



## PROXMOX VE管理ガイド

リリース8.4.0



2025年4月9日

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

著作権© 2025 Proxmox Server Solutions GmbH

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

ライセンスのコピーは「GNU Free Documentation License」の項に含まれています。

# 内容

<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	統合バックアップリストア	3
1.4	高可用性クラスタ	3
1.5	柔軟なネットワーク	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	4
1.7.1	Proxmox VEによるハイパーコンバージドインフラ (HCI) のメリット	4
1.7.2	ハイパーコンバージドインフラストラクチャストレージ	5
1.8	なぜオープンソースなのか	5
1.9	Proxmox VEのメリット	5
1.10	ヘルプ	6
1.10.1	プロックスモックス VE ウィキ	6
1.10.2	コミュニティ支援フォーラム	6
1.10.3	メーリングリスト	6
1.10.4	商業サポート	6
1.10.5	バグトラッカー	6
1.11	プロジェクトの歴史	6
1.12	Proxmox VE ドキュメントの改善	8
1.13	Proxmox VEの翻訳	8
1.13.1	gitを使った翻訳	8
1.13.2	gitを使わない翻訳	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>
2.1 システム要件	10
2.1.1 評価のための最低要件	10
2.1.2 推奨動作環境	11
2.1.3 シンプルなパフォーマンスの概要	11
2.1.4 ウェブ・インターフェイスへのアクセスでサポートされているウェブ・ブラウザ	11
2.2 インストールメディアの準備	12
2.2.1 インストール媒体としてのUSBフラッシュドライブの準備	12
2.2.2 GNU/Linux用説明書	12
2.2.3 macOS用説明書	13
2.2.4 Windows用説明書	14
2.3 Proxmox VEインストーラの使用法	14
2.3.1 インストール後の管理インターフェイスへのアクセス	22
2.3.2 高度なLVM構成オプション	23
2.3.3 高度なZFS設定オプション	24
2.3.4 高度なBTRFS設定オプション	25
2.3.5 ZFSパフォーマンスのヒント	25
2.3.6 nomodesetカーネル・パラメータの追加	25
2.4 無人設置	25
2.5 DebianにProxmox VEをインストール	26
<b>3 ホストシステム管理</b>	<b>27</b>
3.1 パッケージリポジトリ	27
3.1.1 Proxmox VEのリポジトリ	27
3.1.2 Proxmox VEエンタープライズリポジトリ	29
3.1.3 Proxmox VE ノーサブスクリプションリポジトリ	29
3.1.4 Proxmox VE テストリポジトリ	29
3.1.5 Ceph Squidエンタープライズリポジトリ	30
3.1.6 Ceph Squidサブスクリプションなしリポジトリ	30
3.1.7 Ceph Squidテストリポジトリ	30
3.1.8 Ceph Reefエンタープライズリポジトリ	30
3.1.9 Ceph Reefの無サブスクリプションリポジトリ	31
3.1.10 Ceph Reefテストリポジトリ	31
3.1.11 Ceph Quincy Enterpriseリポジトリ	31
3.1.12 Ceph Quincyサブスクリプションなしリポジトリ	31
3.1.13 Ceph Quincyテストリポジトリ	32

3.1.14 古いCephリポジトリ .....	32
3.1.15 Debian フームウェアリポジトリ .....	32
3.1.16 セキュアアパート .....	32
3.2 システムソフトウェアの更新 .....	33
3.3 フームウェアの更新 .....	33
3.3.1 永続的フームウェア .....	33
3.3.2 ランタイムフームウェアファイル .....	34
3.3.3 CPUマイクロコードの更新 .....	34
3.4 ネットワーク構成 .....	36
3.4.1 ネットワーク変更の適用 .....	37
3.4.2 命名規則 .....	37
3.4.3 ネットワーク構成の選択 .....	40
3.4.4 ブリッジを使用したデフォルト構成 .....	41
3.4.5 ルート構成 .....	42
3.4.6 iptablesによるマスカレード (NAT) .....	43
3.4.7 Linuxボンド .....	44
3.4.8 VLAN 802.1Q .....	46
3.4.9 ノードのIPv6の無効化 .....	48
3.4.10 ブリッジでのMAC学習の無効化 .....	49
3.5 時間同期 .....	49
3.5.1 カスタム NTP サーバーの使用 .....	49
3.6 外部メトリックサーバー .....	51
3.6.1 Graphiteサーバーの構成 .....	51
3.6.2 Influxdbプラグインの設定 .....	52
3.7 ディスクの健全性監視 .....	52
3.8 論理ボリュームマネージャ (LVM) .....	53
3.8.1 ハードウェア .....	54
3.8.2 ブートローダー .....	54
3.8.3 ボリュームグループの作成 .....	54
3.8.4 var/lib/vz用の追加LVの作成 .....	55
3.8.5 シンプールのサイズ変更 .....	55
3.8.6 LVM-thinプールの作成 .....	55
3.9 Linux上のZFS .....	56
3.9.1 ハードウェア .....	56
3.9.2 ルートファイルシステムとしてのインストール .....	57

---

3.9.3	ZFS RAIDレベルの考察	58
3.9.4	ZFS dRAID	59
3.9.5	ブートローダー	60
3.9.6	ZFS管理	60
3.9.7	電子メール通知の設定	64
3.9.8	ZFSメモリ使用量の制限	64
3.9.9	ZFS上のSWAP	65
3.9.10	暗号化されたZFSデータセット	66
3.9.11	ZFSの圧縮	67
3.9.12	ZFS専用デバイス	68
3.9.13	ZFSプールの特徴	69
3.10	BTRFS	70
3.10.1	ルートファイルシステムとしてのインストール	70
3.10.2	BTRFSアドミニストレーション	71
3.11	Proxmoxノード管理	73
3.11.1	ウェイクオンLAN	73
3.11.2	タスク履歴	74
3.11.3	バルクゲスト電源管理	74
3.11.4	ファーストゲストブートディレイ	75
3.11.5	一括ゲスト移行	75
3.11.6	バルーン時のRAM使用目標	75
3.12	証明書管理	75
3.12.1	クラスタ内通信証明書	75
3.12.2	API および Web GUI 用の証明書	75
3.12.3	カスタム証明書のアップロード	76
3.12.4	Let's Encrypt (ACME) による信頼できる証明書	76
3.12.5	ACME HTTPチャレンジプラグイン	78
3.12.6	ACME DNS API チャレンジプラグイン	79
3.12.7	ACME証明書の自動更新	80
3.12.8	pvenodeを使用したACMEの例	80
3.13	ホスト・ブートローダー	83
3.13.1	インストーラーが使用するパーティション方式	84
3.13.2	proxmox-boot-toolを使ったESPの内容の同期	84
3.13.3	使用されるブートローダーの決定	87
3.13.4	GRUB	88

---

---

3.13.5 システムブート .....	88
3.13.6 カーネル・コマンドラインの編集 .....	89
3.13.7 次のブートでカーネルバージョンを上書き .....	89
3.13.8 セキュアブート .....	90
3.14 カーネル・サムページ・マージング (KSM) .....	93
3.14.1 KSMの意味 .....	93
3.14.2 KSMの無効化 .....	93
<b>4 グラフィカル・ユーザー・インターフェース</b>	<b>94</b>
4.1 特徴 .....	94
4.2 ログイン .....	95
4.3 GUIの概要 .....	95
4.3.1 ヘッダー .....	96
4.3.2 マイ・セッティング .....	97
4.3.3 リソースツリー .....	97
4.3.4 ログパネル .....	98
4.4 コンテンツパネル .....	98
4.4.1 データセンター .....	99
4.4.2 ノード数 .....	100
4.4.3 ゲスト .....	101
4.4.4 ストレージ .....	103
4.4.5 プール .....	104
4.5 タグ .....	105
4.5.1 Style Configuration .....	106
4.5.2 パーミッション .....	106
4.6 同意バナー .....	107
<b>5 クラスター・マネージャー</b>	<b>108</b>
5.1 必要条件 .....	108
5.2 ノードの準備 .....	109
5.3 クラスタの作成 .....	109
5.3.1 ウェブGUIで作成 .....	110
5.3.2 コマンドラインによる作成 .....	110
5.3.3 同一ネットワーク内の複数のクラスタ .....	110
5.4 クラスタへのノードの追加 .....	111
5.4.1 GUIによるクラスタへのノード参加 .....	111

---

---

5.4.2 コマンドラインによるクラスタへのノード参加.....	112
5.4.3 クラスタネットワークを分離したノードの追加.....	113
5.5 クラスタノードの削除.....	113
5.5.1 再インストールなしのノードの分離.....	116
5.6 定足数.....	117
5.7 クラスター・ネットワーク.....	117
5.7.1 ネットワーク要件.....	117
5.7.2 クラスタ別ネットワーク.....	118
5.7.3 コロシンクの住所.....	121
5.8 コロシンクの冗長性.....	122
5.8.1 既存のクラスタへの冗長リンクの追加.....	122
5.9 Proxmox VEクラスタにおけるSSHの役割.....	124
5.9.1 SSHセットアップ.....	124
5.9.2 .bashrcとその兄弟の自動実行による落とし穴.....	125
5.10 コロシンク外部投票サポート.....	125
5.10.1 QDevice技術概要.....	125
5.10.2 対応セットアップ.....	126
5.10.3 QDevice-Netのセットアップ.....	126
5.10.4 よくある質問.....	127
5.11 コロシンクの構成.....	128
5.11.1 corosync.confの編集.....	128
5.11.2 トラブルシューティング.....	129
5.11.3 Corosync 設定用語集.....	129
5.12 クラスターコールドスタート.....	130
5.13 ゲストVMID自動選択.....	130
5.14 ゲスト移住.....	130
5.14.1 マイグレーションタイプ.....	130
5.14.2 移住ネットワーク .....	131
<b>6 Proxmox クラスタファイルシステム (pmxcfs)</b>	<b>133</b>
6.1 POSIX互換性.....	133
6.2 ファイルアクセス権.....	134
6.3 テクノロジー.....	134
6.4 ファイルシステムのレイアウト.....	134
6.4.1 ファイル.....	134
6.4.2 シンボリックリンク .....	135
6.4.3 デバッグ用の特別なステータス・ファイル (JSON) .....	135

---

6.4.4	デバッグの有効化／無効化	136
6.5	回復	136
6.5.1	クラスタ構成の削除	136
6.5.2	故障したノードからのゲストの回復/移動	136
<b>7</b>	<b>プロックスモックスVEストレージ</b>	<b>138</b>
7.1	ストレージの種類	138
7.1.1	シン・プロビジョニング	139
7.2	ストレージ構成	139
7.2.1	ストレージ・プール	140
7.2.2	一般的なストレージ特性	140
7.3	巻数	142
7.3.1	数量所有権	142
7.4	コマンドラインインターフェイスの使用	142
7.4.1	例	143
7.5	ディレクトリバックエンド	144
7.5.1	構成	145
7.5.2	ファイルの命名規則	145
7.5.3	ストレージ機能	146
7.5.4	例	146
7.6	NFSバックエンド	147
7.6.1	構成	147
7.6.2	ストレージ機能	148
7.6.3	例	148
7.7	CIFSバックエンド	148
7.7.1	構成	149
7.7.2	ストレージ機能	150
7.7.3	例	150
7.8	Proxmoxバックアップサーバ	150
7.8.1	構成	151
7.8.2	ストレージ機能	152
7.8.3	暗号化	152
7.8.4	例CLIによるストレージの追加	153
7.9	GlusterFSバックエンド	154
7.9.1	構成	154
7.9.2	ファイルの命名規則	154
7.9.3	ストレージ機能	154

---

7.9.4	例	155
7.10	ローカルZFSプール・バックエンド	155
7.10.1	構成	155
7.10.2	ファイルの命名規則	156
7.10.3	ストレージ機能	156
7.10.4	例	156
7.11	LVMバックエンド	157
7.11.1	構成	157
7.11.2	ファイルの命名規則	157
7.11.3	ストレージ機能	158
7.11.4	例	158
7.12	LVMシン・バックエンド	158
7.12.1	構成	158
7.12.2	ファイルの命名規則	159
7.12.3	ストレージ機能	159
7.12.4	例	159
7.13	オープンiSCSIイニシエータ	160
7.13.1	構成	160
7.13.2	ファイルの命名規則	160
7.13.3	ストレージ機能	161
7.13.4	例	161
7.14	ユーザーモード iSCSI バックエンド	161
7.14.1	構成	161
7.14.2	ストレージ機能	161
7.15	Ceph RADOSブロックデバイス (RBD)	162
7.15.1	構成	162
7.15.2	認証	163
7.15.3	Cephクライアントの構成(オプション)	164
7.15.4	ストレージ機能	164
7.16	Cephファイルシステム (CephFS)	164
7.16.1	構成	165
7.16.2	認証	166
7.16.3	ストレージ機能	166
7.17	BTRFSバックエンド	167
7.17.1	構成	167
7.17.2	スナップ写真	167

---

---

7.18 ZFS over iSCSIバックエンド	168
7.18.1 構成	168
7.18.2 ストレージ機能	170
<b>8 ハイパーコンバージドCephクラスタの展開</b>	<b>171</b>
8.1 はじめに	171
8.2 用語解説	172
8.3 健全なCephクラスタの推奨事項	172
8.4 Cephの初期インストールと構成	175
8.4.1 ウェブベースのウィザードの使用	175
8.4.2 CephパッケージのCLIインストール	177
8.4.3 CLIによるCephの初期構成	177
8.5 Cephモニター	178
8.5.1 モニター作成	178
8.5.2 モニター破棄	179
8.6 Cephマネージャ	179
8.6.1 クリエイトマネージャー	179
8.6.2 デストロイ・マネージャー	179
8.7 Ceph OSD	180
8.7.1 OSDの作成	180
8.7.2 OSDの破棄	182
8.8 セフ・プール	183
8.8.1 プールの作成と編集	183
8.8.2 消去符号化プール	185
8.8.3 デストロイ・プールズ	187
8.8.4 PGオートスケーラー	187
8.9 Ceph CRUSHおよびデバイスクラス	188
8.10 Cephクライアント	190
8.11 CephFS	191
8.11.1 メタデータ・サーバー (MDS)	191
8.11.2 CephFSの作成	192
8.11.3 CephFSの破棄	193
8.12 Cephメンテナンス	193
8.12.1 OSDの交換	193
8.12.2 トリム/廃棄	194
8.12.3 スクラブ & ディープスクラブ	194
8.12.4 Proxmox VE+ Ceph HCI Clusterのシャットダウン	194

---

8.13 Cephの監視とトラブルシューティング	195
8.13.1 トラブルシューティング	195
<b>9 ストレージ・レプリケーション</b>	<b>197</b>
9.1 対応ストレージタイプ	197
9.2 スケジュール形式	198
9.3 エラー処理	198
9.3.1 考えられる問題	198
9.3.2 エラー時のゲストの移行	198
9.3.3 例	198
9.4 求人管理	199
9.5 ネットワーク	200
9.6 コマンドラインインターフェイスの例	200
<b>10 QEMU/KVM仮想マシン</b>	<b>201</b>
10.1 エミュレートされたデバイスと準仮想化デバイス	201
10.2 仮想マシンの設定	202
10.2.1 一般設定	202
10.2.2 OSの設定	203
10.2.3 システム設定	203
10.2.4 ハードディスク	205
10.2.5 CPU	207
10.2.6 メモリー	212
10.2.7 メモリの暗号化	214
10.2.8 ネットワーク機器	216
10.2.9 表示	218
10.2.10 USBバススルー	219
10.2.11 BIOS と UEFI	219
10.2.12 トラステッド・プラットフォーム・モジュール (TPM)	220
10.2.13 VM間共有メモリ	221
10.2.14 オーディオ機器	221
10.2.15 VirtIO RNG	222
10.2.16 ヴァーチョフ	222
10.2.17 デバイスの起動順序	224
10.2.18 仮想マシンの自動起動とシャットダウン	224
10.2.19 QEMUゲストエージェント	225
10.2.20 SPICEの強化	227
10.3 マイグレーション	228

---

10.3.1 オンライン移行.....	228
10.3.2 オフライン移行.....	229
10.4 コピーとクローン .....	229
10.5 仮想マシンテンプレート .....	230
10.6 VMジェネレーションID .....	231
10.7 仮想マシンのインポート .....	231
10.7.1 インポートウィザード.....	232
10.7.2 CLIによるOVF/OVAのインポート .....	234
10.8 クラウドイニットサポート .....	235
10.8.1 Cloud-Initテンプレートの準備 .....	236
10.8.2 Cloud-Initテンプレートのデプロイ .....	238
10.8.3 カスタムクラウドイニット構成.....	238
10.8.4 Windowsでのクラウドイニット .....	239
10.8.5 Cloudbase-Initテンプレートの準備.....	239
10.8.6 Cloudbase-InitとSysprep .....	240
10.8.7 クラウドイニット固有のオプション .....	241
10.9 PCI(e)バススルー .....	242
10.9.1 一般要件 .....	243
10.9.2 ホスト・デバイス・バススルー .....	245
10.9.3 SR-IOV .....	248
10.9.4 媒介デバイス (vGPU、GVT-g) .....	248
10.9.5 クラスターでの使用 .....	249
10.9.6 vIOMMU (エミュレートされたIOMMU) .....	249
10.10 フックスクリプト .....	250
10.11 冬眠.....	250
10.12 リソースマッピング .....	251
10.13 qm による仮想マシンの管理 .....	253
10.13.1 CLI の使用例 .....	254
10.14 構成 .....	254
10.14.1 ファイル形式 .....	255
10.14.2 スナップショット .....	255
10.14.3 オプション .....	256
10.15 ロック .....	283
<b>11 Proxmoxコンテナツールキット</b>	<b>285</b>
11.1 技術概要 .....	285
11.2 ディストリビューション .....	286
11.2.1 アルバイン・リナックス .....	286
11.2.2 アーチリナックス .....	286
11.2.3 CentOS, Almalinux, Rocky Linux.....	287
11.2.4 デビアン .....	287

---

---

11.2.5 デヴァン	288
11.2.6 フェドラ	288
11.2.7 ジェンツー	288
11.2.8 オープンソース	288
11.2.9 ウブントゥ	288
11.3 コンテナ画像	289
11.4 コンテナ設定	290
11.4.1 一般設定	290
11.4.2 CPU	291
11.4.3 メモリー	292
11.4.4 マウント・ポイント	293
11.4.5 ネットワーク	296
11.4.6 コンテナの自動スタートとシャットダウン	297
11.4.7 フックスクリプト	298
11.5 セキュリティ	298
11.5.1 AppArmor	299
11.5.2 対照群 (cgroup)	299
11.6 ゲストオペレーティングシステムの構成	300
11.7 コンテナ保管	302
11.7.1 ヒューズマウント	302
11.7.2 コンテナ内でのクォータの使用	302
11.7.3 コンテナ内でのACLの使用	303
11.7.4 コンテナマウントポイントのバックアップ	303
11.7.5 コンテナのマウントポイントの複製	303
11.8 バックアップと復元	304
11.8.1 コンテナ・バックアップ	304
11.8.2 コンテナ・バックアップの復元	304
11.9 pctによるコンテナの管理	305
11.9.1 CLI の使用例	305
11.9.2 デバッグログの取得	306
11.10 マイグレーション	306
11.11 構成	307
11.11.1 ファイル形式	307
11.11.2 スナップ写真	308
11.11.3 オプション	308
11.12 ロック	314

---

<b>12 ソフトウェア定義ネットワーク</b>	<b>315</b>
12.1 はじめに	315
12.2 サポート状況	315
12.2.1 沿革	315
12.2.2 現在の状況	316
12.3 設置場所	316
12.3.1 SDNコア	316
12.3.2 DHCP IPAM	316
12.3.3 FRルーティング	317
12.4 構成の概要	317
12.5 技術と構成	317
12.6 ゾーン	318
12.6.1 共通オプション	318
12.6.2 シンプルゾーン	318
12.6.3 VLANゾーン	319
12.6.4 QinQゾーン	319
12.6.5 VXLANゾーン	319
12.6.6 EVPNゾーン	320
12.7 Vネット	321
12.8 サブネット	322
12.9 コントローラ	323
12.9.1 EVPNコントローラ	323
12.9.2 BGPコントローラ	323
12.9.3 ISISコントローラ	324
12.10 アイパム	324
12.10.1 PVE IPAM プラグイン	324
12.10.2 NetBox IPAM プラグイン	325
12.10.3 phplIPAM プラグイン	325
12.11 DNS	325
12.11.1 PowerDNS プラグイン	326
12.12 DHCP	326
12.12.1 構成	326
12.12.2 プラグイン	327
12.13 ファイアウォールの統合	328
12.13.1 VNetsとサブネット	328
12.13.2 アイパム	329
12.14 例	330

---

12.14.1 シンプルゾーンの例	330
12.14.2 ソースNATの例	330
12.14.3 VLAN の設定例	331
12.14.4 QinQセットアップの例	331
12.14.5 VXLAN セットアップの例	332
12.14.6 EVPN セットアップの例	333
12.15 備考	334
12.15.1 複数のEVPN出口ノード	334
12.15.2 VXLAN IPSEC暗号化	335
<b>13 Proxmox VE ファイアウォール</b>	<b>336</b>
13.1 アクセス&ゾーン	336
13.1.1 道順	336
13.1.2 ゾーン	337
13.2 設定ファイル	337
13.2.1 クラスターワイドセットアップ	337
13.2.2 ホスト固有の設定	339
13.2.3 VM/コンテナの構成	341
13.2.4 VNetの設定	342
13.3 ファイアウォールのルール	343
13.4 セキュリティグループ	344
13.5 IPエイリアス	345
13.5.1 標準IPエイリアス local_network	345
13.6 IPセット	345
13.6.1 標準IPセット管理	346
13.6.2 標準IPセットブラックリスト	346
13.6.3 標準IPセット ipfilter-net*	346
13.7 サービスとコマンド	346
13.8 デフォルトのファイアウォールルール	347
13.8.1 データセンター発着 DROP/REJECT	347
13.8.2 VM/CT着信/発信 DROP/REJECT	348
13.9 ファイアウォールルールのログ	348
13.9.1 ユーザー定義ファイアウォールルールのログ	349
13.10 ヒントとコツ	349
13.10.1 FTPを許可する方法	349
13.10.2 スリカータIPS統合	350
13.11 IPv6に関する注意事項	350

---

13.12 Proxmox VEが使用するポート	350
13.13 nftables	351
13.13.1 インストールと使用方法	351
13.13.2 使用方法	352
13.13.3 役立つコマンド	352
<b>14 ユーザー管理</b>	<b>354</b>
14.1 ユーザー	354
14.1.1 システム管理者	355
14.2 グループ	355
14.3 APIトークン	355
14.4 リソースプール	355
14.5 認証領域	356
14.5.1 Linux PAM 標準認証	356
14.5.2 Proxmox VE 認証サーバー	356
14.5.3 LDAP	357
14.5.4 Microsoft Active Directory (AD)	358
14.5.5 LDAP ベースのレルムの同期	358
14.5.6 OpenID Connect	361
14.6 二要素認証	363
14.6.1 利用可能なセカンドファクター	363
14.6.2 レルム強制二要素認証	364
14.6.3 二要素認証の制限とロックアウト	364
14.6.4 ユーザーによる TOTP 認証の設定	364
14.6.5 TOTP	366
14.6.6 ウェブオート	366
14.6.7 リカバリーキー	366
14.6.8 サーバーサイドWebauthnの設定	367
14.6.9 サーバー側U2Fの設定	367
14.6.10 ユーザーとしてのU2Fの有効化	368
14.7 許可管理	368
14.7.1 役割	368
14.7.2 特典	369
14.7.3 オブジェクトとパス	371
14.7.4 プール	372
14.7.5 どの権限が必要ですか？	372
14.8 コマンドラインツール	373

---

<b>14.9 実例</b>	374
14.9.1 管理者グループ	374
14.9.2 監査役	374
14.9.3 ユーザー管理の委任	374
14.9.4 モニタリング用限定APIトークン	375
14.9.5 リソースプール	375
<b>15 高い可用性</b>	376
15.1 必要条件	377
15.2 リソース	378
15.3 管理タスク	378
15.4 仕組み	379
15.4.1 サービス団	380
15.4.2 地域資源マネージャー	381
15.4.3 クラスタリソースマネージャ	382
15.5 HAシミュレータ	383
15.6 構成	384
15.6.1 リソース	384
15.6.2 グループ	386
15.7 フェンシング	388
15.7.1 ブロックスモックスVEフェンス	388
15.7.2 ハードウェア・ウォッチドッグの設定	389
15.7.3 フェンスサービス	389
15.8 スタート失敗ポリシー	390
15.9 エラーリカバリー	390
15.10 パッケージの更新	390
15.11 ノードのメンテナンス	391
15.11.1 メンテナンスマード	391
15.11.2 シャットダウン・ポリシー	392
15.12 クラスタリソースのスケジューリング	394
15.12.1 基本的なスケジューラ	394
15.12.2 静的負荷スケジューラ	394
15.12.3 CRSスケジューリングポイント	395
<b>16 バックアップと復元</b>	396
16.1 バックアップモード	396
16.1.1 VMバックアップの偽り取り	398

---

---

16.1.2 CT変化検出モード	398
16.2 バックアップファイル名	399
16.3 バックアップファイルの圧縮	399
16.4 バックアップ暗号化	400
16.5 バックアップの仕事	400
16.6 バックアップ保持	402
16.6.1 プルーンシミュレータ	403
16.6.2 保持設定例	403
16.7 バックアップ保護	404
16.8 バックアップノート	404
16.9 リストア	405
16.9.1 帯域制限	405
16.9.2 ライブ・レストア	406
16.9.3 単一ファイルの復元	406
16.10 構成	407
16.11 フックスクリプト	410
16.12 ファイルの除外	410
16.13 例	411
<b>17 お知らせ</b>	<b>412</b>
17.1 概要	412
17.2 通知対象	412
17.2.1 センドメール	413
17.2.2 SMTP	414
17.2.3 ゴティファイ	415
17.2.4 ウェブフック	416
17.3 通知マッチャー	418
17.3.1 マッチャーのオプション	419
17.3.2 カレンダーのマッチングルール	419
17.3.3 フィールドマッチング規則	419
17.3.4 重症度照合ルール	420
17.3.5 例	420
17.4 通知イベント	420
17.5 システムメール転送	421
17.6 アクセス許可	421
17.7 通知モード	421

---

<b>18 重要なサービス・デーモン</b>	<b>422</b>
18.1 pvedaemon - Proxmox VE API デーモン	422
18.2 pveproxy - Proxmox VE API プロキシデーモン	422
18.2.1 ホストベースのアクセス制御	422
18.2.2 リスニングIPアドレス	423
18.2.3 SSL暗号スイート	424
18.2.4 対応TLSバージョン	424
18.2.5 ディフィー・ヘルマン・パラメーター	424
18.2.6 代替HTTPS証明書	425
18.2.7 レスポンス・コンプレッション	425
18.2.8 リアルクライアントIPログ	425
18.3 pvestatd - Proxmox VE ステータスデーモン	426
18.4 spiceproxy - SPICE プロキシサービス	426
18.4.1 ホスト・ベースのアクセス制御	426
18.5 pvescheduler - Proxmox VE スケジューラデーモン	426
<b>19 便利なコマンドラインツール</b>	<b>427</b>
19.1 pvesubscription - サブスクリプション管理	427
19.2 pveperf - Proxmox VE ベンチマークスクリプト	427
19.3 Proxmox VE API用シェルインターフェイス	428
19.3.1 使用例	428
<b>20 よくある質問</b>	<b>429</b>
<b>21 書誌</b>	<b>432</b>
21.1 ブロックスマックスVEに関する書籍	432
21.2 関連技術に関する書籍	432
21.3 関連書籍	433
<b>A コマンドラインインターフェース</b>	<b>434</b>
A.1 一般	434
A.2 出力フォーマットのオプション [FORMAT_OPTIONS]	434
A.3 pvesm - Proxmox VEストレージマネージャ	435
A.4 pvesubscription - Proxmox VEサブスクリプションマネージャ	449
A.5 pveperf - Proxmox VEベンチマークスクリプト	450
A.6 pveceph - Proxmox VEノードでCEPHサービスを管理	450
A.7 pvenode - Proxmox VEノード管理	458
A.8 pvesh - Proxmox VE API用シェルインターフェイス	466

<b>A.9</b> <code>qm</code> - QEMU/KVM 仮想マシンマネージャ	467
<b>A.10</b> <code>qmrestore</code> - QemuServer <code>vzdump</code> バックアップのリストア	513
<b>A.11</b> <code>pct</code> - Proxmox コンテナツールキット	513
<b>A.12</b> <code>pveam</code> - Proxmox VE アプライアンスマネージャ	538
<b>A.13</b> <code>pvecm</code> - Proxmox VE クラスタマネージャ	539
<b>A.14</b> <code>pvesr</code> - Proxmox VEストレージレプリケーション	542
<b>A.15</b> <code>pveum</code> - Proxmox VE ユーザーマネージャ	547
<b>A.16</b> <code>vzdump</code> - VMとコンテナのバックアップユーティリティ	563
<b>A.17</b> <code>ha-manager</code> - Proxmox VE HA マネージャ	566
<b>B</b> サービスデーモン	571
<b>B.1</b> <code>pve-firewall</code> - Proxmox VE ファイアウォールデーモン	571
<b>B.2</b> <code>pvedaemon</code> - Proxmox VE API デーモン	572
<b>B.3</b> <code>pveproxy</code> - Proxmox VE API プロキシデーモン	573
<b>B.4</b> <code>pvestatd</code> - Proxmox VE ステータスデーモン	574
<b>B.5</b> <code>spiceproxy</code> - SPICE プロキシサービス	574
<b>B.6</b> <code>pmxcfs</code> - Proxmox クラスタファイルシステム	575
<b>B.7</b> <code>pve-ha-crm</code> - クラスタリソースマネージャデーモン	575
<b>B.8</b> <code>pve-ha-lrm</code> - ローカルリソースマネージャデーモン	576
<b>B.9</b> <code>pvescheduler</code> - Proxmox VE スケジューラデーモン	577
<b>C</b> 設定ファイル	578
<b>C.1</b> 一般	578
<b>C.1.1</b> 物件名のキャッシング	578
<b>C.2</b> データセンターの構成	578
<b>C.2.1</b> ファイル形式	578
<b>C.2.2</b> オプション	579
<b>D</b> カレンダー イベント	584
<b>D.1</b> スケジュール形式	584
<b>D.2</b> 詳細仕様	584
<b>D.2.1</b> 例	585
<b>E</b> QEMU vCPUリスト	587
<b>E.1</b> はじめに	587
<b>E.2</b> インテルCPUの種類	587
<b>E.3</b> AMD CPUの種類	588
<b>F</b> ファイアウォール・マクロの定義	589

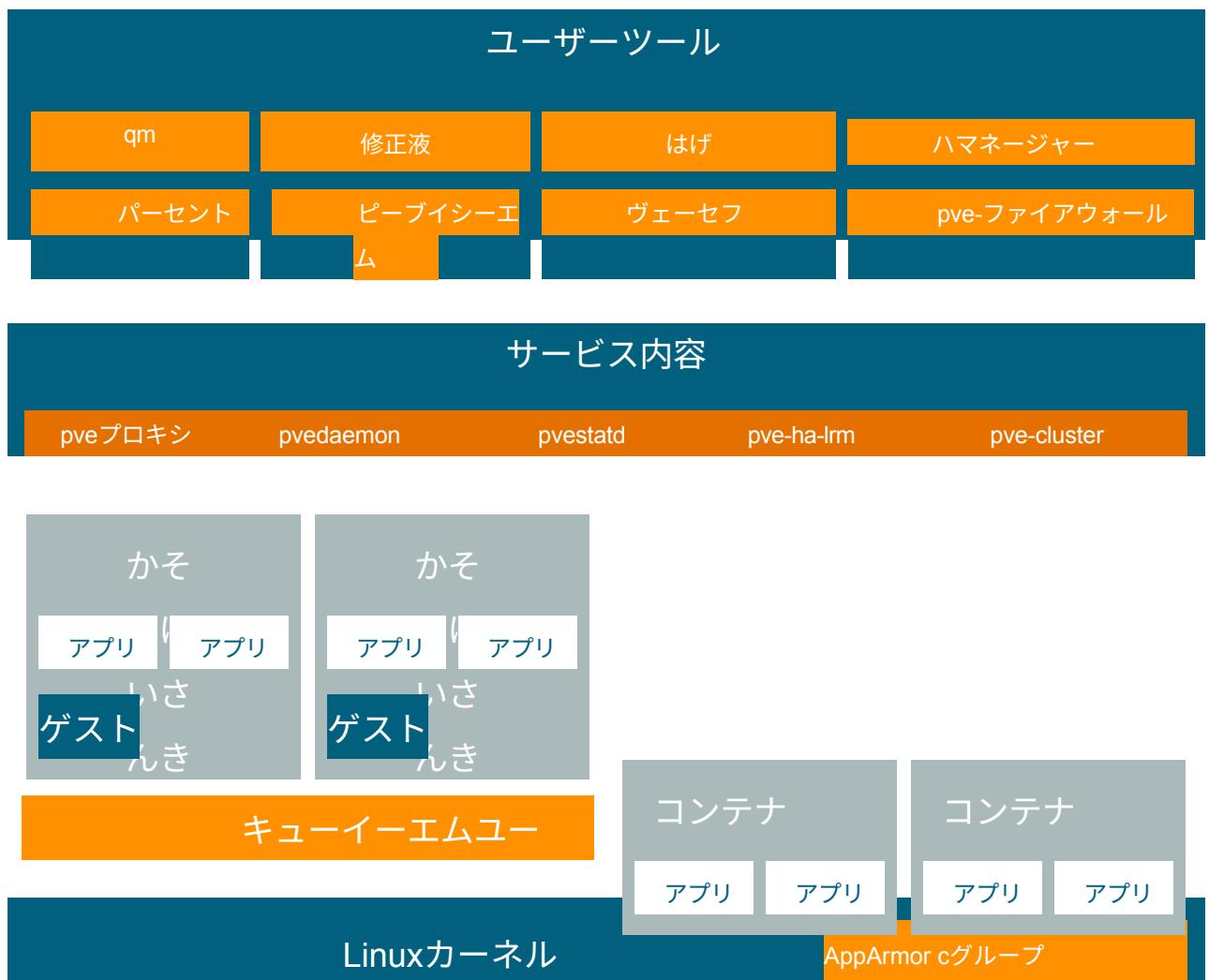
---

<b>G マークダウン入門</b>	<b>604</b>
G.1 マークダウンの基本	604
G.1.1 見出し	604
G.1.2 強調	604
G.1.3 リンク	605
G.1.4 リスト	605
G.1.5 テーブル	606
G.1.6 ブロック名言集	606
G.1.7 コードとスニペット	606
<b>H GNU自由文書ライセンス</b>	<b>608</b>

## 第1章 はじめに

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

主な設計目標の1つは、管理をできるだけ簡単にすることでした。Proxmox VEは単一ノードで使用することも、多数のノードからなるクラスタを構築することもできます。すべての管理タスクはWebベースの管理インターフェイスを使用して行うことができ、初心者ユーザーでもProxmox VEのセットアップとインストールを数分で行うことができます。



## 1.1 中央管理

多くの人はシングルノードから始めますが、Proxmox VEは大規模なクラスタノードにスケールアウトできます。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出荷されます。

### ユニークなマルチマスター・デザイン

統合されたWebベースの管理インターフェイスにより、すべてのKVMゲストやLinuxコンテナ、さらにはクラスタ全体の概要をすっきりと見渡すことができます。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。複雑で高価な管理サーバを別途インストールする必要はありません。

### Proxmox クラスタファイルシステム (pmxcfs)

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

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

### ウェブベースの管理インターフェイス

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

### コマンドライン

UnixシェルやWindows Powershellの快適さに慣れている上級ユーザのために、Proxmox VEは仮想環境のすべてのコンポーネントを管理するコマンドラインインターフェースを提供します。このコマンドラインインターフェースは、インテリジェントなタブ補完とUNIX manページ形式の完全なドキュメントを備えています。

### REST API

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

### 役割ベースの管理

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

### 認証領域

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

## 1.2 柔軟なストレージ

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

共有ストレージにVMを格納する主な利点の1つはクラスタ内のすべてのノードがVMディスクイメージに直接アクセスできるため、ダウントIMEなしに実行中のマシンをライブマイグレーションできることです。

現在、以下のネットワーク・ストレージ・サポートしています：

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

サポートされるローカル・ストレージ・タイプは以下の通りです：

- 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・キャッシング・デバイス)
- ディレクトリ (既存のファイルシステム上のストレージ)
- セットエフェス

## 1.3 統合されたバックアップリストア

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

KVMライブバックアップは、NFS、CIFS、iSCSI LUN、Ceph RBD上のVMイメージを含むすべてのストレージタイプで動作します。新しいバックアップフォーマットは、VMバックアップを高速かつ効果的に保存するために最適化されています（疎なファイル、順序のずれたデータ、最小化されたI/O）。

## 1.4 高可用性クラスタ

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

## 1.5 柔軟なネットワーキング

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

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

## 1.6 統合ファイアウォール

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

## 1.7 ハイパーコンバージドインフラ

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

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

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

ハイパーコンバージドインフラ (HCI) は、高いインフラ需要が低い管理予算に見合う展開、リモートオフィスやブランチオフィス環境などの分散セットアップ、仮想プライベートクラウドやパブリッククラウドに特に有効です。

HCIには次のような利点があります：

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

## 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を展開するためのサポートを緊密に統合しています。例えば、ウェブインターフェースのみを使用して、以下の2つのストレージテクノロジを展開および管理できます：

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

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

## 1.8 なぜオープンソースなのか

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

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

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

## 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
- 迅速な設置と使いやすさ
- ウェブベースの管理インターフェイス
- REST API
- 巨大で活発なコミュニティ
- 低い管理コストとシンプルな導入

## 1.10 ヘルプ

### 1.10.1 プロックスモックスVE ウィキ

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

### 1.10.2 コミュニティ・サポート・フォーラム

Proxmox VE自体は完全にオープンソースであるため、[Proxmox VEコミュニティフォーラム](#)を使用して、ユーザが議論し、ノウハウを共有することを常に奨励しています。このフォーラムはProxmoxのサポートチームによって管理されており、世界中から多くのユーザーが参加しています。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

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

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

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

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

### 1.10.4 コマーシャルサポート

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

ボリュームディスクアント、または一般的な詳細については、[sales@proxmox.com](mailto:sales@proxmox.com)までお問い合わせください。

### 1.10.5 バグトラッカー

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

## 1.11 プロジェクトの歴史

プロジェクトは2007年に始まり、2008年に最初の安定版がリリースされました。当時、コンテナにはOpenVZを、仮想マシンにはKVMとQEMUを使用していました。クラスタリング機能は限定的で、ユーザーインターフェイスはシンプル（サーバーが生成するウェブページ）でした。

しかし、私たちはCorosyncクラスタスタックを使った新しい機能をすぐに開発しました。新しいProxmoxクラスタファイルシステム (pmxefs) の導入は大きな前進でした。16ノードのクラスタを管理するのは、1つのノードを管理するのと同じくらい簡単です。

JSON-Schemaで記述された完全な宣言的仕様の新しいREST APIを導入したこと、他の人々がProxmox VEをインフラストラクチャに統合することが可能になり、追加サービスを簡単に提供できるようになりました。

また、新しいREST APIによって、オリジナルのユーザーインターフェイスをJavaScriptを使っ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、古いJavaベースのVNCコンソールコードをnoVNCに置き換えました。そのため、VMを管理するために必要なのはウェブブラウザだけです。

様々なストレージタイプのサポートも大きな課題です。特に、Proxmox VEは2014年にLinux上でZFSをデフォルトで出荷した最初のディストリビューションです。もう1つのマイルストーンは、ハイパーバイザーノード上でCephストレージを実行・管理できるようになったことです。このようなセットアップは非常に費用対効果が高いです。

私たちのプロジェクトが始まったとき、私たちはKVMの商用サポートを提供する最初の企業のひとつでした。KVMプロジェクト自体は継続的に進化し、今では広く使われているハイパーバイザーです。リリースのたびに新しい機能が追加されています。私たちはKVMライブバックアップ機能を開発し、どのようなストレージタイプでもスナップショットバックアップを作成できるようにしました。

バージョン4.0で最も注目すべき変更は、OpenVZからLXCへの移行です。コンテナが深く統合され、仮想マシンと同じストレージとネットワーク機能を使用できるようになりました。同時に、使いやすいハイアベイラビリティ (HA) マネージャを導入し、高可用性セットアップの設定と管理を簡素化しました。

Proxmox VE 5の開発中に、非同期ストレージレプリケーションとACME/Let's Encryptを使用した自動認証管理が導入されました。

Software Defined Network (SDN) スタックは私たちのコミュニティと協力して開発されました。バージョン 6.2 で実験的な機能としてウェブインターフェースに統合され、高度なネットワーク設定の管理を簡素化しました。バージョン 8.1 以降、SDN の統合は完全にサポートされ、デフォルトでインストールされます。

2020年、Rustプログラミング言語で書かれた新しいプロジェクトProxmox Backup Serverが<sup>1</sup>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、重複排除などの実装によりバックアップ機能を大幅に向上させます。

もう一つの新しいツールであるProxmox Offline Mirrorは2022年にリリースされ、公衆インターネットに接続されていないシステム用のサブスクリプションを可能にしました。

2023年、要望の多かったWebインターフェースのダークテーマが導入されました。同年末、バージョン8.0はCephエンタープライズリポジトリへのアクセスを統合しました。現在、最も安定したCephリポジトリへのアクセスは、Proxmox VEのサブスクリプションに付属しています。

バージョン8.2では、公式ISOインストーラの自動インストールと無人導入され、Proxmox VEの大規模な導入が大幅に簡素化されました。

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、VMware ESXi<sup>(1)</sup>のような他のハイパーバイザーからゲストを簡単かつ効率的に直接移行することができます。さらに、Open Virtualization Format (OVF/OVA)のアーカイブをファイルベースのストレージからWebインターフェースで直接インポートできるようになりました。

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

## 1.12 Proxmox VEドキュメントの改善

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

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

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

- ウィキ特定のセットアップ、ハウツーガイド、チュートリアルについては、wikiが貢献するのに適したオプションです。
- リファレンス・ドキュメントすべてのユーザーに役立つ一般的な内容については、リファレンスドキュメンテーションをご提案ください。これにはProxmox VEの機能のインストール、設定、使用方法、トラブルシューティングに関するすべての情報が含まれます。リファレンスドキュメントはasciidoc形式で書かれています。リファレンス・ドキュメントはasciidoc形式で書かれています。  
ドキュメントを編集するには、`://git.proxmox.com/git/pve-docs` にある git リポジトリをクローンする必要があります。  
をクリックし、[README.adoc](#) ドキュメントに従ってください。

---

### 備考

Proxmox VEのコードベースに興味がある方は、[Developer Documentation](#) wikiの記事から始めることができます。

---

## 1.13 Proxmox VEの翻訳

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

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

### 1.13.1 gitを使った翻訳

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

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

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

または、お好みのエディタを使用して、既存の翻訳を編集することができます：

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

## 1.13.2 gitを使わない翻訳

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

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

## 1.13.3 翻訳のテスト

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

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

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

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

```
# make deb
```



### 重要

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

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

## 1.13.4 翻訳の送信

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

## 第2章

# Proxmox VEのインストール

Proxmox VEはDebianをベースにしています。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パッケージが含まれています。

### チップ

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

インストーラは、ローカルディスクのパーティション設定、基本的なシステム設定 (例: タイムゾーン、言語、ネットワーク) の適用、必要なパッケージのインストールなど、セットアップをガイドします。このプロセスは数分もかかりません。提供された ISO を使ってインストールするのが、新規および既存のユーザに推奨される方法です。

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

## 2.1 システム要件

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

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

### 2.1.1 評価のための最低条件

これらの最小要件は評価のみを目的としており、環境では使用しないでください。

- CPU64ビット (インテル64またはAMD64)
- KVM完全仮想化対応のIntel VT/AMD-V対応CPU/マザーボード
- RAM: 1 GB RAM、およびゲストに必要な追加RAM
- ハードドライブ
- ネットワークカード (NIC) 1枚

## 2.1.2 推奨システム要件

- Intel 64またはIntel VT/AMD-V CPUフラグを持つAMD64。
- メモリOSおよびProxmox VEサービス用に最低2GB、さらにゲスト用に指定されたメモリが必要です。CephとZFSについては、使用するストレージのTBごとに約1GBのメモリが必要です。
- 高速で冗長性のあるストレージは、SSDで最高の結果が得られます。
- OSストレージ：バッテリ保護ライトキャッシュ（「BBU」）付きハードウェアRAIDを使用するか、ZFS（ZIL用オプションSSD）付き非RAIDを使用します。
- VMストレージ：
  - ローカルストレージには、バッテリバックアップライトキャッシュ(BBU)付きのハードウェアRAIDを使用するか、ZFSとCephには非RAIDを使用します。ZFSもCephもハードウェアRAIDコントローラとは互換性がありません。
  - 共有および分散ストレージが可能です。
  - 良好な得るには、Power-Loss-Protection (PLP) 付きのSSDを推奨します。民生用SSDの使用は推奨されません。
- 冗長(マルチ)Gbit NIC。ご希望のストレージ技術やクラスタ設定に応じてNICを追加できます。
- PCI(e)バススルーのためには、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 シンプルなパフォーマンスの概要

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

---

### 備考

これは一般的なベンチマークです。特にシステムのI/O性能については、より詳細なテストをお勧めします。

---

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

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

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

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

## 2.2 インストールメディアの準備

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

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

- CDやDVDに書き込める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)にコピー可能な生セクタ(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を使用することをお勧めします。

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

フラッシュ、少なくとも1GBのストレージが必要です。

---

#### 備考

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

---



#### 重要

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

---

### 2.2.2 GNU/Linux用説明書

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

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

---

#### 備考

/dev/XYZを正しいデバイス名に置き換え、入力ファイル名（もしあれば）のパスを合わせるようにしてください。

---



#### 注意

間違ったディスクを上書きしないよう、十分注意してください！

---

### 正しいUSBデバイス名の検索

USBフラッシュ・ドライブの名前を調べるには、2つの方法があります。1つ目は、フラッシュ・ドライブを接続する前と後のdmesgコマンド出力の最終行を比較する方法です。もう1つは、lsblkコマンドの出力を比較する方法です。ターミナルを開いて実行します：

```
# lsblk
```

その後、USBフラッシュ・ドライブを接続し、もう一度コマンドを実行してください：

```
# lsblk
```

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

### 2.2.3 macOS用説明書

ターミナルを開きます。

hdiutilのconvert .isoファイルを.dmg形式に変換します：

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

---

#### チップ

macOSは出力ファイル自動的に.dmgを付ける傾向があります。

---

現在のデバイスのリストを取得するには、次のコマンドを実行します：

```
# diskutil list
```

USBフラッシュ・ドライブを挿入し、このコマンドをもう一度実行して、どのデバイス・ノードが割り当てられているかを確認します。(例：/dev/diskX)。

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

---

#### 備考

を最後のコマンドの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
```

---

#### 備考

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

---

## 2.2.4 Windows用説明書

### エッチャーの使用

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

### ルーファスの使用

Rufusはより軽量な代替品ですが、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。インストールするか、ポータブル版を使用してください。インストール先ドライブと Proxmox VE ISO ファイルを選択します。



#### 重要

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

## 2.3 Proxmox VEインストーラの使用

インストーラーのISOイメージには以下が含まれています：

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

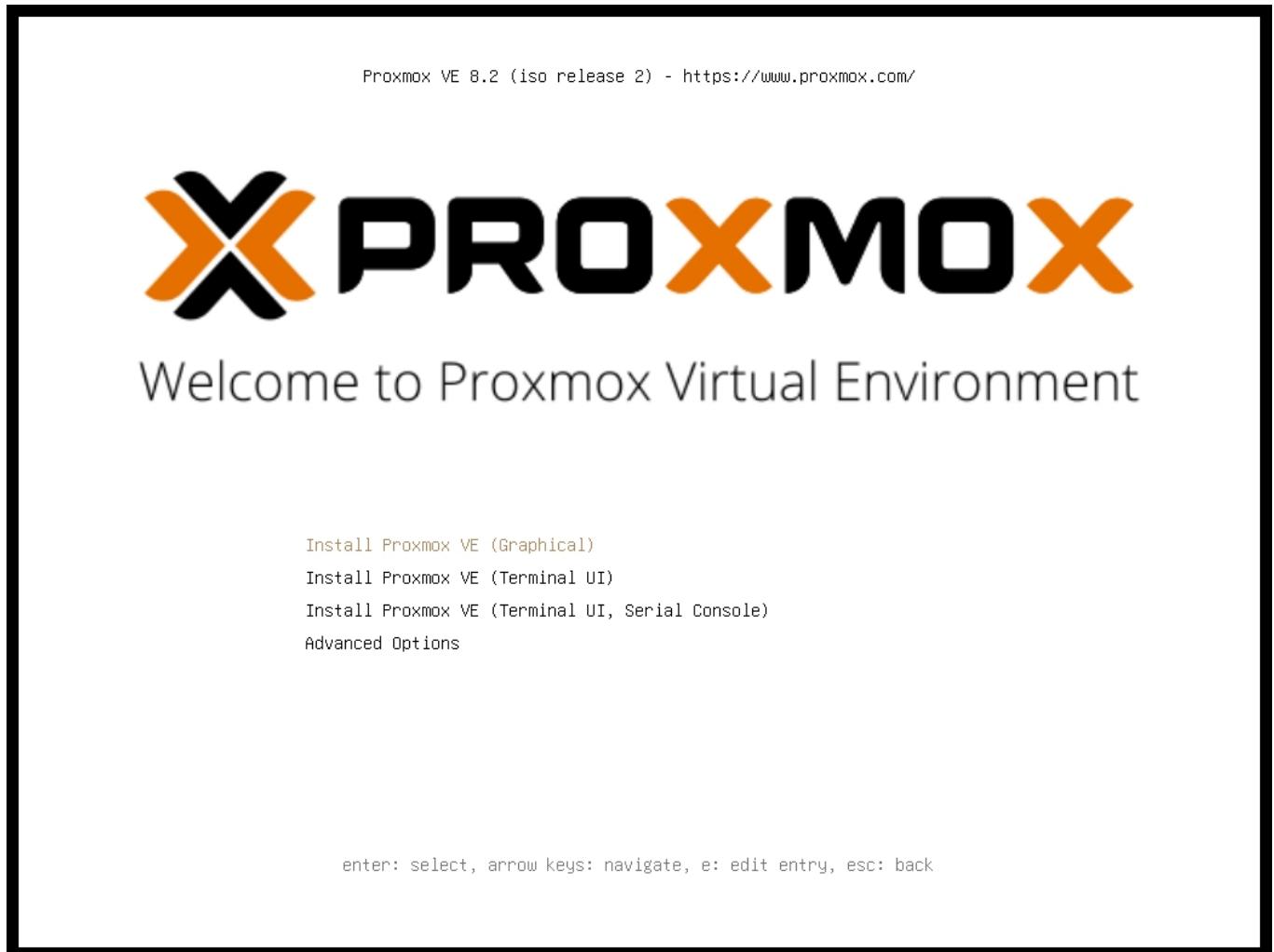
#### 備考

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

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

#### チップ

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



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

#### Proxmox VE (グラフィカル)のインストール

通常のインストールを開始します。

#### チップ

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

#### Proxmox VE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起動します。グラフィカルインストーラと同じ全体的なインストールエクスペリエンスを提供しますが、非常に古いハードウェアや非常に新しいハードウェアとの互換性が一般的に優れています。

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

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



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

#### チップ

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

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

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

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

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

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

ターミナル・ベースのデバッグ・モードと同じですが、Linuxカーネルがマシンの（最初の）シリアル・ポートを入出力に使うように設定します。

#### 高度なオプション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、無人モードでインストーラを起動します。このオプションは、ハードウェアの詳細を収集したり、自動インストールのセットアップをデバッグするのに便利です。詳細は、[無人インストール](#)を参照してください。

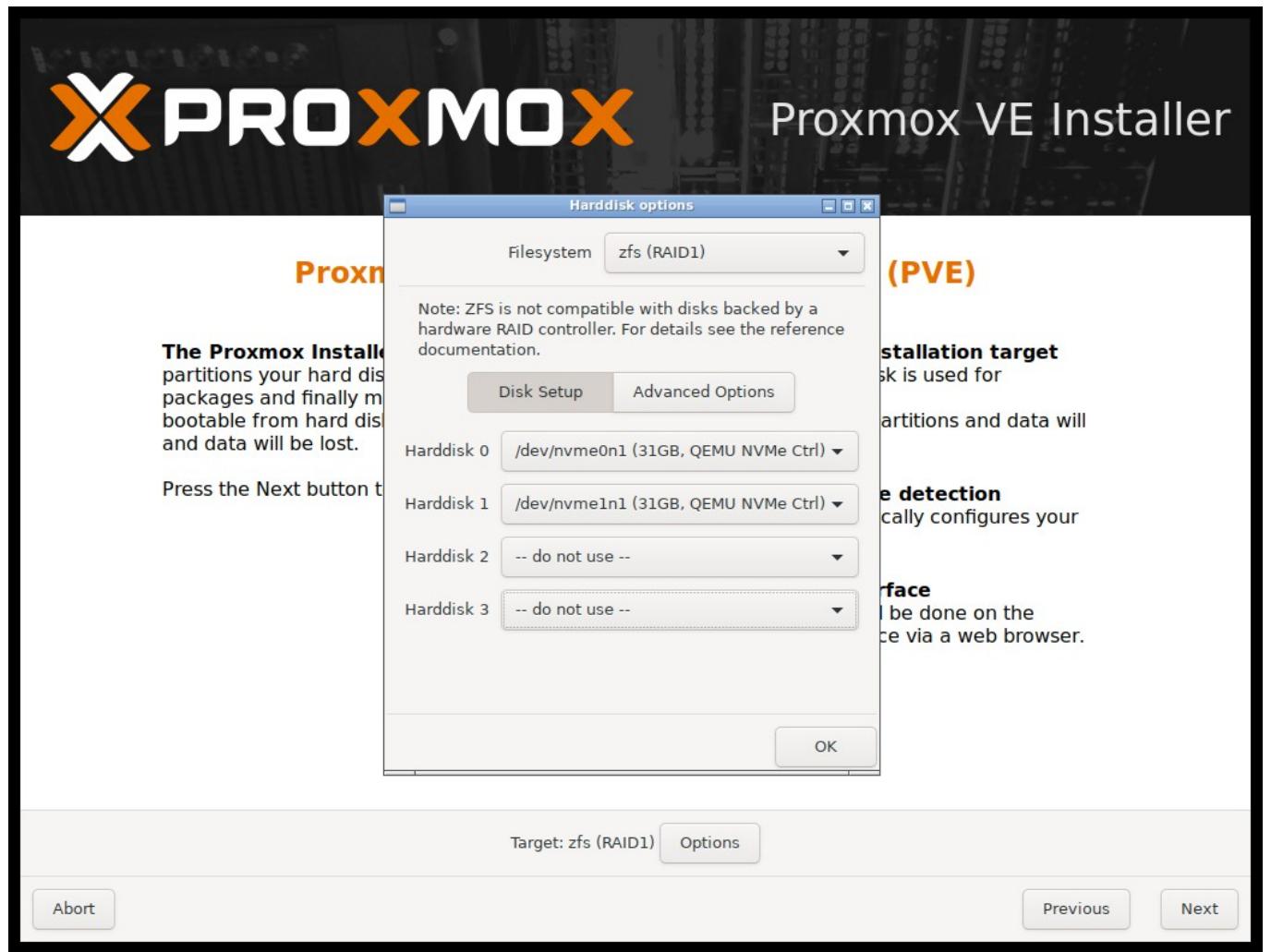
### 詳細オプションレスキューブート

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

### 高度なオプションメモリのテスト (memtest86+)

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

通常はProxmox VEのインストール（グラフィカル）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 (End User License Agreement)を読むことです。続いて、インストール先のハードディスクを選択します。



#### 注意

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

[オプション] ボタンで、ターゲットファイルシステムを選択できます。デフォルトは ext4 です。ファイルシステムとして ext4 または xfs を選択した、インストーラは LVM を使用し、LVM 領域を制限する追加オプションを提供します ([下記参照](#))。

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

**警告**

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

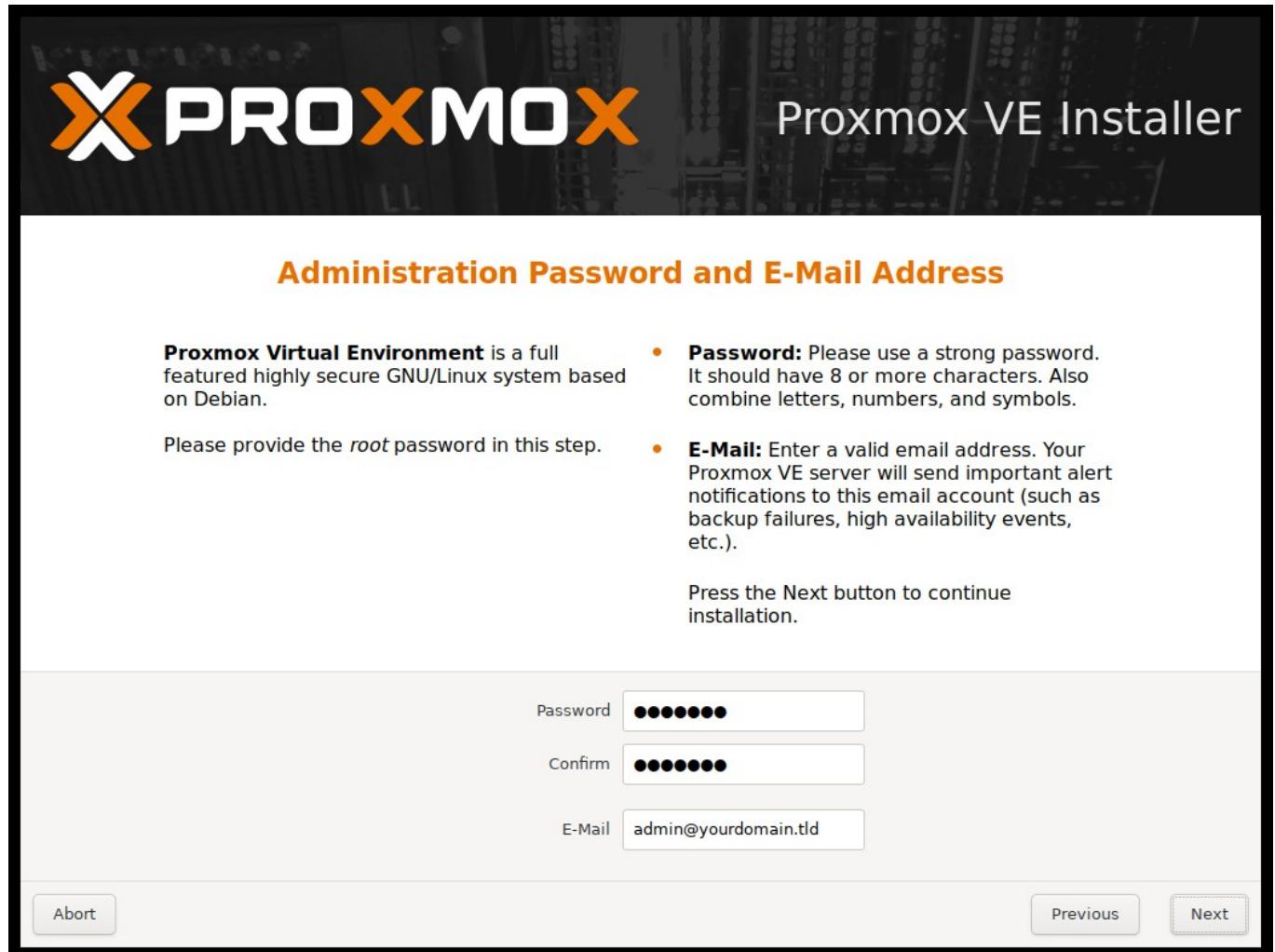
The screenshot shows the Proxmox VE Installer interface. At the top left is the Proxmox logo. To its right, the text "Proxmox VE Installer" is displayed. Below this, the title "Location and Time Zone selection" is centered in orange. On the left, there is a text block about the installer's automatic optimizations and a note to press "Next" to continue. On the right, there is a list of three configuration options with bullet points. At the bottom, there are dropdown menus for "Country" (set to Austria), "Time zone" (set to Europe/Vienna), and "Keyboard Layout" (set to German). Navigation buttons "Abort", "Previous", and "Next" are at the bottom.

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

Press the Next button to continue installation.

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

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

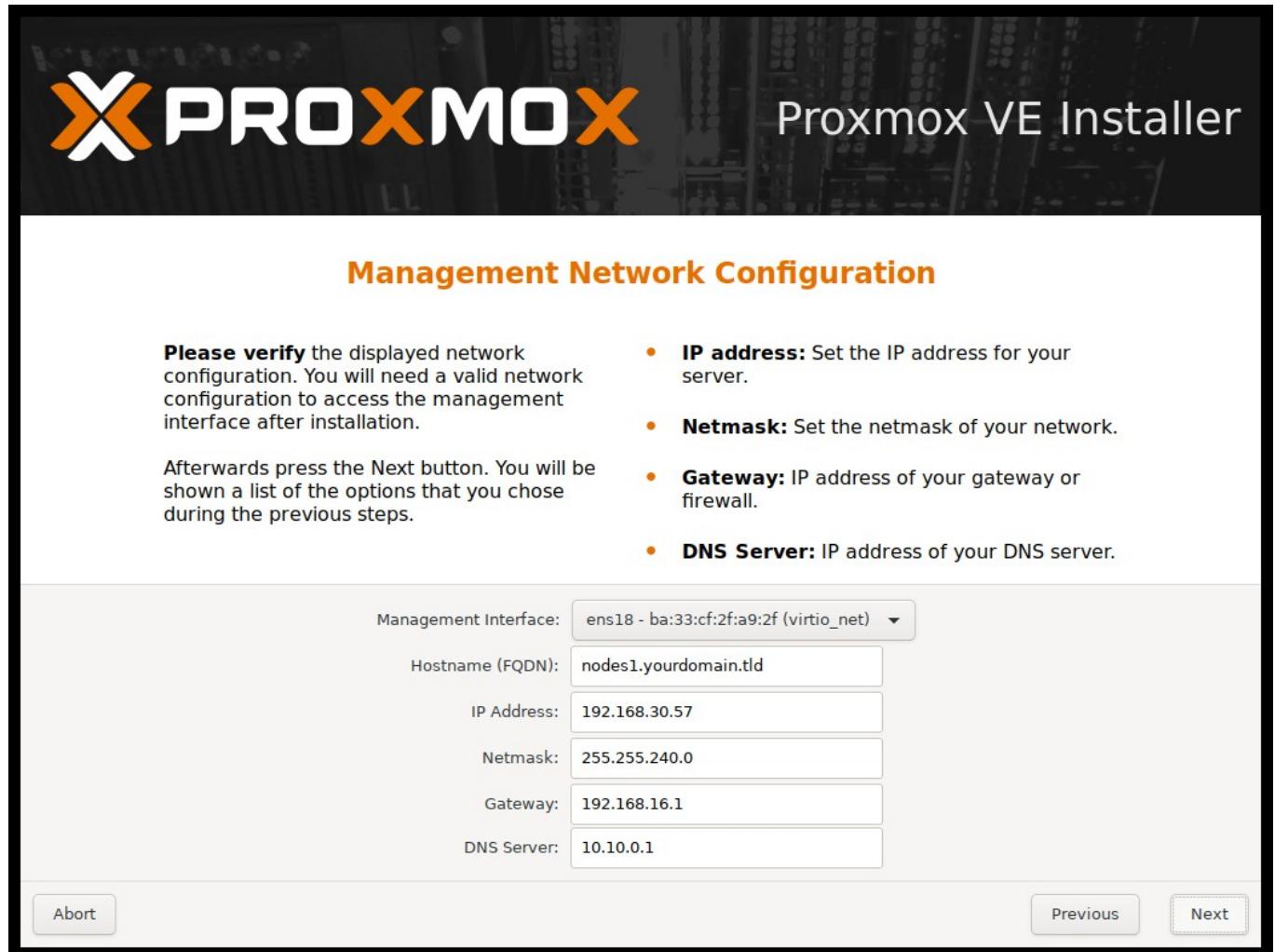


次に、スーパーユーザー (root) のパスワードとメールアドレスを指定する必要があります。パスワードは少なくとも8文字で構成されなければなりません。より強力なパスワードを使用することを強くお勧めします。いくつかのガイドラインがあります:

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

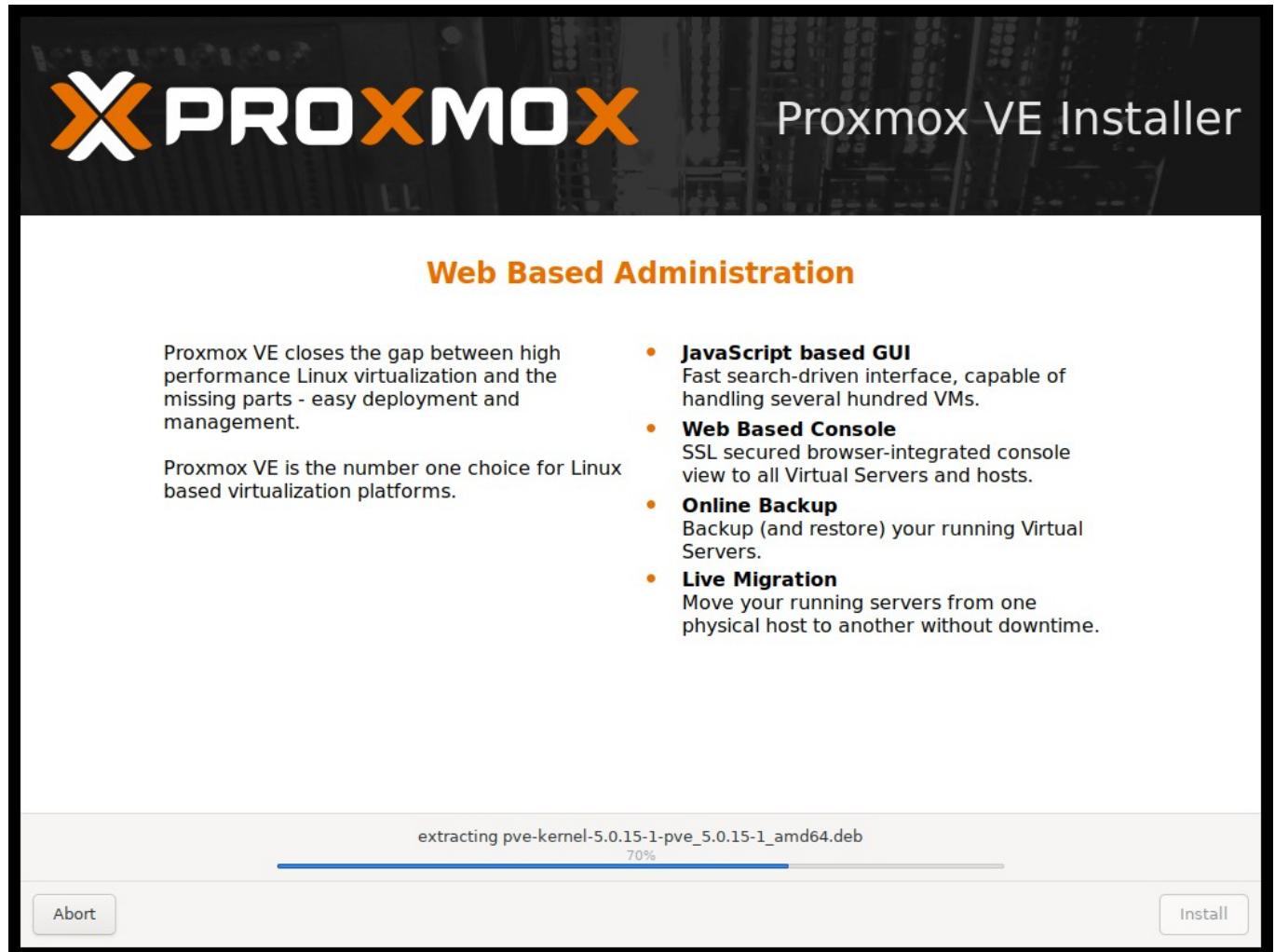
電子メールアドレスは、システム管理者に通知を送信するために使用されます。例えば

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



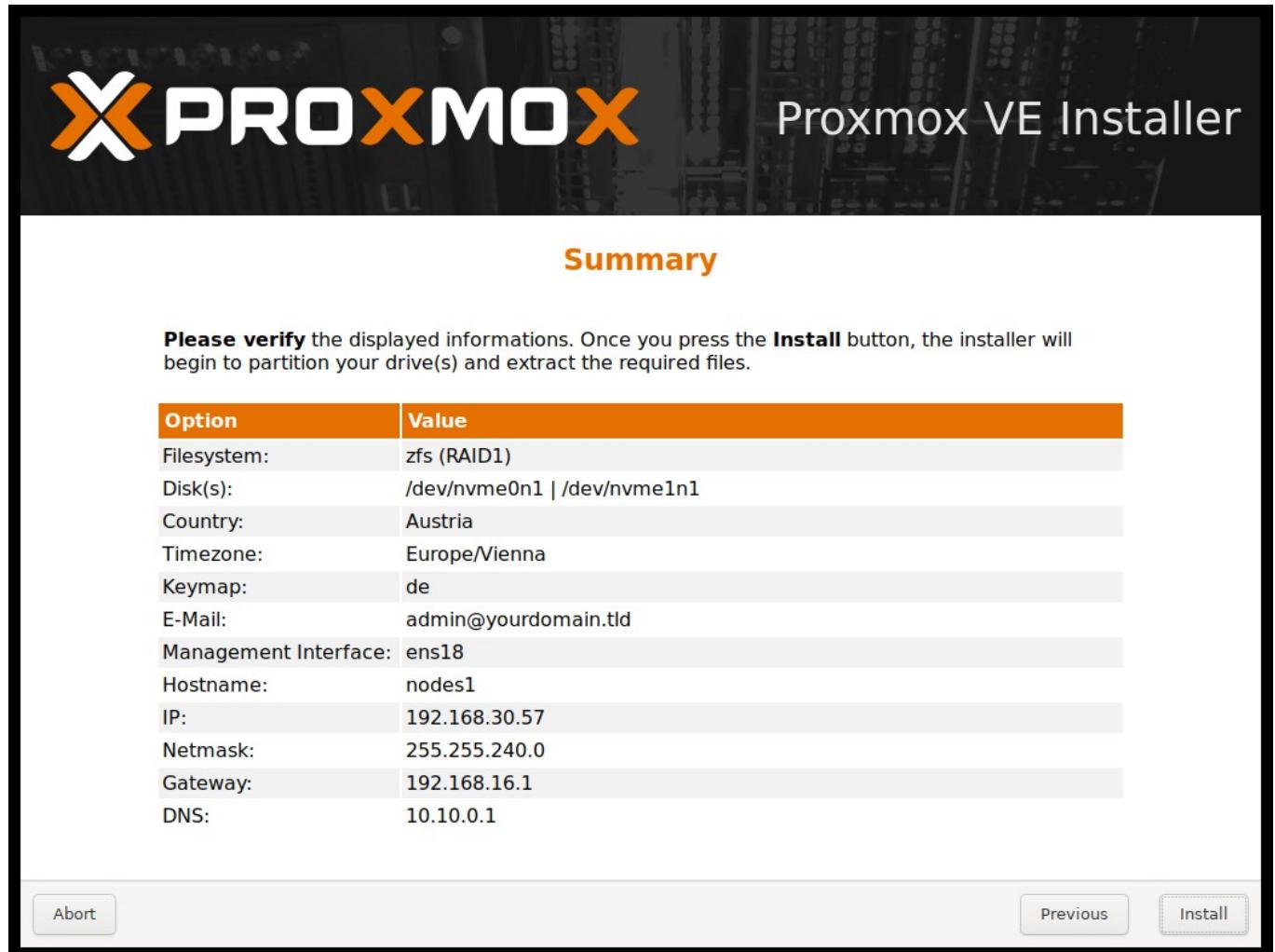
これらの通知メールはすべて指定されたメールアドレスに送信されます。

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



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

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



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

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

#### インストールの失敗

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

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

### 2.3.1 インストール後の管理インターフェイスへのアクセス

Proxmox VE Login

User name:

Password:

Realm: Linux PAM standard authentication

Language: English

Save User name:  Login

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

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

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

#### ハードサイズ

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

#### スワップサイズ

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

---

#### 備考

0に設定すると、スワップ・ボリュームは作成されません。

---

#### マックスルート

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

#### マックスズズ

データボリュームの最大サイズを定義します。データボリュームの実際のサイズは `datasize= hdszie - rootsize - swapsize - minfree` ここで `datasize` は `maxvz` より大きくできません。

---

#### 備考

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

---

---

**備考**

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

---

**ミンフリー**

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

---

**備考**

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

---

### 2.3.3 高度なZFS設定オプション

ZFS を使用する場合、インストーラは ZFS プール rpool を作成します。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にスワップ用のパーティション化されていない領域を確保することができます。インストール後に swap zvol を作成することもできますが、これは問題につながる可能性があります ([ZFS swap notes を参照](#))。

**アッシュシフト**

作成されるプールの ashift 値を定義します。ashift は少なくとも sector- に設定する必要があります。

ディスクのサイズ (2 の ashift 乗がセクタサイズ)、またはプールに入れられる可能性のあるディスク (欠陥のあるディスクの交換など)。

**湿布**

rpool で圧縮を有効にするかどうかを定義します。

**チェックサム**

rpool にどのチェックサムアルゴリズムを使用するかを定義します。

**コピー**

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

**ARC最大サイズ**

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

**ハードサイズ**

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

### 2.3.4 高度なBTRFS設定オプション

BTRFSを使用する場合、スワップ領域は作成されませんが、スワップ用にインストールディスク上のパーティション化されていない領域を確保することができます。`btrfs filesystem mkswapfile` コマンドを使って、別パーティション、BTRFS サブボリューム、またはスワップファイルを作成できます。

#### 湿布

BTRFS サブボリュームで圧縮を有効にするかどうかを定義します。`on` (`zlib`と同等)、`zlib`、`lzo`、`zstd`の異なる圧縮アルゴリズムがサポートされています。デフォルトは`off`です。

#### ハードサイズ

使用するハードディスクの合計サイズを定義します。これは、ハードディスク上の空き領域を保存して、さらにパーティション分割（例えば、スワップパーティションの作成）を行うのに便利です。

### 2.3.5 ZFSパフォーマンスのヒント

ZFSは大容量のメモリで最もよく動作します。ZFSを使用する、十分なRAMを用意してください。TBのRAWディスク容量に対して、4GBと1GBのRAMを追加するのが良い計算です。

ZFSは、ZFS Intent Log (ZIL) と呼ばれる専用ドライブを書き込みキャッシュとして使用できます。これには高速ドライブ (SSD) を使用します。インストール

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

後に以下のコマンドで追加できます：

### 2.3.6 nomodesetカーネル・パラメーターの追加

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

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

次に `Ctrl-X` または `F10` を押して設定をブートします。

## 2.4 無人設置

自動インストール方式により、Proxmox VEを無人でインストールできます。これにより、ベアメタル上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 のような自動化ツールを使用してインストールをさらに設定できます。

インストーラに必要なオプションは、アンサーファイルで提供する必要があります。このファイルにより、ファイルタールを使用して、使用するディスクとネットワークカードを決定できます。

自動インストールを使用するには、まず、アンサーファイルの取得元を選択し、その選択でインストール ISO を準備する必要があります。

ISO が準備されると、最初のブートメニューに *Automated Installation* という新しいブートエントリーが表示されます。

10秒間のタイムアウトの後、自動的に選択されます。

無人インストールの詳細と情報については、[ウィキをご覧ください](#)。

## 2.5 DebianにProxmox VEをインストール

Proxmox VEはDebianパッケージのセットとして出荷され、標準的なDebianインストールの上にインストールすることができます。リポジトリを設定した後、以

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

以下のコマンドを実行する必要があります：

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

一般的に、特にLVMやZFSを使用する場合、これは些細なことではありません。詳細なステップバイステップのハウツーが[wiki](#)にあります。

## 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および運用に関するProxmox VEの仕様について説明します。

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

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

## 3.1 パッケージリポジトリ

Proxmox VEは、他のDebianベースの同様に、パッケージ管理ツールとしてAPTを使用します。

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

### 3.1.1 Proxmox VEのリポジトリ

リポジトリはソフトウェアパッケージの集合体であり、新しいソフトウェアをインストールするために使用できますが、新しいアップデートを取得するためにも重要です。

---

#### 備考

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

---

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

## リポジトリ管理

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

基本的なリポジトリ管理、たとえばリポジトリの有効化・無効化もサポートされています。

## ソースリスト

`sources.list`、各行がパッケージ・リポジトリを定義します。優先するソースが最初になければなりません。空行は無視されます。行のどこかに # 文字があると、その行の残りがコメントとしてマークされます。リポジトリから利用可能なパッケージは、`apt-get update` を実行して取得します。アップデートは `apt-get` を使って直接インストールすることも、GUI (Node→ Updates) を使ってインストールすることもできます。

### ファイル `/etc/apt/sources.list`

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

# セキュリティ・アップデート
deb http://security.debian.org/debian-security bookworm-security main'

コントリビューション
```

Proxmox VEは3つの異なるパッケージリポジトリを提供します。

### 3.1.2 Proxmox VEエンタープライズリポジトリ

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

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

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

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

#### 備考

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

### 3.1.3 Proxmox VE ノーサブスクリプションリポジトリ

名前が示すようにこのリポジトリにアクセスするためにサブスクリプションキーは必要ありません。テストや本番環境以外での使用に使用できます。本番サーバーでの使用は推奨されません。なぜなら、これらのパッケージは常に厳重にテスト・検証されているわけではないからです。

このリポジトリは /etc/apt/sources.list で設定することをお勧めします。

#### ファイル /etc/apt/sources.list

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

# Proxmox VE pve-no-subscription リポジトリは proxmox.com によって提供されています # 本番環境での利用は推奨されていません
デブ http://download.proxmox.com/debian/pve 本の虫 pve-no-subscription

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

### 3.1.4 Proxmox VE テストリポジトリ

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

#### pvetestのsources.listエントリ

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



##### 警告

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

### 3.1.5 Ceph Squidエンタープライズリポジトリ

このリポジトリには、エンタープライズProxmox VE Ceph 19.2 Squidパッケージがあります。います。Proxmox VEでCephクライアントまたは完全なCephクラスタを実行する場合は、このリポジトリを使用します。

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

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

### 3.1.6 Ceph Squidサブスクリプションなしリポジトリ

このCephリポジトリには、エンタープライズリポジトリに移動する前のCeph 19.2 Squidパッケージと、テストリポジトリに移動した後のCeph 19.2 Squidパッケージが格納されています。

#### 備考

本番マシンにはエンタープライズリポジトリを使用することをお勧めします。

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

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

### 3.1.7 Ceph Squidテストリポジトリ

このCephリポジトリには、メインリポジトリに移動する前のCeph 19.2 Squidパッケージが格納されています。Proxmox VEでの新しいCephリリースのテストに使用されます。

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

```
deb http://download.proxmox.com/debian/ceph-squid ブックワームテスト
```

### 3.1.8 Ceph Reefエンタープライズリポジトリ

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

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

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

### 3.1.9 Ceph Reefサブスクリプションなしリポジトリ

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

**備考**

本番マシンにはエンタープライズリポジトリを使用することをお勧めします。

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

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

### 3.1.10 Ceph Reefテストリポジトリ

このCephリポジトリには、メインリポジトリに移動する前のCeph 18.2 Reefパッケージが格納されています。Proxmox VEでの新しいCephリリースのテストに使用されます。

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

```
deb http://download.proxmox.com/debian/ceph-reef ブックワームテスト
```

### 3.1.11 Ceph Quincy Enterpriseリポジトリ

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

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

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

### 3.1.12 Ceph Quincyサブスクリプションなしリポジトリ

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

**備考**

本番マシンにはエンタープライズリポジトリを使用することをお勧めします。



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

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

### 3.1.13 Ceph Quincy テストリポジトリ

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

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

```
deb http://download.proxmox.com/debian/ceph-quincy ブックワーム テスト
```

### 3.1.14 古いCephリポジトリ

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

詳細については、[各アップグレードガイド](#)を参照してください。

### 3.1.15 Debian ファームウェアリポジトリ

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

[Early OS Microcode Updates](#) をセットアップする場合や、プリインストールパッケージ pve-firmware に含まれていないランタイムファームウェアファイルが必要な場合は、このリポジトリを有効にしてください。

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

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

### 3.1.16 セキュアアパート

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

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

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

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

その後、sha512sum CLIツールでチェックサムを検証します:

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

またはmd5sum CLIツール:

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

## 3.2 システムソフトウェアの更新

Proxmoxはすべてのリポジトリに対して定期的にアップデートを提供します。アップデートをインストールするには、WebベースのGUIまたは以下のCLIコマンド

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

を使用します:

---

### 備考

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

---

### チップ

定期的なアップデートは、最新のパッチとセキュリティ関連の修正を取得するために不可欠です。主要なシステムアップグレードは [Proxmox VEコミュニティフォーラム](#) で発表されます。

---

## 3.3 ファームウェア・アップデート

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

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

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

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

### 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しています。通常、BIOS/UEFI アップデートに含まれる更新されたマイクロコードはマザーボードに保存され、その他のファームウェアはそれぞれのデバイスに保存されます。この永続的な方法は、ブート時に更新されたマイクロコードをできるだけ早く定期的にロードできるようにするため、CPUにとって特に重要です。



#### 注意

BIOS/UEFIやストレージコントローラなどの一部のアップデートでは、デバイスの設定がリセットされる可能性があります。ベンダーの指示に注意深く従い、現在の設定をバックアップしてください。

---

どのアップデート方法が利用可能か、ベンダーにご確認ください。

- サーバーの便利な更新方法には、デルのLifecycle ManagerやHPEのサービスパックがあります。
- Linux利用できる場合があります。  
例えば、NVIDIA ConnectX用の*mlxup*や  
*bnxtnvm/niccli* は Broadcom ネットワークカード用です。
- ハードウェアベンダーとの協力があり、[サポートされているハードウェアが](#)使用されている場合、[LVFS](#) もオプションです。このための技術的条件は、システムが2014年以降に製造され、UEFI経由で起動することです。

Proxmox VE は、Proxmox 署名キーによるセキュアブートサポートを有効にするため、独自のバージョンの fwupd パッケージを出荷しています。このパッケージは、ハイパーバイザー上の udisks2 の使用に関する問題が確認されているため、udisks2 package への依存の推奨を意識的に取りやめています。これは、例えば /etc/fwupd/daemon.conf で EFI パーティションの正しいマウントポイントを明示的に設定する必要があることを意味します：

#### ファイル /etc/fwupd/daemon.conf

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

#### チップ

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

### 3.3.2 ランタイム・ファームウェア・ファイル

このメソッドは Proxmox VE オペレーティングシステムにファームウェアを保存し、[パーシステッド ファームウェアが](#)最新でないデバイスにファームウェアを渡します。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サポートされていますが、マザーボードやハードディスクなどのパーシステッドファームウェアに依存しているデバイスではサポートされていません。

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

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

追加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した場合、APTはインストールを中断します。もしかしたら、そのファームウェアは別の方法で入手できるかもしれません。

### 3.3.3 CPUマイクロコードアップデート

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

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

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

### 早期OSマイクロコードアップデートの設定

Linuxカーネルによってブート時に早期に適用されるマイクロコード・アップデートを設定するには、次のことが必要です：

1. [Debian フームウェアリポジトリの有効化](#)
2. 利用可能な最新のパッケージを取得 `apt update` (またはウェブインターフェースの Node→Updates 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します：
  - インテルCPUの場合: `apt install intel-microcode`
  - AMD CPUの場合: `apt install amd64-microcode`
4. Proxmox VEホストの再起動

今後マイクロコードのアップデートを行う場合も、ロードのために再起動が必要になります。

### マイクロコードバージョン

比較やデバッグの目的で、現在実行中のマイクロコード・リビジョンを取得します：

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

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

新しいマイクロコードパッケージをインストールして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に組み込まれたバージョンとマザーボードのファームウェアのバージョンの両方よりも新しい場合システムログに "microcode updated early" というメッセージが表示されます。

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

### トラブルシューティング

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

1. ホストが[安全に](#)再起動できることを確認してください。
2. ホストを再起動してGRUBメニューを表示します。

3. を押します。
4. linuxで始まる行に行き、スペースで区切って追加 **dis\_ucode\_ldr**
5. CTRL-Xを押して、Early OS Microcode Updateなしで起動します。

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

この例で示すように、Debian リポジトリに以前の microcode パッケージバージョンがあれば、ダウングレードは可能です：

```
# apt list -a intel-microcode Listing...完了
intel-microcode/stable-security,now 3.20230808.1~deb12u1 amd64 [インストール済み] intel-microcode/stable
3.20230512.1 amd64
```

```
# apt install intel-microcode=3.202305*
...
intel-/にバージョン「3.20230512.1」(Debian:12.1/stable [amd64])を選択。
マイクロコード
...
dpkg: 警告: intel-microcode を 3.20230808.1~deb12u1 から'にダウングレードしています
3.20230512.1
...
intel-microcode: マイクロコードは次のブート時に更新されます。
...
ホストが安全に再起動できることを（もう一度）確認してください。CPU タイプ用のマイクロコード・潜在的に含まれている古いマイクロコードを適用するには、今すぐ再起動してください。
```

---

#### チップ

ダウングレードしたパッケージをしばらくの間保持し、後日より新しいバージョンを試してみるのは理にかなっています。将来パッケージのバージョンが同じになったとしても、その間にシステムアップデートによって経験した問題が修正されているかもしれません。

```
# apt-mark hold intel-microcode intel-
microcode を保留にします。
```

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

---

## 3.4 ネットワーク構成

Proxmox VEはLinuxネットワークスタックを使用しています。このため、Proxmox VEノードのネットワーク設定に柔軟性があります。コンフィギュレーションは、GUI 経由でも、手動でファイル /etc/network/interfaces はネットワーク全体の設定を含んでいます。interfaces (5)

マニュアルのページに完全なフォーマットの説明があります。すべてのProxmox VEツールは、ユーザーが直接修正できるように努力していますが、GUIを使用する方がエラーから保護されるため、まだ好ましいです。

Linux ブリッジ・インターフェース（一般に `vmbrX` と呼ば）は、ゲストを基礎となる物理ネットワークに接続するために必要です。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を考えることができます。このセクションでは、[ボンドによる冗長化](#)、[VLAN](#)、ルーティングや[NAT](#)のセットアップなど、さまざまなユースケースに対応するためにネットワークをどのようにセットアップできるか、いくつかの例を示します。

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

#### 警告

従来の Debian ツール `ifup` と `ifdown` は、`vmbrX` の `ifdown` すべてのゲストのトラフィックを中断させるが、後で同じブリッジで `ifup` を実行したときにそれらのゲストを再接続しないといった落とし穴があるので、よくわからない場合は使うのをお勧めしません。

### 3.4.1 ネットワーク変更の適用

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

#### ifupdown2によるライブリロードネットワーク

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

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

#### 備考

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

#### ノードを再起動して適用

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

### 3.4.2 命名規則

現在、デバイス名には以下の命名規則を使用しています：

- イーサネットデバイス: en\*、systemd ネットワークインターフェース名。この命名法はバージョン5.0以降、Proxmox VEの新規インストールに使用されます。
- イーサネットデバイス: eth[N], 0≤ N (eth0, eth1, ...) この命名法は、5.0リリース以前にインストールされたProxmox VEホストに使用されます。5.0にアップグレードしても、名前はそのままです。
- ブリッジ名: 一般的にはvmbr [N] で、0≤ N≤ 4094 (vmbr0 - vmbr4094)ですが、文字で始まり、最大10文字までの英数字文字列を使用できます。
- ボンド: bond[N]、ここで 0≤ N (bond0, bond1, ...)
- VLAN: VLAN番号をピリオドで区切ってデバイス名に追加するだけです (eno1.50, bond1.30)。デバイス名がデバイスのタイプを意味するため、ネットワークの問題をデバッグしやすくなります。

### Systemd ネットワークインターフェース名

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

- o<index>[n<phys\_port\_name>|d<dev\_port>] - ボード上のデバイス。
- s<slot>[f<function>] [n<phys\_port\_name>|d<dev\_port>] - ホットプラグIDによるデバイス。
- [P<domain>]p<bus>s<slot>[f<function>] [n<phys\_port\_name>|d<dev\_port>] - バスIDでデバイスを指定します。
- 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ましょう:

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

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

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

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

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

## 特定のネーミングスキームのバージョンをピン留め

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

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

8.0のインストールでは、以下のカーネル・コマンドライン・パラメータを追加します:

```
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する必要があります。

## ネットワークデバイス名の上書き

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

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

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]` セクションでそのデバイスを一意かつ恒久的に識別する方法が必要です。`[MACAddress]` オプションを使ってデバイスのMACアドレスと一致させることもできます。

`[Match]` セクションには、同じMACアドレスを持つブリッジ/ボンド/VLANインターフェイスではなく、期待される物理インターフェイスのみにマッチするように、Typeオプションも含める必要があります。ほとんどのセットアップでは、Type イーサネットデバイスだけにマッチするように `ether` に設定されるべきですが、セットアップによっては他の選択が必要になるかもしれません。詳細は [systemd.link\(5\) man ページ](#) を参照してください。

次に、[リンク]セクションの[名前]オプションを使用して名前を割り当てることができます。

リンク・ファイルはinitramfsにコピーされるため、リンク・ファイルを追加、変更、削除した後はinitramfsをリフレッシュすることをお勧めします:

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

例えば、`enwan0` という名前を MAC アドレス `aa:bb:cc:dd:ee:ff` のイーサネットデバイスに割り当てるには、以下の内容の `/etc/systemd/network/10-enwan0.link` ファイルを作成します:

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

```
ーテル
```

```
[リンク]
```

新しい名前を使うように `/etc/network/interfaces` を調整し、initramfsをリフレッシュすることを忘れないでください。を使用してください。変更を有効にするには、ノードを再起動する必要があります。

### 備考

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

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

### 3.4.3 ネットワーク構成の選択

現在のネットワーク構成とリソースに応じて、ブリッジ、ルーティング、マスカレード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、インターネットへのアクセスには外部ゲートウェイを使用

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

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

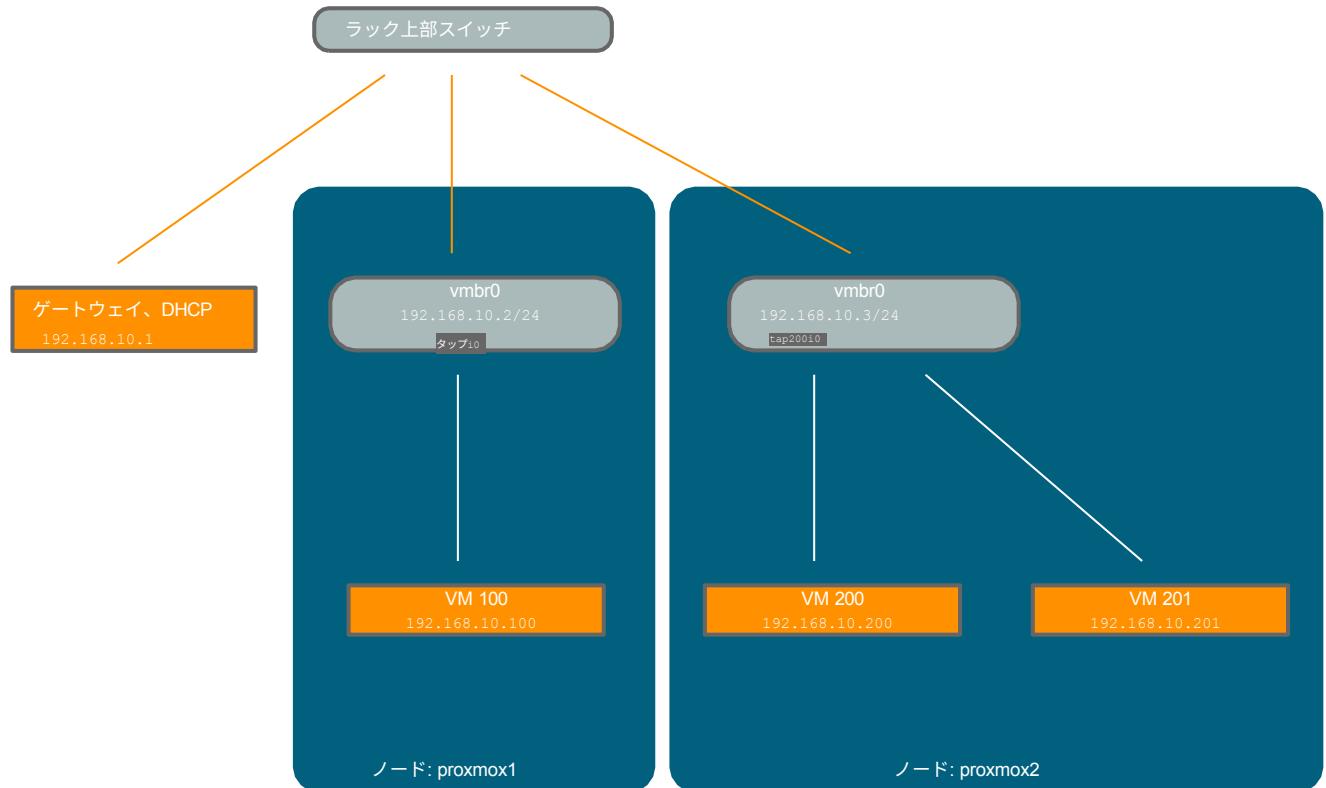
このセットアップでは、プロバイダーの許可に応じて、ブリッジ型またはルーティング型のいずれかを使用できます。

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

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

さらに柔軟性を高めるために、VLAN (IEEE 802.1q) やネットワークボンディング（別名「リンクアグリゲーション」）を設定することができます。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できます。

### 3.4.4 ブリッジを使用したデフォルト構成



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は単一のブリッジを共有できますが、複数のブリッジを作成してネットワークドメインを分けることもできます。各ホストは最大 4094 個のブリッジを持つことができます。

インストール・プログラムは vmbr0 という名前のブリッジを1つ作成し、最初のイーサネット・カードに接続します。etc/network/interfacesの対応する設定は以下のようになります：

```
オートロー
iface lo inet loopback iface

en0 inet manual

自動 vmbr0
iface vmbr0 inet static
    アドレス 192.168.10.2/24 ゲートウェイ
    インターフェース en0 bridge-ports en0
    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。ネットワークは、すべての仮想マシンをネットワークに接続しているネットワーク・ケーブルが1本しかないにもかかわらず、各仮想マシンを独自のMACを持っていると見なします。

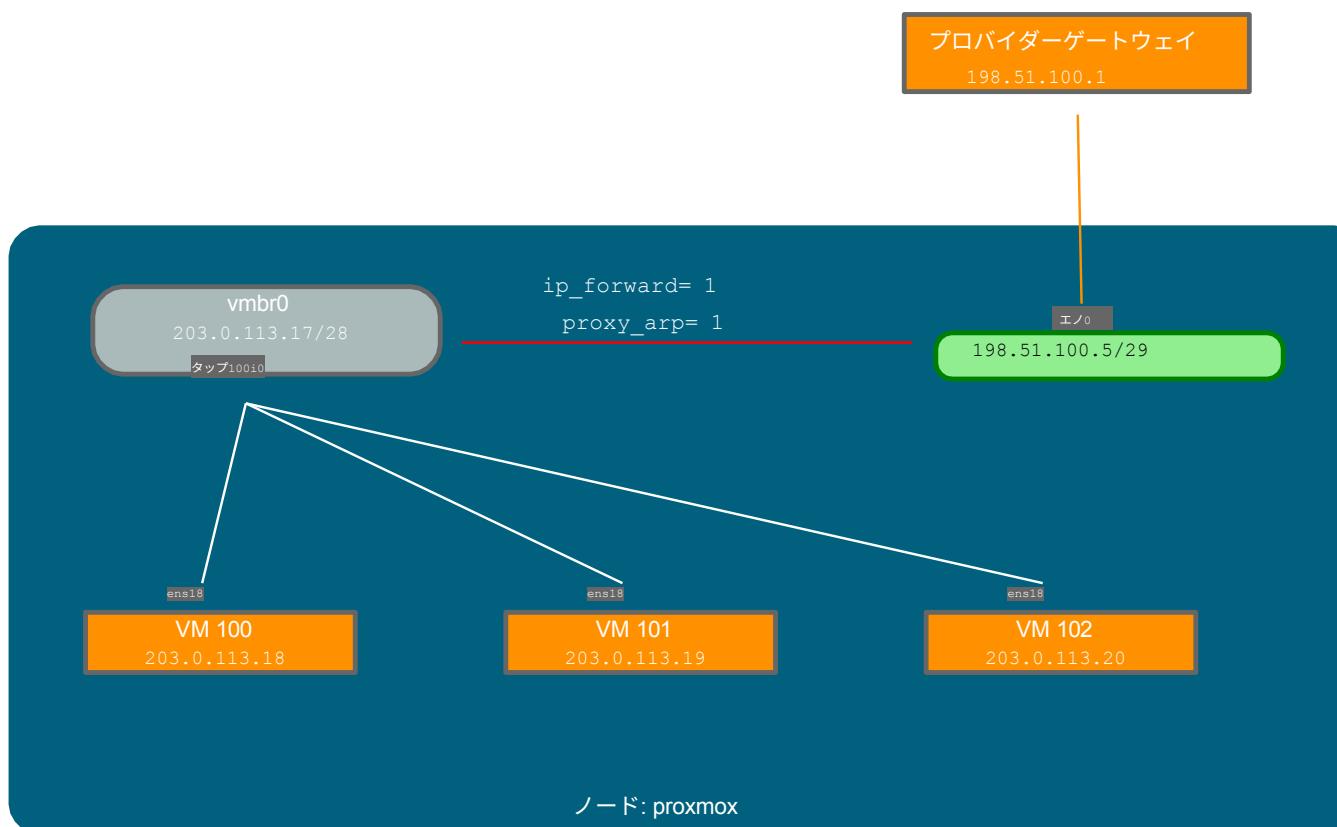
### 3.4.5 ルーティング構成

ほとんどのホスティングプロバイダー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、1つのインターフェイスで複数のMACアドレスを検出するとすぐにネットワークを無効にします。

#### チップ

プロバイダによっては、管理インターフェイスから追加のMACを登録できるものもあります。これにより問題は回避されますが、VMごとにMACを登録する必要があるため、設定が面倒になる可能性があります。

すべてのトラフィックを1つのインターフェイス経由で「ルーティング」することで、この問題を回避することができます。これにより、すべてのネットワークパケットが同じMACアドレスを使用するようになります。



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

#### オートロー

```
iface lo inet ループバック
```

#### オートエノ0

```
iface eno0 inet static
    アドレス 198.51.100.5/29 ゲートウェイ
    198.51.100.1
    post-up echo 1> /proc/sys/net/ipv4/ip_forward
    post-up echo 1> /proc/sys/net/ipv4/conf/eno0/proxy_arp
```

#### 自動 vmbr0

```
iface vmbr0 inet static
```

```
アドレス 203.0.113.17/28 bridge-
ports none bridge-stp off
bridge-fd 0
```

### 3.4.6 iptablesによるマスカレード (NAT)

マスカレードは、プライベートIPアドレスしか持たないゲストが、送信トラフィックにホストIPアドレスを使用してネットワークにアクセスすることを可能にします。各送信パケットはiptablesによってホストから発信されているように書き換えられ、レスポンスもそれに応じて書き換えられ、元の送信者にルーティン

オートロー

```
iface lo inet ループバック
```

オートエノ1

```
#実IPアドレス iface eno1 inet
static
    アドレス 198.51.100.5/24 ゲートウェイ
    198.51.100.1
```

自動 vmbr0

```
#プライベートサブネットワーク iface vmbr0
inet static
    アドレス 10.10.10.1/24 bridge-
    ports none bridge-stp off
    bridge-fd 0

    ポストアップ echo 1 > /proc/sys/net/ipv4/ip_forward
    ポストアップ 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
```

グされます。

#### 備考

ファイアウォールが有効になっている一部のマスカレード・セットアップでは、発信接続にコントラック・ゾーンが必要な場合があります。そうしないと、ファイアウォールは（MASQUERADEではなく）VMブリッジのPOSTROUTINGを優先するため、発信接続をブロックする可能性があります。

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

```
ポストアップ iptables -t raw -I PREROUTING -i fwbr+ -j CT --zone 1 post-down iptables -
raw -D PREROUTING -i fwbr+ -j CT --zone 1
```

これについては、以下のリンクを参照してください： [ネットフィルターのパケットフロー](#)

[conntrackゾーンを導入するnetdev-listのパッチ](#)

[生テーブルでTRACEを使用することで良い説明があるブログ記事](#)

### 3.4.7 Linuxボンド

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

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

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

ボンディングには7つのモードがあります：

- **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イス (NIC) スレーブから最後のスレーブまで、順次ネットワークパケットを送信します。このモードは負荷分散とフォールトトレランスを提供します。
- **アクティブバックアップ (active-backup)**：ボンド内の1つのNICスレーブのみがアクティブになります。アクティブスレーブに障害が発生した場合のみ、別のスレーブがアクティブになります。単一の論理ボンディングインターフェイスのMACアドレスは、ネットワークスイッチの歪みを避けるために、1つのNIC（ポート）のみで外部に表示されます。このモードはフォールトトレランスを提供します。
- **XOR (balance-xor)**：送信元MACアドレスと送信先MACアドレスのXOR] モジュロNICスレーブ数]に基づいてネットワークパケットを送信します。これは、各宛先MACアドレスに対して同じNICスレーブを選択します。このモードは負荷分散とフォールトトレランスを提供します。
- **ブロードキャスト (放送)**：すべてのスレーブ・ネットワーク・インターフェースでネットワーク・パケットを送信します。このモードはフォールトトレランスを提供します。
- **IEEE 802.3ad ダイナミックリンクアグリゲーション(802.3ad)(LACP)**：同じ速度とデュプレックス設定を共有するアグリゲーショングループを作成します。802.3ad仕様に従って、アクティブアグリゲーターグループ内のすべてのスレーブネットワークインターフェースを使用します。
- **アダプティブ・トランスマッシュ・ロードバランシング(balance-tlb)**：特別なネットワークスイッチのサポートを必要としないLinuxのボンディングドライバモード。発信ネットワークパケットトラフィックは、各ネットワークインターフェイススレーブの現在の負荷（速度に対して計算される）に従って分配されます。着信トラフィックは、現在指定されている1つのスレーブ・ネットワーク・インターフェースで受信されます。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。
- **適応型ロードバランシング (balance-alb)**：balance-tlbにIPV4トラフィックの受信負荷分散(rlb)を加えたもので、特別なネットワークスイッチのサポートは必要ありません。受信負荷分散はARPネゴシエーションによって達成されます。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リプライをインターセプトし、異なるネットワークピアがネットワークパケットトラフィックに異なるMACアドレスを使用するように、単一の論理ボンディングインターフェース内のNICスレーブの1つの固有のハードウェアアドレスでソースハードウェアアドレスを上書きします。

お使いのスイッチがLACP（IEEE 802.3ad）プロトコルをサポートしている場合は、対応するボンディングモード（802.3ad）を使用することをお勧めします。そうでない場合は、通常アクティブバックアップ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お勧めします。Corosyncはネットワークの冗長化のためのボンドを必要としません。

以下のボンド構成は、分散/共有ストレージネットワークとして使用できます。は、より高速になり、ネットワークがフォールトトレラントになることです。

### 例固定IPアドレスでボンドを使用

```

オートロー
iface lo inet loopback iface

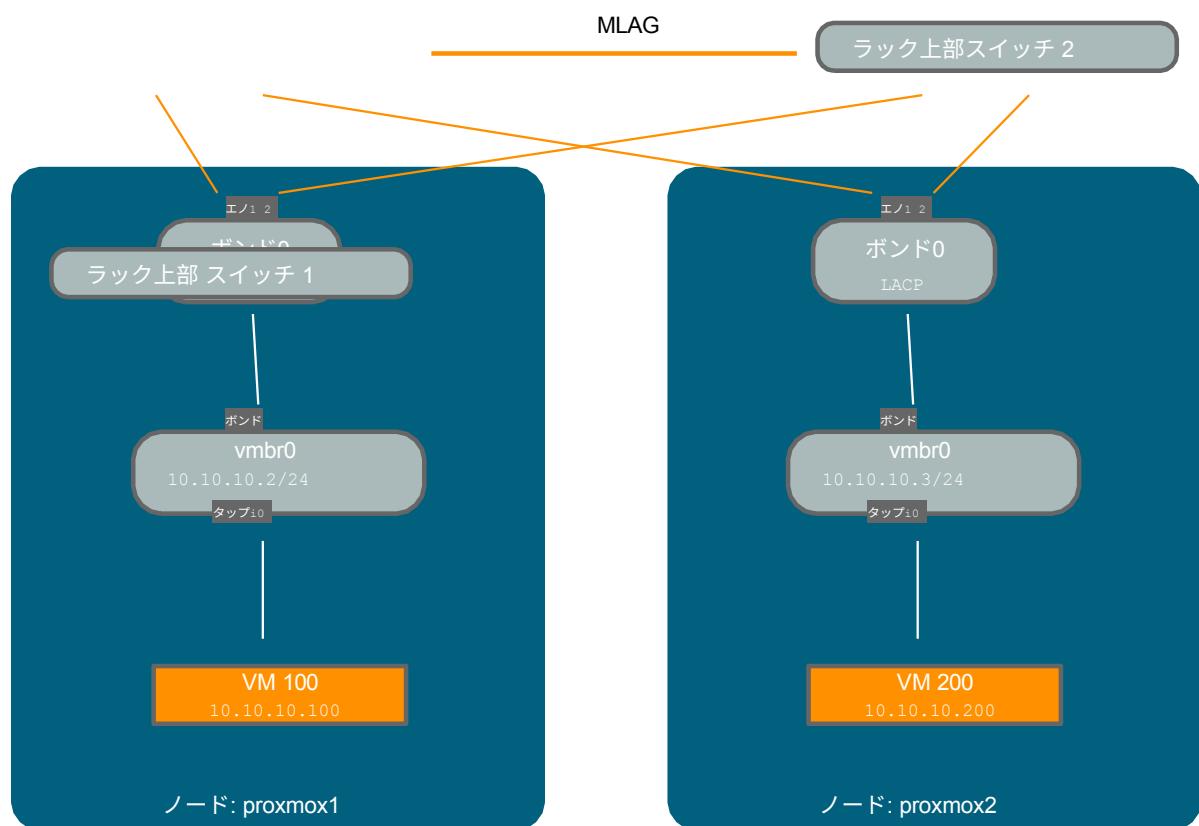
en01 inet manual iface eno2 inet

iface eno3 inet manual

オートボンド0
iface bond0 inet static bond-slaves
    eno1 eno2
        アドレス 192.168.1.2/24 bond-
        miimon 100
        ボンドモード 802.3ad
        bond-xmit-hash-policyレイヤー2+3

自動 vmbr0
iface vmbr0 inet static
    アドレス 10.10.10.2/24 ゲートウェイ
    10.10.10.1 bridge-ports eno3
    bridge-stp off
    bridge-fd 0

```



もう1つの可能性は、ブリッジポートとしてボンドを直接使用することです。これは、ゲストネットワークをフォールトトレラントにするために使用できます。

#### 例ブリッジポートとしてボンドを使用

```
オートロー
iface lo  inet  loopback  iface
en0  inet  manual  iface eno2  inet
manual

オートボンド0
iface bond0  inet  manual  bond-slaves
    eno1  eno2  bond-mimon 100
    ボンドモード 802.3ad
    bond-xmit-hash-policyレイヤー2+3

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

### 3.4.8 VLAN 802.1Q

仮想LAN(VLAN)はプロードキャスト・ドメインで、レイヤ2のネットワーク内で分割・分離されます。そのため、物理ネットワーク内に複数のネットワーク（4096）を持つことができ、それぞれが他のネットワークから独立しています。

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。ネットワークパッケージは、どの仮想ネットワークに属するかを識別するためにタグ付けされます。

#### ゲストネットワーク用VLAN

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

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

## ホスト上のVLAN

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

たとえば、ホスト管理アドレスを別のVLANに配置したいデフォルトの構成では、次のようにになります。

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

```
オートロード
iface lo inet loopback iface eno1
```

```
inet manual iface eno1.5 inet manual
```

自動 vmbr0v5

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

自動 vmbr0

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

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

```
オートロード
iface lo inet loopback iface
```

```
eno1 inet manual
```

自動 vmbr0.5

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

自動 vmbr0

```
iface vmbr0 inet マニュアル
    bridge-ports eno1
    bridge-stp off bridge-fd
    0
    ブリッジ-vlan-aware yes
```

### ブリッジビデオ 2-4094

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

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

オートロー

```
iface lo inet loopback iface  
en0 inet manual iface eno2 inet  
manual
```

オートボンド0

```
iface bond0 inet manual bond-slaves  
    eno1 eno2 bond-mimon 100  
    ボンドモード 802.3ad  
    bond-xmit-hash-policy layer2+3 iface bond0.5  
inet manual
```

自動vmbr0v5

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

自動 vmbr0

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

### 3.4.9 ノードのIPv6の無効化

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

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

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

内容で：

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

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

### 3.4.10 ブリッジでの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をスムーズに使用できるように、ブリッジでMAC学習が有効になっています。

しかし、環境によってはこれが望ましくないこともあります。Proxmox VE 7.3以降では、ブリッジの'bridge-disable-mac-learning 1'設定を`/etc/network/interfaces`などで設定することで、ブリッジのMAC学習を無効にできます：

```
# ...

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

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

## 3.5 時間同期

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

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



#### 重要

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

chrony、ntp または openntpd。

### 3.5.1 カスタムNTPサーバーの使用

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

クロニーを使用するシステムの場合：

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

```
サ 一 バ      ntp1.example.com サ 一 バ  
ntp2.example.com サ 一 バ      ntp3.example.com  
iburst
```

クロニーを再起動してください:

```
# systemctl restart chronyd
```

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

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

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

**systemd-timesyncd を使用しているシステムの場合:**

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

#### 時間

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

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

```
...  
Oct 07 14:58:36 node1 systemd[1]: ネットワーク時刻同期を停止しています...Oct 07 14:58:36 node1 systemd[1]: ネットワ  
ーク時刻同期の開始...Oct 07 14:58:36 node1 systemd[1]: ネットワーク時刻同期を開始しました: ネットワーク時刻同期を開始しま  
した。  
Oct 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。  
Oct 07 14:58:36 node1 systemd-timesyncd[13514]: interval/delta/delay/jitter:  
    /drift 64s/-0.002s/0.020s/0.000s/-31ppm  
...
```

## 3.6 外部メトリックサーバー

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

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

現在サポートされているのは

- グラフィット (<https://graphiteapp.org> を参照)
- InfluxDB (<https://www.influxdata.com/time-series-platform/influxdb/> を参照)

外部メトリックサーバ定義は`/etc/pve/status.cfg`に保存され、ウェブインターフェースで編集できます。

### 3.6.1 Graphiteサーバーの構成

Create: Graphite

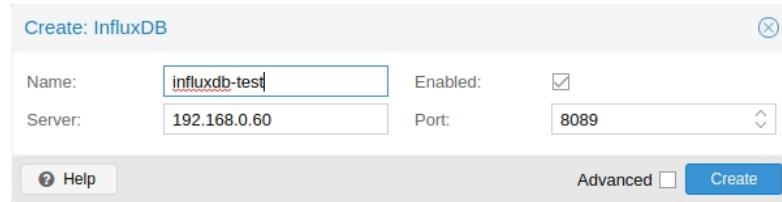
Name:	<input type="text" value="graphite-test"/>	Enabled:	<input checked="" type="checkbox"/>
Server:	<input type="text" value="192.168.0.50"/>	Port:	<input type="text" value="2003"/>
		Path:	<input type="text" value="proxmox"/>

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

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

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

### 3.6.2 Influxdbプラグインの設定



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

以下は influxdb (influxdb サーバー) の設定例です：

```
[udp] ]。
enabled= true
バインドアドレス= "0.0.0.0:8089" データベース =
"proxmox"
batch-size = 1000 batch-
timeout= "1s"
```

この設定では、サーバーはポート8089ですべてのIPアドレスをリッスンし、データを  
**Proxmox**データベース

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

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

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

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。

InfluxDB1.xでは意味がありません。

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

## 3.7 ディスクヘルス監視

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

Proxmox VE 4.3から、smartmontoolsパッケージ<sup>1</sup>がインストールされ、必須となりました。これは、ローカルハードディスクのS.M.A.R.T.システムを監視および制御するためのツールセットです。

ディスクのステータスは、以下のコマンドで取得できます：

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

/dev/sdXはローカルディスクのパスです。出力が

```
SMARTのサポートは無効
```

コマンドで有効にできます：

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

smartctlの使い方については、man smartctlを参照してください。

デフォルトでは、smartmontoolsデーモンsmartdがあり、/dev/sdX以下のディスクをスキャンします。

/dev/hdXを30分ごとにエラーと警告をチェックし、問題があればrootにメールを送ります。

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。

ハードディスクをハードウェア raid コントローラーで使用している場合、ほとんどの場合 raid アレイ内のディスクとアレイ自体を監視するツールがあります。これに関する詳細については、ご使用の raid コントローラーのベンダーにお問い合わせください。

## 3.8 論理ボリュームマネージャ (LVM)

ほとんどの場合、Proxmox VEはローカルディスクに直接インストールされます。Proxmox VEのインストールCDには、ローカルディスク管理のためのいくつかのオプションが用意されており、現在のデフォルトのセットアップではLVMが使用されています。インストーラでは、このようなセットアップに使用するディスクを1つ選択でき、そのディスクをボリュームグループ(VG) pve の物理ボリュームとして使用します。以下の出力は、8GB の小型ディスクを使用したテストインストールによるものです：

```
# PV
/dev/sda3      ブイ   Fmt Attr PSize PFree
                ジー   lvm2 a--    7.87g 876.00m
                レベ
                ル
# vgs
VG            #PV #LV #SN Attr          VSサイズ VFree
レベ          1     3     0 WZ--N- 7.87G 876.00M
ル
```

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

```
# lvs
レベル ブイジー 属性          LSサイズ プール原産地      データ メタ
データ レベル  ツイ・ア・ツ      4.38g
ルート レベル  -ウィアオ      1.75g
スワップ レベル  -ウィアオ    896.00m
```

### ルート

フォーマット形式はext4で、オペレーティングシステムが入っています。

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

## スワップ

スワップ・パーティション

### データ

このボリュームはLVM-thinを使用し、VMイメージの保存に使用されます。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、このタスクに適しています。

Proxmox VEのバージョン4.1までの場合、インストーラは「data」という標準論理ボリュームを作成し、/var/lib/vzにマウントします。

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

### 3.8.1 ハードウェア

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

LVM自体は特別なハードウェアを必要とせず、メモリ要件も非常に低くなっています。

### 3.8.2 ブートローダー

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

### 3.8.3 ボリュームグループの作成

空のディスク/dev/sdbがあり、そこに"vmdata"という名前のボリュームグループを作成します。



#### 注意

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

まずパーティションを作成します。

```
# 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にします。

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

「vmdata」という名前のボリュームグループを/dev/sdb1に作成 # vgcreate

```
vmdata /dev/sdb1
```

### 3.8.4 var/lib/vz用の追加LVの作成

これは新しい薄いLVを作成することで簡単にできます。

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

実例です:

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

ここで、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
```

ついにこれを搭載しなければなりません。



#### 警告

var/lib/vzがであることを確認してください。デフォルトのインストールではそうではありません。

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

```
# 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 +<size[“M,G,T”]> --poolmetadatasize +<size[“M,G”]><←’  
VG>/<LVThin_pool>.
```

---

#### 備考

データ・プールを拡張する場合は、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プールの作成

シンプルはボリューム上に作成する必要があります。ボリュームグループの作成方法は、セクションLVMを参照してください。

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

## 3.9 Linux上のZFS

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

- すべてのパッケージが含まれています。

ZFSを使用することで、低予算のハードウェアで最大限のエンタープライズ機能を実現するだけでなく、SSDキャッシングやSSDのみのセットアップを活用することで、高性能なシステムを実現することも可能です。ZFSは、CPUとメモリ負荷の軽減と容易な管理を組み合わせることで、コスト高なハードウェア 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のGUIとCLIで簡単な設定と管理。
- 信頼できる
- データ破損からの保護
- ファイルシステムレベルでのデータ圧縮
- スナップ写真
- コピーオンライトクローン
- 様々なレイドレベルRAID0、RAID1、RAID10、RAIDZ-1、RAIDZ-2、RAIDZ-3、dRAID、dRAID2、dRAID3
- キャッシュにSSDを使用可能
- セルフヒーリング
- 繙続的な完全性チェック
- 大容量収納に対応した設計
- ネットワーク経由の非同期レプリケーション
- オープンソース
- 暗号化
- ...

### 3.9.1 ハードウェア

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

専用のキャッシュおよび/またはログディスクを使用する場合は、エンタープライズクラスのSSDを使用する必要があります。これにより、全体的なパフォーマンスが大幅に向上します。

#### 重要

独自のキャッシュ管理を持つハードウェアRAIDコントローラの上でZFSを使用しないでください。ZFSはディスクと直接通信する必要があります。HBAアダプタか、「IT」モードでフラッシュされたLSIコントローラのようなものがより適切です。

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

### 3.9.2 ルートファイルシステムとしてのインストール

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

RAID0 「ストライピング」とも呼ばれます。このようなボリュームの容量は、すべてのディスクの容量の合計です。しかし、RAID0は冗長性を追加しないため、1台のドライブが故障ボリュームは使用できなくなります。

RAID1 「ミラーリング」とも呼ばれます。データはすべてのディスクに同じように書き込まれます。このモードでは、同じサイズのディスクが少なくとも2台必要です。その結果、容量は1台のディスクの容量になります。

RAID10 RAID0とRAID1の組み合わせ。少なくとも4台のディスクが必要。RAIDZ-1 RAID-5のバ

リエーションで、シングルパーティ。少なくとも3台のディスクが必要。RAIDZ-2 RAID-5のバリエーション、ダブル

パーティ。少なくとも4台のディスクが必要。RAIDZ-3 RAID-5のバリエーション、トリプルパーティ。少なくと

も5台のディスクが必要。

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

VMイメージを格納するために、`rpool/data`という別のサブボリュームが作成されます。これをProxmox VEツールで使用するために、インストーラは`/etc/pve/storage.cfg`に以下の設定エントリを作成します:

```
zfspool: local-zfs
    プール rpool/data スペース
    コンテンツイメージ、ルートディレクトリ
```

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

```
# zpool status pool:
  rpool
状态オンライン
scan: none リクエストされた設定
:
```

名称	状態	読む	ライト	シークサム
リップル	オンライン	0	0	0
ミラーゼロ	オンライン	0	0	0
エスダツー	オンライン	0	0	0
sdb2	オンライン	0	0	0
ミラー1	オンライン	0	0	0
秒秒	オンライン	0	0	0

へん ちょ	オンライン	0	0	0
エラー既知のデータエラーなし				

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

```
# zfs list
名称          中古      AVAIL    リファー    マウントポイント
リプール        4.94G   7.68T     96K /rpool
rpool/ROOT      702M    7.68T     96K /rpool/ROOT
rpool/ROOT/pve-1 702M    7.68T    702M /
rpool/データ      96K    7.68T     96K /rpool/データ
rpool/swap       4.25G  7.69T     64K -
```

### 3.9.3 ZFS RAIDレベルの考察

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

#### パフォーマンス

各 vdev タイプは、それぞれ異なるパフォーマンス動作をします。注目すべき 2 つのパラメータは、IOPS (In-put/Output Operations per Second) と、データを書き込んだり読み込んだりできる帯域幅です。

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

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

どの冗長レベルの RAIDZ でも、IOPS に関してはシングルディスクのように動作し、帯域幅は多くなります。どの程度の帯域幅になるかは、RAIDZ vdev のサイズと冗長レベルに依存します。

dRAID プールは同等の RAIDZ プールのパフォーマンスと同等であるべきです。VM を実行する場合、ほとんどの

状況では IOPS の方が重要です。

#### サイズ、スペース使用量、冗長性

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

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

RAIDZ レベルを使用する際のもう 1 つの重要な要素は、VM ディスクに使用される ZVOL データセットの動作です。各データ・ブロックに対して、プールは少なくとも最小ブロック・サイズのパリティ・データを必要とします。

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

この動作は、ZVOLの以下のプロパティをチェックすると確認できます：

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

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

volsize はVMに表示されるディスクのサイズであり、refreservation は、パリティ・データに必要な予期されるスペースを含むプールの予約スペースを示します。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。動作を観察するもう 1 つの方法は、VM内の使用済みディスク領域と使用済みプロパティを比較することです。スナップショットによって値が歪むことに注意してください。

スペースの増加に対抗するために、いくつかの選択肢があります：

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

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

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

ミラーVdev (RAID1、RAID10)はVMワークロードに有利な動作をします。RAIDZのパフォーマンス特性が許容できるような特殊なニーズや特性を持つ環境でない限り、これらを使用してください。

### 3.9.4 ZFS dRAID

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

#### 備考

dRAID 10-15 台以上のディスクを想定しています。RAIDZ セットアップの方が、ほとんどのユースケースにおいて、より少量のディスクに適しています。

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

### 備考

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

- dRAID1 または dRAID: 少なくとも2台のディスクが必要。
- dRAID2: 少なくとも3台のディスクが必要。
- dRAID3: 少なくとも4台のディスクが必要で、データが失われる前に3台が故障する可能性があります。

その他の情報はマニュアルのページをご覧ください:

```
# man zpoolconcepts
```

### スペアとデータ

スペアの数は、ディスク障害が発生した場合に備えて、システムが準備しておくディスクの数を示します。デフォルト値は 0 スペアです。スペアがないと、再構築の速度が向上しません。

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

### 3.9.5 ブートローダー

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

### 3.9.6 ZFS管理

このセクションでは、一般的なタスクの使用例を紹介します。ZFS自体は本当に強力で、多くのオプションを提供しています。ZFSを管理する主なコマンドはzfsとzpoolです。どちらのコマンドにも素晴らしいマニュアルページが付属しています:

```
# 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つのディスクが必要です。ashift は、基礎となるディスクと同じセクタサイズ (ashift の 2乗) 以上でなければなりません。

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

## チップ

プール名は以下の従わなければなりません:

- 頭文字 (a-z または A-Z)
- 英数字、`\_`、`.`、`:` または ``(スペース) 文字のみを含みます。
- mirror、raidz、draid、spareのいずれかで始まってはいけません。
- ログであってはなりません

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

ディスク1枚以上

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

### RAID-1で新しいプールを作成

最低2ディスク

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

### RAID-10で新しいプールを作成

最低4ディスク

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

### RAIDZ-1で新しいプールを作成

ディスク3枚以上

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

### RAIDZ-2で新しいプールを作成

最低4ディスク

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

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

## キャッシュ（L2ARC）付きプールを新規作成

第2レベルキャッシュとして専用デバイスまたはパーティションを使用することで、パフォーマンスを向上させることができます。このようなキャッシュ・デバイスは、ほとんどが静的なデータのランダム・リード・ワークロードに特に役立ちます。キャッシュ・デバイスは、実際のストレージとインメモリARCの間の追加キャッシュ層として機能するため、メモリ制約のためにARCを削減しなければならない場合にも役立ちます。

### オンディスク・キャッシュ付きZFSプールの作成

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

キャッシュ・デバイスにはミラーやレイド・モディが存在せず、それらはすべて単純に累積されることに注意してください。

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は透過的にそのリクエストを基礎となるストレージ層に迂回させます。

### ログ(ZIL)付きプールを新規作成

ZFS Intent Log (ZIL) 専用ドライブまたはパーティションを使用することも可能です。ZIL は主に安全な同期トランザクションを提供するために使用され、データベースなどのパフォーマンスが重要なパスや、fsync 操作を頻繁に発行するその他のプログラムでよく使用されます。

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

ディスクを直接、またはパーティションを通してログ・デバイスとして使用する場合は、次のことをお勧めします：

- 停電保護機能付きの高速SSDを使用すると、コミット・レイテンシが非常に小さくなります。
- 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を使用しますが、インストールされているメモリの半分以上を使用しても、実際の利点は得られません。

### 個別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>が使用されていますが、[RAIDを使用して新しいプールを作成する](#)セクションで説明されているように、これを他のRAIDバリエーションと組み合わせることもできます。

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

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

## 既存のプールにキャッシュとログを追加

キャッシュとログがないプールがあっても、その両方、または片方だけを追加することができます。

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

ログデバイスの最大サイズは、インストールされている物理メモリの約半分である必要があるため、ZILはSSDの比較的小さな部分しか占有しない可能性が高く、残りの領域はキャッシュとして使用できます。

まず、partedまたはgdiskを使って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。それからプールに追加します：

**独立したログ・デバイスと第2レベル・キャッシュの両方を既存のプールに追加します。**

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

<pool>、<device-part1>、<device-part2>をプール名と2つの /dev/disk/by-id/ のパスをパーティションに追加します。ZILと

キャッシュを別々に追加することもできます。

## 既存のZFSプールへのログデバイスの追加

```
# 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>.
```

## 失敗したブータブルデバイスの変更

Proxmox VE のインストール方法によって、systemd-boot を使うか、proxmox-boot 経由で GRUB を使います。

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

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

パーティションテーブルのコピー、GUID の再発行、ZFS パーティションの交換という最初のステップは同じです。新しいディスクからシステムを起動可能にするには、使用するブートローダによって異なる手順が必要です。

```
# sgdisk <健康なブータブルデバイス> -R <新しいデバイス> # sgdisk -G <新しいデバイス>
# zpool replace -f <pool> <old zfs partition> <new zfs partition>.
```

---

### 備考

zpool status -v コマンドを使用して、新しいディスクの resilvering プロセスがどの程度進んでいるかを監視します。

---

<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

**proxmox-boot-toolを使用します:**

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

**備考**

ESPはEFIシステムパーティションの略で、バージョン5.4以降のProxmox VEインストーラを使用する場合、ブータブルディスクのパーティション#2として設定されます。詳細については、[同期されたESPとして使用するための新しいパーティションの設定](#)を参照してください。

**備考**

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

**プレーンなGRUBで:**

```
# grub-install <新しいディスク
```

**備考**

プレーン GRUB は Proxmox VE 6.3 以前でインストールされ、proxmox-boot-tool を使用するように手動で移行しないシステムでのみ使用されます。

### 3.9.7 電子メール通知の設定

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

デーモンの設定は、/etc/zfs/zed.d/zed.rc ファイルからお好みのエディタで行えます。電子メール通知に必要な設定はZED\_EMAIL\_ADDRで、デ

```
ZED_EMAIL_ADDR="root"
```

フォルトではrootに設定されています。

Proxmox VEはroot宛のメールをroot設定されたメールアドレスに転送しますのでご注意ください。

### 3.9.8 ZFSメモリ使用量の制限

ZFSは、デフォルトでホストメモリの50%をAdaptive Replacement Cache (ARC)に使用します。Proxmox VE 8.1以降の新規インストールでは、ARCの使用制限がインストールされた物理メモリの10%に設定され、最大16GiBにクランプされます。この値は /etc/modprobe.d/zfs.conf に書き込まれます。

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

また、ZFSは最小値64MBを強制します。

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

`zfs_arc_max` モジュール・パラメータを直接指定します:

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

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

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

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

### 重要

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

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

この設定例では（一時的に）、総メモリ量が256GiBを超えるシステムでの使用量を8GiB ( $8 * 2^{30}$ ) に制限しています。

### 重要

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

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

これらの有効にするには、再起動する必要があります。

## 3.9.9 ZFS上のSWAP

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

メモリ不足に陥らないよう、十分なメモリを使用することを強くお勧めします。スワップを追加したい場合は、物理ディスク上にパーティションを作成し、スワップデバイスとして使用することをお勧めします。インストーラの詳細オプションで、この目的のために空き領域を残すことができます。さらに、"swappiness"値を下げるこどもできます。サーバに適した値は10です：

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

スワップを永続的にするには、`/etc/sysctl.conf`をお好みのエディターで開き、以下の行を追加します：

```
vm.swappiness= 10
```

表 3.1: Linuxカーネルのスワッピネス・パラメーター値

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

表3.1: (続き)

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

### 3.9.10 暗号化されたZFSデータセット

**警告**

Proxmox VEのネイティブZFS暗号化は実験的なものです。既知の制限と問題には、暗号化されたデータセットでのリプリケーション<sup>a</sup>、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があります。<sup>b</sup>

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

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

```
# zpool get feature@encryption tank
  名称      不動産          値           ソース
  タンク    暗号化機能      使用禁止      ローカル

# zpool set feature@encryption(enabled)
# zpool get feature@encryption tank
  名前プロパティ          値           ソース
  タンク機能@暗号化有効      使用禁止      ローカル
```

**警告**

暗号化されたデータセットを持つプールから GRUB を使ってブートすることは現在サポートされていません。暗号化をサポートしていない古いバージョンの ZFS では、保存されたデータを復号化することはできません。

**備考**

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、起動時のロック解除に必要なキー素材を `zfs load-key` に渡すカスタムユニットを作成することをお勧めします。

**警告**

本番データの暗号化を有効にする前に、バックアップ手順を確立し、テストしてください。暗号化されたキー・マテリアル//パスフレーズ/キー・ファイルを紛失した場合、暗号化されたデータへのアクセスは不可能になります。

暗号化はデータセット/zvolsの作成時に設定する必要があり、デフォルトで子データセットに継承されます。例えば、暗号化データセット tank/encrypted\_dataを作成し、Proxmox VEのストレージとして設定するには、以下のコマンドを実行します：

```
# zfs create -o encryption=on -o keyformat=passphrase tank/encrypted_data パスフレーズを入力してください:  
パスフレーズを再入力します:
```

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

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

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

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

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

```
# dd if=/dev/urandom of=/path/to/keyfile bs=32 count=1  
  
# zfs change-key -o keyformat=raw -o keylocation=file:///path/to/keyfile!  
タンク/暗号化データ
```

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



#### 警告

キーファイルを使用する場合は、不正アクセスや偶発的な紛失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できません！

暗号化データセットの下に作成されたゲストボリュームは、それに応じてencryptionrootプロパティが設定されます。鍵マテリアルは、encryptionrootごとに1回だけロードする必要があり、その下にあるすべての暗号化データセットで使用できるようになります。

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

### 3.9.11 ZFSの圧縮

データセットで圧縮を有効にすると、ZFSは新しいブロックをすべて圧縮してから書き込み、読み込み時に解凍しようとします。既に存在するデータは遡って圧縮されません。

で圧縮を有効にできます：

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

lz4 アルゴリズムの使用を推奨します。lzbj や gzip-N (N 1 (最速) から 9 (最高の圧縮率) までの整数) のような他のアルゴリズムも利用可能です。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性能が向上することもあります。

でいつでも圧縮を無効にできます：

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

繰り返しますが、この変更の影響を受けるのは新しいブロックだけです。

### 3.9.12 ZFS専用デバイス

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

特別なデバイスは、メタデータの変更が多い低速回転のハードディスクで構成されるプールの速度を向上させることができます。例えば、大量のファイルを作成、更新、または削除するようなワークロードでは、特別なデバイスは、特別な小さなファイル全体を保存するように設定することができ、パフォーマンスをさらに向上させることができます。特殊高速SSDを使用してください。

**重要**

特殊デバイスの冗長性は、プールの冗長性と一致させる必要があります。

デバイスはプール全体の障害点となります。

**警告**

プールに特別なデバイスを追加すると、元に戻すことはできません！

特別なデバイスとRAID-1でプールを作成します：

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

既存のRAID-1プールに特別なデバイスを追加します：

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

ZFSデータセットでは、`special_small_blocks=<size>`プロパティを公開しています。`size`には、小さなファイル・ブロックを特殊保存しないようにする0、または512Bから1Mの範囲の2のべき乗を指定します。このプロパティを設定すると`size`よりも小さなファイル・ブロックが特別な割り当てられます。

**重要**

`special_small_blocks`の値が`recordsize`以上の場合（デフォルトは

データセットの128K）に書き込まれると、すべてのデータが特別なデバイスに書き込まれますので、注意してください！

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

プール全体で4Kブロックより小さいファイルすべてにオプトインします:

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

単一のデータセットに対して小さなファイルブロックを選択します:

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

単一のデータセットに対する小さなファイルブロックからオプトアウトします:

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

### 3.9.13 ZFSプールの特徴

ZFSのオンディスク・フォーマットへの変更は、メジャー・バージョンの変更の間にのみ行われ、**features**を通して指定されます。すべての機能は、一般的なメカニズムと同様に、zpool-features (*man* ページで十分に文書化されています)。

新しい機能を有効にすると、古いバージョンの ZFS ではプールがインポートできなくなる可能性があるため、これは、管理者がプール上で `zpool upgrade` を実行して、積極的に行う必要があります (`zpool-upgrade(8)` *man* ページを参照)。

新機能を使用する必要がない限り、新機能を有効にすることにプラス面はありません。実際、新機能を有効にすることにはマ

イナス面もあります:

- GRUBでZFSの実装に互換性がないため、GRUBを使用して起動するZFS上のrootを持つシステムは、rpoolで新しい機能がアクティブになると起動できなくなります。
- 古いZFSモジュールが同梱されている古いカーネルで起動すると、システムはアップグレードされたプールをインポートできません。
- 起動しないシステムを修復するために古いProxmox VE ISOを起動しても同様に動作しません。

**重要**

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

ZFSプールの新機能を有効にします:

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

## 3.10 ビーティーアールエフエス



### 警告

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

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

#### BTRFSの一般的な利点

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

#### 注意事項

- RAIDレベル5/6は実験的で危険です。

### 3.10.1 ルートファイルシステムとしてのインストール

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

RAID0	「ストライピング」とも呼ばれます。このようなボリュームの容量は、すべてのディスクの容量の合計です。しかし、RAID0は冗長性を追加しないため、1台のドライブが故障ボリュームは使用できなくなります。
RAID1	「ミラーリング」とも呼ばれます。データはすべてのディスクに同じように書き込まれます。このモードでは、同じサイズのディスクが少なくとも 2 台必要です。その結果、容量は 1 台のディスクの容量になります。
RAID10	RAID0 と RAID1 の組み合わせ。少なくとも 4 台のディスクが必要です。

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

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

/etc/pve/storage.cfg:

ディレクトリ：ローカル

```
パス /var/lib/vz  
コンテンツiso,vztmpl,バックアップ無効
```

btrfs：ローカルbtrfs

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

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

btrfsコマンドは、BTRFSファイルシステムの設定と管理に使用します。インストール後、次のコマンドで追加のサボリュームをすべて一覧表示します：

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

## 3.10.2 BTRFSアドミニストレーション

このセクションでは、一般的なタスクの使用例を紹介します。

### BTRFSファイルシステムの作成

BTRFSファイルシステムを作成するには、mkfs.btrfsを使用します。d パラメーターと -m パラメーターは、それぞれメタデータとデータのプロファイルを設定するために使用します。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。

一般に、以下のモードがサポートされています: シングル、raid0、raid1、raid10。単一ディスク/dev/sdb上に、My-

Storageというラベルを付けて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にRAID1を作成します:

```
# 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、例えば手動でマウントすることができます:

```
# 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 に追加して、起動時に自動的にマウントすることもできます。ブロック・デバイス・パスの使用は避け、mkfs.btrfs コマンドが outputする UUID 値を使用することをお勧めします。

例えば

## ファイル /etc/fstab

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

### チップ

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

その後、最初のマウントを実行ことができます：

```
マウント /my-storage
```

次のリブート後は、ブート時にシステムによって自動的に実行されます。

## Proxmox VEへのBTRFSファイルシステムの追加

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

```
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 サブボリューム作成 /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逆に、サブボリュームは btrfs コマンドで削除するために空である必要はありません。

```
# btrfs サブボリューム削除 /some/path
```

### サブボリュームのスナップショットの作成

BTRFS は実際にはスナップショットと通常のサブボリュームを区別しないため、スナップショットを作成することはサブボリュームの任意のコピーを作成することと見なすこともできます。慣例では、Proxmox VEはゲストディスクまたはサブボリュームのスナップショットを作成するときに読み取り専用フラグを使用しま

```
# btrfs subvolume snapshot -r /some/path /a/new/path  
ですが、このフラグは変更することもできます。
```

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

読み取り専用 (-r) オプションをすると、両方のサブボリュームが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` `mount` オプションを追加します。すでに書き込まれたデータは後から圧縮されないことに注意してください。

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

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

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

上記の:

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

この変更は再起動後に有効になります。

## スペースの使用状況の確認

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

`btrfs` ファイルシステムの使用法 /PATH :

```
# 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再ソースを制御できます。

現在 `pvenode` では、ノードの説明を設定したり、ノードのゲストに対して様々な一括操作を実行したり、ノードのタスク履歴を表示したり、ノードのSSL証明書を管理したりすることができます。

### 3.11.1 ウェイクオンLAN

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

`pvenode`を使用すると、WoL経由でクラスタのスリープ・メンバーをスリープ解除できます:

```
pvenode wakeonlan <ノード>。
```

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

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

WoLパケットを送信するインターフェースは、デフォルト・ルートから決定されます。以下のコマンドで `bind-interface` を設定することで上書きできます

```
pvenode config set -wakeonlan XX:XX:XX:XX:XX:XX,bind-interface=<iface-name>を設定します。  
す:
```

WoLパケットを送信する際に使用されるブロードキャストアドレス（デフォルトは255.255.255.255）は、以下のコマンドを使用してブロードキャス

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

トアドレスを明示的に設定することでさらに変更できます：

### 3.11.2 タスク履歴

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

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

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

タスクのログは、そのUPIDを使って印刷できます：

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

### 3.11.3 一括ゲスト電源管理

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

例えば、VM100、101、102を起動するには、オンブレットセットの有無に関係なく、次のようにします：

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

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

```
ブベノードストップオール
```

#### 備考

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

### 3.11.4 ファーストゲストブートディレイ

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

以下のように設定することで実現できます（ここで10は秒単位の遅延を表します）：

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

### 3.11.5 ゲストの一括移行

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

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

```
pvenode migrateall pve2 --vms 100,101,102 --ローカルディスクあり
```

### 3.11.6 バルーニングのRAM使用量目標

[自動メモリ割り当ての目標パーセンテージ](#)のデフォルトは80%です。このターゲットは、ballooning-target プロパティを設定することで、ノードごとにカスタマイズできます。たとえば、ホスト・メモリ使用率を90%に設定することもできます：

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

## 3.12 証明書管理

### 3.12.1 クラスタ内通信証明書

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

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

### 3.12.2 APIおよびWeb GUIの証明書

REST API および Web GUI は、各ノードで実行される pveproxy サービスによって提供されます。pveproxy で使用する証明書には以下のオプションがあります：

- 既定では、/etc/pve/nodes/NODENAME/pve-ssl.pem にあるノード固有の証明書が使用されます。この証明書はクラスタ CA によって署名されているため、ブラウザおよびオペレーティング システムからは自動的に信頼されません。

2. 外部から提供された証明書（商用CAによって署名されたものなど）を使用します。
3. ACME（Let's Encrypt）を使用して、自動更新付きの信頼できる証明書を取得してください。

オプション2と3では、`/etc/pve/local/pveproxy-ssl.pem` ファイル（および`/etc/pve/local/pveproxy-ssl.key` ファイル）を使用します。

これはパスワードなしである必要があります）が使用されます。

#### 備考

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

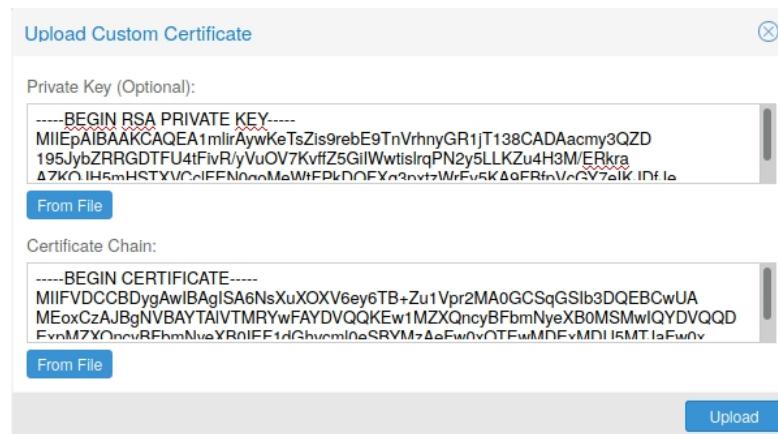
証明書は Proxmox VE Node 管理コマンドで管理します（`pvenode(1)` man-page を参照）。

#### 警告

しない 置換 または 手動 修正 自動 生成 ノード 証明書 ファイル で  
`/etc/pve/local/pve-ssl.pem` および `/etc/pve/local/pve-ssl.key`、または `/etc/pve/pve-root-ca.pem` および  
`/etc/pve/priv/pve-root-ca.key` にあるクラスタ CA ファイル。

### 3.12.3 カスタム証明書のアップロード

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



証明書の鍵ファイルが提供される場合、パスワードで保護されてはならないことに注意してください。

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

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

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

## ACMEアカウント

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

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

ACME アカウントの登録と停止は、Web インターフェース Datacenter -> ACME または pvenode コマンドラインツールを使用して行うことができます。

```
pvenode acme アカウント登録 アカウントmail@example.com
```

### チップ

レート制限のため、実験やACMEを使用する場合はLEステージングを使用する必要があります。

## ACMEプラグイン

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

たとえば、http-では、ウェブサーバーがドメインを管理していることを証明するために、特定の内容のファイルを提供します。技術的な制限や、レコードのアドレスが公共のインターネットから到達できない場合など、これが不可能なこともあります。dns-01チャレンジは、このような使用できます。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実行されます。

The screenshot shows the 'Accounts' section with two entries: 'default' and 'staging'. Below it is the 'Challenge Plugins' section, which contains one entry: 'pdns-example.com' associated with the 'pdns' API.

Add	Edit	Remove
Name ↑		
default		
staging		
Challenge Plugins		
Add	Edit	Remove
Plugin ↑		API
pdns-example.com		pdns

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

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

#### ノード・ドメイン

各ドメインはノード固有です。ノード] -> [証明書]、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を追加したり、既存のドメインエントリを管理したりできます。

The dialog box has the following fields:

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

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

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

更新は[自動的に行われます](#)。

### 3.12.5 ACME HTTPチャレンジプラグイン

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

### 備考

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

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

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

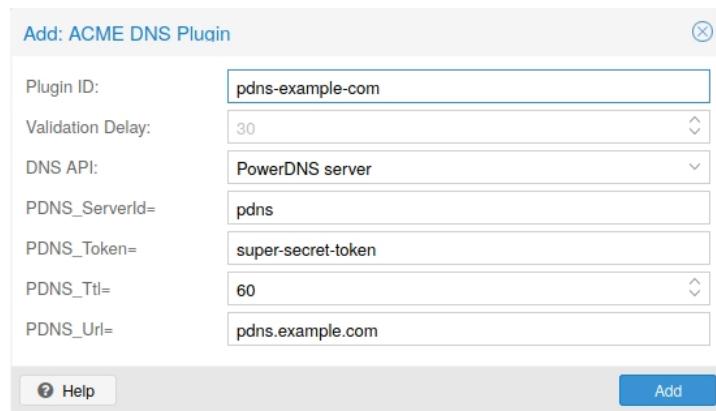
### 3.12.6 ACME DNS APIチャレンジプラグイン

http-01メソッド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いシステム上では、dns-01検証メソッドを使用する可能です。この検証方法では、API経由でTXTレコードをプロビジョニングできるDNSサーバーが必要です。

#### 検証のためのACME DNS APIの設定

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

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



チャレンジタイプとしてDNSを選択します。次に、API プロバイダを選択し、API 経由でアカウントにアクセスするためのクレデンシャルデータを入力します。

検証遅延は、DNSレコードを設定してからACMEプロバイダに検証を促すまでの時間を秒単位で決定します。

#### チップ

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

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

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

#### CNAMEエイリアスによるDNS検証

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

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

#### プラグインの組み合わせ

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

---

#### チップ

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

---

### 3.12.7 ACME証明書の自動更新

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

---

#### 備考

短命の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は、再起動後に証明書の更新が行われないように、pve-daily-update.timer ユニットのランダム遅延を無効にすることをお勧めします。

---

### 3.12.8 pvenodeを使用したACMEの例

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

```
root@proxmox:~# pvenode acme account register defaultmail@example.invalid Directory エンドポイント:  
0) Let's Encrypt V2 (https://acme-v02.api.letsencrypt.org/directory)  
1) Let's Encrypt V2 ステージング (https://acme-staging-v02.api.letsencrypt.org/'  
    ディレクトリ)  
2) カスタム  
選択を入力します: 1
```

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

全ドメイン有効!
...
証明書のダウンロード
pveproxy 証明書と鍵の設定 pveproxy の再起動
タスクOK
```

#### 例ドメインを検証するためのOVH APIの設定

##### 備考

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

##### 備考

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

```
root@proxmox:~# cat /path/to/api-token OVH_AK=XXXXXXXXXXXXXXXXXXXX
OVH_AS=YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
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": "/ドメイン/ゾーン/*/レコード/*"}。
  ]
}'
```

(検証URLを開き、指示に従ってアプリケーションキーを"でリンクしてください。

### アカウント/コンシューマー・キー)

これでACMEプラグインを設定することができます。

```
root@proxmox:~# pvenode acme plugin add dns example_plugin --api ovh --data!
```

/パス/to/api token

```
root@proxmox:~# pvenode acme plugin config example_plugin
```

アピ &#x2502; ovh &#x2502;

```
&#x2502; data &#x2502; OVH_AK=XXXXXXXXXXXXXXXXXXXX  
&#x2502;   &#x2502;   OVH_AS=YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY  
OVH CK=ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ &#x2502;
```

&#x2502; ダイジェスト &#x2502; 867fcf556363ca1bea866863093fcab83edf47a1 &#x2502;

&#x2502; plugin &#x2502; example plugin &#x2502;

タイプ &#x2502; dns &#x2502; &#x2502;

最後に、証明書を取得したいドメインを設定し、そのドメインの証明書を注文します：

```
root@proxmox:~# pvenode config set -acmedomain0 example.proxmox.com,plugin='
```

## 例プラグイン

```
root@proxmox:~# pvenode acme cert order ACMEアカウント詳細の
```

□ —

ACMEの発注

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

```
https://acme-staging-v02.api.letsencrypt.org/acme/authz-v3/33333333'.
```

example.proxmox.comの検証は保留中です!

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

2020] 認証の確認 [Wed Apr 22 09:25:30 CEST 2020] Consumer key は OK です。

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

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

一秒間睡眠

54 面對民

ステータスは「有効」です!

```
[Wed Apr 22 09:25:48 CEST 2020] OVH endpoint: ovh-eu の使用 [Wed Apr 22 09:25:48 CEST 2020] 認証の確認 [Wed Apr 22 09:25:48 CEST 2020] Consumer key は OK です。  
TXTレコードを削除します: _acme-challenge.example.proxmox.comすべてのドメインが検証されました!
```

ました!

CSRの作成 注文状況の確認

ご注文の準備が整いました!

証明書のダウンロード

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

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

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

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

```
root@proxmox:~# pvenode acme account deactivate default  
アカウントファイルの名前を '/etc/pve/priv/acme/default' から '/etc/pve/priv/'に変更しました。  
acme/_deactivated_default_4  
タスクOK
```

```
root@proxmox:~# pvenode acme account register default example@proxmox.com Directory エンドポイント:  
0) Let's Encrypt V2 (https://acme-v02.api.letsencrypt.org/directory)  
1) Let's Encrypt V2 ステージング (https://acme-staging-v02.api.letsencrypt.org/  
ディレクトリ)  
2) カスタム  
選択を入力してください: 0
```

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

上記の同意しますか? [y|N]y

...

タスクOK

## 3.13 ホスト・ブートローダー

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

ZFSをルートファイルシステムとしてインストールしたEFIシステムでは、Secure Bootが有効になっていない限り、`systemd-boot`が使用されます。それ以外の展開では、標準のGRUBブートローダを使用します(これは通常、Debian上にインストールされたシステムにも当てはまります)。

### 3.13.1 インストーラーが使用するパーティション方式

Proxmox VEのインストーラーは、インストール用に選択されたすべてのディスクに3つのパーティションを作成します。作成

されるパーティションは次のとおりです：

- 1 MBのBIOSブートパーティション(gdiskタイプEF02)
- 512 MB EFIシステムパーティション(ESP、gdiskタイプEF00)
- 設定された`hdsizze`パラメータまたは選択されたストレージタイプで使用される残りのスペースにまたがる第3のパーティション

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

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

### 3.13.2 `proxmox-boot-tool`を使ったESPの内容の同期

`proxmox-boot-tool`は、EFIシステムパーティションの内容を適切に設定し、同期させるために使用される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、`vfat`フォーマットのESPからブートするようにそれぞれのブートローダを設定します。ルートファイルシステムとしてのZFSの文脈では、これは、GRUBのZFS実装にも存在するサブセットや、別の小さなブートプールを作成しなければならない代わりに、ルートプールですべてのオプション機能を使用できることを意味します<sup>(6)</sup>。

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

通常の操作では、ESPはマウントされたままにはなりません。これにより、システムがクラッシュした場合に`vfat`フォーマットされたESPのファイルシステムが破壊されるのを防ぎ、プライマリブートデバイスが故障した場合に`/etc/fstab`を手動で変更する必要がなくなります。

`proxmox-boot-tool`は以下の処理します：

- 新しいパーティションのフォーマットと設定
- 新しいカーネル・イメージと`initrd`イメージをリストされたすべてのESPにコピーし、設定します。
- カーネル・アップグレードやその他のメンテナンス作業時の設定の同期
- 同期されているカーネルバージョンのリスト管理

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

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

- 特定のカーネルバージョンをブートするようにブートローダを設定する（pinning） 現在設定されている

ESPとその状態は、以下を実行することで確認できます：

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

#### 同期ESPとして使用するための新しいパーティションの設定

パーティションを同期された ESP としてフォーマット・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、同期メカニズム以前の既存のシステムを接続する場合、proxmox-kerne の proxmox-boot-tool を使用できます。



##### 警告

format<パーティション>をフォーマットします!

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

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

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

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

または

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

を使うと、例えばセキュアブートをサポートするために、systemd-boot ではなく GRUB で初期化するように強制できます。

その後、/etc/kernel/proxmox-boot-uuidsに新しく追加されたパーティションのUUIDの行が追加されるはずです。initコマンドは、設定されているすべてのESPのリフレッシュも自動的に行います。

#### すべてのESPでの設定の更新

すべてのブータブルカーネルをコピーして設定し、/etc/kernel/proxmox-boot-uuidsにリストされたすべてのESPを保持するには  
同期が取れていれば、あとは走るだけです：

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

(rootでext4またはxfsでupdate-grubシステムを実行するのと同じです)。

これは、カーネルコマンドラインに変更を加えたり、すべてのカーネルとinitrdsを同期させたい場合に必要です。

---

##### 備考

update-initramfsもaptも（必要であれば）自動的にリフレッシュを起動します。

---

### proxmox-boot-tool が考慮するカーネルバージョン

以下のカーネルバージョンがデフォルトで設定されています:

- 現在実行中のカーネル
- パッケージ更新時に新しくインストールされるバージョン
- インストール済みの最新カーネル
- 該当する場合は、最後から2番目のカーネルシリーズの最新バージョン（5.0、5.3など）
- 手動で選択したカーネル

### 手動でカーネルをブート可能に

特定のカーネルと initrd イメージをブート可能なカーネルのリストに追加したい場合は proxmox-boot-tool kernel add を使ってください。

たとえば、以下を実行して、ABIバージョン5.0.15-1-pveのカーネルを、すべてのESPにインストールして同期しておくカーネルのリストに追加します:

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

proxmox-boot-tool kernel list は現在起動用に選択されている全てのカーネルバージョンを一覧表示します:

```
# proxmox-boot-tool kernel list 手動で選択した  
カーネル: 5.0.15-1-pve
```

自動的に選択されたカーネル5.0.12-1-pve

```
4.15.18-18-pve
```

proxmox-boot-tool kernel remove を実行して、手動で選択したカーネルのリストからカーネルを削除してください:

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

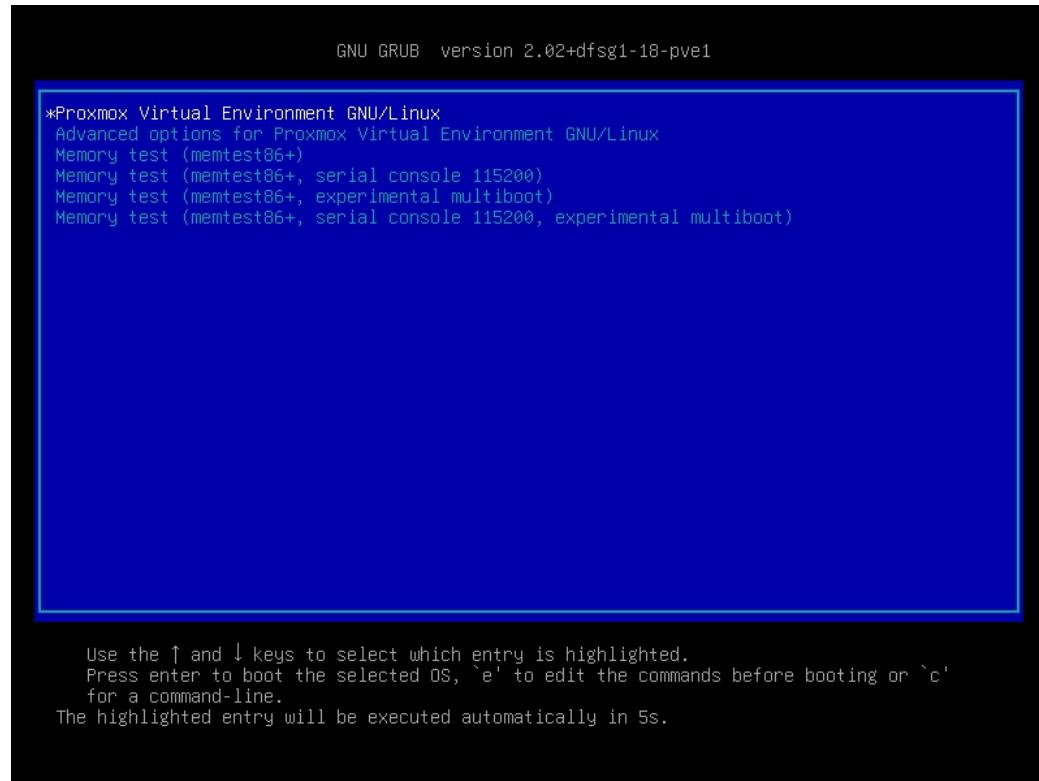
---

### 備考

手動でカーネルを追加または削除した後、すべてのEFIシステムパーティション(ESP)を更新するために proxmox-boot-tool refresh を実行する必要があります。

---

### 3.13.3 使用するブートローダーの決定



どのブートローダが使用されているかを判断する最も簡単で確実な方法は、Proxmox VEノードのブートプロセスを見ることです。

GRUBの青いボックスか、白地に黒のシンプルなsystemd-bootが表示されます。



実行中のシステムからブートローダを判断するのは100%正確ではないかもしれません。最も安全な方法は、以下のコマンドを実行することです：

```
# efibootmgr -v
```

UEFI 変数がサポートされていないというメッセージを返した場合、GRUB は BIOS/Legacy モードで使用されます。出力に以下のような行が含まれる場合、GRUB は UEFI モードで使用されます。

Boot0005\* proxmox [...] File(\EFI\proxmox\Grubx64.efi)

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

```
Boot0006* Linux ブートマネージャ [...] File(| | | | | | |)
```

走ることによって:

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

`proxmox-boot`-設定されているかどうかを調べることができます。

### 3.13.4 グルップ

GRUBは、長年Linuxシステムを起動するための事実上の標準であり、非常によく文書化されています。

7

構成

GRUBの設定変更は、`/etc/default/grub`の`defaults`ファイルまたは`/etc/default/grub.d`の`config snip-pets`を介して行われます。

```
# update-grub
```

### 3.13.5 システムブート

`systemd-boot` は軽量な EFI ブートローダです。インストールされている EFI サービスパーティション (ESP) からカーネルと `initrd` イメージを直接読み込みます。ESP から直接カーネルを読み込む主な、ストレージにアクセスするためのドライバを再実装する必要がないことです。Proxmox VE [proxmox-boot](#)、ESP 上の設定を同期させるために使用されます。

構成

systemd-、EFIシステムパーティション(ESP)のルートディレクトリにあるloader/loader.confファイルで設定します。詳細は `loader.conf(5)` man ページを参照してください。

各ブートローダ・エントリは、loader/entries/ ディレクトリにあるそれ自身のファイルに置かれます。

entry.confの例は以下のようになります（//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>

`proxmox-boot-tool`を使用しているシステムは、`update-grub`時に`proxmox-boot-tool refresh`を呼び出します。

```
タイトル      Proxmox バージョン  
5.0.15-1-pve  
オプション      root=ZFS=rpool/ROOT/pve-1 boot=zfs  
リナックス      /EFI/proxmox/5.0.15-1-pve/vmlinuz-5.0.15-1-pve initrd  
                  /EFI/proxmox/5.0.15-1-pve/initrd.img-5.0.15-1-pve
```

### 3.13.6 カーネル・コマンドラインの編集

カーネルコマンドラインは、使用するブートローダに応じて以下の箇所で変更できます：

#### グループ

カーネルコマンドラインは、次のファイルの変数GRUB\_CMDLINE\_LINUX\_DEFAULTに置く必要があります。

/etc/default/grubに追加します。update-grubを実行すると、その内容が/boot/grub/gのすべてのlinuxエントリに追加されます。

#### システムブート

カーネルコマンドラインは /etc/kernel/cmdline に1行として配置する必要があります。変更を適用するには、proxmox-boot-tool refresh を実行して、loader/entries/proxmox-\*.confのすべての設定ファイルのオプション行として設定します。

カーネルパラメータの完全なリストは、<https://www.kernel.org/doc/html/v<YOUR-KERNEL-VERSION>/admin-guide/kernel-parameters.html>にあります。<YOUR-KERNEL-VERSION>をメジャーバージョンと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は次のようにになります: <https://www.kernel.org/doc/html/v6.5/admin-guide/kernel-parameters.html>

カーネル・バージョンは、ウェブ・インターフェイス (Node→Summary) で確認するか、次のコマンドを実行して調べることができます。

```
# uname -r
```

出力の最初の2つの数字を使用してください。

### 3.13.7 次のブートのためにカーネルバージョンを上書きします。

現在デフォルトでない選択するには、以下のいずれかの方法があります：

- ブートプロセスの最初に表示されるブートローダーメニューを使用します。
- proxmox-boot-tool を使ってシステムをカーネルバージョンに固定します

これは、新しいカーネルバージョンとハードウェア間の非互換性を回避するのに役立つはずです。

#### 備考

このようなピンはできるだけ早く取り外し、最新のカーネルのセキュリティパッチがすべてシステムに適用されるようにしてください。

例えばバージョン5.15.30-1-pveを恒久的に選択して起動するには、次のように実行します:

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

#### チップ

もしあなたのシステムが同期に proxmox-boot-tool を使用しないのであれば、最後に proxmox-boot-tool のリフレッシュコールをスキップすることもできます。

また、カーネルバージョンを次のシステム起動時のみ起動するように設定することもできます。これは例えば、最初にバージョンを固定原因となった問題が、更新されたカーネルによって解決されたかどうかをテストするのに便利です:

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

固定されているバージョン設定を削除するには、unpin サブコマンドを使用します:

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

unpinには--next-bootありますが、これは--next-bootで設定したpinされたバージョンをクリアするために使用します。これはブート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。

ピン留めされたバージョンを設定またはクリアした後は、refreshサブコマンドを実行してESPのコンテンツと設定を同期させる必要があります。

#### チップ

proxmox-boot-toolが管理されているシステムで、対話的にツールを呼び出すと、自動的にプロンプトが表示されます

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

### 3.13.8 セキュアブート

Proxmox VE 8.1 以降、セキュアブートは、署名付きパッケージと以下の統合により、すぐにサポートされます。

proxmox-boot-tool。

セキュアブートの動作には以下のパッケージが必要です。proxmox-secure-boot-support' メタパッケージを使えば一度にインストールできます。

- shim-signed (Microsoftによって署名されたshimブートローダ)
- shim-helpers-amd64-signed フォールバックブートローダと MOKManager、Proxmox により署名済み)
- grub-efi-amd64-signed (GRUB EFI ブートローダ、Proxmox により署名済み)
- proxmox-kernel-6.X.Y-Z-pve-signed (Proxmox により署名されたカーネルイメージ)

他のブートローダは現在セキュアブートコード署名のないため、GRUBのみがブートローダとしてサポートされて。

Proxmox VEを新規インストールすると、自動的に上記のパッケージがすべて含まれます。

セキュアブートの仕組みやセットアップのカスタマイズ方法の詳細については、[wikiをご覧ください](#)。

## 既存のインストールをセキュアブートに切り替える

### 警告

これは、正しく行われないと、場合によっては起動不能なインストールにつながる可能性があります。ホストを再インストールすると、セキュアブートが利用可能であれば、余分な操作なしに自動的にセットアップが行われます。**Proxmox VEホストのバックアップが正常に動作し、テスト済みであることを確認してください!**

Proxmox VE をゼロから再インストールしなくても、必要に応じて既存の UEFI インストールをセキュアブートに切り替えることができます。

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

## システムブート

`systemd-boot`がブートローダとして使用されている場合(どのブートローダが使用されているかを判断するを参照)、いくつかの追加設定が必要です。これは Proxmox VEがZFS-on-rootでインストールされた場合のみです。

後者を確認するには

```
# findmnt /
```

ホストが本当にルートファイルシステムとしてZFSを使用している場合、FSTYPE列には`zfs`が含まれるはずです:

ターゲット・ソース	FSTYPEオプション
/	<code>rpool/ROOT/pve-1 zfs</code> <code>rw,relatime,xattr,noacl,ケースセンシティブ</code>

次に、適切なESP (EFIシステムパーティション) をなりません。これは `lsblk`

コマンドを次のように実行します:

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

出力は次のようになるはずです:

名称	MAJ:MIN	RM	サイズ	ro	タイプ	マウントポイント	fstype
エスデ	8:0	0	32G	0	ディスク		
&#x251c;&#x2500;sda1			8:1	0	1007K	0	部分
&#x251c;&#x2500;sda2			8:2	0	512M	0	部分
&#x2514;&#x2500;sda3			8:3	0	31.5G	0	部分
セロトニン	8:16	0	32G	0	ディスク		
&#x251c;&#x2500;sdb1			8:17	ク	1007K	0	部分
					0		
&#x251c;&#x2500;sdb2			8:18	0	512M	0	部分
&#x2514;&#x2500;sdb3			8:19	0	31.5G	0	部分

この場合、パーティションsda2とsdb2がターゲットです。パーティションのサイズは512Mで、FSTYPEは`vfat`です。

これらのパーティションは、`proxmox-boot-tool` を使って GRUB で起動するように適切に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 に対して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、セットアップの正常性をチェックできます:

```
# efibootmgr -v
```

このリストには、次のようなエントリーが含まれているはずです:

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

#### 備考

古い `systemd-boot` ブートローダは維持されますが、GRUB が優先されます。これにより、セキュアブートモードで GRUB を使用してブートしても何らかの理由で動作しない場合でも、セキュアブートをオフにした `systemd-boot` を使用してシステムをブートすることができます。

これでホストを再起動し、UEFI フームウェアセットアップユーティリティでセキュアブートを有効にすることができます。

再起動すると、`proxmox` という新しいエントリーが UEFI フームウェアブートメニューで選択できるようになっていて、署名済みの EFI shim を使って起動するようになっているはずです。

何らかの理由で UEFI ブートメニューに `proxmox` エントリーが見つからない場合、(フームウェアでサポートされていれば) カスタムブートエントリとしてファイル `proxmox | shimx64.efi` を追加することで、手動で追加してみることができます。

#### 備考

いくつかの UEFI フームウェアは再起動時に `proxmox` ブートオプションを落とすことが知られています。これは `proxmox` のブートエントリがディスク上の GRUB インストールを指していて、ディスク自体がブートオプションになっていない場合に起こります。可能であれば、UEFI フームウェアセットアップユーティリティでディスクをブートオプションに追加して `proxmox-boot-tool` をもう一度実行してください。

#### チップ

カスタムキーを登録するには、付属の [Secure Boot wiki ページ](#)を参照してください。

### セキュアブートでの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になっているシステムでは、カーネルは信頼できる鍵で署名されていないモジュールのロードを拒否します。カーネルパッケージとともに出荷されるデフォルトのモジュールセットは、カーネルイメージに埋め込まれたエフェメラルキーで署名されており、特定のバージョンのカーネルイメージで信頼されています。

DKMS または手動でビルドしたモジュールなど、他のモジュールをロードするには、セキュアブートスタックに信頼された鍵で署名する必要があります。これを実現する最も簡単な方法は、`mokutil` を使用してそれらをマシンオーナーキー (MOK) として登録することです。

`dkms`ツールは自動的に鍵ペアと証明書を `/var/lib/dkms/mok.key` と

`/var/lib/dkms/mok.pub`をビルドして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

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

を作成し、以下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 input password:  
もう一度パスワードを入力してください:
```

mokutilコマンドは(一時的な)パスワードを2回要求しますので、このパスワードはプロセスの次のステップでもう1回入力する必要があります! システムを再起動すると、自動的にMOKManager EFIバイナリが起動し、mokutilを使用して登録を開始するときに選択したパスワードを使用して、キー/証明書を検証し、登録を確認することができます。その後、カーネルは（登録されたMOKで署名された）DKMSでビルドされたモジュールのロードを許可するはずです。必要であれば、MOKはカスタムEFIバイナリやカーネルイメージに署名するためにも使用できます。

DKMSで管理されていないカスタム/サードパーティモジュールにも同じ手順を使用できますが、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・サムページ・マージング (KSM)

Kernel Samepage Merging (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、Proxmox VEではデフォルトで有効になっています。KSMは、物理メモリページの範囲をスキャンして同一のコンテンツを探し、それらにマップされている仮想ページを特定することで機能します。同一のページが場合、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、それへの書き込みはメモリの新しい領域に書き込まれ、共有された物理ページはそのまま残されます。

### 3.14.1 KSMの意味

KSMは、仮想化環境におけるメモリ使用量を最適化することができます。これは、同様のオペレーティング・システムやワークロードを実行する複数のVM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。

しかし、KSMはメモリ使用量を削減できる一方で、VMをサイドチャネル攻撃にさらす可能性があるため、セキュリティ上のリスクも伴います。研究では、KSMの特定の特性を悪用することで、同じホスト上の2つ目のVMを介して、実行中のVMに関する情報を推測することが可能であることが示されています。

したがって、ホスティングサービスを提供するためにProxmox VEを使用している場合は、ユーザーに追加のセキュリティを提供するために、KSMを無効にすることを検討する必要があります。さらに、KSMを無効にすることが法律で義務付けられている場合がありますので、お住まいの国の規制を確認してください。

### 3.14.2 KSMの無効化

KSMがアクティブかどうかを確認するには、以下の出力をチェックします:

```
# systemctl status ksmtuned
```

もしもそうなら、すぐに無効に:

```
# systemctl disable --now ksmtuned
```

最後に、現在マージされているすべてのアンマージするには、以下を実行してください:

```
# echo 2> /sys/kernel/mm/ksm/run
```

## 第4章

# グラフィカル・ユーザー・インターフェース

Proxmox VEはシンプルです。別の管理ツールをインストールする必要はなく、すべてウェブブラウザ（最新のFirefoxまたはGoogle Chromeが望ましい）から行うことができます。ゲストコンソールへのアクセスには、組み込みのHTML5コンソールが使用されます。別の方針として、SPICEを使用することもできます。

Proxmoxクラスタファイルシステム(pmxcfs)を使用しているため、任意のノードに接続してクラスタ全体を管理することができます。各ノードはクラスタ全体を管理できます。専用のマネージャノードは必要ありません。

ウェブベースの管理インターフェイスは、最新のブラウザで使用できます。Proxmox VEがモバイルデバイスからの接続を検出すると、よりシンプルなタッチベースのユーザーインターフェースにリダイレクトされます。

ウェブインターフェイスには、<https://youripaddress:8006>（デフォルトログインはrootで、パスワードはインストールプロセス中に指定されます）を介してアクセスできます。

### 4.1 特徴

- Proxmox VEクラスタのシームレスな統合と管理
- リソースの動的更新のためのAJAX技術
- SSL暗号化(https)によるすべての仮想マシンとコンテナへのセキュアなアクセス
- 数百から数千のVMを処理できる高速な検索主導型インターフェース
- 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（VM、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（ローカル、MS ADS、LDAPなど）のサポート
- 二要素認証（OATH、Yubikey）
- ExtJS 7.x JavaScriptフレームワークをベースにしています。

## 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は様々な認証バックエンド(*Realm*)をサポートしており、ここで言語を選択できます。GUIは20以上の言語に翻訳されています。

### 備考

下部のチェックボックスを選択すると、クライアント側でユーザー名を保存できます。これにより、次回ログイン時に入力の手間が省けます。

## 4.3 GUIの概要

**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つの領域から構成されています。

ヘッダー	上部。ステータス情報が表示され、最も重要なアクションのボタンがあります。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。
コンテンツパネル	中央の領域。選択されたオブジェクトは、ここに設定オプションとステータスを表示します。	
ログパネル	下部にあります。最近のタスクのログエントリが表示されます。これらのログエントリをダブルクリックして詳細を表示したり、実行中のタスクを中止したりできます。	

#### 備考

リソースツリーとログパネルのサイズを縮小・拡大、ログパネルを完全に非表示にすることができます。これは、小さなディスプレイで作業していく、他のコンテンツを表示するためのスペースが欲しい場合に便利です。

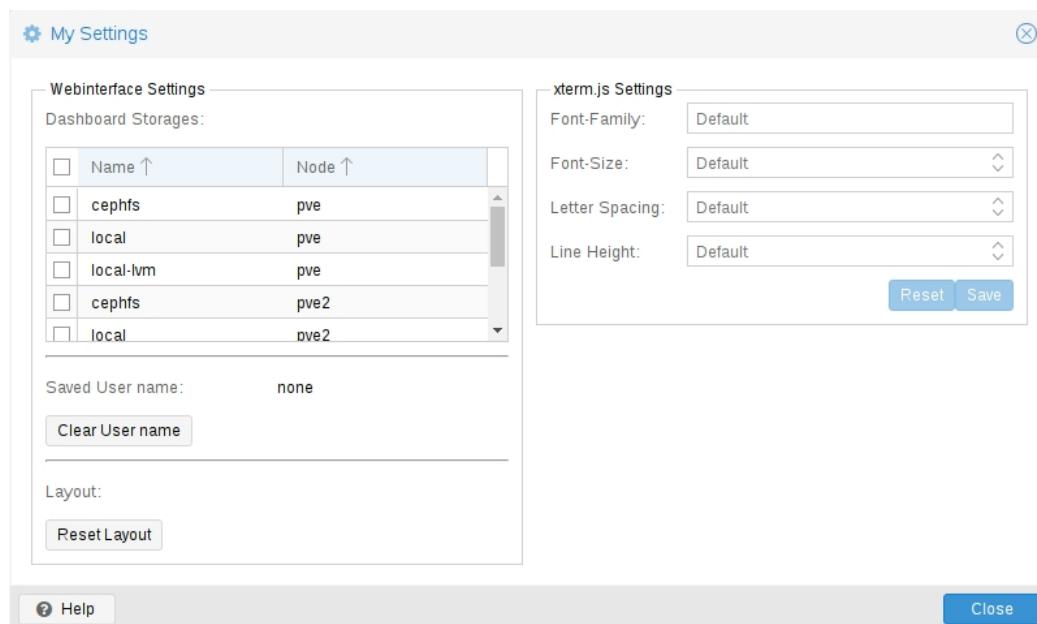
### 4.3.1 ヘッダー

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

ヘッダーの右側には4つのボタンがあります：

ドキュメント	リファレンス 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。仮想マシンの作成
	仮想マシンの作成ウィザードを開きます。
CTの作成	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IDが表示され、クリックするとユーザー固有のオプション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）とノードスワードセルフサービスのショートカットがあります。また、言語とカラーテーマを変更するオプションもあります。最後に、メニューの一番下にログアウトオプションがあります。

### 4.3.2 マイセッティング



「マイ設定」ウィンドウでは、ローカルに保存された設定を設定できます。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・サマリーで表示される合計量にカウントする特定のストレージを有効または無効にできます。どのストレージにもチェックが入っていない場合は、すべてのストレージの合計になります。

ダッシュボード設定の下には、保存されたユーザー名とそれをクリアするボタン、GUI内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 Settings* があります。これらには以下のオプションがあります: Font-Family xterm.jsで使用するフ

ォント（例: Arial）。

**フォントサイズ** 使用するフォントサイズ。

**文字間隔** テキストの文字間隔を増減します。行の高さ 行の高さの絶対値を指定します。

### 4.3.3 リソースツリー

これがメインのナビゲーションツリーです。ツリーの上部では、いくつかの定義済みビューを選択することができます。デフォルトのビューはサーバビューで、以下のオブジェクトタイプが表示されます:

**データセンター** クラスタ全体の設定（すべてのノードに関連）を含みます。

**ノード** ゲストが実行されるクラスタ内のホストを表します。ゲスト VM、コンテナ、テ

ンプレート。

ストレージ	データストレージ。
プール	プールを使用してゲストをグループ化し、管理を簡素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	すべての種類のオブジェクトをオブジェクトの種類別に表示します。プール
ビュー	VMとコンテナをプールごとにグループ化して表示します。
タグビュー	VMとコンテナをタグでグループ化して表示します。

#### 4.3.4 ログパネル

ログパネルの主な目的は、クラスタで現在何が行われているかを表示することです。新しいVMを作成するようなアクションはバックグラウンドで実行され、このようなバックグラウンドジョブをタスクと呼びます。

このようなタスクからの出力はすべて別のログファイルに保存されます。タスクログエントリーをダブルクリックするだけで、そのログを見ることができます。そこで実行中のタスクを中止することもできます。

そのため、他のクラスタノードで誰かが作業しているのをリアルタイムで確認できます。

---

##### 備考

リストを短くするために、古いタスクと終了したタスクをログパネルから削除します。しかし、ノードパネル内のタスク履歴でこれらのタスクを見つけることができます。

---

短時間で実行されるアクションの中には、すべてのクラスタメンバにログを送信するだけのものもあります。これらのメッセージはクラスターログパネル。

#### 4.4 コンテンツパネル

リソースツリーから項目を選択すると、対応するオブジェクトの設定とステータス情報がコンテンツパネルに表示されます。以下のセクションでは、この機能の概要を説明します。より詳細な情報については、リファレンス・ドキュメントの対応する章を参照してください。

## 4.4.1 データセンター

The screenshot shows the Proxmox VE Datacenter interface. On the left, there's a sidebar with a tree view of clusters: Server View, Datacenter (prod-eu-central), prod1, prod2, prod3, and development. The Datacenter node is expanded. Below the sidebar is a search bar and a navigation bar with Documentation, Create VM, Create CT, and admin@pve buttons.

The main area is titled "Datacenter" and contains a table of resources. The columns are Type, Description, Disk usage..., Memory us..., CPU usage, and Uptime. The table lists various resources including lxc, node, pool, qemu, storage, and local storage. Some rows are highlighted in blue, such as the row for "storage cp (prod3)".

Below the table is a "Cluster log" section with tabs for Tasks and Cluster log. The Tasks tab shows a table of log entries with columns Start Time, End Time, Node, User name, Description, and Status. The log entries show VM creation, destruction, and configuration changes for nodes prod1 and prod2.

データセンター・レベルでは、クラスタ全体の設定とアクセスできます。

- 検索: ノード、VM、コンテナ、ストレージデバイス、プールをクラスタ全体で検索します。
- Summary:** クラスタの健全性とリソースの使用状況の概要を示します。
- クラスタ:** クラスタの作成または参加に必要な機能と情報を提供します。
- オプション:** クラスタ全体のデフォルト設定を表示および管理します。
- ストレージ:** クラスタストレージを管理するためのインターフェイスを提供します。
- バックアップ:** バックアップジョブをスケジュールします。これはクラスタ全体で動作するので、スケジューリング時にVM/コンテナがクラスタ上のどこにあるかは関係ありません。
- レプリケーション:** レプリケーションジョブの表示と管理。
- パーミッション:** ユーザー、グループ、APIトークンのパーミッション、LDAP、MS-AD、Two-Factor認証を管理します。
- HA:** Proxmox VE High Availability を管理します。

- **ACME:** サーバーノードにACME (Let's Encrypt) 証明書を設定します。
- **ファイアウォール:** Proxmoxファイアウォールのクラスタ全体の設定とテンプレートの作成。
- **メトリックサーバ:** Proxmox VEの外部メトリックサーバを定義します。
- **通知:** Proxmox VEの通知動作とターゲットを設定します。
- **サポート:** サポート契約に関する情報を表示します。

## 4.4.2 ノード

The screenshot shows the Proxmox VE 6.0-4 interface. On the left, the 'Server View' sidebar lists the Datacenter (prod-eu-central) with nodes prod1, prod2 (selected), prod3, and development. The main area displays the 'Summary' for node prod2, which has been up for 11 days and 00:07:07. It shows CPU usage (1.92% of 4 CPU(s)), IO delay (0.19%), Load average (0.50, 0.28, 0.21), RAM usage (25.12% of 7.88 GiB of 31.38 GiB), HD space (root) (6.20% of 2.49 GiB of 40.11 GiB), KSM sharing (0 B), and SWAP usage (0.00% of 0 B of 8.00 GiB). Below this are details about the CPU (4 x Intel(R) Xeon(R) CPU E3-1220 v3 @ 3.10GHz (1 Socket)), Kernel Version (Linux 5.0.15-1-pve #1 SMP PVE 5.0.15-1 (Wed, 03 Jul 2019 10:51:57 +0200)), and PVE Manager Version (pve-manager/6.0-4/779fa8d98). Two charts are present: 'CPU usage' showing a peak around July 15, 2019, at 19:57:00, and 'Server load' showing a value of 0.6. At the bottom, there are sections for 'Tasks' (listing recent VM operations like Destroy and Create) and 'Cluster log'.

クラスタ内のノードは、この個別に管理できます。

トップヘッダにはReboot、Shutdown、Shell、Bulk Actions、Helpといった便利なボタンがあります。ShellにはnoVNC、SPICE、xterm.jsのオプションがあります。Bulk Actionsには、Bulk Start、Bulk Shutdown、Bulk Migrateのオプションがあります。

- **検索:** ノードのVM、コンテナ、ストレージデバイス、プールを検索します。
- **Summary:** ノードのリソース使用状況の概要を表示します。
- **注釈:** 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・インターフェースへのアクセス。
- システム:** ネットワーク、DNS、時刻の、 syslogへのアクセス。
- アップデート:** システムをアップグレードし、利用可能な新しい表示します。
- ファイアウォール:** 特定のノードのProxmoxファイアウォールを管理します。
- ディスク:** 接続されているディスクの概要を把握し、その使用方法を管理します。
- Ceph:** は、ホストにCephサーバをインストールしている場合にのみ使用します。この場合、ここでCephクラスタを管理し、ステータスを確認できます。
- レプリケーション:** レプリケーションジョブの表示と管理。
- タスク履歴:** 過去のタスクのリストを見ることができます。
- サブスクリプション:** サブスクリプションキーをアップロードし、サポートケースで使用するためのシステムレポートを生成します。

#### 4.4.3 ゲスト

The screenshot shows the Proxmox VE 6.0-4 interface. On the left, the 'Server View' sidebar lists datacenters, nodes, and virtual machines. A specific VM named '99999' is selected. The main area displays the 'Summary' tab for this VM, showing details such as 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 is a 'CPU usage' chart with a single data point at 0.8. At the bottom, there are tabs for 'Tasks' and 'Cluster log', which show a history 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、どちらもテンプレートに変換できます。一つはカーネルベースの仮想マシン(KVM)で、もう一つはLinuxコンテナ(LXC)です。これらのナビゲーションはほとんど同じです。

さまざまなゲスト管理インターフェイ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、コンソールへのアクセスとタイプ、クローニング、HA、ヘルプなどのコマンドを含んでいます。これらのボタンのいくつかはドロップダウンメニューを含んでおり、例えば **Shutdown** には他の電源オプションも含まれています：コンソールには、*SPICE*、*noVNC*、*xterm.js* などのコンソールタイプがあります。

右側のパネルには、左側のメニューから選択された項目のインターフェイスが含まれています。利用可能なインターフェースは以下の通りです。

- **Summary:** VMのアクティビティの簡単な概要と、[Markdown構文](#)のコメント用のNotesフィールドを提供します。
- **コンソール:** VM/コンテナの対話型コンソールにアクセスできます。
- **(KVM)Hardware:** KVM VMで使用可能なハードウェアを定義します。
- **(LXC)Resources:** LXCで使用可能なシステムリソースを定義します。
- **(LXC)Network:** コンテナのネットワーク行います。
- **(LXC)DNS:** コンテナのDNS構成します。
- **オプション:** ゲストのオプションを管理します。
- **タスク履歴:** 選択したゲストに関連する過去のすべてのタスクを表示します。
- **(KVM) モニタ:** KVMプロセスへの対話型通信インターフェース。
- **バックアップ:** システムバックアップの作成と復元。
- **レプリケーション:** 選択したゲストのレプリケーションジョブを表示および管理します。
- **スナップショット:** VMスナップショットの作成とリストア。
- **ファイアウォール:** VM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イスと同様に、ストレージのインターフェイスは、左側の特定のストレージエレメント用のメニューと、右側のこれらのエレメントを管理するためのインターフェイスで構成されています。

このビューでは、2つのパーティションに分かれています。左側にはストレージオプションがあり、選択したオプションの内容が表示されます。

- 概要:** ストレージの種類、使用状況、保存されているコンテンツなど、ストレージに関する重要な情報が表示されます。
- コンテンツ:** バックアップ、ISOイメージ、CTテンプレートなど、ストレージが保存する各コンテンツタイプのメニュー項目。
- 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. On the left, the 'Server View' sidebar lists nodes: prod1, prod2, prod3, and development. prod2 is expanded to show storage volumes: 100 (VM 100), cephfs (prod2), cp (prod2), iso (prod2), local (prod2), and local-lvm (prod2). The main area is titled 'Resource Pool: development'. The 'Summary' tab is selected, showing a status bar with 'Status' and a comment field containing 'IT development pool'. Below this is a 'Tasks' section with a table of recent operations:

Start Time ↓	End Time	Node	User name	Description	Status
Jul 15 20:35:56	Jul 15 20:35:57	prod1	admin@pve	VM 9000 - Destroy	OK
Jul 15 20:35:53	Jul 15 20:35:53	prod1	admin@pve	VM 9000 - Create	OK
Jul 15 20:19:51	Jul 15 20:19:51	prod1	admin@pve	VM 99999 - Destroy	OK
Jul 15 20:19:45	Jul 15 20:19:45	prod1	admin@pve	VM 99999 - Create	OK
Jul 15 20:19:02	Jul 15 20:19:03	prod1	admin@pve	VM 9000 - Destroy	OK

繰り返しますが、プールビューは2つのパーティションで構成されています：左側にメニュー、右側に各メニュー項目に対応するインターフェース。

- **概要：** プールの説明を表示します。
- **Members：** プールメンバー（ゲストとストレージ）の表示と管理。
- **Permissions：** プールの権限を管理します。

## 4.5 タグ

The screenshot shows the Proxmox VE web interface with the following details:

- Virtual Machine Summary:** DemoVM (ID 99999) on node 'prod1'. 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).
- Notes:** No notes present.
- Logs:** CPU usage chart from 2022-11-18 14:52:00 to 2022-11-18 15:57:00, showing values between 0% and 0.5%.
- Logs:** Memory usage chart from 2022-11-18 14:52:00 to 2022-11-18 15:57:00, showing values between 0 Bytes and 0.5 Bytes.

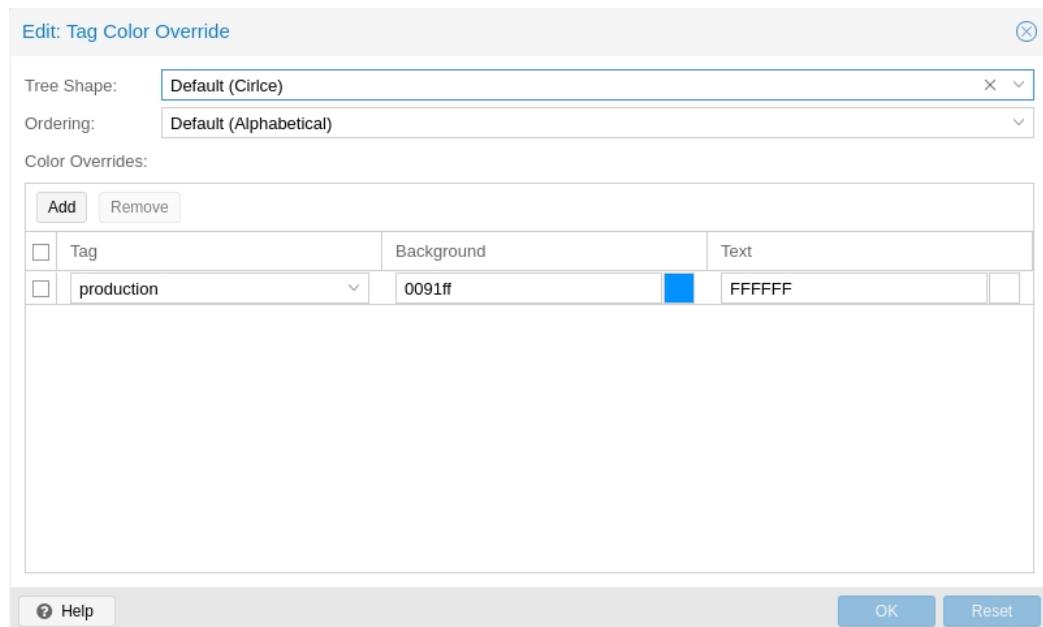
組織的な目的のために、ゲストにタグを設定することができます。現在のところ、これらは情報価値を提供するだけです。タグは、リソースツリーと、ゲストが選択されたときのステータス行の2つの場所に表示されます。

タグは、鉛筆アイコンをクリックして、ゲストのステータスラインで追加、編集、削除することができます。+ボタンを押すと複数のタグを追加でき、- 押すと削除できます。変更を保存またはキャンセルするには、それぞれ✓ と ✕ ボタンを使用できます。

複数のタグをセミコロンで区切ってCLIで設定することもできます。例えば

```
# qm set ID --tags myfirsttag;mysecondtag
```

## 4.5.1 スタイル構成



デフォルトでは、タグの色はテキストから決定論的に導かれます。色、リソースツリー内の形状、大文字小文字の区別、およびタグの並べ替え方法は、カスタマイズできます。これは、Webインターフェイスの Datacenter → Options → Tag Style Override から実行できます。または、CLI 経由で行うこともできます。例えば

```
# pvesh set /cluster/options --tag-style color-map=example:000000:FFFFFF
```

タグ例の背景色を黒 (#000000) に、文字色を白 (#FFFFFF) に設定します。

## 4.5.2 アクセス許可

Keyboard Layout	German (de)
HTTP proxy	none
Console Viewer	Default (xterm.js)
Email from address	root@\$hostname
MAC address prefix	none
Migration Settings	network=10.3.0.64/24,type=insecure
HA Settings	shutdown_policy=migrate
U2F Settings	None
WebAuthn Settings	None
Bandwidth Limits	None
Maximal Workers/bulk-action	4
Next Free VMID Range	Default
Tag Style Override	Tree Shape: Default (Circle), Ordering: Default (Alphabetical), production
User Tag Access	Mode existing, Pre-defined: allowed
Registered Tags	backup

デフォルトでは、ゲスト (/vms/) の VM.Config.Options 権限を持つユーザーは、以下のように任意のタグを設定できます。

この動作を制限したい場合は、「Datacenter Options User Tag Access: User Tag Access」で適切な権限を設定できます。この動作を制限したい場合は、*Datacenter*→*Options*→*User Tag Access*で適切なパーミッションを設定できます：

- *free*: ユーザーはタグの設定を制限されません(デフォルト)
- *リスト*: ユーザーは事前に定義されたタグのリストに基づいてタグを設定できます。
- *既存*: リストのようなものですが、既存のタグを使用することもできます。
- *none*: ユーザーはタグの使用を制限されます。

CLIでも同じことができます。

登録タグのリストは、*Datacenter Options*で編集できます。登録されたタグのリストは、*Datacenter*→*Options*→*Registered Tags*またはCLIで編集できます。

正確なオプションとCLIでの呼び出し方法の詳細については、「[データセンターの構成](#)」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、*Datacenter*→*Options*で設定できます。

→ 同意テキスト。コンテンツがない場合、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 クラスタマネージ

ヤ

Proxmox VEクラスタマネージャは、物理サーバのグループを作成するためのツールです。このようなグループを**クラスタ**と呼びます。信頼性の高いグループ通信にはCorosync Cluster Engineを使用します。クラスタ内のノード数に明確な制限はありません。実際には、実際に可能なノード数はホストとネットワークのパフォーマンスによって制限される可能性があります。現在（2021年）、50ノードを超えるクラスタ（ハイエンドのエンタープライズハードウェアを使用）が稼働しているという報告があります。

、新しいクラスタの作成、クラスタへのノードの参加、クラスタからの離脱、ステータス情報の取得、およびその他の様々なクラスタ関連タスクの実行に使用できます。Proxmox Cluster File System ("pmxefs") は、クラスタ構成をすべてのクラスタノードに透過的に配布するために使用されます。

ノードをクラスタにグループ化すると、次のような利点があります：

- ウェブベースの集中管理
- マルチマスタークラスター：各ノードがすべての管理タスクを実行可能
- データベース駆動型ファイルシステムpmxefsを設定ファイルの保存に使用し、corosyncを使用して全ノードでリアルタイムにレプリケートされます。
- 物理ホスト間での仮想マシンとコンテナの容易な移行
- 迅速な展開
- ファイアウォールやHAなどのクラスタ全体のサービス

### 5.1 必要条件

- corosyncが動作するためには、すべてのノードがUDPポート5405-5412を介して相互に接続できる必要があります。
- 日付と時刻は同期していなければなりません。
- ノード間でTCPポート22のSSHトンネルが必要です。
- 高可用性に興味がある場合は、信頼できるクオーラムのために少なくとも3つのノードが必要です。すべてのノードが同じバージョンである必要があります。
- 特に共有ストレージを使用する場合は、クラスタ・トラフィック用に専用のNICを使用することをお勧めします。

- ノードを追加するには、クラスタ・ノードのルート・パスワードが必要です。
- 仮想マシンのオンラインマイグレーションは、ノードに同じベンダーのCPUが搭載されている場合にのみサポートされます。それ以外の場合は動作する可能性がありますが、保証はされません。

---

**備考**

Proxmox VE 3.x以前とProxmox VE 4.Xのクラスタノードを混在させることはできません。

---

**備考**

Proxmox VE 4.4とProxmox VE 5.0のノードを混在させることは可能ですが、本番構成としてはサポートされていません。

---

**備考**

Proxmox VE 6.xのクラスタを以前のバージョンで実行することはできません。Proxmox VE 6.xとそれ以前のバージョンのクラスタプロトコル(corosync)が根本的に変更されました。Proxmox VE 5.4用のcorosync 3パッケージは、Proxmox VE 6.0へのアップグレード手順のみを対象としています。

---

## 5.2 ノードの準備

まず、すべてのノードにProxmox VEをインストールします。各ノードが最終的なホスト名とIP設定でインストールされていることを確認してください。クラスタ作成後にホスト名とIPを変更することはできません。

`etc/hosts`ですべてのノード名とそのIPを参照するのが一般的ですが（あるいは他の手段で名前を解決できるようにする）、これはクラスタを動作させるために必要なことではありません。というのも、あるノードから別のノードにSSHで接続する場合、覚えやすいノード名を使うことができるからです（[リンクアドレスの種類も参照してください](#)）。クラスタ構成では常にIPアドレスでノードを参照することをお勧めします。

## 5.3 クラスタの作成

クラスタの作成は、コンソール上で行うか(`ssh`でログイン)、Proxmox VEのWebインターフェース(*Datacenter*→ *Cluster*)を使用してAPIから行うことができます。

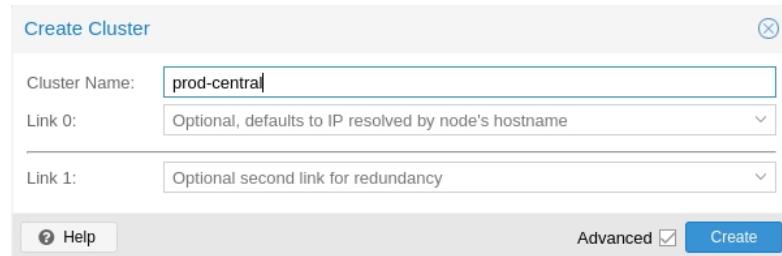
---

**備考**

クラスタには一意の名前を付けます。この名前は後で変更できません。クラスタ名はノード名と同じ規則に。

---

### 5.3.1 ウェブGUIで作成



*Datacenter→ Cluster】で、【Create Cluster】をクリックします。クラスタ名を入力し、ドロップダウン・リストからメイン・クラスタ・ネットワーク（リンク0）として使用するネットワーク・コネクションを選択します。デフォルトは、ノードのホスト名で解決されたIPです。*

Proxmox VE 6.2では、クラスタに最大8つのフォールバックリンクを追加できます。冗長リンクを追加するには、*Add*ボタンをクリックし、それぞれのフィールドからリンク番号とIPアドレスを選択します。Proxmox VE 6.2以前では、フォールバックとして2つ目のリンクを追加するには、*Advanced*チェックボックスを選択し、追加のネットワークインターフェース（リンク1、[Corosyncの冗長性](#)も参照）を選択します。

#### 備考

クラスタ通信用に選択したネットワークが、ネットワークストレージやライブマイグレーションなど、トラフィックの多い目的に使用されていないことを確認してください。クラスタネットワーク自体は少量のデータを生成しますがレイテンシに非常に敏感です。[クラスタネットワーク要件](#)の詳細を確認してください。

### 5.3.2 コマンドラインでの作成

最初のProxmox VEノードにssh経由でログインし、以下のコマンドを実行します:

```
hp1# pvecm create CLUSTERNAME
```

新しいクラスタの状態を確認するには、以下を使用します:

```
hp1# pvecmステータス
```

### 5.3.3 同一ネットワーク内の複数のクラスタ

同じ物理ネットワークまたは論理ネットワークに複数のクラスタを作成することができます。この場合、クラスタ通信スタックでの衝突を避けるため、各クラスタに一意の名前を付ける必要があります。さらに、クラスタを明確に区別できるようにすることで、人間の混乱を避けることができます。

corosyncクラスタの帯域幅要件は比較的低いものの、パッケージのレイテンシと1秒あたりのパッケージ（PPS）レートが制限なります。同じネットワーク内の異なるクラスタは、これらのリソースをめぐって互いに競合する可能性があるため、大規模なクラスタには別の物理ネットワークインフラを使用することが理にかなっている場合があります。

## 5.4 クラスタへのノードの追加

### 注意

クラスタに参加すると、`/etc/pve`にある既存の設定はすべて上書きされます。特に、ゲストIDが衝突する可能性があるため、参加ノードはゲストを保持できず、ノードはクラスタのストレージ構成を継承します。既存のゲストを持つノードに参加するには、回避策として各ゲストのバックアップを作成し(`vzdump`を使用)、参加後に異なるIDでリストアします。ノードのストレージレイアウトが異なる場合は、ノードのストレージを再追加し、ストレージが実際に利用可能なノードに反映されるように各ストレージのノード制限を適応させる必要があります。

### 5.4.1 GUIによるクラスタへのノード参加

**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:	eyJpcEFkZHJlc3MiOiixOTluMTY4LjI2LjIyNSIslmZpbmdlcnByaW50ijoNTY6QUi6Qtg6REQ6NEU6RjQ6ODU6Rkl6QkQ6Qzk6OTM6NTU6MUM6QzI6Nkl6RDc6ODg6NTQ6QjY6MDU6Qjk6MTg6MDI6OUE6MjU6MEQ6QjU6Mzk6RDc6Rku6N0M6RDYiLCJyaW5nX2FkZHliOlsimTkyLjE2OC4yNi4yMjUiLG51bGxdLCJ0b3RlbSI6eyJjbHvzrdGVvX25hbWUjOjwcm9kLWV1LWNlbnRvYWwiLCJpbnRlcmbhY2UiOnsiMCi6evJsaW5rbnVtYmVlioiMCJ

**Copy Information**

既存のクラスタノードでWebインターフェイスにログインします。`Datacenter → Cluster` で、上部の【Join Information】ボタンをクリックします。次に、【情報をコピー】ボタンをクリックします。または、【情報】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: eyJpcEFkZHJlc3MiOiixOTluMTY4LjI2LjIyNSIslmZpbmdlcnByaW50ijoNTY6QUi6Qtg6REQ6NEU6RjQ6ODU6Rkl6QkQ6Qzk6OTM6NTU6MUM6QzI6Nkl6RDc6ODg6NTQ6QjY6MDU6Qjk6MTg6MDI6OUE6MjU6MEQ6QjU6Mzk6RDc6Rku6N0M6RDYiLCJyaW5nX2FkZHliOlsimTkyLjE2OC4yNi4yMjUiLG51bGxdLCJ0b3RlbSI6eyJjbHvzrdGVvX25hbWUjOjwcm9kLWV1LWNlbnRvYWwiLCJpbnRlcmbhY2UiOnsiMCi6evJsaW5rbnVtYmVlioiMCJ

Peer Address: 192.168.26.225

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

**Join**

次に、追加するノードのWebインターフェースにログインします。`Datacenter → Cluster` で、【Join Cluster】をクリックします。【情報】フィールドに、先ほどコピーした【参加情報】テキストを入力します。クラスタへの参加に必要なほとんどの設定が自動的に入力されます。セキュリティ上の理由から、クラスタのパスワードは手動で入力する必要があります。

### 備考

必要なデータをすべて手動で入力するには、Assisted Joinチェックボックスを無効にします。

Joinボタンをクリックすると、クラスタ参加プロセスが直ちに開始されます。ノードがクラスタに参加すると、現在のノード証明書はクラスタ認証局(CA)から署名されたものに置き換えられます。つまり、現在のセッションは数秒後に機能しなくなります。この場合、Webインターフェースを強制的にリロードし、クラスタ認証情報で再度ログインする必要があります。

これで、`Datacenter → Cluster`にノードが表示されるはずです。

## 5.4.2 コマンドラインによるクラスタへのノード参加

既存のクラスタに参加させたいノードにsshでログインします。

```
# pvecm add IP-ADDRESS-CLUSTER
```

IP-ADDRESS-CLUSTERには、既存のクラスタ・ノードのIPまたはホスト名を使用します。IP推奨されます(「[リンクアドレスの種類](#)」を参照)。

クラスタの状態を確認するには

```
# pvecmステータス
```

### 4ノード追加後のクラスタステータス

```
# クラスタ情報
~~~~~
名前 prod-central
Config Version: 3 トランスポート
ネット
安全な認証 オン

定足数情報
~~~~~
日付 9月14日 (火) 11:06:47 2021
クオーラムプロバイダ: corosync_votequorum ノード数: 4
ノードID 0x00000001
リングID 1.1a8
定足数 はい

定足数情報
~~~~~
予想得票数 4
予想最高4
総得票数 4
定足数 3
フラグ 定足数

会員情報
~~~~~

ノードID 投票 名称
0x00000001 1 192.168.15.91
0x00000002 1 192.168.15.92 (現地)
0x00000003 1 192.168.15.93
0x00000004 1 192.168.15.94
```

全ノードのリストだけが必要な、次のようにします:

```
# pvecm ノード
```

### クラスタ内のノードを一覧表示

```
# pvecm ノード
```

#### 会員情報

ノードID	投票	名称
1	1	馬力
2	1	hp2 (ローカル)
3	1	馬力
4	1	馬力4

### 5.4.3 分離されたクラスタネットワークでのノードの追加

クラスタ・ネットワークが分離されているクラスタにノードを追加する場合は、*link0*パラメータをそのネットワーク上のノードのアドレスを設定する必要があります

```
# pvecm add IP-ADDRESS-CLUSTER --link0 LOCAL-IP-ADDRESS-LINK0
```

:

クロノスネットのトランスポート・レイヤーに内蔵されている冗長性を使用する場合は、*link1*パラメータも使用してください。GUIを使用して、Cluster Joinの対応するLink Xフィールドから正しいインターフェイスを選択することができます。

ダイアログ

### 5.5 クラスタノードの削除



#### 注意

手続きを進める前に注意深くお読みください。

ノードからすべての仮想マシンを移動します。ローカル データまたはバックアップのコピーが作成されている確認します。さらに、削除するノードへのスケジュールされたレプリケーション ジョブをすべて削除してください。

#### 注意

ノードを削除する前にそのノードへのレプリケーションジョブを削除しないと、レプリケーションジョブが削除できなくなります。特に、レプリケートされたVMがマイグレーションされるとレプリケーションの方向が自動的に切り替わるため、削除するノードからレプリケートされたVMをマイグレーションすることで、そのノードへのレプリケーションジョブが自動的に設定されることに注意してください。

削除するノードがCeph用に構成されている場合:

- 稼働中のOSDを持つProxmox VEノードが十分に存在することを確認します。

**備考**

デフォルトでは、Cephサイズ/min\_sizeは3/2で、オブジェクトバランサCRUSHの障害ドメインはフルノードです。そのため、実行中のOSDを持つノードがサイズ(3)に満たない場合、データの冗長性が低下します。min\_size未満のノードしかオンラインでない場合、プールI/Oはブロックされ、影響を受けたゲストはクラッシュする可能性があります。

2. 十分なモニタ、マネージャ、およびCephFSを使用している場合はメタデータサーバが利用可能であることを確認します。
3. データの冗長性を維持するため、OSD、特にノード上の最後のOSDが破壊されるたびに、データのリバランスが発生します。したがって、残りのノード上のOSDに十分な空き領域が残っていることを確認してください。
4. 削除するノードからCephを削除するには、まずOSDを1破棄します。
5. CEPHのステータスが再びHEALTH\_OKになったら、次のように進みます：

1. メタデータサーバは、Ceph→ CephFSのWebインターフェースを介して、または実行によって破壊されます：

```
# pveceph mds destroy <ローカルホスト名>.
```

2. モニター破壊
3. 監督潰し

6. 最後に空のバケツ(削除するProxmox VEノード)をCRUSH階層から削除します：

```
# ceph osd crush remove <ホスト名>.
```

次の例では、クラスタからノードhp4を削除します。

別のクラスタ・ノード(hp4ではない)にログインし、pvecm nodesコマンドを実行して削除するノードIDを特定します：

```
hp1# pvecm ノード

会員情報
~~~~~ ノードID 投票 名称
      1   1 hp1 (ローカル)
      2   1 馬力2
      3   1 馬力
      4   1 馬力4
```

この時点でhp4の電源を切り、現在の設定のまま（ネットワーク内で）再び電源が入らないようにする必要があります。

**重要**

 上述したように、ノードを取り外す前に電源を切り、現在の構成で（既存のクラスタネットワーク内で）電源が再投入されないことを確認することが重要です。そのままノードの電源を入れると、クラスタが壊れてしまい、機能する状態に復元するのが困難になる可能性があります。

ノードhp4の電源を切ったら、安全にクラスタから削除できます。

```
hp1# pvecm delnode hp4 ノード4をキル
```

#### 備考

この時点で、Could not kill node (error = CS\_ERR\_NOT\_EXIST) というエラー・メッセージが表示される可能性があります。これはノードの削除に失敗したのではなく、オフラインのノードを kill しようとした corosync に失敗したことを意味します。したがって、これは安全に無視できます。

pvecm nodes または pvecm status を使ってノードのリストをもう一度確認してください。以下のなるはずです：

```
hp1# pvecm ステータス
```

...

#### 定足数情報

```
~~~~~ 予想得票数
```

3

予想最高3

総得票数 3

定足数 2

フラグ 定足数

#### 会員情報

```
~~~~~
```

##### ノードID 投票 名称

0x00000001	1	192.168.15.90 (ローカル)
0x00000002	1	192.168.15.91
0x00000003	1	192.168.15.92

何らかの理由で、このサーバーを再び同じクラスタに参加させたい場合は、そうしなければなりません：

- Proxmox 新規インストールしてください、
- そして、前のセクションで説明したように、それを結合します。

削除されたノードの設定ファイルはまだ `/etc/pve/nodes/hp4` に存在します。まだ必要な設定を回復してから、ディレクトリを削除してください。

#### 備考

ノードの削除後も、その SSH フィンガープリントは他のノードの `known_hosts` に残っています。同じ IP またはホスト名を持つノードに再参加した後に SSH エラーが発生した場合は、再参加したノードで `pvecm updatecerts` を 1 回実行してフィンガープリントをクラスタ全体で更新します。

## 5.5.1 再インストールせずにノードを分離



### 注意

これは推奨される方法ではありません。不安な、前の方法を使用してください。

ノードを再インストールせずに、クラスタからノードを分離することもできます。ただし、クラスタからノードを削除した後も、そのノードは共有ストレージにアクセスできます。これは、クラスタからノードの削除を開始する前に解決しておく必要があります。Proxmox VEクラスタは他のクラスタとまったく同じストレージを共有することはできません。さらに、VMIDの競合につながる可能性もあります。

分離したいノードだけがアクセスできる新しいストレージを作成することをお勧めします。例えば、NFS上の新しいエクスポートや新しいCephプールなどです。まったく同じストレージが複数のクラスタからアクセスされないようにすることが重要です。このストレージを設定したら、すべてのデータとVMをノードからそのストレージに移動します。これで、クラスタからノードを分離する準備が整いました。



### 警告

すべての共有リソースがきれいに分離されていることを確認してください！ そうしないと、コンフリクトや問題が発生します。

まず、ノードのcorosyncとpve-clusterサービスを停止します：

```
systemctl stop pve-cluster
systemctl stop corosync
```

クラスタファイルシステムをローカル再度起動します：

```
pmxcfs -l
```

corosync設定削除します：

```
rm /etc/pve/corosync.conf
rm -r /etc/corosync/*
```

これで、ファイルシステムを通常の再開できます：

```
killall pmxcfs
systemctl start pve-cluster
```

これでノードはクラスタから切り離されました。クラスタの残りのノードから削除するには

```
pvecm delnode oldnode
```

残りのノードのクオーラムが失われたためにコマンドが失敗した場合、回避策として予想得票数を1に設定することができます：

```
pvecm 予想1
```

そして `pvecm delnode` コマンドを繰り返します。

次に、分離したノードに戻り、そのノードに残っているクラスタファイルをすべて削除します。これにより、ノードを問題なく別のクラスタに追加できるようになります。

```
rm /var/lib/corosync/*
```

他のノードの設定ファイルがまだクラスタ ファイル システムに残っているため、それらもクリーンアップします。ノード名が正しいことを確認したら、`/etc/pve/nodes/NODENAME` からディレクトリ全体を再帰的に削除します。

#### 注意

ノードのSSH鍵は `authorized_key` ファイルに残ります。これは、ノードがまだ公開鍵認証で互いに接続できることを意味します。これを修正するには、`/etc/pve/priv/authorized_keys` ファイルからそれぞれの鍵を削除します。

## 5.6 定足数

Proxmox VEは、クオーラムベースの技術を使用して、すべてのクラスタノード間で一貫性のある状態を提供します。

クオーラム（定足数）とは、分散システムにおいて分散トランザクションが操作を実行するために最低限必要な投票数のことです。

- Wikipediaより クオーラム（分散コンピューティング）

ネットワーク・パーティショニングの場合、状態の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です。クオーラムを失うと、クラスタは読み取り専用モードに切り替わります。

#### 備考

Proxmox VEはデフォルトで各ノードに1票を割り当てます。

## 5.7 クラスター・ネットワーク

クラスター・ネットワークはクラスタの中核です。これを介して送信されるすべてのメッセージは、それぞれの順序ですべてのノードに確実に配信されなければなりません。Proxmox VEでは、この部分は高性能、低オーバーヘッド、高可用性の開発ツールキットの実装である corosync によって行われます。これは分散型設定ファイルシステム (`pmxcfs`) に対応しています。

### 5.7.1 ネットワーク要件

Proxmox VE クラスタスタックを安定して動作させるには、全ノード間のレイテンシが5ミリ秒以下（LANパフォーマンス）の信頼性の高いネットワークが必要です。ノード数が少ないセットアップでは、より高いレイテンシのネットワークでも動作する可能性がありますが、これは保証されたものではなく、ノード数が3つ以上、レイテンシが約10ミリ秒を超えると、むしろ可能性が低くなります。

corosyncはあまり帯域幅を使用、レイテンシのジッターの影響を受けやすいので、ネットワークは他のメンバーによって多用されるべきではありません。特に、corosyncとストレージに共有ネットワークを使わないでください（冗長構成で優先順位の低いフォールバックの可能性を除いて）。

クラスタをセットアップする前に、ネットワークがそのいるかどうかを確認することは良い習慣です。クラスタ・ネットワーク上でノードが互いに接続できることを確認するには、pingツールでノード間の接続性をテストします。

Proxmox VEファイアウォールが有効な場合、corosyncのACCEPTルールが自動的に生成されます。

#### 備考

Corosyncは、バージョン3.0（Proxmox VE 6.0で導入）以前はマルチキャストを使用していました。最近のバージョンでは、クラスタ通信は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](#)でトランスポートをudpまたはudpuに設定することで、マルチキャストまたはレガシーユニキャストを有効にすることはできますが、暗号化と冗長性のサポートがすべて無効になることに注意してください。これは推奨されません。

## 5.7.2 クラスタネットワーク

パラメータなしでクラスタを作成する場合、corosyncクラスタのネットワークは通常、WebインターフェースおよびVMのネットワークと共有されます。セットアップによっては、ストレージトラフィックも同じネットワーク上で送信されるかもしれません。corosyncはタイムクリティカルでリアルタイムなアプリケーションなので、これを変更することをお勧めします。

#### 新しいネットワークの設定

まず、新しいネットワークインターフェイスをセットアップする必要があります。これは物理的に別のネットワーク上にある必要があります。ネットワークがクラスタ・ネットワークの要件を満たしていることを確認してください。

#### クラスタ作成時の分離

これは、新しいクラスタの作成に使用される `pvecm create` コマンドの `linkX` パラメータで可能です。

10.10.10.1/25の静的アドレスで追加のNICをセットアップし、すべてのクラスタ通信をこのインターフェイスで送受信したい場合は、次のように実行します：

```
pvecm create test --link0 10.10.10.1
すべてが正しく機能しているかどうかを確認するには、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、[クラスタ・ネットワークを分離したノードを追加します](#)。

## クラスタ作成後の分離

既にクラスタを作成していて、クラスタ全体を再構築することなく、その通信を別のネットワークに切り替える場合にこれを行うことができます。この変更により、ノードはcorosyncを再起動し、新しいネットワーク上で次々に立ち上がる必要があるため、クラスタ内でクオーラムが短時間失われる可能性があります。

まず、[corosync.confファイルの編集](#)方法を確認してください。そして、それを開くと、以下のようなファイルがあるはずです：

```
logging { debug:  
    off  
    to_syslog: yes  
}
```

### ノーデリスト

```
ノード  
  name: due  
  nodeid: 2  
  quorum_votes: 1 ring0_addr: due  
}
```

```
ノード  
  name: tre  
  nodeid: 3  
  quorum_votes: 1 ring0_addr: tre  
}
```

```
ノード  
  name: uno  
  nodeid: 1  
  quorum_votes: 1 ring0_addr: uno  
}
```

```
}
```

### クオーラム

```
プロバイダ: corosync_votequorum  
}
```

### トーテム

```
cluster_name: testcluster  
config_version: 3 ip_version: ipv4-6  
secauth: on  
バージョン: 2  
interface {.  
    リンク番号: 0  
}  
}
```

---

#### 備考

`ringX_addr` は実際には corosync リンクアドレスを指定します。`ring` "という名前は、古いバージョンの corosync の名残で、後方互換性のために残されています。

---

最初に行なうことは、もしまだ表示されていなければ、ノード・エントリーに名前プロパティを追加することです。これらはノード名と一致していなければなりません。

次に、すべてのノードの`ring0_addr`プロパティからすべてのアドレスを新しいアドレスに置き換えます。ここでは、プレーンIPアドレスまたはホスト名を使用できます。ホスト名を使用する場合は、すべてのノードから解決可能であることを確認してください（[リンクアドレスの種類](#)も参照してください）。

この例では、クラスタ通信を10.10.10.0/25ネットワークに切り替えます。

それぞれのノードの`ring0_addr`。

---

#### 備考

全く同じ手順で、他の`ringX_addr`値も変更できます。しかし、一度に1つのリンクアドレスだけを変更することをお勧めします。

---

`config_version`プロパティを増やすと、新しいコンフィギュレーション・ファイルは次のようになります：

```
logging { debug: off
    to_syslog: yes
}
```

#### ノードリスト

```
    name: due
    nodeid: 2
    定足数_投票1
    ring0_addr: 10.10.10.2
}
```

#### ノード

```
    name: tre
    nodeid: 3
    定足数_投票1
    ring0_addr: 10.10.10.3
}
```

#### ノード

```
    name: uno
    nodeid: 1
    定足数_投票1
    ring0_addr: 10.10.10.1
}
```

```
}
```

#### クオーラム

```
プロバイダ: corosync_votequorum
}

トーテム
cluster_name: testcluster
config_version: 4 ip_version: ipv4-6
secauth: on
バージョン: 2
interface {.
    リンク番号: 0
}

}
```

そして、変更した情報がすべて正しいことを最終確認した後、保存し、もう一度、[corosync.confファイルの編集](#)セクションに従って、有効にします。

変更はライブで適用されるので、corosyncの再起動は厳密には必要ありません。他の設定も変更した場合や、corosync が文句を言っているのに気づいた場合は、オプションで再起動をトリガーできます。

単一ノードで実行します：

```
systemctl restart corosync
```

では、問題がないか確認してください：

```
systemctl status corosync
```

corosyncが再び動作し始めたら、他の全てのノードでもcorosyncを再起動してください。その後、新しいネットワーク上でクラスタメンバーシップに1つずつ参加します。

### 5.7.3 コロシンクアドレス

corosyncリンクアドレス（後方互換性のため、corosync.confでは*ringX\_addr*と表記）は、2つの方法で指定できます：

- IPv4/v6アドレスは直接使用できます。これらは静的で、通常は不用意に変更されることがないため、推奨されます。
- ホスト名はgetaddrinfoを使用して解決されます。つまり、デフォルトでは、IPv6アドレスが利用可能であれば最初に使用されます (man gai.conf も参照してください)。特に既存のクラスタをIPv6にアップグレードする場合は、この点に注意してください。

---

 注意

ホスト名の解決先アドレスは、corosyncやそれが動作しているノードに触れることなく変更することができるため、注意して使用する必要があります。

---

ホスト名を優先する場合は、corosync専用の別個の静的ホスト名を推奨します。また、クラスタ内のすべてのノードがすべてのホスト名を正しく解決できることを確認してください。

Proxmox VE 5.1以降、サポートされている間は、ホスト名は入力時に解決されます。解決されたIPのみが設定に保存されます。

以前のバージョンでクラスタに参加したノードは、corosync.confで未解決のホスト名をまだ使用している可能性があります。上記のように、IPまたは別のホスト名で置き換えるのが良いかもしれません。

## 5.8 コロシンクの冗長性

Corosyncは、統合されたKronosnetレイヤーを介した冗長ネットワーキングをデフォルトでサポートしています（レガシーの/トランスポートではサポートされていません）。複数のリンク指定することで、有効にすることができます。

pvecm の `--linkX` パラメータ、GUI で **Link 1** クラスタ作成中または新規ノード追加中)、または corosync.conf で複数の `ringX_addr` を指定します。

### 備考

有用なフェイルオーバーを提供するためには、すべてのリンクが独自の物理ネットワーク接続上にある必要があります。

リンクは優先度の設定に従って使用されます。この優先順位は、corosync.conf の対応する interface セクションで `knet_link_priority` を設定するか、pvecm でクラスタを作成するときに `priority` パラメータを使うことで設定できます：

```
# pvecm create CLUSTERNAME --link0 10.10.10.1,priority=15 --link1'  
    10.20.20.1,priority=20
```

この場合、優先順位が高いlink1が先に使われることになります。

優先順位が手動で設定されていない場合（または2つのリンクの優先順位が同じ場合）、リンクは番号順に使用され、番号の小さい方が優先順位が高くなります。

すべてのリンクが動作していても、最も優先度の高いリンクだけがコロシンク・トラフィックを見ることになります。リンクの優先度を混在させることはできません。つまり、優先度の異なるリンクは通信することができません。

優先順位の低いリンクは、優先順位の高いリンクがすべて故障しない限りトラフィックが発生しないため、他のタスク（VM、ストレージなど）に使用するネットワークを優先順位の低いリンクに指定することは有効な戦略になります。最悪の場合、待ち時間が長かったり混雑していたりする接続の方が、全く接続しないよりはましかもしれません。

### 5.8.1 既存のクラスタへの冗長リンクの追加

実行中のコンフィギュレーションに新しいリンクを追加するには、まず [corosync.confファイルの編集](#) 方法を確認してください。

次に、nodelist セクションのすべてのノードに新しい `ringX_addr` をします。はどのノードに追加しても同じで、各ノードで一意であることを確認してください。

最後に、Xを上記で選択したリンク番号に置き換えて、下図のような新しいインターフェイスをトーテムセクションに追加します。

1番のリンクを追加したと仮定すると、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  
}
```

#### ノデリスト

```
ノード  
  name: due  
  nodeid: 2  
  定足数_投票1  
    ring0_addr: 10.10.10.2  
    ring1_addr: 10.20.20.2  
}
```

```
ノード  
  name: tre  
  nodeid: 3  
  定足数_投票1  
    ring0_addr: 10.10.10.3  
    ring1_addr: 10.20.20.3  
}
```

```
ノード  
  name: uno  
  nodeid: 1  
  定足数_投票1  
    ring0_addr: 10.10.10.1  
    ring1_addr: 10.20.20.1  
}  
}
```

#### クオーラム

```
プロバイダ: corosync_votequorum  
}
```

#### トーム

```
cluster_name: testcluster  
config_version: 4 ip_version: ipv4-6  
secauth: on  
バージョン: 2  
interface {  
    リンク番号: 0  
}  
インターフェース { linknumber: 1  
}  
}
```

最後の手順に従って[corosync.confファイルを編集](#)すると、すぐに新しいリンクが有効になります。再起動

は必要ありません。corosyncが新しいリンクを読み込んだかどうかは、以下の方法で確認できます：

```
journalctl -b -u corosync
```

1つのノードで古いリンクを一時的に切断し、切断中もステータスがオンラインであることを確認して、新しいリンクをテストするのがよいでしょう：

pvecmステータス

健全なクラスタ状態が表示された場合は、新しいリンクが使用されていることを意味します。

## 5.9 Proxmox VEクラスタにおけるSSHの役割

Proxmox VEは様々なSSHトンネルを利用しています。

- コンソール/シェルセッションのプロキシ（ノードとゲスト）

ノード接続している状態でノードBのシェルを使用する場合、ノードAのターミナルプロキシに接続し、そのターミナルプロキシは非インタラクティブSSHトンネルを介してノードBのログインシェルに接続します。

- セキュアモードでのVMおよびCTのメモリとローカルストレージのマイグレーション。

移行中、移行情報を交換し、メモリとディスクの内容を転送するために、1つ以上のSSHトンネルが移行元と移行先のノード間で確立されます。

- ストレージのレプリケーション

### 5.9.1 SSHセットアップ

Proxmox VE、SSH設定/セットアップに以下の変更が行われます：

- rootユーザーのSSHクライアントがChaCha20よりもAESを優先するように設定されている場合。
- rootユーザのauthorized\_keysファイルは/etc/pve/priv/authorized\_keysにリンクされます。
- sshdはパスワードを使ってrootとしてログインできるように設定されています。

---

#### 備考

古いシステムでは、/etc/ssh/ssh\_known\_hosts が次の場所を指すシンボリックリンクとして設定されていることもあります。

/このシステムは pve-cluster <<INSERT VERSION>> の明示的なホスト鍵固定に置き換えられました。このシステムは pve-cluster <<INSERT VERSION>> の明示的なホスト鍵の固定に置き換えられ、シンボリックリンクは pvecm updatecerts --unmerge-known-hosts を実行することで解除できます。

---

## 5.9.2 .bashrcとその兄弟の自動実行による落とし穴

カスタム .bashrc や、設定されたシェルによってログイン時に実行される同様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、自動的にそれを実行します。これは予期せぬ動作を引き起こす可能性があります。なぜなら、これらのコマンドは上記のどの操作に対しても root 権限で実行される可能性があるからです。これは問題となる副作用を引き起こす可能性があります！

このようなややこしいことを避けるためには、/root/.bashrc に、セッションが対話型であることを確認チェックを追加し、その上で .bashrc コマンドを実行することをお勧めします。

このスニペットを .bashrc ファイルの先頭に追加してください：

```
# 副作用を避けるため、対話的に実行しない場合は早めに終了します case $- in
  *i*) ;;
    *) return;;
エサック
```

## 5.10 コロシンク外部投票サポート

このセクションでは、Proxmox VEクラスタに外部ボータを配置する方法について説明します。このように構成すると、クラスタはクラスタ通信の安全特性に違反することなく、より多くのノード障害に耐えることができます。

これが機能するためには、2つのサービスがです：

- 各Proxmox VEノードで実行されるQDeviceデーモン
- 独立したサーバー上で動作する外部投票デーモン

その結果、小規模なセットアップ（例えば2+1ノード）でも高い可用性を実現できます。

### 5.10.1 QDevice技術概要

Corosync Quorum Device (QDevice)は、各クラスタノード上で実行されるデーモンです。外部で実行されるサードパーティのアビトリーターの決定に基づいて、クラスタのクオーラムサブシステムに設定された投票数を提供します。主な用途は、標準的なクオーラム規則で許容されるよりも多くのノード故障にクラスタが耐えられるようにすることです。外部デバイスはすべてのノードを見るため、投票を行うノードのセットを1つだけ選択することができ、安全に行うことができます。これは、サードパーティの投票を受けた後、そのノードのセットがクオーラムを（再び）持つことができる場合にのみ実行されます。

現在、QDevice Netのみがサードパーティ・アビトリーターとしてサポートされています。これは、ネットワーク経由でパーティションメンバーに到達できる場合、クラスタパーティションに投票を提供するデーモンです。常にクラスタの1つのパーティションにしか票を与えません。複数のクラスタをサポートするように設計されており、設定や状態はほとんど自由です。新しいクラスタは動的に処理され、QDeviceを実行しているホスト上に設定ファイルは必要ありません。

外部ホストの唯一の要件は、クラスタへのネットワークアクセスが必要であることと、corosync-qnetdパッケージが利用可能であることです。私たちはDebianベースのホスト用のパッケージを提供していますが、他のLinuxディストリビューションもそれぞれのパッケージマネージャからパッケージ入手できるはずです。

#### 備考

corosync自体とは異なり、QDeviceはTCP/IPでクラスタに接続します。デーモンはクラスタのLAN実行でき、corosyncの低レイテンシ要件に制限されません。

## 5.10.2 対応セットアップ

偶数ノードのクラスタではQDevicesをサポートしており、より高い可用性を提供する必要がある場合は2ノードクラスタでの使用を推奨しています。奇数ノード数のクラスタについては、現在のところQDeviceの使用を推奨していません。その理由は、クラスタタイプごとにQDeviceが提供する投票数に差があるためです。偶数のクラスタには追加の票が1票ありますが、これは可用性を高めるだけで、QDevice自体に障害が発生した場合は、QDeviceをまったく使用しない場合と同じ状態になるからです。

一方、クラスタサイズが奇数の場合、QDeviceは $(N-1)$ 票を提供します。この代替動作は理にかなっています。もしQDeviceに1票しか追加の票がなかった場合、クラスタは分裂状態に陥る可能性があります。このアルゴリズムでは、1つのノードを除くすべてのノード（そして当然QDevice自体も）が故障する可能性があります。しかし、これには2つの欠点があります：

- QNetデーモン自体に障害が発生した場合、他のノードに障害が発生するか、クラスタが直ちにクオーラムを失います。たとえば、15台のノードを持つクラスタでは、クラスタがクオーラムを失う前に7台が故障する可能性があります。しかし、ここでQDeviceが構成され、それ自体が故障した場合、15ノードのうち1ノードも故障することはありません。この場合、QDeviceはほぼ単一障害点として機能します。
- 1つのノードとQDeviceを除くすべてのノードに障害が発生する可能性があるという事実は、最初は有望に聞こえますが、その結果、HAサービスが大量に復旧し、残りの1つのノードに過負荷がかかる可能性があります。さらに、 $((N-1)/2)$ ノード以下しかオンラインでない場合、Cephサーバはサービスの提供を停止します。

欠点とその意味を理解すれば、奇数クラスタのセットアップでこの技術を使うかどうかを自分で決めることができます。

## 5.10.3 QDevice-Netのセットアップ

corosync-qdeviceへの投票を提供するデーモンは、非特権ユーザーとして実行することをお勧めします。Proxmox VEとDebianはその設定されたパッケージを提供しています。Proxmox VEにQDeviceを安全かつセキュアに統合するには、デーモンとクラスタ間のトラフィックを暗号化する必要があります。

まず、外部サーバーにcorosync-qnetdパッケージをインストールします。

```
外部# apt install corosync-qnetd
```

とcorosync-qdeviceパッケージを全クラスタノードにインストールします。

```
pve# apt install corosync-qdevice
```

これを実行した後、クラスタ内のすべてのノードがオンラインであることを確認します。

Proxmox VEノードの1つで以下のコマンドを実行して、QDeviceをセットアップできます：

```
pve# pvecm qdevice setup <QDEVICE-IP> .
```

クラスタのSSHキーが自動的にQDeviceにコピーされます。

### 備考

外部サーバーのrootユーザーに鍵ベースのアクセスを設定するか、セットアップ段階で一時的にrootにパスワード付きでログインできるようにしてください。この段階で「*Host key verification failed.*」などのエラーが表示された場合は、pvecm updatecertsを実行すると問題が解決する可能性があります。

すべてのステップが正常に完了すると、「完了」と表示されます。でQDeviceがセットアップされたことを確認できます：

```
pve# pvecm ステータス
...
定足数情報
~~~~~ 予想得票数
3
予想最高3
総得票数 3
定足数 2
フラグ クオーレート Qdevice

会員情報
~~~~~ ノードID 投票 Qデバイス 名称
0x00000001 1 A、V、NMW 192.168.22.180 (ローカル)
0x00000002 1 A、V、NMW 192.168.22.181
0x00000000 1 Qデバイス
```

#### Qデバイス・ステータス・フラグ

上記のように、QDeviceのステータス出力には通常3つの列が含まれます：

- A / NA: Alive または Not Alive。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がノードに投票する場合。ノード間のcorosync接続がダウンしているが、両方のノードが外部のcorosync-qnetdデーモンとまだ通信できるようなスプリットブレインの状況では、1つのノードだけが投票権を得ます。
- MW / NMW: マスターが勝つか (MW) 、勝たないか (NMW) 。デフォルトはNMW。<sup>1</sup> 参照。
- NR: QDeviceが登録されていません。

---

#### 備考

QDeviceがNot Alive (上記の出力でNA) と表示されている場合は、外部サーバーのポート5403 (qnetdサーバーのデフォルトポート) がTCP/IP経由で到達可能であることを確認してください！

---

## 5.10.4 よくある質問

### タイブレーク

2つの同じ大きさのクラスターパーティションが互いに見えないがQDeviceは見えるという同点の場合、QDeviceはそれらのパーティションの1つをランダムに選び、そのパーティションに票を提供します。

---

<sup>1</sup> votequorum\_qdevice\_master\_wins マニュアルページ [https://manpages.debian.org/bookworm/libvotequorum-dev/-votequorum\\_qdevice\\_master\\_wins.3ja.html](https://manpages.debian.org/bookworm/libvotequorum-dev/-votequorum_qdevice_master_wins.3ja.html)

## 否定的な影響の可能性

ノード数が偶数であるクラスタでは、QDeviceを使用してもマイナスの影響はありません。QDeviceが機能しない場合は、QDeviceがないのと同じです。

## QDeviceセットアップ後のノードの追加と削除

QDeviceが設定されたクラスタに新しいノードを追加したり、既存のノードを削除したりする場合は、まずQDeviceを削除する必要があります。その後、普通にノードを追加または削除できます。ノード数が均等クラスタに戻ったら、前述の方法でQDeviceを再度セットアップできます。

## QDeviceの取り外し

公式のpvecmツールを使ってQDeviceを追加した場合はそれを削除することができます：

```
pve# pvecm qdevice remove
```

## 5.11 コロシンクの設定

etc/pve/corosync.confファイルはProxmox VEクラスタで中心的な役割を果たします。このファイルはクラスタのメンバーシップと制御します。corosync.confの詳細については、corosync.confのマニュアル参照してください：

```
man corosync.conf
```

ノード・メンバーシップについては、常にProxmox VEが提供するpvecm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かもしれません。以下はそのためのベストプラクティスのヒントです。

### 5.11.1 corosync.conf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2つあり、1つは

/etc/pve/corosync.confと/etc/corosync/corosync.confにあります。クラスタ・ファイル・システムにあるものを編集すると、ローカルのものに変更が伝搬されますが、その逆はありません。

ファイルが変更されるとすぐに、設定は自動的に更新されます。つまり、実行中のcorosyncに統合できる変更は即座に反映されます。従って、編集中にファイルを保存する際に意図しない変更を引き起こさないように、常にコピーを作成し、代わりにそれを編集する必要があります。

```
cp /etc/pve/corosync.conf /etc/pve/corosync.conf.new
```

次に、すべてのProxmox VEノードにプリインストールされているnanoやvim.tinyなどのお気に入りのエディタで設定ファイルを開きます。

#### 備考

コンフィギュレーションをした後は、必ずconfig\_version番号をインクリメントしてください。

必要な変更を行った後、現在の作業用設定コピーをもう1つ作成します。これは、新しい設定の適用に失敗したり、その他の問題が発生した場合のバックアップとして機能します。

```
cp /etc/pve/corosync.conf /etc/pve/corosync.conf.bak
```

そして、古い設定ファイルを新しい置き換えます:

```
mv /etc/pve/corosync.conf.new /etc/pve/corosync.conf
```

変更が自動的に適用されるかどうかは、以下のコマンドで確認できます:

```
systemctl status corosync journalctl  
-b -u corosync
```

変更が自動的に適用されない場合は、corosyncサービスを再起動する必要があります:

```
systemctl restart corosync
```

エラーについては、以下のトラブルシューティングのセクションを確認してください。

## 5.11.2 トラブルシューティング

問題: `quorum.expected_votes` を設定する必要があります。

corosyncが失敗し始め、システムログに以下のメッセージが表示された場合:

```
[...]  
corosync[1647]: 初期化      [QUORUM] 定数プロバイダ: corosync_votequorum が失敗しました。  
          。  
corosync[1647]:           [サーブ     ] サービスエンジン 'corosync_quorum' のロードに失敗しました。          ←'  
    訳あって  
    'configuration error: nodelist or quorum.expected_votes must be'  
    構成された!」。  
[...]
```

これは、コンフィギュレーションで`corosync ringX_addr`に設定したホスト名が解決できなかったことを意味します。

### 非クオーレート時のコンフィギュレーションの書き込み

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、自分が何をしているかを理解している場合は、これを使用します:

```
pvecm 予想1
```

これで予想される投票数が1に設定され、クラスタがクオーレートになります。その後、設定を修正したり、最後に動作したバックアップに戻したりすることができます。

corosyncが起動できなくなった場合は、これだけでは不十分です。この場合、`/etc/corosync/corosync.conf`にあるcorosync設定のローカルコピーを編集し、corosyncが再び起動できるようにするのが最善です。スプリットブレインの状況を避けるために、すべてのノードでこのコンフィギュレーションが同じ内容であることを確認してください。

## 5.11.3 Corosync設定用語集

### リングXアドレス

これは、ノード間のクロノスネット接続のための異なるリンクアドレスに名前を付けます。

## 5.12 クラスターコールドスタート

すべてのノードがオフラインの場合、クラスタがクォーレートでないことは明らかです。これは停電後によくあるケースです。

---

### 備考

無停電電源装置 ("UPS"、"バッテリー・バックアップ"とも呼ばれます) を使ってこのような状態にならないようにするのは、特にHAを望む場合には常に良い考え方です。

---

ノードの起動時に `pve-guests` サービスが開始され、クオーラムを待ちます。定足数が満たされると、`onboot` フラグが設定されているすべてのゲストが起動します。

ノードの電源を入れたとき、または停電後に電源が回復したとき、一部のノードが他のノードよりも速く起動する可能性があります。クオーラムに達するまでゲストの起動が遅れることを覚えておいてください。

## 5.13 ゲストVMID自動選択

新しいゲストを作成する際、ウェブインターフェースは自動的にバックエンドに空いているVMIDを尋ねます。デフォルトの検索範囲は 100 から 1000000 (スキーマによって強制される最大許容 VMID より低い) です。

例えば、一時的なVMと手動でVMIDを選択するVMを簡単に分けたい場合などです。例えば、一時的なVMと手動でVMIDを選択するVMを簡単に分けたい場合などです。また、安定した長さのVMIDを提供したい場合もあり、そのような場合は下限を例えば100000に設定することで、より余裕を持たせることができます。

このユースケースに対応するために、`datacenter.cfg`で下限、上限、または両方の境界を設定することができます。

設定ファイルは、ウェブ・インターフェイスの `「Datacenter → Options」` で編集できます。

---

### 備考

この範囲はnext-id API呼び出しにのみ使用されるので、厳しい制限ではありません。

---

## 5.14 ゲスト移住

仮想ゲストを他のノードに移行することは、クラスタにおいて便利な機能です。このようなマイグレーションの動作を制御する設定があります。これは設定ファイル`datacenter.cfg`を介して、またはAPIまたはコマンドラインパラメータを介して特定のマイグレーションに対して行うことができます。

ゲストがオンラインかオフラインか、またはローカルリソース (ローカルディスクなど) を持っているかどうかに違いがあります。仮想マシンの移行の詳細については、[QEMU/KVM 移行の章](#)を参照してください。

コンテナ移行の詳細については、[コンテナ移行の章](#)を参照してください。

### 5.14.1 マイグレーションタイプ

マイグレーションタイプは、マイグレーションデータを暗号化された（安全な）チャネルで送信するか、暗号化されていない（安全でない）送信するかを定義します。マイグレーションタイプを安全でないものに設定すると

仮想ゲストも暗号化されずに転送されるため、ゲスト内部の重要なデータ（パスワードや暗号化キーなど）の情報漏えいにつながる可能性があります。

従って、ネットワークを完全に制御できず、誰にも盗聴されていないことを保証できない場合は、安全なチャネルを使用することを強くお勧めします。

#### 備考

ストレージの移行はこの設定に従いません。現在、ストレージコンテンツは常に安全なチャネルで送信されます。

暗号化には多くの計算パワーが必要な、この設定はパフォーマンスを向上させるためにしばしば安全でない設定に変更されます。最近のシステムはハードウェアでAES暗号化を実装しているため、その影響は小さくなっています。パフォーマンスへの影響は、10Gbps以上の転送が可能な高速ネットワークで特に顕著です。

## 5.14.2 移住ネットワーク

デフォルトでは、Proxmox VEはクラスタ通信が行われるネットワークを使用して移行トラフィックを送信します。これは、機密性の高いクラスタトラフィックが中断される可能性があることと、このネットワークがノードで利用可能な最良の帯域幅を持たない可能性があるため、いずれも最適ではありません。

移行ネットワークパラメータを設定すると、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ます。メモリに加えて、これはオフライン移行のストレージトラフィックにも影響します。

移行ネットワークはCIDR表記を使用したネットワークとして設定されます。これには、各ノードに個別のIPアドレスを設定する必要がないという利点があります。Proxmox VEは、CIDR形式で指定されたネットワークから移行先ノードの実アドレスを判断できます。これを有効にするには、各ノードがそれぞれのネットワークで正確に1つのIPを持つようにネットワークを指定する必要があります。

#### 例

ここでは、3つのノードがあり、3つのネットワークがあると仮定します。1つはインターネットとのパブリック通信用、もう1つはクラスタ通信用、そしてもう1つはマイグレーション用の専用ネットワークとして使いたい非常に高速なものです。

このようなセットアップのネットワーク構成は以下のようになります：

```
iface eno1 inet マニュアル

# public network 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
```

```
アドレス 10.1.1.1/24
```

```
# 高速ネットワーク自動工
```

```
ノ3
```

```
iface eno3 inet 静的アドレス  
    10.1.2.1/24
```

ここでは、移行ネットワークとしてネットワーク10.1.2.0/24を使用します。1回の移行では、コマンドラインツールのmigration\_networkパラメータを使って

```
# qm migrate 106 tre --online --migration_network 10.1.2.0/24
```

これを行うことができます:

これをクラスタ内のすべてのマイグレーション用のデフォルト・ネットワークとして設定するには、次のように

/etc/pve/datacenter.cfgファイル:

```
# use dedicated migration network migration:  
secure, network=10.1.2.0/24
```

---

#### 備考

その マイグレーション タイプ は 常に でなければなりません。 セット を設定する必要があります。 設定する必要があります。

移行 ネットワーク が 設定 で

/etc/pve/datacenter.cfg。

---

## 第6章

# Proxmox クラスタファイルシステム (pmxcfs)

Proxmox Clusterファイルシステム("pmxcfs")は、設定ファイルを保存するためのデータベース駆動型のファイルシステムで、corosyncを使用してすべてのクラスタノードにリアルタイムでレプリケートされます。Proxmox VEに関連するすべての設定ファイルを保存するために使用します。

ファイルシステムはすべてのデータをディスク上の永続データベース内に格納しますが、データのコピーはRAMに存在します。このため、最大サイズには制限があり、現在は128メガバイトです。これでも数千台の仮想マシンの設定を保存するには十分です。

このシステムには次のような利点があります：

- すべての設定をリアルタイムで全ノードにシームレスにレプリケーション
- VM IDの重複を避けるための強力な一貫性チェックを提供します。
- ノードがクオーラムを失った場合の読み取り専用
- 全ノードへのcorosyncクラスタ構成の自動更新
- 分散ロック機構付き

## 6.1 POSIX互換性

ファイルシステムはFUSEをベースにしているので、動作はPOSIXライクです。しかし、いくつかの機能は単に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は変更できません（この方がVMIDが一意であることを保証しやすくなるからです）。
- ファイルのパーミッションは変更できません。
- `O_EXCL`の作成は（古いNFSのように）アトミックではありませんでした。
- `O_TRUNC`の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

すべてのファイルとディレクトリは、ユーザー `root` が所有し、グループ `www-data` を持っています。書き込み権限を持つのは `root` だけですが、グループ `www-data` はほとんどのファイルを読むことができます。以下のパス以下のファイルは、`root` によってのみアクセス可能です：

```
/etc/pve/priv/
/etc/pve/nodes/${NAME}/priv/ です。
```

## 6.3 テクノロジー

クラスタ通信には [Corosync Cluster Engine](#) を、データベースファイルには [SQLite](#) を使用しています。ファイルシステムは [FUSE](#) を使用してユーザ空間に実装されています。

## 6.4 ファイルシステムのレイアウト

ファイルシステムは

```
/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> シンボリックリンク）
<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のデータセンター全体の構成 (キーポードレイアウト、プロキシ、...)
<code>domains.cfg</code>	Proxmox VE認証ドメイン
ファイアウォール/クラスタ. <code>fw</code>	すべてのノードに適用されるファイアウォール設定
ファイアウォール/ <code>&lt;NAME&gt;.fw</code>	各ノードのファイアウォール設定
ファイアウォール/ <code>&lt;VMID&gt;.fw</code>	VMとコンテナのファイアウォール設定
<code>ha/crm_commands</code>	現在実行中の HA オペレーションを表示します。 CRMが実施
<code>ha/manager_status</code>	HAに関するJSON形式の情報 クラスタ上のサービス
<code>ha/resources.cfg</code>	高可用性によって管理されるリソースとその 併せ
<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
<code>nodes/&lt;NAME&gt;/openvz/</code> です。	Proxmox VE 4.0以前はコンテナに使用 設定データ（非推奨。近日中に削除されます）
<code>nodes/&lt;NAME&gt;/pve-ssl.key</code>	<code>pve-ssl.pem</code> のSSL秘密鍵

nodes/<NAME>/pve-ssl.pem	ウェブサーバのパブリックSSL証明書（署名 クラスタCA）
nodes/<NAME>/pveproxy-ssl.key	pveproxy-ssl.pem の SSL 密鑑 (オプション)
nodes/<NAME>/pveproxy-ssl.pem	ウェブサーバの公開SSL証明書（チェーン (オプションで pve-ssl.pem をオーバーライドします)
ノード/<NAME>/qemu-server/<VMID>.conf	KVM VM用のVMM構成データ
priv/authkey.key	チケットシステムで使用される秘密鍵
priv/authorized_keys	認証用クラスタメンバーのSSHキー
priv/ceph*	Ceph認証キーと関連 資格
priv/known_hosts	検証用クラスタ・メンバーのSSH鍵
priv/lock/*	様々なサービスで使用されるファイルをロックし、安全性を確保 クラスタ全体の運営
priv/pve-root-ca.key	クラスタCAの秘密鍵
priv/shadow.cfg	PVE Realmユーザー用シャドウパスワードファイル
priv/storage/<ストレージID>.pw	ストレージのパスワードがブレーンテキストで格納されています。
priv/tfa.cfg	Base64エンコードされた二要素認証 構成
priv/token.cfg	すべてのAPIトークンの秘密
pve-root-ca.pem	クラスタ CA のパブリック証明書
pve-www.key	CSRF トークンの生成に使用される秘密鍵
sdn/*	ソフトウェア定義のための共有設定ファイル ネットワーキング (SDN)
ステータス.cfg	Proxmox VEの外部メトリクス・サーバー構成
storage.cfg	Proxmox VEのストレージ構成
ユーザー設定	Proxmox VEのアクセス制御設定 (ユーザー/グループ/...)
仮想ゲスト/cpu-models.conf	カスタムCPUモデル保存用
vzdump.cron	クラスタ全体のvzdumpバックアップジョブスケジュール

## 6.4.2 シンボリックリンク

クラスタ・ファイル・システム内の特定のディレクトリでは、ノード独自のファイルを指すためにシンボリックリンクが使用されます。したがって、以下の表で指すファイルはクラスタの各ノードで異なるファイルを指します。

ローカル	ノード/<LOCAL_HOST_NAME>
エルエックス	nodes/<LOCAL_HOST_NAME>/lxc/
オープンベツツ	nodes/<LOCAL_HOST_NAME>/openvz/です。 (非推奨、近日削除)
qemu-server	ノード/<LOCAL_HOST_NAME>/qemu-server

## 6.4.3 デバッグ用の特別なステータス・ファイル (JSON)

バージョン	ファイルのバージョン（ファイルの変更を検出）
メンバー	クラスタメンバーに関する情報
.vmlist	すべてのVMのリスト

クラスタログ .rrd	クラスター ログ (直近50件) RRDデータ (最新のもの)
----------------	------------------------------------

## 6.4.4 デバッグの有効/無効

で冗長なsyslogメッセージを有効にできます:

```
echo "1" >/etc/pve/.debug
```

で、冗長なsyslogメッセージを無効にします:

```
echo "0" >/etc/pve/.debug
```

## 6.5 回復

Proxmox VEホストに大きな問題がある場合（ハードウェアの問題など）、pmxcfsデータベースファイル/var/lib/pve-cluster/config.dbをコピーし、新しいProxmox VEホストに移動すると便利です。新しいホストでは（何も実行していない状態で）、pve-clusterサービスを停止し、config.dbファイルを置き換える必要があります（必要なパーミッションは0600）。これに続いて、/etc/hostnameと失われたProxmox VEホストに従って/etc/hostsを設定し、再起動して確認してください（VM/CTデータも忘れないでください）。

### 6.5.1 クラスタ構成の削除

推奨される方法は、クラスタからノードを削除した後にノードを再インストールすることです。これにより、すべての秘密クラスタ/sshキーと共有設定データが確実に破棄されます。

場合によっては、[再インストール](#)せずにノードをローカルモードに戻したいこともあります。

### 6.5.2 故障したノードからのゲストの回復/移動

nodes/<NAME>/qemu-server/ (VM)およびnodes/<NAME>/lxc/ (コンテナ)内のゲスト設定ファイルについて、Proxmox VEはそれぞれのゲストの所有者として含まれるノード<NAME>を認識します。これにより、ゲスト設定の同時変更を防止するために、高価なクラスタ全体のロックの代わりにローカルロックを使用できます。

その結果、ゲストの所有ノードに障害が発生した場合(たとえば、停電、フェンシングイベントなど)、(オフラインの)所有ノードのローカルロックが取得できないため、(すべてのディスクが共有ストレージにある場合でも)通常のマイグレーションはできません。Proxmox VEの高可用性スタックには、フェンスで保護されたノードからのゲストの正しい自動回復を保証するために必要な（クラスタ全体の）ロックとウォッチドッグ機能が含まれているため、これはHA管理されたゲストの問題ではありません。

HA管理されていないゲストが共有ディスクのみを持つ場合(そして、障害ノードでのみ利用可能な他のローカルリソースがない場合)、/etc/pve/内の障害ノードのディレクトリからオンラインノードのディレクトリにゲスト設定ファイルを移動するだけで、手動での復旧が可能です(これにより、ゲストの論理的な所有者または場所が変更されます)。

たとえば、ID 100のVMをオフラインのノード1から別のノードnode2にリカバリするには、クラスタの任意のメンバーノードでrootとして次のコマンドを実行します:

```
mv /etc/pve/nodes/node1/qemu-server/100.conf /etc/pve/nodes/node2/'  
qemu-server/
```

**警告**

のようにゲストを手動でリカバリする前に、障害が発生したソースノードが本当に電源オフ/フェンスされていることを絶対に確認してください。そうしないと、Proxmox VE のロックの原則が mv コマンドによって破られ、予期しない結果を招く可能性があります。

**警告**

ローカルディスク(またはオフラインノードでのみ利用可能な他のローカルリソース)を持つゲストはこの方法では復旧できません。障害が発生したノードがクラスタに再参加するのを待つか、バックアップからそのようなゲストをリストアしてください。

## 第7章

# プロックスモックスVEストレージ

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

VMを共有ストレージに格納する主な利点の1つは、クラスタ内のすべてのノードがVMディスクイメージに直接アクセスできるため、ダントンタイム稼働中のマシンをライブマイグレーションできることです。VMイメージ・データをコピーする必要がないため、ライブ・マイグレーションは非常に高速です。

ストレージライブラリ(libpve-storage-perl パッケージ)は、柔軟なプラグインシステムを使用して、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。これは、さらなるストレージタイプを追加するために簡単に採用することができます。

## 7.1 ストレージの種類

基本的に2つのタイプがあります：

### ファイルレベルのストレージ

ファイル・レベル・ベースのストレージ技術では、完全な機能を備えた（POSIX）ファイル・システムにアクセスできます。一般的に、ブロックレベルのストレージ（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、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。

### ブロックレベルのストレージ

大きなRAW画像を保存できます。通常、このようなストレージに他のファイル（ISO、バックアップなど）を保存することはできません。最近のブロックレベルのストレージ実装のほとんどは、スナップショットとクローンをサポートしています。RADOSとGlusterFSは、ストレージデータを異なるノードに複製する分散システムです。

表7.1：利用可能なストレージタイプ

説明	プラグインタイプ	レベル	共有	スナップ写真	安定
ZFS（ローカル）	ゼットスプール	両方 <sup>1</sup>	いいえ	はい	はい
ディレクトリ	監督	ファイル	いいえ	いいえ <sup>2</sup>	はい
ビーティーアールエフエス	ビットトランフ	ファイル	いいえ	はい	テクノロジー プレビュー

表7.1: (続き)

説明	プラグインタイプ	レベル	共有	スナップ写真	安定
ネットワークファイルシステム	エヌエフエス	ファイル	はい	いいえ <sup>2</sup>	はい
CIFS	cifs	ファイル	はい	いいえ <sup>2</sup>	はい
Proxmoxバックアップ	pbs	どちらも	はい	該当なし	はい
GlusterFS	グロスタフ	ファイル	はい	いいえ <sup>2</sup>	はい
セフエフエス	ケフフ	ファイル	はい	はい	はい
エルブイエム	エルブイエム	ブロック	いいえ <sup>3</sup>	いいえ	はい
LVMシン	ラヴムティン	ブロック	いいえ	はい	はい
iSCSI/カーネル	イッシ	ブロック	はい	いいえ	はい
iSCSI/libiscsi	イシディレック	ブロック	はい	いいえ	はい
Ceph/RBD	アールブイ	ブロック	はい	はい	はい
iSCSI上のZFS	zfs	ブロック	はい	はい	はい

<sup>(1)</sup>: VMのディスクイメージはZFSボリューム (zvol) データセットに格納され、ブロックデバイス機能を提供します。

<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。

<sup>3</sup>: iSCSIやFCベースのストレージの上でLVMを使うことは可能です。そうすれば、共有LVMストレージが得られます。

### 7.1.1 シン・プロビジョニング

多くのストレージと QEMU イメージフォーマット qcow2 はシンプロビジョニングをサポートしています。シンプロビジョニングを有効にすると、ゲストシステムが実際に使用するブロックのみがストレージに書き込まれます。

例えば、32GBのハードディスクでVMを作成し、ゲストシステムOSをインストールした後、VMのルートファイルシステムに3GBのデータが含まれているとします。この場合、ゲストVMが32GBのハードディスクを見ていたとしても、ストレージには3GBしか書き込まれません。このように、シンプロビジョニングでは、現在利用可能なストレージブロックよりも大きなディスクイメージを作成することができます。VMのために大きなディスクイメージを作成し、必要性が生じたときに、VMのファイルシステムのサイズを変更することなく、ストレージにディスクを追加することができます。

「スナップショット」機能を持つすべてのストレージタイプは、シンプロビジョニングもサポートしています。

#### 注意

ストレージがいっぱいになると、そのストレージ上のボリュームを使用しているすべてのゲストが IO エラーを受け取ります。これはファイルシステムの不整合を引き起こし、データを破損する可能性があります。そのため、ストレージリソースの過剰なプロビジョニングを避けるか、空き領域を注意深く観察して、このような状態を回避することをお勧めします。

## 7.2 ストレージ構成

Proxmox VEに関するすべてのストレージ設定は、/etc/pve/storage.cfgの1つのテキストファイルに保存されます。このファイルは /etc/pve/ 内にあるため、すべてのクラスタノードに自動的に配布されます。そのため、すべてのノードで同じストレージ構成が共有されます。

共有ストレージの構成は、すべてのノードから同じ「共有」ストレージにアクセスできるため、共有ストレージでは完全に理にかなっています。しかし、ローカルストレージタイプにも有効です。この場合、そのようなローカルストレージはすべてのノードで利用できますが、物理的に異なり、まったく異なるコンテンツを持つことができます。

## 7.2.1 ストレージプール

各ストレージプールは<type>を持ち、<STORAGE\_ID>で一意に識別されます。プール構成は次のようにになります：

```
<タイプ>: <ストレージID>
  <プロパティ> <値>
  <プロパティ> <値>
  <プロパティ>
  ...
  ...
```

<type>: <STORAGE\_ID> 行がプール定義を開始し、その後にプロパティのリストが続きます。ほとんどのプロパティは値を必要とします。いくつかのプロパティは、妥当なデフォルトを持ち、その場合、値を省略することができます。

具体的には、インストール後のデフォルトのストレージ構成を見てください。`local` という名前の特別なローカルストレージプールが 1 つ含まれており、これは `/var/lib/vz` ディレクトリを参照し、常に使用可能です。Proxmox VEのインストーラは、インストール時に選択したストレージタイプに応じて、追加のストレージエントリを作成します。

### デフォルトのストレージ設定 (`/etc/pve/storage.cfg`)

```
ディレクトリ: ローカル
  パス /var/lib/vz
  コンテンツ iso, vztmpl, バックアップ

# LVM ベースのインストールにおけるデフォルトのイメージストア lvmthin: local-lvm
  thinpoolデータ vgname
    pve
    コンテンツ rootdir, images

# ZFS ベースのインストールにおけるデフォルトのイメージストア zfspool: local-zfs
  プール rpool/data スペース
  コンテンツイメージ、ルートディレクトリ
```

#### 注意

 全く同じストレージを指す複数のストレージ構成があるのは問題です。このようなエイリアスされたストレージ構成では、2つの異なるボリュームID (`volid`) がまったく同じディスクイメージを指すことになります。Proxmox VEは、イメージのボリュームIDが一意であることを期待します。エイリアスされたストレージ構成に異なるコンテンツタイプを選択することは問題ありませんが、推奨されません。

## 7.2.2 一般的なストレージ特性

いくつかのストレージ特性は、異なるストレージタイプ間で共通です。

## ノード

このストレージが使用可能/アクセス可能なクラスタ・ノード名のリスト。このプロパティを使用して、ストレージへのアクセスを限られたノードに制限できます。

## 内容

例えば、仮想ディスクイメージ、CDROM ISOイメージ、コンテナテンプレート、コンテナルートディレクトリなどです。すべてのストレージタイプがすべてのコンテンツタイプをサポートするわけではありません。このプロパティを設定して、このストレージが何に使用されるかを選択できます。

## イメージ

QEMU/KVM VMイメージ。

## ルートディレクトリ

コンテナデータの保存を許可します。

## vztmpl

コンテナテンプレート。

## バックアップ

バックアップファイル (vzdump)。

## アイソ

ISOイメージ

## スニペット

スニペットファイル (ゲストフックスクリプトなど)

## シェアード

すべてのノード（またはnodesオプションにリストされているすべて）で同じ内容を持つ単一のストレージであることを示します。ローカルストレージの内容が自動的に他のノードからアクセスできるようになるわけではなく、すでに共有されているストレージをそのようにマークするだけです！

## 無効にする

このフラグを使うと、ストレージを完全に無効することができます。

## マックスファイル

非推奨。代わりに prune-backups を使用してください。VMごとのバックアップファイルの最大数。使用方法 0なら無制限。

## バックアップの削除

バックアップの保持オプション。詳細については、「[バックアップの保持](#)」を参照してください。

## フォーマット

デフォルトの画像フォーマット (raw | qcow2 | vmdk)

## ブレアロケーション

ファイルベースストレージ上の raw および qcow2 画像に対するブレアロケーションモード (off | metadata | fallow | full)。デフォルトはメタデータで、raw off と同じように扱われます。大きな qcow2 イメージと組み合わせてネットワークストレージを使用する場合、off を使用するとタイムアウトを回避できます。



## 警告

異なるProxmox VEクラスタで同じストレージプールを使用することはお勧めできません。ストレージ操作によってはストレージへの排他的なアクセスが必要なため、適切なロックが必要です。これはクラスタ内では実装されていますが、異なるクラスタ間では機能しません。

## 7.3 卷数

ストレージ・データのアドレス指定には特別な表記を使用します。ストレージ・プールからデータを割り当てるとき、そのようなボリューム識別子に変わります。ボリュームは、<STORAGE\_ID>の後に、コロンで区切られたストレージタイプに依存するボリューム名で識別されます。有効な<VOLUME\_ID>は次のようにになります:

```
local:230/example-image.raw local:iso/debian-501-
amd64-netinst.iso

local:vztmpl/debian-5.0-joomla_1.5.9-1_i386.tar.gz

iscsi-storage:0.0.2.scsi-14!
f504e46494c4500494b5042546d2d646744372d31616d61
```

<VOLUME\_ID>のファイルシステム・パスを取得するには、次のようにします:

```
pvesm/パス <VOLUME_ID
```

### 7.3.1 数量所有権

イメージ・タイプのボリュームには所有関係が存在します。このようなボリュームは、それぞれVMまたはコンテナによって所有されます。例えば、ボリューム local:230/example-image.raw は VM 230 が所有します。ほとんどのストレージバックエンドは、この所有者情報をボリューム名にエンコードします。

VMまたはコンテナを削除すると、そのVMまたはコンテナが所有する関連ボリュームもすべて削除されます。

## 7.4 コマンドラインインターフェイスの使用

ストレージ・プールやボリューム識別子の概念に慣れておくことをお勧めしますが、実際の運用では、コマンドラインでそのような低レベルの操作を行う必要はありません。通常、ボリュームの割り当てと削除はVMとコンテナの管理ツールによって行われます。

とはいって、pvesm（「Proxmox VE Storage Manager」）と呼ばれるコマンドラインツールがあり、一般的なストレージ管理タスクを実行できます。

## 7.4.1 例

ストレージプールの追加

```
pvesm add <TYPE> <STORAGE_ID> <OPTIONS>.  
pvesm add dir <STORAGE_ID> --path <PATH>  
pvesm add nfs <STORAGE_ID> --path <PATH> --server <SERVER> --export'  
<エクスポート>  
pvesm add lvm <STORAGE_ID> --vgname <VGNAME>  
pvesm add iscsi <STORAGE_ID> --portal <HOST[:PORT]> --target <TARGET>  
>
```

ストレージプールの無効化

```
pvesm set <STORAGE_ID> --disable 1
```

ストレージプールの有効化

```
pvesm set <STORAGE_ID> --disable 0
```

ストレージオプションの変更/設定

```
pvesm set <STORAGE_ID> <OPTIONS> pvesm set  
<STORAGE_ID> --shared 1 pvesm set local --format  
qcow2 pvesm set <STORAGE_ID> --content iso
```

ストレージプールを削除します。これはいかなるデータも削除しませんし、何かを切断したりアンマウントしたりもしません。ストレージ構成を削除するだけです。

pvesm remove <STORAGE\_ID> です。

ボリュームの割り当て

```
pvesm alloc <STORAGE_ID> <VMID> <name> <size> [--format <raw|qcow2>].
```

ローカルストレージに 4G ボリュームを割り当てます。<name> として空の文字列を渡した場合、名前は自動生成されます pvesm alloc local  
<VMID> '' 4G

全巻無料

```
pvesm free <VOLUME_ID>
```



警告

これは本当にすべてのボリューム破壊します。

保管状況一覧

pvesmステータス

## 収納内容一覧

pvesm list <STORAGE\_ID> [--vmid <VMID>] です。

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>
```

リストisoイメージ

```
pvesm list <STORAGE_ID> --content iso
```

リスト・コンテナ・テンプレート

```
pvesm list <STORAGE_ID> --content vztmpl
```

ボリュームのファイルシステムのパスを表示

```
pvesm/パス <VOLUME_ID>
```

ボリュームlocal:103/vm-103-disk-0.qcow2をファイルターゲットにエクスポートします。これは主にpvesmインポートで内部的に使用されます。ストリーム形式qcow2+sizeはqcow2形式とは異なります。そのため、エクスポートしたファイルを単純にVMにアタッチすることはできません。これは他の形式でも同様です。

```
pvesm export local:103/vm-103-disk-0.qcow2 qcow2+size target --with-'
```

スナップ写真1

## 7.5 ディレクトリバックエンド

ストレージプールのタイプ: dir

Proxmox VEは、ローカルディレクトリまたはローカルにマウントされた共有をストレージとして使用できます。ディレクトリはファイルレベルのストレージであるため、仮想ディスクイメージ、コンテナ、テンプレート、ISOイメージ、バックアップファイルなど、あらゆる種類のコンテンツを保存できます。

### 備考

標準的なlinuxの/etc/fstabを使って追加ストレージをマウントし、そのマウントポイントにディレクトリストレージを定義することができます。こうすることで、Linuxでサポートされているあらゆるファイルシステムを使用することができます。

このバックエンドは、基礎となるディレクトリがPOSIX互換であることを前提としていますが、それ以外のことは想定していません。これは、ストレージ・レベルでスナップショットを作成できないことを意味します。しかし、qcow2ファイル・フォーマットを使用しているVMイメージについては、このフォーマットが内部的にスナップショットをサポートしているため、回避策があります。

### チップ

ストレージ・タイプによってはO\_DIRECTをサポートしていないものがあり、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・ライトバックを使用してください。

異なるコンテンツタイプを異なるサブディレクトリに格納するために、定義済みの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使用されます。

表7.2: ディレクトリのレイアウト

コンテンツタイプ	サブディレクトリ
VMイメージ	images/<VMID>/
ISOイメージ	テンプレート/iso/
コンテナテンプレート	テンプレート/キャッシュ
バックアップファイル	ダンプ
スニペット	スニペット

## 7.5.1 構成

このバックエンドは、一般的なストレージプロパティをすべてサポートし、さらに2つのプロパティを追加します。パスプロパティを使用してディレクトリを指定します。これは絶対ファイル・システム・パスである必要があります。

オプションの `content-dirs` プロパティでは、デフォルトのレイアウトを変更できます。このプロパティは、コンマで区切られた識別子のリストで構成されます：

`vtype=パス`

`vtype` はストレージに許可されているコンテンツ・タイプの1つで、`path` はストレージのマウントポイントからの相対パスです。

**設定例 (/etc/pve/storage.cfg)**

```
dir:バックアップ
    パス /mnt/backup コンテンツ
    のバックアップ
    バックアップの削除 keep-last=7 最大保護バ
    ックアップ数 3
    content-dirs backup=custom/backup/dir
```

上記の構成では、`backup` というストレージ・プールを定義しています。このプールを使用して、VMごとに最大7つの通常バックアップ (`keep-last=7`) と3つの保護バックアップを保存できます。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 ファイルの命名規則

このバックエンドは、VMイメージに対して明確に定義された命名スキームを使用します：

`vm-<VMID>-<NAME>.<フォーマット`

**<VMID>**

これはオーナーVMを指定します。

**<名前>**

バックエンドは `disk-[N]` をデフォルトでは、[N] は名前を一意にするために整数で置き換えられます。

## <フォーマット

画像フォーマット (raw | qcow2 | vmdk) を指定します。

VM テンプレートを作成すると、すべての VM イメージの名前が変更され、読み取り専用になり、クローンのベースイメージとして使用できるようになります：

ベース<VMID>-<NAME>.<FORMAT>

### 備考

このようなベース画像は、クローン画像を生成するために使用されます。そのため、これらのファイルは読み取り専用で、決して変更されないことが重要です。バックエンドはアクセスモードを0444に変更し、ストレージが対応していればimmutableフラグ(chattr +i)を設定します。

## 7.5.3 ストレージ機能

上述したように、ほとんどのファイルシステムは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に、このバックエンドはqcow2内部のスナップショット機能を使うことができます。

クローンも同様です。バックエンドはqcow2のベースイメージ機能を使ってクローンを作成します。

表7.3：バックエンドディールのストレージ機能

コンテンツの種類	画像フォーマット	共有	スナップ写真	クローン
イメージ	生のqcow2	いいえ	qcow2	qcow2
ルートディレクトリ vztmpl iso	vmdkサブボリューム			
バックアップ				
スニペット				

## 7.5.4 例

```
以下のコマンドでローカルストレージに4GBのイメージを割り当ててください: # pvesm alloc local 100
vm-100-disk10.raw 4G
'/var/lib/vz/images/100/vm-100-disk10.raw', fmt=raw size'をフォーマットします。
=4294967296
local:100/vm-100-disk10.raw' の作成に成功しました。
```

### 備考

画像名は上記の命名規則に従わなければなりません。

実際のファイルシステムのパスは

```
# pvesm パス local:100/vm-100-disk10.raw  
/var/lib/vz/images/100/vm-100-disk10.raw
```

で画像を削除できます:

```
# pvesm free local:100/vm-100-disk10.raw
```

## 7.6 NFSバックエンド

ストレージプールのタイプ: nfs

NFSバックエンドはディレクトリバックエンドをベースにしているため、ほとんどのプロパティを共有しています。ディレクトリのレイアウトやファイルの命名規則も同じです。主な利点は、NFSサーバーのプロパティを直接設定できるので、バックエンドが自動的に共有をマウントできることです。

/etc/fstab を使用します。バックエンドはまた、サーバがオンラインかどうかをテストし、エクスポートされた共有をサーバに問い合わせるメソッドを提供します。

### 7.6.1 構成

バックエンドは、常に設定される共有フラグを除く、すべての一般的なストレージ・プロパティをサポートしています。さらに、以下のプロパティがNFSサーバの設定に使用されます:

#### サーバー

サーバーIPまたはDNS名。DNSルックアップの遅延を避けるために、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。

/etc/hostsファイル。

#### 輸出

NFS エクスポート・パス (pvesm nfsscan によってリストされます)。

NFSマウントオプションも設定できます:

#### パス

ローカルマウントポイント (デフォルトは /mnt/pve/<STORAGE\_ID>/)。

#### コンテンツディレクトリ

デフォルトのディレクトリレイアウトを上書きします。オプション。

#### オプション

NFS マウントオプション (man nfs を参照)。

**設定例 (/etc/pve/storage.cfg)**

```
nfs: iso-テンプレート
  パス /mnt/pve/iso-templates サーバー
  10.0.0.10
  export /space/iso-templates options
  vers=3,ソフトコンテンツiso,vztmpl
```

**チップ**

NFSリクエストがタイムアウトした後、NFSリクエストはデフォルトで無期限に再試行されます。これは、クライアント側で予期しないハングアップを引き起こす可能性があります。読み取り専用コンテンツの場合は、再試行回数を3回に制限するNFSソフト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の機能を使っ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

コンテンツの種類	画像フォーマット	共有	スナップ写真	クローン
イメージ	生のqcow2	はい	qcow2	qcow2
ルートディレクトリ	ブイエムディーケー			
vztmpl iso				
バックアップ				
スニペット				

## 7.6.3 例

でエクスポートされたNFS共有のリストを取得できます:

```
# pvesm scan nfs <server>
```

## 7.7 CIFSバックエンド

ストレージプールの種類: cifs

CIFSバックエンドはディレクトリバックエンドを拡張し、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を通して直接追加することができ、サーバのハートビートチェックやエクスポートされた共有の快適な選択など、すべてのバックエンドの利点を利用することができます。

## 7.7.1 構成

バックエンドは、常に設定される共有フラグを除く、すべての一般的なストレージプロパティをサポートしています。さらに、以下のCIFS特殊プロパティも使用できます：

### サーバー

サーバーIPまたはDNS名。必須

---

### チップ

DNSルックアップの遅延を避けるため、通常はDNS名ではなくIPアドレスを使用するのが望ましいです-よほど信頼できるDNSサーバーがあるか、ローカルの/etc/hostsサーバーをリストしている場合を除きます。

---

### シェア

使用する CIFS 共有 (pvesm scan cifs <address> または Web UI で利用可能なもの取得)。必要です。

### ユーザー名

CIFS ストレージのユーザー名。オプションで、デフォルトは「guest」です。

### パスワード

ユーザーのパスワード。オプション。このパスワードは、root によってのみ読み取り可能なファイル (/etc/pve/priv/storage/) に保存されます。

### ドメイン

このストレージのユーザードメイン（ワークグループ）を設定します。オプションです。

### スミバージョン

SMBプロトコルのバージョン。オプション、デフォルトは3。SMB1はセキュリティ上のサポートされていません。

### パス

ローカルのマウントポイント。オプションで、デフォルトは /mnt/pve/<STORAGE\_ID>/です。

### コンテンツディレクトリ

デフォルトのディレクトリレイアウトを上書きします。オプション。

### オプション

追加のCIFSマウントオプション (man mount.cifsを参照)。一部のオプションは自動的に設定されるため、ここで設定する必要はありません。

Proxmox VEは常にソフトオプションを設定します。設定に応じて、これらのオプションは自動的に設定されます：ユーザー名、資格情報、ゲスト、ドメイン、vers。

### サブディレクトリ

マウントする共有のサブディレクトリ。オプションで、デフォルトは共有のルートディレクトリです。

#### 設定例 (/etc/pve/storage.cfg)

```
cifs: バックアップ
  パス /mnt/pve/backup サーバー
    10.0.0.11 共有 VMData コンテンツ バックアップ
    オプション noserverino,echo_interval=30 ユーザー名 anna
    smbversion 3
    subdir /data
```

### 7.7.2 ストレージ機能

CIFSはストレージレベルでのスナップショットをサポートしていません。しかし、スナップショットやクローン機能を使用したい場合は、qcow2バックティングファイルを使用することができます。

表7.5: バックエンドcifsのストレージ機能

コンテンツの種類	画像フォーマット	共有	スナップ写真	クローン
イメージ	生のqcow2	はい	qcow2	qcow2
ルートディレクトリ	ブイエムディーケー			
vztmpl iso				
バックアップ				
スニペット				

### 7.7.3 例

でエクスポートされたCIFS共有のリストを取得できます:

```
# pvesm scan cifs <server> [--username <username>] [--password].
```

次に、これらの共有の1つをストレージとしてProxmox VEクラスタ全体に追加:

```
# pvesm add cifs <storagename> --server <server> --share <share> [--'
  ユーザー名 <ユーザー名>] [パスワード]
```

## 7.8 Proxmoxバックアップサーバー

ストレージ・プール・タイプ: pbs

このバックエンドを使用すると、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のAPI、CLI、またはWebインターフェイスから直接追加できます。

## 7.8.1 構成

バックエンドは、常に設定される共有フラグを除くすべての一般的なストレージプロパティをサポートします。さらに、Proxmox Backup Serverには以下の特別なプロパティがあります：

### サーバー

サーバーIPまたはDNS名。必須

### ポート

デフォルトのポート（8007）ではなく、このポートを使用します。オプション。

### ユーザー名

Proxmox Backup Serverストレージのユーザー名。必須です。

---

### チップ

ユーザー名にレルムを追加することを忘れないでください。例えば、root@pamやarchiver@pbsのように。

---

### パスワード

ユーザ・パスワード。この値は、/etc/pve/priv/storage/<STORAGE-ID>以下のファイルに保存されます。  
で、アクセスはroot制限されています。必要です。

### データストア

使用するProxmox Backup ServerデータストアのID。必須。

### フィンガープリント

Proxmox Backup Server API TLS 証明書のフィンガープリントです。Servers Dashboardまたはproxmox-backup-manager cert infoコマンドで取得できます。自己署名証明書またはホストがサーバーのCAを信頼していない他の証明書に必要です。

### 暗号キー

クライアントバックアップデータを暗号化するためのキー。現在、非パスワード保護（キー派生関数（kdf）なし）のみがサポートされています。  
etc/pve/priv/storage/<STORAG>下のファイルに保存され、アクセスはroot制限されます。マジックバリュー autogen を使って自動的に  
proxmox-backup-client key create --kdf none <path>を使って新しい鍵を作成します。オプション。

### マスター・暗号キー

バックアップ・タスクの一部としてバックアップ暗号化キーを暗号化するために使用される公開RSAキー。  
etc/pve/priv/storage/<STORAGE-ID>.master.pemに保存されます。バックアップ暗号化キーの暗号化コピーは各バックアップに追加され、リカバリ用にProxmox Backup Serverインスタンスに保存されます。オプションで、encryption-keyが必要です。

**設定例 (/etc/pve/storage.cfg)**

```

pbs: バックアップ
  データストアメイン
  サーバー enya.proxmox.com コンテンツ
  バックアップ
  指紋 09:54:ef:...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
  暗号化キー a9:ee:c8:02:13:...snip...:2d:53:2c:98 マスター・パブリックキー 1

```

## 7.8.2 ストレージ機能

Proxmox Backup Serverはブロックレベルまたはファイルレベルのバックアップのみをサポートします。Proxmox VEは仮想マシンにブロックレベル、コンテナにファイルレベルを使用します。

表 7.6: バックエンド pbs のストレージ機能

コンテンツの種類	画像フォーマット	共有	スナップ写真	クローン
バックアップ	該当なし	はい	該当なし	該当なし

## 7.8.3 暗号化



オプションで、GCM モードで AES-256 を使用したクライアント側の暗号化を設定することができます。暗号化は Web インターフェイス経由、または CLI で `encryption-key` オプション（上記参照）を使用して設定できます。鍵は、root ユーザのみがアクセスできる `/etc/pve/priv/storage/<STORAGE-ID>.enc` ファイルに保存されます。

**警告**

キがないと、バックアップにアクセスできなくなります。したがって、キーは順序よく、バックアップされるコンテンツとは別の場所に保管する必要があります。たとえば、システム全体をバックアップし、そのシステムのキーを使用することができます。何らかの理由でシステムにアクセスできなくなり、復元する必要が生じた場合、暗号化キーは壊れたシステムとともに失われてしまうため、復元は不可能です。

迅速な災害復旧のために、鍵は安全な場所に保管し簡単にアクセスできるようにしておくことをお勧めします。そのため、すぐに復旧できるパスワード・マネージャー内に保管するのが最適です。このバックアップとして、キーをUSBフラッシュドライブに保存し、安全な保管してください。こうすることで、どのシステムからも切り離された状態になりますが、それでも緊急時には簡単に復旧できます。最後に、最悪のシナリオに備えて、キーの紙コピーを安全な場所に保管しておくことも検討してください。paperkeyサブコマンドを使えば、QRコード化された鍵のバージョンを作成することができます。次のコマンドは、paperkeyコマンドの出力をテキストファイルに送信し、簡単に印刷できるようにします。

```
# proxmox-backup-client key paperkey /etc/pve/priv/storage/<STORAGE-ID>.enc'
--output-format text > qrkey.txt
```

暗号化バックアップを行うすべてのクライアントが単一のパブリック・マスター・キーを使用するように設定すると、それ以降のすべての暗号化バックアップには、使用されたAES暗号化キーのRSA暗号化コピーが含まれます。対応するプライベート・マスター・キーは、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キーの復元とバックアップの復号化を可能にします。

#### 警告

マスター・キー・ペアには、通常の暗号化キーと同じ保管ルールが適用されます。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、安全な物理的場所に保管するために、秘密鍵のマスター・キーのコピーを作成します。

暗号化はクライアント側で管理されるため、暗号化されていないバックアップと暗号化されたバックアップが異なるキーで暗号化されていても、サーバ上の同じデータストアを使用することができます。ただし、異なるキーで暗号化されたバックアップ間の重複排除はできないため、データストアは別々に作成した方がよい場合が多いです。

#### 備考

例えば、信頼できるネットワークでローカルにサーバを実行している場合など、暗号化によるメリットがない場合は暗号化を使用しないでください。暗号化されていないバックアップからの復旧の方が常に簡単です。

### 7.8.4 例CLIによるストレージの追加

を使用して、使用可能なProxmox Backup Serverデータストアのリストを取得できます:

```
# pvesm scan pbs <server> <username> [--password <string>] [--fingerprint<←'> string>] です。
```

次に、これらのデータストアの1つをストレージとしてProxmox VEクラスタ全体に追加:

```
# pvesm add pbs <id> --server <server> --datastore <datastore> --username<←'> <username> --fingerprint 00:B4:... --password
```

## 7.9 GlusterFSバックエンド

ストレージプールのタイプ: `glusterfs`

GlusterFSはスケーラブルなネットワークファイルシステムです。このシステムはモジュール設計を採用しており、コモディティハードウェア上で動作し、低コストで可用性の高いエンタープライズストレージを提供することができます。このシステムは数ペタバイトまで拡張可能で、数千のクライアントを扱うことができます。

### 備考

ノード/ブリックのクラッシュ後、GlusterFSはデータの一貫性を確認するために完全なrsyncを行います。これは大きなファイルでは非常に時間がため、このバックエンドは大きなVMイメージの保存には適していません。

### 7.9.1 構成

バックエンドはすべての一般的なストレージプロパティをサポートし、以下のGlusterFS固有の追加します：

#### サーバー

GlusterFS volfileサーバのIPまたはDNS名。

#### サーバー2

バックアップファイルサーバーのIPまたはDNS名。

#### ボリューム

GlusterFSボリューム。

#### 輸送

GlusterFSトランSPORT: `tcp`、`unix`または`rdma`

#### 設定例 (/etc/pve/storage.cfg)

```
glusterfs: クラスタ
    サーバー 10.2.3.4
    server2 10.2.3.5 volume
    glustervol content
    images,iso
```

### 7.9.2 ファイルの命名規則

ディレクトリのレイアウトとファイルの命名規則は `dir` バックエンドから継承しています。

### 7.9.3 ストレージ機能

ストレージはファイルレベルのインターフェイスを提供しますが、ネイティブのスナップショット/クローン実装はありません。

表7.7: バックエンドglusterfsのストレージ機能

コンテンツの種類	画像フォーマット	共有	スナップ写真	クローン
イメージ vztmpl iso	生のqcow2 ブイエムディーケー	はい	qcow2	qcow2
バックアップ				
スニペット				

## 7.9.4 例

で利用可能なGlusterFSボリュームのリストを取得できます:

```
# pvesm scan glusterfs <server>
```

## 7.10 ローカルZFSプール・バックエンド

ストレージプールのタイプ: `zfspool`

このバックエンドを使うと、ローカルのZFSプール（またはプール内のZFSファイルシステム）にアクセスできます。

### 7.10.1 構成

バックエンドは、一般的なストレージ・プロパティである`content`、`nodes`、`disable`と、以下のZFS固有のプロパティをサポートしています:

#### プール

ZFSプール/ファイルシステムを選択します。すべての割り当てはそのプール内で行われます。

#### ブロックサイズ

ZFSブロックサイズ・パラメータを設定します。

#### まばら

ZFSシン・プロビジョニングを使用します。スペースボリュームとは、予約がボリュームサイズに等しくないボリュームのことです。

#### マウントポイント

ZFSプール/ファイルシステムのマウントポイント。これを変更しても、`zfs`が見るデータセットのマウントポイントプロパティには影響しません。デフォルトは `/<pool>` です。

#### 設定例 (`/etc/pve/storage.cfg`)

```
zfspool: vmdatas
    プール tank/vmdatas コンテンツ
    rootdir,images sparse
```

## 7.10.2 ファイルの命名規則

バックエンドでは、VMイメージに以下のような命名スキームを使用します：

```
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>

これはオーナーVMを指定します。

### <名前>

これは空白のない任意の名前(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] は名前を一意にするために整数に置き換えられています。

## 7.10.3 ストレージ機能

ZFSは、スナップショットとクローニングに関して、おそらく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（フォーマットraw）とコンテナデータ（フォーマットsubvol）の両方にZFSデータセットを使用します。ZFSプロパティは親データセットから継承されるので、親データセットにデフォルトを設定するだけです。

表7.8：バックエンドzfsのストレージ機能

コンテンツの種類	画像フォーマット	共有	スナップ写真	クローン
イメージ	生サブヴォル	いいえ	はい	はい
ルートディレクトリ				

## 7.10.4 例

VMイメージを保存するために、追加のZFSファイルシステムを作成することをお勧めします：

```
# zfs create tank/vmdata
```

新しく割り当てられたファイルシステムで圧縮を有効にするには

```
# zfs set compression=on tank/vmdata
```

で利用可能なZFSファイルシステムのリストを取得できます：

```
# pvesm scan zfs
```

## 7.11 LVMバックエンド

ストレージ・プールのタイプ: lvm

LVMは、ハードディスクとパーティションの上にある軽いソフトウェアレイヤーです。利用可能なディスクスペースをより小さな論理ボリュームに分割するため使用できます。LVMはLinuxで広く使用されており、ハードドライブの管理を容易にします。

もう1つの使用例は、大きなiSCSI LUNの上にLVMを置くことです。そうすれば、iSCSI LUN上のスペースを簡単に管理することができます。iSCSI仕様ではスペース割り当て用の管理インターフェイスが定義されていないため、他の方法では不可能です。

### 7.11.1 構成

LVMバックエンドは、content、nodes、disableの一般的なストレージ・プロパティと、以下のLVM固有のプロパティをサポートしています：

#### ブグネーム

LVMボリューム・グループ名。これは既存のボリューム指す必要があります。

#### ベース

ベースボリューム。このボリュームは、ストレージにアクセスする前に自動的にアクティブ化されます。これは、LVMボリュームグループがリモートのiSCSIサーバに存在する場合に便利です。

#### セーフリムーブ

ウェブUIで「削除したボリュームのワイプ」を呼び出します。LVの削除時にデータをゼロにします。ボリュームを削除する際に、すべてのデータが消され、後から作成された他のLV（たまたま同じ物理エクステントが割り当てられている）からアクセスできないようにします。これはコストのかかる操作ですが、特定の環境ではセキュリティ対策として必要な場合があります。

#### saferemove\_throughput

ワイプスループット (cstream -tパラメータ値)。

設定例 (/etc/pve/storage.cfg)

```
lvm: マイスペース
      vgname myspace コンテンツ
      rootdir,images
```

### 7.11.2 ファイルの命名規則

バックエンドは、基本的にZFSプールのバックエンドと同じ命名規則を使用します。

vm-<VMID>-<NAME> // 通常のVMイメージ

### 7.11.3 ストレージ機能

LVMは典型的なブロックストレージですが、このバックエンドはスナップショットとクローンをサポートしていません。残念ながら、通常のLVMスナップショットは、スナップショット時間中にボリュームグループ全体のすべての書き込みを妨害するため、非常に非効率的です。

大きな利点は、iSCSI LUNなどの共有ストレージの上で使えることです。バックエンド自体が適切なクラスタ全体のロックを実装しています。

#### チップ

新しいLVM-thin/バックエンドはスナップショットとクローンを可能にしますが、共有ストレージをサポートしません。

表7.9: バックエンドlvmのストレージ機能

コンテンツの種類	画像フォーマット	共有	スナップ写真	クローン
イメージ ルートディレクトリ	生	可能	いいえ	いいえ

### 7.11.4 例

利用可能なLVMボリュームグループのリストは

```
# pvesm scan lvm
```

## 7.12 LVMシン・バックエンド

ストレージ・プールのタイプ: lvmthin

LVMは通常、ボリュームの作成時にブロックを割り当てます。LVMのシンプルルは、代わりに、書き込み時にブロックを割り当てます。ボリュームは物理的に利用可能なスペースよりもはるかに大きくなる可能性があるため、この動作はシン・プロビジョニングと呼ばれます。

通常のLVMコマンドラインツールを使用して、LVMシンブルを管理および作成できます(詳細については、man lvmthinを参照してください)。pveというLVMボリュームグループが既にあると仮定すると、以下のコマンドは、dataという新しいLVMシンブル(サイズ100G)を作成します:

```
lvcreate -L 100G -n data pve lvconvert --type thin-
pool pve/data
```

### 7.12.1 構成

LVMシンバックエンドは、一般的なストレージプロパティであるcontent、nodes、disableと、以下のLVM固有のプロパティをサポートします:

### ブグネーム

LVMボリューム・グループ名。これは既存のボリューム指す必要があります。

### シンプール

LVMシンプールの名前。

#### 設定例 (/etc/pve/storage.cfg)

```
lvmthin: ローカル lvm
    thinpool データ vgname
    pve
    コンテンツ rootdir, images
```

### 7.12.2 ファイルの命名規則

バックエンドは、基本的にZFSプールのバックエンドと同じ命名規則を使用します。

vm-<VMID>-<NAME> // 通常のVMイメージ

### 7.12.3 ストレージ機能

LVM thinはブロックストレージですが、スナップショットとクローンを効率的にサポートします。新しいボリュームは自動的にゼロで初期化されます。

LVMシン・プールは複数のノード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できます。

表 7.10: バックエンド lvmthin のストレージ機能

コンテンツの種類	画像フォーマット	共有	スナップ写真	クローン
イメージ ルートディレクトリ	生	いいえ	はい	はい

### 7.12.4 例

ボリュームグループ利用可能なLVMシンプールのリストを取得できます:

```
# pvesm scan lvmthin pve
```

## 7.13 Open-iSCSIイニシエータ

ストレージプールの種類: 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。ほとんどすべてのストレージベンダーが iSCSI をサポートしています。また、Debian ベースの [OpenMediaVault](#) など、オープンソースの iSCSI ターゲットソリューションもあります。

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 Debian の標準パッケージですが、リソースを節約するためにデフォルトではインストールされません。

```
# apt-get install open-iscsi
```

低レベルのiscsi管理タスクはiscsiadm行うことができます。

### 7.13.1 構成

バックエンドは、一般的なストレージ・プロパティであるcontent、nodes、disable、および以下のiSCSI固有のプロパティをサポートしています：

#### ポータル

iSCSIポータル（IPまたはDNS名とオプションのポート）。

#### ターゲット

iSCSI ターゲット。

#### 設定例 (/etc/pve/storage.cfg)

```
ISCSI: マイナス
  ポータル 10.10.10.1
  ターゲット iqn.2006-01.openfiler.com:tsn.dcb5aaaddd コンテンツ none
```

#### チップ

iSCSIの上で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を作成することができます。

### 7.13.2 ファイルの命名規則

iSCSIプロトコルは、データを割り当てたり削除したりするインターフェースを定義していません。代わりに、それはターゲット側で行われる必要があります、ベンダー固有です。ターゲットは単に番号付きLUNとしてエクスポートします。そのため、Proxmox VEのiSCSIボリューム名は、Linuxカーネルから見たLUNに関する情報をエンコードしているだけです。

### 7.13.3 ストレージ機能

iSCSIはブロック・レベル・タイプのストレージで、管理インターフェイスを提供しません。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の上にLVMをセットアップするのがベストです。その後、LVMプラグインを使ってiSCSI LUN上のストレージを管理することができます。

表7.11: バックエンドiscsiのストレージ機能

コンテンツの種類	画像フォーマット	共有	スナップ写真	クローン
画像なし	生	はい	いいえ	いいえ

### 7.13.4 例

リモートのiSCSIポータルをスキャンして可能なターゲットのリストを取得できます:

```
pvesm scan iscsi <ホスト[:ポート]>。
```

## 7.14 ユーザーモードiSCSIバックエンド

ストレージプールの種類: iscidirect

このバックエンドは基本的に Open-iSCSI バックエンドと同じ機能を提供しますが、実装にはユーザレベルのライブラリを使用します。このバックエンドを使用するには libiscsi-bin パッケージをインストールする必要があります。

カーネル・ドライバが関与していないので、これはパフォーマンスの最適化と見なすことができることに留意すべきです。しかしこれには、このようなiSCSI LUNの上でLVMを使用できないという欠点があります。そのため、ストレージサーバー側ですべてのスペースの割り当てを管理する必要があります。

### 7.14.1 構成

ユーザーモード iSCSI バックエンドは、Open-iSCSI バックエンドと同じ設定オプションを使用します。

設定例 (/etc/pve/storage.cfg)

```
iscsidirect: ファストストア・ポータ
ル10.10.10.1
ターゲット iqn.2006-01.openfiler.com:tsn.dcb5aaadd
```

### 7.14.2 ストレージ機能

#### 備考

このバックエンドはVMでのみ動作します。コンテナはこのドライバを使用できません。

表7.12: バックエンドiscsidirectのストレージ機能

コンテンツの種類	画像フォーマット	共有	スナップ写真	クローン
イメージ	生	はい	いいえ	いいえ

## 7.15 Ceph RADOSブロックデバイス (RBD)

ストレージ・プールのタイプ: rbd

Cephは、優れたパフォーマンス、信頼性、スケーラビリティを提供するように設計された分散オブジェクトストアおよびファイルシステムです。RADOSブロックデバイスは、豊富な機能を備えたブロックレベルのストレージを実装しており、次のような利点があります：

- シンプロビジョニング
- サイズ変更可能なボリューム
- 分散および冗長（複数のOSDにストライピング）
- 完全なスナップショットとクローン機能
- 自己治癒
- 単一障害点なし
- エクサバイトレベルまで拡張可能
- カーネルおよびユーザー空間の実装が可能

### 備考

小規模な導入では、Proxmox VEノードでCephサービスを直接実行することも可能です。最近のハードウェアはCPUパワーとRAMに余裕があるため、ストレージサービスとVMを同じノードで実行できます。

### 7.15.1 構成

このバックエンドは、一般的なストレージ・プロパティであるノード、ディセーブル、コンテンツ、および以下をサポートしています。

RBD特有の特性：

#### モノホスト

モニターデーモンのIPのリスト。オプション。CephがProxmox VEクラスタで実行されていない場合にのみ必要です。

#### プール

Cephプール名。

### ユーザー名

RBDユーザID。オプション。CephがProxmox VEクラスタで実行されていない場合にのみ必要です。ユーザIDのみを使用する注意してください。client. "タイプの接頭辞は省略する必要があります。

### クルプド

krbdカーネルモジュールを通してradosブロックデバイスへのアクセスを強制します。オプション。

### 備考

コンテナは、オプション値とは無関係にkrbdを使用します。

### 外部Cephクラスタの構成例(/etc/pve/storage.cfg)

#### RBD: セフ外部

```
monhost 10.1.1.20 10.1.1.21 10.1.1.22
pool ceph-external content
images username admin
```

### チップ

rbdユーティリティを使って低レベルの管理を行うことができます。

## 7.15.2 認証

### 備考

Proxmox VEクラスタにCephがローカルにインストールされている場合、ストレージの追加時に以下が自動的に実行されます。

デフォルトで有効になっているcephxを使用する場合、外部のCephクラスタからキーリングを提供する必要があります。

CLIでストレージを構成するには、まず、キーリングを含むファイルを利用できるようにする必要があります。1つの方法は、外部CephクラスタからProxmox VEノードの1つにファイルを直接コピーすることです。次の例では、実行するノードの/rootディレクトリにコピーします：

```
# scp <外部 cephserver>:/etc/ceph/ceph.client.admin.keyring /root/rbd.<'
キーholder
```

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します。たとえば、以下のようになります：

```
# pvesm add rbd <名前> --monhost "10.1.1.20 10.1.1.21 10.1.1.22" --content'
images --keyring /root/rbd.keyring
```

GUIで外部RBDストレージを設定する場合、キーリングをコピーして適切フィールドに貼り付けることができます。

キーholderは

```
# /etc/pve/priv/ceph/<STORAGE_ID>.keyring
```

#### チップ

外部クラスタに接続する場合は、必要な機能のみを持つキーリングを作成することをお勧めします。Cephのユーザ管理の詳細については、Cephのドキュメント<sup>(a)</sup>を参照してください。

<sup>a</sup> Cephユーザ管理

### 7.15.3 Cephクライアント構成(オプション)

外部Cephストレージに接続しても、外部クラスタ上のconfig DBでクライアント固有のオプションを設定できるとは限りません。Cephキーリングの横にceph.confを追加して、ストレージのCephクライアント設定を変更できます。

ceph.confはストレージと同じ名前にする必要があります。

```
# /etc/pve/priv/ceph/<STORAGE_ID>.conf
```

可能な、RBD設定リファレンス<sup>1</sup>を参照してください。

#### 備考

これらの設定を軽率に変更しないでください。Proxmox VEは<STORAGE\_ID>.confをストレージ設定にマージします。

### 7.15.4 ストレージ機能

rbdバックエンドはブロックレベルのストレージで、完全なスナップショットとクローン機能を実装しています。

表7.13: バックエンドrbdのストレージ機能

コンテンツの種類	画像フォーマット	共有	スナップ写真	クローン
イメージ	生	はい	はい	はい
ルートディレクトリ				

### 7.16 Cephファイルシステム (CephFS)

ストレージプールのタイプ: cephfs

CephFSはPOSIX準拠のファイルシステムを実装し、Cephストレージクラスタを使用してデータを格納します。CephFSはCephをベースに構築されているため、Cephの特性のほとんどを共有します。これには、冗長性、スケーラビリティ、自己回復、高可用性が含まれます。

<sup>1</sup> RBD構成参考資料 <https://docs.ceph.com/en/quincy/rbd/rbd-config-ref/>

## チップ

Proxmox VEはCephのセットアップを管理できるため、CephFSストレージの構成が容易になります。最新のハードウェアは多くの処理能力とRAMを備えているため、ストレージサービスとVMを同じノードで実行してもパフォーマンスに大きな影響はありません。

CephFSストレージプラグインを使用するには、Debian純正のCephクライアントを置き換えて、[Cephリポジトリ](#)を追加する必要があります。追加したら、`apt update`を実行し、続いて`apt dist-upgrade`を実行して、最新のパッケージを取得します。



### 警告

他のCephリポジトリが設定されていないことを確認してください。そうしないと、インストールに失敗するか、ノード上にパッケージのバージョンが混在して予期しない動作になります。

## 7.16.1 構成

このバックエンドは、一般的なストレージプロパティのノード、無効化、コンテンツに加え、以下のcephfs固有のプロパティもサポートします：

### エフエスネーム

Ceph FSの名前。

### モノホスト

モニターモンタードレスのリスト。オプション。CephがProxmox VEクラスタで実行されていない場合にのみ必要です。

### パス

ローカルのマウントポイント。オプションで、デフォルトは`/mnt/pve/<STORAGE_ID>/`です。

### ユーザー名

CephユーザID。オプション。CephがProxmox VEクラスタで実行されていない場合にのみ必要で、デフォルトは`admin`です。

### サブディレクトリ

マウントするCephFSサブディレクトリ。オプション、デフォルトは`/`です。

### ヒューズ

カーネルクライアントの代わりにFUSEを介してCephFSにアクセスします。オプション、デフォルトは`0`。

### 外部Cephクラスタの構成例(`/etc/pve/storage.cfg`)

```
cephfs: cephfs-external
    monhost 10.1.1.20 10.1.1.21 10.1.1.22
    パス /mnt/pve/cephfs-external コンテンツのバッ
    クアップ
    ユーザ名 admin fs-名前
    cephfs
```

**備考**

cephxが無効になっていない場合は、クライアントの秘密鍵ファイルを設定することを忘れないでください。

## 7.16.2 認証

**備考**

Proxmox VEクラスタにCephがローカルにインストールされている場合、ストレージの追加時に以下が自動的に実行されます。

デフォルトで有効になっているcephxを使用する場合、外部Cephクラスタからシークレットを提供する必要があります。

CLIでストレージを設定するには、まず、シークレットを含むファイルを利用できるようにする必要があります。1つの方法は、外部CephクラスタからProxmox VEノードの1つにファイルを直接コピーすることです。次の例では、実行するノードの/rootディレクトリにコピーします：

```
# scp <外部 cephserver>:/etc/ceph/cephfs.secret /root/cephfs.secret
```

次に、pvsm 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、コピーしたシークレットファイルへのパスを指定します。たとえば、以下のようになります：

```
# pvsm add cephfs <name> --monhost "10.1.1.20 10.1.1.21 10.1.1.22" --!
  content backup --keyring /root/cephfs.secret
```

GUIで外部RBDストレージを設定する場合、適切なフィールドにシークレットをコピー&ペーストできます。

rbdバックエンドには[client.userid]も含まれているのとは対照的に、シークレットはキーそのものだけです。  
セクションをご覧ください。

秘密は

```
# /etc/pve/priv/ceph/<STORAGE_ID>.secret
```

useridはクラスタにアクセスするように設定されたクライアントIDです。Cephのユーザ管理の詳細については、Cephのドキュメント[\(a\) を](#)参照してください。

```
# ceph auth get-key client.userid> cephfs.secret
```

## 7.16.3 ストレージ機能

cephfsバックエンドは、Cephクラスタ上のPOSIX準拠ファイルシステムです。

表7.14: バックエンドcephfsのストレージ機能

コンテンツの種類	画像フォーマット	共有	スナップ写真	クローン
vztmpl iso バックアップ スニペット	皆無	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、スナップショットは十分なテストが行われていないため、安定性は保証されていません。

## 7.17 BTRFSバックエンド

ストレージプールの種類: 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ので、一般的な概要についてはディレクトリバックエンドのセクションを参照してください。

主な違いは、このストレージタイプでは、スナップショットを取得し、スナップショットを保持したままオフラインストレージの移行をサポートするために、一つフォーマットディスクがサブボリュームに配置されることです。

### 備考

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。  
そうでない場合はチェックサムエラーが発生します。

### 7.17.1 構成

このバックエンドは、ディレクトリ・ストレージと同様に設定します。それ自身がマウントポイントでもないディレクトリをBTRFSストレージとして追加する場合は、is\_mountpointオプションで実際のマウントポイントを指定することをお勧めします。

たとえば、BTRFSファイルシステムが /mnt/data2 にマウントされ、その pve-storage/ サブディレクトリ（スナップショットである可能性があり、これを推奨）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: データ2
  パス /mnt/data2/pve-storage コンテンツ
    rootdir,images is_mountpoint /mnt/data2
```

### 7.17.2 スナップ写真

サブボリュームまたはRAWスナップショットを作成する場合、スナップショットは、同じパスに@とスナップショット名を付けた読み取り専用のサブボリュームとして作成されます。

## 7.18 ZFS over iSCSIバックエンド

ストレージプールのタイプ: zfs

このバックエンドは、ストレージとして ZFS プールを持ち、iSCSI ターゲットの実装を持つリモートマシンに ssh 経由でアクセスします。各ゲストディスクに対してZVOLを作成し、iSCSI LUNとしてエクスポートします。このLUNはProxmox VEによってゲストディスクに使用されます。

以下のiSCSIターゲット実装がサポートされています:

- LIO (リナックス)
- IET (Linux)
- ISTGT (FreeBSD)
- コムスター (ソラリス)

### 備考

このプラグインを使用して、通常のストレージアプライアンス/SAN上にZFSプールを作成することはできません。

### 7.18.1 構成

ZFS over iSCSI プラグインを使用するには、リモートマシン (ターゲット) が Proxmox VE ノードからの ssh 接続を受け入れるように設定する必要があります。

Proxmox VEはターゲットに接続してZVOLを作成し、iSCSI経由でエクスポートします。に保存されたsshキー(パスワード保護なし)を使用して認証します。

/etc/pve/priv/zfs/<target\_ip>\_id\_rsa

以下の手順でsshキーを作成し、IP 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.pubroot@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:

#### プール

iSCSIターゲット上のZFSプール/ファイルシステム。すべての割り当てはそのプール内で行われます。

#### ポータル

iSCSIポータル (IPまたはDNS名とオプションのポート)。

#### ターゲット

iSCSI ターゲット。

#### アイソサイプロバイダ

リモートマシンで使用されるiSCSIターゲット実装

## コムスター

コムスター・ビューのターゲット・グループ。

## comstar\_hg

comstarビューのホストグループ。

## lio\_tpg

Linux LIOターゲット用ターゲットポータルグループ

## ナウライトキャッシュ

ターゲットの書き込みキャッシュを無効にします。

## ブロックサイズ

ZFSブロックサイズ・パラメータを設定します。

## まばら

ZFSシン・プロビジョニングを使用します。スペースボリュームとは、予約がボリュームサイズに等しくないボリュームのことです。

## 設定例 (/etc/pve/storage.cfg)

```
zfs: リオ
    ブロックサイズ 4k
    iscsiprovider LIOプールタ
    ンク
    ポータル 192.0.2.111
    ターゲット iqn.2003-01.org.linux-iscsi.lio.x8664:snxxxxxxxxx コンテンツイメージ
    lio_tpg tpg1 スペース 1

zfs: ソラリスのブロックサイズ
    4k
    target iqn.2010-08.org.illumos:02:xxxxxxxx-xxxxxx-xxxxxxxxxxxx: ←'
        タンク1 プ
        ールタンク
        iscsiprovider comstar portal
        192.0.2.112 コンテンツイメージ

zfs: freebsd ブロックサイズ 4k
    対象 iqn.2007-09.jp.ne.peach.isttgt:tank1 プールタンク
    iscsiprovider isttgt portal
    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プールタンク
iscsiprovider ietポータル
192.0.2.114 コンテンツイメー
ジ
```

## 7.18.2 ストレージ機能

ZFS over iSCSIプラグインは、スナップショットが可能な共有ストレージを提供します。ZFS アプライアンスが展開における単一障害点にならないようにする必要があります。

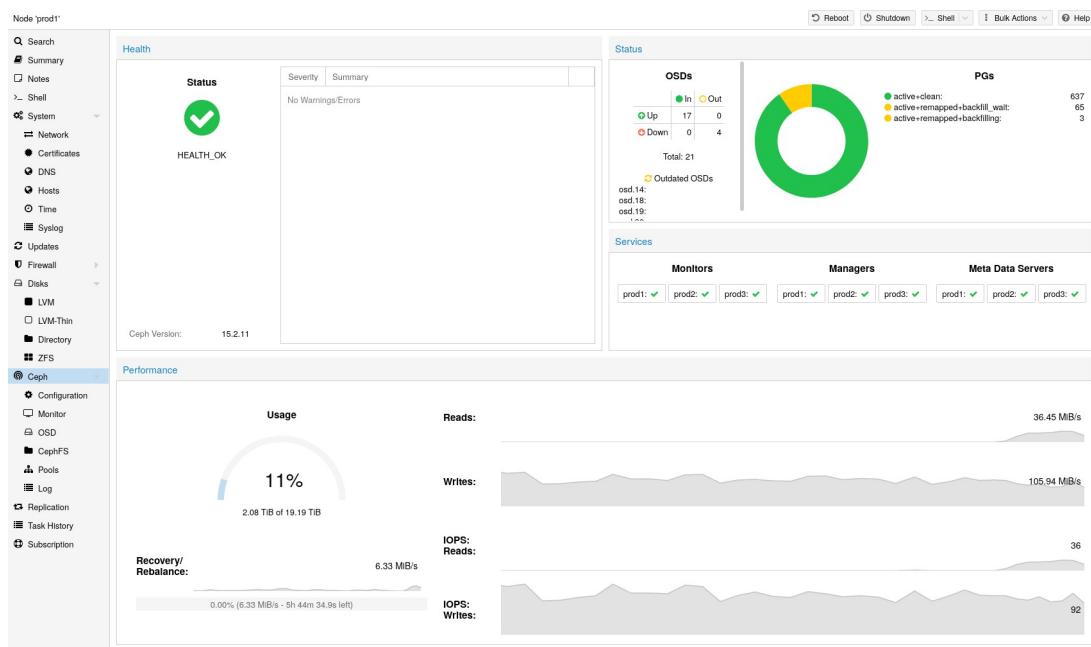
表7.15: バックエンドiscsiのストレージ機能

コンテンツの種類	画像フォーマット	共有	スナップ写真	クローン
イメージ	生	はい	はい	いいえ

## 第8章

# ハイパー・コンバージド Ceph クラスタの展開

## 8.1 はじめに



Proxmox VEは、コンピュートシステムとストレージシステムを統合します。つまり、クラスタ内の同じ物理ノードをコンピューティング（VMとコンテナの処理）とレプリケートされたストレージの両方に使用できます。つまり、クラスタ内の同じ物理ノードを、コンピューティング（VMとコンテナの処理）とレプリケートされたストレージの両方に使用できます。別々のストレージエイジネットワーク（SAN）やネットワークアタッチドストレージ（NAS）を介した接続はなくなります。オープンソースのSoftware-Defined StorageプラットフォームであるCephの統合により、Proxmox VEはハイパー・バーノード上でCephストレージを直接実行・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ように設計された分散オブジェクトストアおよびファイルシステムです。

PROXMOX VE上でのCEPHの利点は以下のとおりです：

- CLIとGUIによる簡単なセットアップと管理
- シン・プロビジョニング
- スナップショットのサポート

- セルフヒーリング
- エクサバイト・レベルまで拡張可能
- ブロック、ファイルシステム、オブジェクトストレージを提供
- パフォーマンスと冗長性の特性が異なるプールの設定
- データは複製されるため、フォールトトレラントです。
- コモディティハードウェアで動作
- ハードウェアRAIDコントローラは不要
- オープンソース

小規模から中規模の導入では、RADOS Block Devices (RBD)またはCephFSを使用するCephサーバをProxmox VEクラスタノードに直接インストールできます (Ceph RADOS Block Devices (RBD)を参照)。最近のハードウェアはCPUパワーとRAMが大きいため、ストレージサービスと仮想ゲストを同じノードで実行できます。

管理を簡素化するために、Proxmox VEは、組み込みのWebインターフェース、または

## 8.2 用語解説

CEPHは複数のデーモンで構成され、RBDストレージとして使用します：

- Cephモニタ (ceph-mon、またはMON)
- Ceph Manager (ceph-mgr、またはMGS)
- Cephメタデータサービス (ceph-mds、またはMDS)
- Ceph Object Storage Daemon (ceph-osd、またはOSD)

### チップ

Ceph<sup>a</sup>、そのアーキテクチャ<sup>b</sup>、語彙<sup>c</sup>についてよく知ることを強くお勧めします。

<sup>a</sup> Ceph intro <https://docs.ceph.com/en/quincy/start/>

<sup>b</sup> Cephアーキテクチャ <https://docs.ceph.com/en/quincy/architecture/>

<sup>c</sup> Ceph glossary <https://docs.ceph.com/en/quincy/glossary>

## 8.3 健全なCephクラスタの推奨事項

ハイパーコンバージドProxmox+ Ceph Clusterを構築するには、セットアップに少なくとも3台の(できれば同一の)サーバを使用する必要があります。

Cephのウェブサイトの推奨事項も確認してください。

## 備考

以下の推奨事項は、ハードウェアを選択する際のおおまかな指針とお考えください。したがって、あなたの特定のニーズに合わせることが不可欠です。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。

## 中央演算処理装置

Cephサービスは2つのカテゴリに分類できます：

- CPUを集中的に使用し、高いCPU基本周波数とマルチコアの恩恵を受けています。このカテゴリのメンバーは以下の通り：
  - 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)
- 複数のCPUコアを必要としない、適度なCPU使用率。これらは
  - モニター (MON) サービス
  - マネージャー (MGR) サービス

単純な経験則として、安定した耐久性のあるCephパフォーマンスに必要な最小限のリソースを提供するために、各Cephサービスに少なくとも1つのCPUコア（またはスレッド）を割り当てる必要があります。

たとえば、1つのノードでCephモニタ、Cephマネージャ、および6つのCeph OSDサービスを実行する予定であれば、基本的に安定したパフォーマンスを目標とする場合、8つのCPUコアをCeph専用に確保する必要があります。

OSDのCPU使用率は、主にディスクの性能に依存することに注意してください。ディスクの IOPS (IO Operations per Second) が高ければ高いほど、OSD サービスで使用できる CPU は増えます。100,000を超える高いIOPS負荷をサブミリ秒のレイテンシで永続的に維持できるNVMeのような最新のエンタープライズSSDディスクの場合、各OSDは複数のCPUスレッドを使用できます。

## メモリー

特にハイパーコンバージドセットアップでは、メモリ消費を注意深く計画し、監視する必要があります。仮想マシンとコンテナのメモリ使用量の予測に加えて、Cephが優れた安定したパフォーマンスを提供するために十分なメモリを利用できることも考慮する必要があります。

経験則として、およそ1TiBのデータに対して、1GiBのメモリがOSDによって使用されます。通常の使用量は少ないかもしれません、リカバリー、再バランス、バックファイルなどの重要な操作時には最も多く使用されます。つまり、通常の運用で使用可能なメモリを最大にすることは避け、むしろ障害に対処するための余裕を残しておく必要があります。

OSDサービス自体にはさらにメモリが必要です。デーモンのCeph BlueStore/バックエンドには、デフォルトで3~5GiBのメモリが必要です(調整可能)。

## ネットワーク

Cephトラフィック専用に、少なくとも10 Gbps以上のネットワーク帯域幅を使用することをお勧めします。10Gbps以上のスイッチがない場合、3~5ノードのクラスタではメッシュ型ネットワーク設定<sup>1</sup>も選択肢になります。

### 重要

特にリカバリ時のトラフィック量は、同じネットワーク上の他のサービスに干渉し、特にレイテンシに敏感なProxmox VEのcorosyncクラスタスタッフが影響を受け、クラスタクオーラムが失われる可能性があります。Cephトラフィックを専用の物理的に分離されたネットワークに移動すると、corosyncだけでなく、仮想ゲストによって提供されるネットワーキングサービスでも、このような干渉を回避できます。

必要な帯域幅を見積るには、ディスクの性能を考慮する必要があります。1台のHDDでは1Gbの飽和させることはできないかもしれません、ノードあたり複数のHDD OSDがあれば、すでに10Gbpsも飽和させることができます。最新のNVMeアタッチドSSDを使用すれば、1台で10 Gbps以上の帯域幅を飽和させることができます。このような高性能セットアップには、少なくとも25Gpbsを推奨します。また、基盤となるディスクの潜在性能をフルに活用するには、40Gbpsまたは100Gbps以上が必要な場合もあります。

不安な場合は、高性能なセットアップのために3つの（物理的な）独立したネットワークを使用することをお勧めします：

- Ceph(内部)クラスタトラフィック用の超高帯域幅(25Gbps以上)ネットワーク1つ。
- cephサーバとcephクライアントのストレージトラフィック間のCeph（パブリック）トラフィック用の高帯域幅（10Gbps以上）ネットワーク1つ。ニーズによっては、仮想ゲストトラフィックとVMライブマイグレーショントラフィックのホストにも使用できます。
- 遅延の影響を受けやすいcorosyncクラスタ通信のために、1つの中帯域幅（1 Gbps）専用。

## ディスク

Cephクラスタのサイズを計画する際は、リカバリ時間を考慮することが重要です。特に小規模なクラスタでは、リカバリに時間がかかる場合があります。リカバリ時間を短縮し、リカバリ中に後続の障害イベントが発生する可能性を最小限に抑えるため、小規模なセットアップではHDDの代わりにSSDを使用することをお勧めします。

一般的に、SSDは回転ディスクよりも多くのIOPSを提供します。この点を考慮すると、コストが高くなることに加えて、[クラスベースのプール分離](#)を実装することが理にかなっている場合があります。OSDを高速化するもう1つの方法はより高速なディスクをジャーナルまたはDB/Write-Ahead-Logデバイスとして使用することです。より高速なディスクを複数のOSDに使用する場合、OSDとWAL/DB(またはジャーナル)ディスクの適切なバランスを選択する必要があります。

ディスクの種類は別として、Cephはノードごとに均等なサイズで均等な量のディスクを使用するのが最適です。たとえば、1 TBディスク1台と250 GBディスク3台の混在セットアップよりも、各ノード内に500 GBディスク4台の方が優れています。

OSD数とシングルOSD容量のバランスも取る必要があります。容量を増やすと、ストレージ密度を高めることができます。OSDが1つ故障すると、Cephは一度に多くのデータを回復しなければならなくなります。

<sup>1</sup> Ceph用フルメッシュネットワーク [https://pve.proxmox.com/wiki/Full\\_Mesh\\_Network\\_for\\_Ceph\\_Server](https://pve.proxmox.com/wiki/Full_Mesh_Network_for_Ceph_Server)

## RAIDを避ける

Cephはデータオブジェクトの冗長性とディスクへの複数の並列書き込み(OSD)を独自に処理するため、通常、RAIDコントローラを使用してもパフォーマンスや可用性は向上しません。それどころか、Cephはディスク全体を単独で処理するように設計されています。RAIDコントローラはCephのワークロード用に設計されておらず、書き込みやキャッシングのアルゴリズムがCephのものと干渉する可能性があるため、状況が複雑になり、場合によってはパフォーマンスが低下することもあります。



### 警告

RAIDコントローラは避けてください。代わりにホストバスアダプタ（HBA）を使用してください。

## 8.4 Cephの初期インストールと構成

### 8.4.1 ウェブベースのウィザードの使用

The screenshot shows the Proxmox VE web interface for node 'nina'. The left sidebar has a 'Ceph' section selected, which includes options like Configuration, Monitor, OSD, CephFS, Pools, Log, Replication, Task History, and Subscription. The main content area is titled 'Health' and shows 'No Warnings/Errors'. Below it is a 'Status' section for Ceph, which displays the following information:

OSDs	In	Out
Up	0	0
Down	0	0

Total: 0

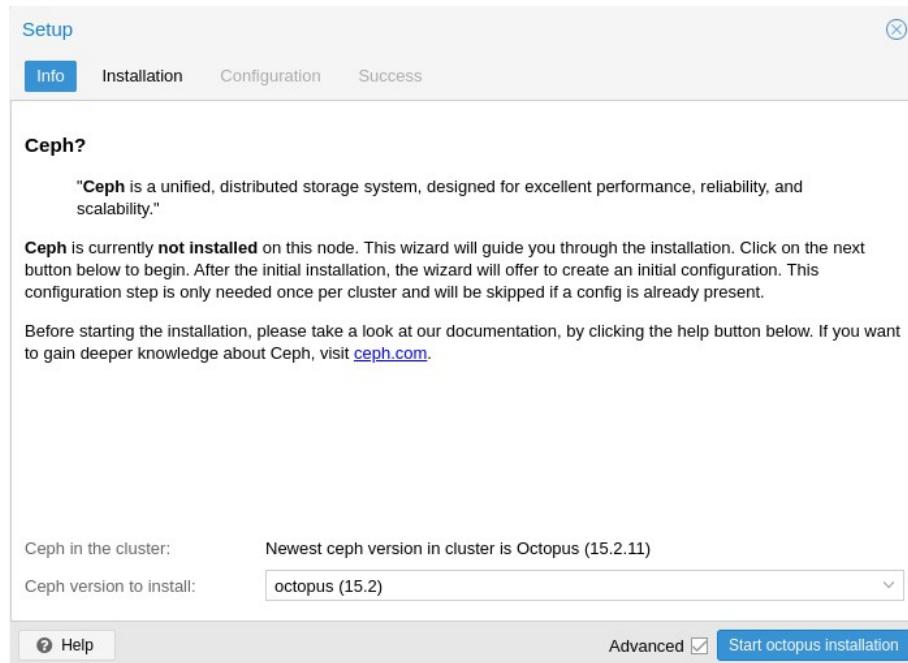
A modal dialog box is overlaid on the status section, containing the message: "Ceph is not installed on this node. Would you like to install it now?". A blue "Install Ceph" button is visible at the bottom of this dialog.

Proxmox VEには、Ceph用の使いやすいインストールウィザードが用意されています。クラスタノードの1つをクリックし、メニューツリーのCephセクションに移動します。Cephがまだインストールされていない場合は、インストールを促すプロンプトが表示されます。

ウィザードは複数のセクションに分かれており、Cephを使用するには、各セクションを正常に終了する必要があります。

最初に、インストールするCephのバージョンを選択する必要があります。他のノードのものを選ぶか、Cephをインストールする最初のノードであれば最新のものを選びます。

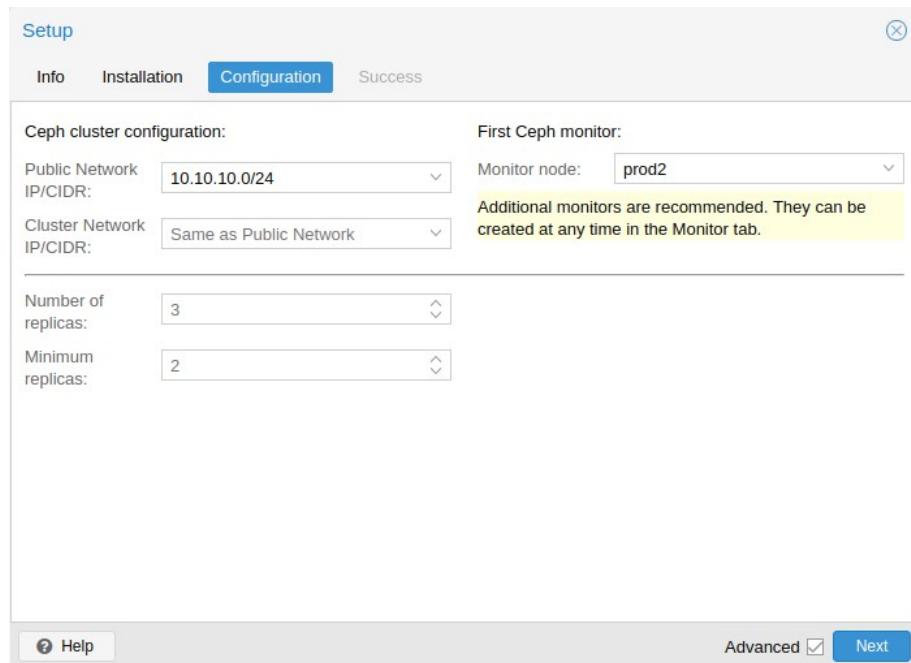
インストールを開始すると、ウィザードがProxmox VEのCephリポジトリから必要なパッケージをすべてダウンロードしてインストールします。



インストール手順が終了したら、構成を作成する必要があります。この構成はProxmox VEのクラスタ構成ファイルシステム(pmxcfs)を通じて残りのすべてのクラスタメンバに自動的に配布されるため、この手順はクラスタごとに1回だけ必要です。

コンフィギュレーション・ステップには以下の設定が含まれます：

- 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(Ceph RBDバックアップディスクやCephFSマウントを使用する仮想マシンなど)、およびさまざまなCephサービス間の通信に使用されます。この設定は必須です。  
CephトライフィックをProxmox VEクラスタ通信(corosync)、および可能であれば仮想ゲストの前面(パブリック)ネットワークから分離することを強く推奨します。そうしないと、Cephの高帯域幅IOトライフィックが他の低レイテンシ依存サービスに干渉する可能性があります。
- クラスタ・ネットワーク：** OSDレプリケーションとハートビートトライフィックも分離するように指定します。この設定はオプションです。  
物理的に分離されたネットワークを使用すると、Cephパブリックと仮想ゲストのネットワークが緩和され、Cephのパフォーマンスも大幅に向上するため、推奨されます。  
Cephクラスタネットワークを構成し、後で物理的に分離された別のネットワークに移動できます。



さらに2つのオプションがありますが、これは上級者向けなので、自分が何をしているかを知っている場合のみ変更してください。

- **レプリカの数:** オブジェクトが複製される頻度を定義します。
- **最小レプリカ数:** I/Oを完了としてマークするために必要なレプリカの最小数を定義します。

さらに、最初のモニターノードを選択する必要があります。このステップは必須です。

これで完了です。最後のステップとして成功ページが表示され、続行方法が表示されます。これで、システムはCephの使用を開始する準備ができました。開始するには、追加の[モニタ](#)、[OSD](#)、および少なくとも1つの[プール](#)を作成する必要があります。

この章の残りの部分では、Proxmox VEベースのCephセットアップを最大限に活用する方法を説明します。これには、前述のヒントのほか、新しいCephクラスターに追加すると便利な[CephFS](#)などが含まれます。

## 8.4.2 CephパッケージのCLIインストール

Webインターフェースで利用可能な推奨のProxmox VE Cephインストールウィザードの代わりに、各ノードで次のCLIコマンドを使用できます：

`pveceph`インストール

これにより、`/etc/apt/sources.list.d/ceph.list` に `apt` パッケージリポジトリが設定され、必要なソフトウェアがインストールされます。

## 8.4.3 CLIによるCephの初期構成

Proxmox VE Cephインストールウィザードを使用するか(推奨)、1つのノードで以下のコマンドを実行します：

```
pveceph init --network 10.10.10.0/24
```

これにより、`/etc/pve/ceph.conf`にCeph専用のネットワークを持つ初期構成が作成されます。このファイルは、[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。このコマンドはまた、`/etc/ceph/ceph.conf`にシンボリックリンクを作成し、このファイルを指します。このため、設定ファイルを指定しなくても、Cephコマンドを単純に実行できます。

## 8.5 Cephモニター

The screenshot shows the Proxmox VE management interface for node 'prod2'. On the left, the navigation tree is expanded to show the Ceph section, with 'Monitor' selected. The main area contains two tables: 'Monitor' and 'Manager'.

**Monitor Table:**

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

**Manager Table:**

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維持します。高可用性を実現するには、少なくとも3つのモニタが必要です。インストールウィザードを使用した場合は、1つのモニタがすでにインストールされています。クラスタが小規模から中規模であれば、3つ以上のモニタは必要ありません。これ以上必要になるのは、本当に大規模なクラスタだけです。

### 8.5.1 モニターの作成

モニタを配置する各ノードで(3つのモニタを推奨)、GUIのCeph→ Monitorタブを使用するか、runしてモニタを作成します：

プロセス・モン・クリエイト

<sup>2</sup> Ceph Monitor <https://docs.ceph.com/en/quincy/rados/configuration/mon-config-ref/>

## 8.5.2 モニターを破壊

GUIでCephモニタを削除するには、まずツリービューでノードを選択し、**Ceph→ Monitor** パネルに表示されます。MON を選択し、**Destroy** ボタンをクリックします。

CLIでCephモニタを削除するには、まずMONが実行されているノードに接続します。次に、次のコマンドを実行します：

```
ヴェセフ・モン討伐
```

### 備考

定足数には少なくとも3名のモニターが必要です。

## 8.6 Cephマネージャ

Managerデーモンはモニタと並行して実行されます。クラスタ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 クリエイトマネージャー

複数のManagerをインストールできますが、アクティブなManagerは1つだけです。

```
を作成します。
```

### 備考

Ceph Managerはモニタノードにインストールすることをお勧めします。高可用性を実現するには、複数のマネージャをインストールします。

### 8.6.2 デストロイ・マネージャ

GUIでCeph Managerを削除するには、まずツリービューでノードを選択し、**Ceph→ Monitor**に移動します。

パネルに表示されます。Managerを選択し、**Destroy**クリックします。

CLIでCephモニタを削除するには、まずManagerが実行されているノードに接続します。次に、次のコマンドを実行します：

```
コンベック破壊
```

### 備考

マネージャはハード依存ではありませんが、PG自動スケーリング、デバイスのヘルスマニタリング、テレメトリなどの重要な機能を処理するため、Cephクラスタにとって重要です。

<sup>3</sup> Ceph Manager <https://docs.ceph.com/en/quincy/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 Object Storage Daemonは、ネットワーク経由でCephのオブジェクトを格納します。物理ディスクごとに1つのOSDを使用することをお勧めします。

### 8.7.1 OSDの作成

OSDはProxmox VEのWebインターフェイスまたはpvecephを使用してCLIで作成できます。例えば

```
pveceph osd create /dev/sd[X].
```

#### チップ

少なくとも3台のノードと、ノード間で均等に分散された少なくとも12台のOSDを備えたCephクラスタを推奨します

ディスクが以前（例えばZFSやOSDとして）使用されていた場合は、まずその使用痕跡をすべて消す必要があります。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の残りを削除するには、以下のコマンドを使用します：

**警告**

上記のコマンドはディスク上のすべてのデータを破壊します！

**Ceph Bluestore**

Ceph Kraken、Bluestore<sup>4</sup>という新しいCeph OSDストレージタイプが導入されました。これは、Ceph Luminous以降のOSD作成時のデフォルトです。

```
pveceph osd create /dev/sd[X].
```

**block.dbとblock.wal**

OSDに別のDB/WALデバイスを使いたい場合は、`-db_dev`と  
`-wal_dev`オプション。別途指定しない場合、WALはDBと一緒に置かれます。

```
pveceph osd create /dev/sd[X] -db_dev /dev/sd[Y] -wal_dev /dev/sd[Z].
```

それぞれ`-db_size`と`-wal_size`パラメータで直接サイズを指定できます。それらが与えられない場合は、以下の値（順番に）が使われます：

- Ceph設定からのbluestore\_block\_{db,wal}\_size。 . .
  - データベース、セクション OSD
  - データベース、グローバルセクション
  - ファイル、セクション *osd*
  - ファイル、セクショングローバル
- OSDサイズの10%(DB)/1%(WAL)

**備考**

DBはBlueStoreの内部メタデータを保存し、WALはBlueStoreの内部ジャーナルまたはライトアヘッド・ログです。パフォーマンスを向上させるために、高速なSSDまたはNVRAMを使用することをお勧めします。

**Cephファイルストア**

Ceph Luminous以前は、Ceph OSDのデフォルトのストレージタイプとしてFilestoreが使用されていました。Ceph Nautilus以降、Proxmox VEでは`pveceph`を使用したこのようなOSDの作成がサポートされなくなりました。ファイルストアOSDを作成する場合は、`ceph-volume`を直接使用してください。

```
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に問題が発生した場合は、まず[トラブルシューティング](#)を行い、必要かどうかを判断してください。

OSDを破棄するには、<ノード> Ceph→ → OSDパネルに移動するか、OSDが配置されているノードで前述の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スペースを持っていることを確認します。Ceph→ OSDパネルで、破棄予定のOSDがまだ稼働中である場合(AVAILでゼロ以外の値)、すべてのOSDのUsed (%) 値がデフォルトの85%のニアフルレシオを大幅に下回っていることを確認します。こうすることで、OSDが満杯になり、CephプールのI/Oがブロックされる可能性のある、今後のリバランシングによるリスクを軽減できます。

CLIで同じ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ツリー
```

2. 破棄するOSDがまだ出でていない場合は、OSDを選択し、「Out」をクリックしてください。これにより、データ配布から除外され、リバランスが開始されます。

次のコマンドも同じです：

```
ceph osd out <id>
```

3. 可能であれば、Cephのリバランスが終了するまで待つと、常に十分なレプリカを確保できます。OSDは空になり、空になると0 PGが表示されます。
4. **停止**をクリックしてください。停止がまだ安全でない場合は、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。しばらくしてからもう一度試してください。

以下のコマンドを使用して、OSDを停止して安全かどうかを確認できます：

```
ceph osd ok-to-stop <id> pveceph stop --  
service osd.<id>
```

5. 最後に

CephからOSDを削除し、すべてのディスクデータを削除するには、まず[More→Destroy](#)をクリックします。クリーンアップオプションを有効にして、パーティションテーブルとその他の構造をクリーンアップします。これにより、Proxmox VEでディスクをすぐに再利用できます。次に、**Remove**をクリックします。

OSDを破壊するCLIは次のとおりです：

```
pveceph osd destroy <id> [--cleanup] です。
```

## 8.8 Cephプール

Name	Size/min	# of Placement Gr...	Optimal # of PGs	Autoscale Mode	CRUSH Rule (ID)	Used (%)	Total Size
cp	3/2	256	256	on	replicated_rule (0)	937.64 GiB (5.85%)	
cephfs_data	3/2	32	32	on	replicated_rule (0)	846.00 GiB (5.31%)	
cephfs_metadata	3/2	32	16	warn	replicated_rule (0)	101.09 MiB (0.00%)	
cp-krbd	3/2	256	256	on	replicated_rule (0)	320.31 GiB (2.08%)	
device_health_metrics	3/2	1	1	on	replicated_rule (0)	147.90 MiB (0.00%)	
ssd	3/2	128	128	on	replicated_ssd (1)	0 B (0.00%)	2.05 TiB

プールは、オブジェクトを格納するための論理グループです。これは、配置グループ（PG、pg\_num）。

### 8.8.1 プールの作成と編集

プールの作成と編集は、Proxmox VEホストのコマンドラインまたはWebインターフェイスから、以下の手順で実行できます。

Ceph→プール。

オプションが指定されていない場合、デフォルトで128個のPG、3個のレプリカ、2個のレプリカの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、レプリカが1つしかないときにオブジェクトに対するI/Oが許可されるため、データ損失、不完全なPG、または未発見のオブジェクトが発生する可能性があります。

PG-Autcalerを有効にするか、セットアップに基づいてPG数を計算することをお勧めします。計算式とPG計算機<sup>5</sup>はオンラインで入手できます。Ceph Nautilus以降では、セットアップ後にPG数<sup>6</sup>を変更できます。

<sup>5</sup> PG電卓 <https://web.archive.org/web/20210301111112/http://ceph.com/pgcalc/>

<sup>6</sup> ブレイスマント・グループ <https://docs.ceph.com/en/quincy/rados/operations/placement-groups/>

PG オートスケーラ<sup>(7)</sup>は、バックグラウンドでプールの PG カウントを自動的にスケーリングできます。ターゲット・サイズまたはターゲット比率のアドバンス・パラメータの設定は、PG オートスケーラがより良い決定を下すのに役立ちます。

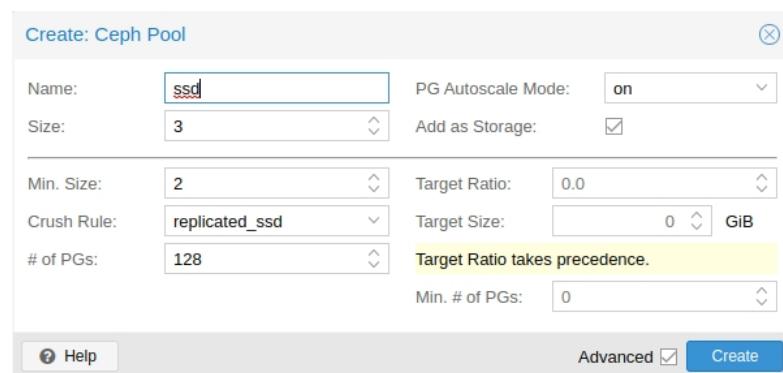
#### CLIでプールを作成する例

```
pveceph pool create <プール名> --add_storages
```

#### チップ

プールにストレージを自動的に定義したい場合は、Webインターフェースで「Add as Storage」チェックボックスをチェックしたままにするか、プール作成時にコマンドラインオプション--add\_storagesを使用します。

#### プールオプション



以下のオプションは、プールの作成時に使用でき、プールの編集時にも一部使用できます。

#### 名称

プールの名前。これは一意でなければならず、後から変更することはできません。

#### サイズ

オブジェクトあたりのレプリカ数。Cephは常にオブジェクトのコピーをこの数にしようとします。デフォルト: 3.

#### PGオートスケールモード

プールの自動 PG スケーリングモード<sup>(7)</sup>。warn に設定すると、プールの PG カウントが最適でない場合に警告メッセージを生成します。デフォルト: warn。

#### ストレージとして追加

新しいプールを使用してVMまたはコンテナ・ストレージを構成します。デフォルト: true (作成時のみ表示されます)。

#### 高度なオプション

#### 最小サイズ

オブジェクトあたりの最小レプリカ数。PGのレプリカ数がこの数未満の場合、Cephはプール上のI/Oを拒否します。デフォルト: 2。

<sup>7</sup> 自動スケーリング <https://docs.ceph.com/en/quincy/rados/operations/placement-groups/#automated-scaling>

## クラッシュルール

クラスタ内のオブジェクト配置のマッピングに使用するルール。これらのルールは、クラスタ内でのデータの配置方法を定義します。デバイスベースのルールについては、[Ceph CRUSH & デバイスクラス](#)を参照してください。

## # PG数

プールが最初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

プールで予想されるデータの比率。PG オートスケーラは、他の比率セットに対する比率を使用します。両方が設定されている場合は、ターゲットサイズよりも優先されます。

## 対象サイズ

プールに予想されるデータ量。PG オートスケーラは、このサイズを使用して最適な PG 数を推定します。

## 最低# PG数

配置最小数。この設定は、そのプールの PG カウントの下限を微調整するために使用されます。PG オートスケーラは、この閾値未満の PG をマージしません。

Ceph プールの処理に関する詳細は、[Ceph プールの操作<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「順方向エラー訂正」符号の一形態で、ある程度のデータ損失から回復することができます。消去符号化されたプールは、複製されたプールと比較してより多くの使用可能領域を提供できますが、その分パフォーマンスが低下します。

一方、消去コード化プールでは、データは  $k$  個のデータチャunkに分割され、さらに  $m$  個のコーディング（チェック）チャunkが追加されます。これらのコーディング・チャunkは、データ・チャunkが欠落した場合にデータを再作成するために使用されます。

符号化チャunkの数  $m$  は、データを失うことなく OSD を失うことができる数を定義します。保存されるオブジェクトの総量は  $k + m$  です。

### EC プールの作成

消去符号化 (EC) プールは、`pveceph` CLIツールを使用して作成できます。EC プールを計画する場合は、レプリケートされたプールとは動作が異なるという事実を考慮する必要があります。

EC プールのデフォルトの `min_size` は  $m$  パラメータに依存します。 $m > 1$  の場合、`min_size` は  $k + 1$  になります。Ceph のドキュメントでは、保守的な `min_size` として  $k + 2$  を推奨しています<sup>9</sup>。

利用可能な OSD が `min_size` 未満の場合、プールへの IO は、再び十分な OSD が利用可能になるまでブロックされます。

<sup>8</sup> Ceph プール操作 <https://docs.ceph.com/en/quincy/rados/operations/pools/>

<sup>9</sup> Ceph Erasure Coded Pool Recovery <https://docs.ceph.com/en/quincy/rados/operations/erasure-code/#erasure-coded-pool-recovery>

---

#### 備考

消去コード化されたプールを計画するときは、`min_size` に注意してください。そうしないと、IOがブロックされてしまいます。

---

たとえば、 $k = 2, m = 1$  の EC、`size = 3, min_size = 2` となり、1つの OSD が故障しても運用を継続します。プールが  $k = 2, m = 2$  で構成されている場合、`サイズ = 4, min_size = 3` となり、1つの OSD が失われた場合でも運用を継続します。

新しい EC プールを作成するには、以下のコマンドを実行します：

```
pveceph pool create <pool-name> --erasure-coding k=2,m=1
```

オプションのパラメータは `failure-domain` と `device-class` 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しいプールを作成する必要があります。

これにより、新しい EC プールと、RBD オマップとその他のメタデータを格納するために必要な複製プールが作成されます。最終的に、<プール名>-データと<プール名>-メタデータ・プールが作成されます。デフォルトの動作では、一致するストレージ構成も作成されます。この動作が不要な場合、`--add_storages 0` パラメータを指定することで無効にすることができます。ストレージ構成を手動で構成する場合、`data-pool` パラメータを設定する必要があることに注意してください。その場合のみ、EC プールがデータ・オブジェクトの格納に使用されます。例えば

---

#### 備考

オプションのパラメータ `-size`、`-min_size`、および `-crush_rule` は、複製されたメタデータ・プールには使用されますが、消去コード化されたデータ・プールには使用されません。データ・プールの `min_size` を変更する必要がある場合は、後で変更できます。`size` および `crush_rule` パラメータは、消去コード化プールでは変更できません。

---

EC プロファイルをさらにカスタマイズする必要がある場合は、Cephツールで直接作成できます。

<sup>10</sup>、`profile` パラメータで使用するプロファイルを指定します。例えば

```
pveceph pool create <pool-name> --erasure-coding profile=<profile-name>.
```

#### ストレージとしての EC プールの追加

既存の EC プールをストレージとして Proxmox VE に追加できます。これは

RBD プールが必要ですが、追加のデータプールオプションが必要です。

```
pvesm add rbd <ストレージ名> --pool <複製プール> --data-pool <ec-pool>
```

---

#### チップ

ローカルの Proxmox VE クラスタで管理されていない外部の Ceph クラスタ用に、`キーリング` と `monhost` オプションを追加することを忘れないでください。

---

<sup>10</sup> Ceph Erasure Code Profile <https://docs.ceph.com/en/quincy/rados/operations/erasure-code/#erasure-code-profiles>

### 8.8.3 プールを破壊

GUIでプールを破棄するには、ツリービューでノードを選択し、**Ceph**→ **Pools**パネルに移動します。破棄するプールを選択し、**[Destroy]**ボタンをクリックします。プールの破棄を確認するには、プール名を入力する必要があります。

以下のコマンドを実行してプールを破棄します。関連するストレージも削除するには、**-remove\_storages** を指定します。

```
pveceph プール破壊 <名前>
```

#### 備考

プールの削除はバックグラウンドで実行されるため、時間がかかることがあります。このプロセスの間、クラスタのデータ使用量が減少していることに気づくでしょう。

### 8.8.4 PGオートスケーラー

PGオートスケーラにより、クラスタは各プールに保存される(予想される)データ量を考慮し適切なpg\_num値を自動的に選択できます。Ceph Nautilusから使用できます。

調整が有効になる前に、PGオートスケーラー・モジュールをアクティブにする必要がある場合があります。

```
ceph mgr module enable pg_autoscaler
```

オートスケーラーはプールごとに構成され、以下のモードがあります:

#### 警告

pg\_numの推奨値が現在の値と大きく異なる場合、健全性の警告が発行されます。

#### on

pg\_numは自動的に調整されます。 pg\_numの自動調整は行われず、もし PG数は最適ではありません。

スケーリングファクターは、将来のデータ保存を容易にするために、target\_size、target\_size\_rat と pg\_num\_min オプションを指定します。

#### 警告

デフォルトでは、オートスケーラは、プールのPGカウントが3倍ずれている場合にチューニングを検討します。これは、データ配置のかなりのシフトにつながり、クラスタに高負荷をもたらす可能性があります。

PGオートスケーラの詳細については、Cephのブログ - [Nautilusの新機能をPG参照してください: マージとオートチューニング](#)。

## 8.9 Ceph CRUSHおよびデバイスクラス

<sup>11</sup> (Controlled Replication Under Scalable Hashing)アルゴリズムはCephの基盤です。

CRUSHはどこにデータを保存し、どこから取得するかを計算します。これには、中央のインデックス作成サービスが必要ないという利点があります。CRUSHはプールのOSD、バケット（デバイスの位置）、ルールセット（データの複製）のマップを使用して動作します。

### 備考

詳細については、CephドキュメントのCRUSH map <sup>a</sup>のセクションを参照してください。

<sup>a</sup> CRUSH マップ <https://docs.ceph.com/en/quincy/rados/operations/crush-map/>

このマップは、異なるレプリケーション階層を反映するように変更できます。オブジェクトのレプリカは、望ましい分散を維持したまま、（障害ドメインなど）分離することができます。

一般的な構成は、異なるCephプールに異なるクラスのディスクを使用することです。このため、Cephはルミナスでデバイスクラスを導入し、ルールセットを簡単に生成できるようになりました。

デバイスクラスは、`ceph osd crush tree`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、以下のコマンドで確認できます。

```
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:

身分証	クラス	重量	タイプ名
<b>明書</b>			
-16	nvme	2.18307	root default~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

特定のデバイスクラス上のオブジェクトのみを配布するようにプールに指示するには、まず、デバイスクラス用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 <ルール名> <ルート> <障害ドメイン><←'クラス
```

<ルール名	プールと接続するためのルール名 (GUI および CLI で表示)
<ルート	どのクラッシュルートに属するか(デフォルトのCephルート「default」)
<障害ドメイン	オブジェクトを配布する障害ドメイン (通常はホスト)
<クラス	使用する OSD バッキングストアのタイプ (例: nvme、sd、hdd)

ルールがCRUSHマップにあれば、そのルールセットを使用するようにプールに指示することができます。

```
ceph osd プールセット<プール名> crush_rule <rule-name>。
```

## チップ

プールにすでにオブジェクトが含まれている場合は、これらを適宜移動する必要があります。セットアップによっては、これはクラスタに大きなパフォーマンスの影響を与える可能性があります。別の方法として、新しいプールを作成し、ディスクを個別に移動することができます。

## 8.10 Cephクライアント

The screenshot shows the Proxmox VE interface with the following details:

- Left Sidebar (Server View):** Shows the Datacenter with three nodes: sumi1, sumi2, and sumi3.
- Main Content Area:**
  - Node 'sumi1':** A detailed log of Ceph cluster events for node sumi1. The log includes:
    - 2017-10-23 09:49:32.166969 mon.sumi1.mon.0 10.10.10.34.6789/0 247 : cluster [WPN] Health check failed: 1 osds down (OSD\_DOWN)
    - 2017-10-23 09:49:38.216306 mon.sumi1.mon.0 10.10.10.34.6789/0 251 : cluster [INF] Health check cleared: OSD\_DOWN (was: 1 osds down)
    - 2017-10-23 09:49:38.216335 mon.sumi1.mon.0 10.10.10.34.6789/0 252 : cluster [INF] Cluster is now healthy
    - 2017-10-23 09:49:38.2657/6 mon.sumi1.mon.0 10.10.10.34.6789/0 253 : cluster [INF] osd.10.10.10.36.6804/6536 boot
    - 2017-10-23 09:49:49.52.250111 mon.sumi1.mon.0 10.10.10.34.6789/0 269 : cluster [WPN] overall HEALTH\_WARN few PGs per OSD (10 < min 30) (TOO\_FEW\_PGS)
    - 2017-10-23 09:49:55.830536 mon.sumi1.mon.0 10.10.10.34.6789/0 272 : cluster [INF] Health check cleared: TOO\_FEW\_PGS (was: few PGs per OSD (24 < min 30))
    - 2017-10-23 09:49:55.830563 mon.sumi1.mon.0 10.10.10.34.6789/0 273 : cluster [INF] Cluster is now healthy
    - 2017-10-23 09:50:52.248928 mon.sumi1.mon.0 10.10.10.34.6789/0 274 : cluster [INF] mon.2 10.10.10.36.6890/0
    - 2017-10-23 09:50:52.248983 mon.sumi1.mon.0 10.10.10.34.6789/0 275 : cluster [INF] mon.1 10.10.10.35.6789/0
    - 2017-10-23 09:50:52.250192 mon.sumi1.mon.0 10.10.10.34.6789/0 276 : cluster [INF] overall HEALTH\_OK
    - 2017-10-23 09:53:50.877957 mon.sumi1.mon.0 10.10.10.34.6789/0 1 : cluster [INF] mon.sumi1 calling new monitor election
    - 2017-10-23 09:53:50.905341 mon.sumi1.mon.0 10.10.10.34.6789/0 2 : cluster [INF] mon.sumi1@0 won leader election with quorum 0,1,2
    - 2017-10-23 09:53:50.911884 mon.sumi1.mon.0 10.10.10.34.6789/0 3 : cluster [INF] overall HEALTH\_OK
    - 2017-10-23 09:53:50.918974 mon.sumi1.mon.0 10.10.10.34.6789/0 4 : cluster [INF] mon.2 10.10.10.36.6789/0
    - 2017-10-23 09:53:50.991996 mon.sumi1.mon.0 10.10.10.34.6789/0 5 : cluster [INF] monmap e3: 3 m ons at (sumi1=10.10.10.34.6789/0,sumi2=10.10.10.35.6789/0,sumi3=10.10.10.36.6789/0)
    - 2017-10-23 09:53:51.032992 mon.sumi1.mon.0 10.10.10.34.6789/0 9 : cluster [INF] monmap e3: 3 m ons at (sumi1=10.10.10.34.6789/0,sumi2=10.10.10.35.6789/0,sumi3=10.10.10.36.6789/0)
    - 2017-10-23 09:54:30.027300 mon.sumi1.mon.0 10.10.10.34.6789/0 1 : cluster [INF] mon.sumi1 calling new monitor election
    - 2017-10-23 09:54:30.052019 mon.sumi1.mon.0 10.10.10.34.6789/0 2 : cluster [INF] mon.sumi1@0 won leader election with quorum 0,1,2
    - 2017-10-23 09:54:30.061673 mon.sumi1.mon.0 10.10.10.34.6789/0 3 : cluster [INF] overall HEALTH\_OK
    - 2017-10-23 09:54:30.067936 mon.sumi1.mon.0 10.10.10.34.6789/0 4 : cluster [INF] mon.2 10.10.10.36.6789/0
    - 2017-10-23 09:54:30.125299 mon.sumi1.mon.0 10.10.10.34.6789/0 5 : cluster [INF] monmap e3: 3 m ons at (sumi1=10.10.10.34.6789/0,sumi2=10.10.10.35.6789/0,sumi3=10.10.10.36.6789/0)
    - 2017-10-23 09:54:30.160747 mon.sumi1.mon.0 10.10.10.34.6789/0 9 : cluster [INF] mon.1 10.10.10.35.6789/0
    - 2017-10-23 09:59:30.062068 mon.sumi1.mon.0 10.10.10.34.6789/0 75 : cluster [INF] mon.1 10.10.10.35.6789/0
    - 2017-10-23 09:59:30.062218 mon.sumi1.mon.0 10.10.10.34.6789/0 76 : cluster [INF] mon.2 10.10.10.36.6789/0
    - 2017-10-23 10:00:00.001132 mon.sumi1.mon.0 10.10.10.34.6789/0 77 : cluster [INF] overall HEALTH\_OK
    - 2017-10-23 10:04:30.062378 mon.sumi1.mon.0 10.10.10.34.6789/0 78 : cluster [INF] mon.1 10.10.10.35.6789/0
    - 2017-10-23 10:04:30.062469 mon.sumi1.mon.0 10.10.10.34.6789/0 79 : cluster [INF] mon.2 10.10.10.36.6789/0
    - 2017-10-23 10:09:30.062524 mon.sumi1.mon.0 10.10.10.34.6789/0 80 : cluster [INF] mon.2 10.10.10.36.6789/0
    - 2017-10-23 10:09:30.062578 mon.sumi1.mon.0 10.10.10.34.6789/0 81 : cluster [INF] mon.1 10.10.10.35.6789/0
    - 2017-10-23 10:14:30.062650 mon.sumi1.mon.0 10.10.10.34.6789/0 82 : cluster [INF] mon.2 10.10.10.36.6789/0
    - 2017-10-23 10:14:30.062750 mon.sumi1.mon.0 10.10.10.34.6789/0 83 : cluster [INF] mon.1 10.10.10.35.6789/0
    - 2017-10-23 10:19:30.062901 mon.sumi1.mon.0 10.10.10.34.6789/0 84 : cluster [INF] mon.1 10.10.10.35.6789/0
    - 2017-10-23 10:20:30.062951 mon.sumi1.mon.0 10.10.10.34.6789/0 85 : cluster [INF] mon.2 10.10.10.36.6789/0
    - 2017-10-23 10:24:30.062972 mon.sumi1.mon.0 10.10.10.34.6789/0 86 : cluster [INF] mon.1 10.10.10.35.6789/0
    - 2017-10-23 10:24:30.063027 mon.sumi1.mon.0 10.10.10.34.6789/0 87 : cluster [INF] mon.2 10.10.10.36.6789/0
    - 2017-10-23 10:29:30.063104 mon.sumi1.mon.0 10.10.10.34.6789/0 88 : cluster [INF] mon.1 10.10.10.35.6789/0
    - 2017-10-23 10:29:30.063171 mon.sumi1.mon.0 10.10.10.34.6789/0 89 : cluster [INF] mon.2 10.10.10.36.6789/0
    - 2017-10-23 10:34:30.063271 mon.sumi1.mon.0 10.10.10.34.6789/0 90 : cluster [INF] mon.1 10.10.10.35.6789/0
    - 2017-10-23 10:34:30.063324 mon.sumi1.mon.0 10.10.10.34.6789/0 91 : cluster [INF] mon.2 10.10.10.36.6789/0
    - 2017-10-23 10:39:30.063434 mon.sumi1.mon.0 10.10.10.34.6789/0 92 : cluster [INF] mon.2 10.10.10.36.6789/0
    - 2017-10-23 10:39:30.063467 mon.sumi1.mon.0 10.10.10.34.6789/0 93 : cluster [INF] mon.1 10.10.10.35.6789/0
    - 2017-10-23 10:44:30.063597 mon.sumi1.mon.0 10.10.10.34.6789/0 94 : cluster [INF] mon.1 10.10.10.35.6789/0
    - 2017-10-23 10:44:30.063630 mon.sumi1.mon.0 10.10.10.34.6789/0 95 : cluster [INF] mon.2 10.10.10.36.6789/0
    - 2017-10-23 10:49:30.063792 mon.sumi1.mon.0 10.10.10.34.6789/0 96 : cluster [INF] mon.2 10.10.10.36.6789/0
    - 2017-10-23 10:49:30.063826 mon.sumi1.mon.0 10.10.10.34.6789/0 97 : cluster [INF] mon.1 10.10.10.35.6789/0
    - 2017-10-23 10:54:30.063842 mon.sumi1.mon.0 10.10.10.34.6789/0 98 : cluster [INF] mon.2 10.10.10.36.6789/0
    - 2017-10-23 10:54:30.063894 mon.sumi1.mon.0 10.10.10.34.6789/0 99 : cluster [INF] mon.1 10.10.10.35.6789/0
    - 2017-10-23 10:59:30.063972 mon.sumi1.mon.0 10.10.10.34.6789/0 100 : cluster [INF] mon.1 10.10.10.35.6789/0
    - 2017-10-23 10:59:30.064008 mon.sumi1.mon.0 10.10.10.34.6789/0 101 : cluster [INF] mon.2 10.10.10.36.6789/0
    - 2017-10-23 11:00:00.000123 mon.sumi1.mon.0 10.10.10.34.6789/0 102 : cluster [INF] overall HEALTH\_OK
    - 2017-10-23 11:04:30.064161 mon.sumi1.mon.0 10.10.10.34.6789/0 103 : cluster [INF] mon.1 10.10.10.35.6789/0
    - 2017-10-23 11:04:30.064211 mon.sumi1.mon.0 10.10.10.34.6789/0 104 : cluster [INF] mon.2 10.10.10.36.6789/0
    - 2017-10-23 11:09:30.064399 mon.sumi1.mon.0 10.10.10.34.6789/0 141 : cluster [INF] mon.1 10.10.10.35.6789/0
    - 2017-10-23 11:09:30.064452 mon.sumi1.mon.0 10.10.10.34.6789/0 142 : cluster [INF] mon.2 10.10.10.36.6789/0
    - 2017-10-23 11:14:30.064570 mon.sumi1.mon.0 10.10.10.34.6789/0 203 : cluster [INF] mon.1 10.10.10.35.6789/0
    - 2017-10-23 11:14:30.064624 mon.sumi1.mon.0 10.10.10.34.6789/0 204 : cluster [INF] mon.2 10.10.10.36.6789/0

前のセクションのセットアップに続いて、Proxmox VEを設定して、このようなプールを使用してVMおよびコンテナイメージを保存できます。GUIを使用して新しいRBDストレージを追加するだけです([Ceph RADOS Block Devices \(RBD\)の項](#)を参照)。

また、外部Cephクラスタ用の定義済みの場所にキーリングをコピーする必要があります。CephがProxmoxノード自体にインストールされている場合、これは自動的に実行されます。

### 備考

ファイル名は`<storage_id> + `keyring``で、`<storage_id>`は`/etc/pve/storage.cfg`の`rbd:`の後に続く表現です。以下の例では、`my-ceph-storage`は

```
<storage_id>:
```

```
mkdir /etc/pve/priv/ceph
cp /etc/ceph/ceph.client.admin.keyring /etc/pve/priv/ceph/my-ceph-storage.keyring
```

キーホルダー

## 8.11 セフエフエス

Cephはファイルシステムも提供し、RADOSブロックデバイスと同じオブジェクトストレージ上で動作します。メタデータサーバ(MDS)を使用して、RADOSでバックアップされたオブジェクトをファイルとディレクトリにマッピングするため、POSIX準拠の複製ファイルシステムを提供できます。これにより、クラスタ化された可用性の高い共有ファイルシステムを簡単に構成できます。Cephのメタデータサーバは、ファイルがCephクラスタ全体に均等に分散されることを保証します。その結果、負荷が高い場合でも、NFSなどの従来の共有ファイルシステムアプローチで問題となる、単一のホストを圧倒することはありません。

Name ↑	Host	Status	Address	Version
mds.prod1	prod1	up:standby	192.168.30.64:6805/3515028757	15.2.11
mds.prod2	prod2	up:active	192.168.30.65:6801/1445057454	15.2.11
mds.prod3	prod3	up:standby	192.168.30.66:6801/1826014728	15.2.11

Proxmox VEは、ハイパーコンバージドCephFSの作成と、バックアップ、ISOファイル、コンテナテンプレートを保存する[ストレージとしてのCephFS](#)の既存の使用の両方をサポートしています。

### 8.11.1 メタデータサーバー (MDS)

CephFSを機能させるには、少なくとも1つのメタデータサーバを構成して実行する必要があります。データシートは、Proxmox VE Web GUIのNode -> CephFSパネルまたはコマンドラインから作成できます：

#### ペケフ・マップ・クリエイト

クラスタには複数のメタデータサーバーを作成できますが、デフォルト設定では一度にアクティブにできるのは1つのみです。データシートまたはそのノードが応答しなくなった（またはクラッシュした）場合、別のスタンバイ データシートがアクティブに昇格します。作成時に`hotstandby`パラメータオプションを使用することで、アクティブとスタンバイのデータシート間のハンドオーバーを高速化できます：

```
mds スタンバイ・リプレイ= true
```

を/etc/pve/ceph.confの各データシートセクションに追加します。この設定を有効にすると、指定されたデータシートはウォーム維持し、アクティブなデータシートにポーリングします。

#### 備考

このアクティブなポーリングは、システムおよびアクティブなデータシートにさらなるパフォーマンスの影響を与えます。

#### 複数のアクティブMDS

Luminous (12.2.x)以降、複数のアクティブなメタデータサーバーを同時に実行することができますが、これは通常、大量のクライアントが並行して実行されている場合にのみ有用です。そうでない場合、データシートがシステムのボトルネックになることはほとんどありません。この設定を行う場合は、Cephのドキュメントを参照してください。<sup>12</sup>

### 8.11.2 CephFSの作成

Proxmox VEのCephFSの統合により、Webインターフェース、CLI、または外部APIインターフェースを使用して簡単にCephFSを作成できます。これを動作させるには、いくつかの前提条件が必要です：

CephFSのセットアップを成功させるための前提条件：

- [Cephパッケージのインストール](#) - 以前に実行済みの場合は、最新のシステムで再実行して、CephFS関連パッケージがすべてインストールされるようにします。
- [モニターの設定](#)
- [OSDの設定](#)
- [少なくとも1つのデータシートを設定](#)

これが完了したら、Web GUIのNode → CephFSのいずれかでCephFSを作成します。

パネルやコマンドラインツール pveceph あります：

```
pveceph fs create --pg_num 128 --add-storage
```

これにより、cephfsという名前のCephFSが作成され、128の配置グループを持つcephfs\_dataという名前のデータ用プールと、データプールの配置グループの4分の1(32)を持つcephfs\_metadataという名前のメタデータ用プールが使用されます。セットアップに適した配置グループ番号(pg\_num)の詳細については、[Proxmox VE managed Ceph poolの章](#)を参照するか、Cephのドキュメントを参照してください。<sup>6</sup> さらに--add-storageパラメータは、CephFSが正常に作成された後、Proxmox VEストレージ構成に追加します。

<sup>12</sup> 複数のアクティブなデータシートデーモンを設定する <https://docs.ceph.com/en/quincy/cephfs/multimds/>

### 8.11.3 CephFSの破棄



警告

CephFSを破棄すると、そのデータはすべて使用できなくなります。これは元に戻せません！

CephFSを完全かつ優雅に削除するには、次の手順が必要です：

- すべての非Proxmox VEクライアントを切断します(たとえば、ゲストのCephFSをアンマウントします)。
- 関連するすべてのCephFS Proxmox VEストレージエントリを無効にします(自動的にマウントされないようにします)。
- 破棄するCephFS上のゲスト(ISOなど)から、使用済みのリソースをすべて削除します。
- を使用して、すべてのクラスタノードのCephFSストレージを手動でアンマウントします。

```
umount /mnt/pve/<STORAGE-NAME>
```

ここで、<STORAGE-NAME>はProxmox VE内のCephFSストレージの名前です。

- メタデータサーバ(MDS)を停止するか破棄して、そのCephFSでメタデータサーバ(MDS)が実行されていないことを確認します。これは、Webインターフェースまたはコマンドラインインターフェースで実行できます。後者の場合は、次のコマンドを実行します：

```
pveceph stop --service mds.NAME  
止めるか
```

```
pveceph mds destroy NAME
```

破壊ために。

アクティブなデータシートが停止または削除されると、スタンバイサーバーは自動的にアクティブに昇格することに注意してください。

- これで、CephFSを次のように破壊できます。

```
pveceph fs destroy NAME --remove-storages --remove-pools
```

これにより、基盤となるCephプールが自動的に破棄され、pve設定からストレージが削除されます。

これらの手順の後、CephFSを完全に削除し、他のCephFSインスタンスがある場合は、停止したメタデータサーバを再度起動してスタンバイとして動作させることができます。

## 8.12 Cephメンテナンス

### 8.12.1 OSDの交換

以下の手順で、OSDのディスクを交換できます。これは、Cephで最も一般的な保守作業の1つです。ディスクが正常であるように見えるOSDに問題がある場合は、最初に[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、同じタイプとサイズの[推奨](#)交換ディスクを入手してください。
2. 問題のOSDを[破棄](#)してください。
3. 古いディスクをサーバーから取り外し、新しい取り付けます。
4. OSDを再度[作成](#)します。
5. 自動リバランス後、クラスタのステータスは[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、次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
```

## 8.12.2 トリム/廃棄

VMやコンテナで`fstrim` (discard) を定期的に実行するのは良い習慣です。これにより、ファイルシステムがもう使用していないデータブロックが解放されます。これにより、データ使用量とリソース負荷が軽減されます。最近のオペレーティング・システムでは、このようなdiscardコマンドを定期的にディスクに発行しています。仮想マシンが[ディスク廃棄オプション](#)を有効にしていることを確認するだけです。

## 8.12.3 スクラップ&ディープスクラップ

Cephは配置グループをスクラップすることで、データの整合性を確保します。CephはPG内のすべてのオブジェクトの健全性をチェックします。スクラップには、毎日の安価なメタデータチェックと毎週のディープデータチェックの2つの形式があります。毎週のディープスクラップはオブジェクトを読み取り、チェックサムを使用してデータの整合性を確保します。実行中のスクラップがビジネス（パフォーマンス）ニーズの妨げになる場合は、スクラップ<sup>(13)</sup>が実行される時間を調整できます。

## 8.12.4 Proxmox VE+ Ceph HCIクラスタのシャットダウン

Proxmox VE + Cephクラスタ全体をシャットダウンするには、まずすべてのCephクライアントを停止します。これらは主にVMとコンテナになります。Ceph FSまたはインストールされているRADOS GWにアクセスする可能性がある追加のクライアントがある場合は、これらも停止します。高可用性のゲストは、Proxmox VEツールを使用してパワーダウンすると、状態が停止に切り替わります。

すべてのクライアント、VM、およびコンテナがオフになるか、Cephクラスタにアクセスしなくなったら、Cephクラスタが健全な状態であることを確認します。

```
ceph -s
```

Web UIまたはCLIのいずれかを使用します：

すべての自己回復アクションを無効にし、CephクラスタのクライアントIOを一時停止するには、Ceph→ OSDパネルまたはCLIで次のOSDフラグを有効にします：

```
ceph osd set noout ceph osd set
norecover
ceph osd set norebalance ceph osd
set nobackfill ceph osd set nodown
ceph osd set pause
```

<sup>13</sup> セフ・スクラビング <https://docs.ceph.com/en/quincy/rados/configuration/osd-config-ref/#scrubbing>

モニター（MON）のないノードのパワーダウンを開始します。これらのノードのパワーダウンが完了したら、モニターのあるノードのパワーダウンを続けます。

クラスタの電源を入れるときは、モニタ(MON)があるノードを最初に起動します。すべてのノードが稼働、すべてのCephサービスが稼働していることを確認してから、

```
ceph osd unset pause ceph osd unset  
nodown ceph osd unset nobackfill  
ceph osd unset norebalance ceph osd  
unset norecover ceph osd unset noout
```

OSDフラグを再度設定解除します：

これでゲストを起動できます。高可用性ゲストは、電源が入ると状態が開始状態に変わります。

## 8.13 Cephの監視とトラブルシューティング

Cephツールを使用するか、Proxmox VE APIを介してステータスにアクセスするかして、Cephデプロイの健全性を最初から継続的に監視することが重要です。

以下のCephコマンドを使用して、クラスタが健全かどうか(*HEALTH\_OK*)、警告があるかどうか(*HEALTH\_WARN*)、またはエラーかどうか(*HEALTH\_ERR*)を確認できます。クラスタが不健全である場合、以下のステータスコマンドによって現在のイベントと取るべきアクションの概要もわかります。これらの実行を停止するには、CTRL-Cを押します。

クラスタの継続的に監視します：

```
watch ceph --status
```

クラスタのステータスを一度だけ表示し（更新されません）、ステータス行を継続的に追加します：

```
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、よく使用されるトラブルシューティング記載されています。詳細については、Cephの公式Web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

- ディスクヘルス監視
- 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：

```
journalctl --since "2日前"
```

- IPMIとRAIDコントローラのログ

Cephサービスのクラッシュは、以下のコマンドを実行して一覧表示し、詳細を確認できます：

<sup>14</sup>Ceph トラブルシューティング <https://docs.ceph.com/en/quincy/rados/troubleshooting/>

```
ceph crash ls  
cephクラッシュ情報<crash_id>。
```

新しいクラッシュとしてマークされたクラッシュは、実行することで確認できます：

```
cephクラッシュアーカイブ
```

より詳細に表示するには、すべてのCephサービスに/var/log/ceph/の下にログファイルがあります。詳細が必要な場合は、ログレベルを調整できます<sup>15</sup>。

## CEPHの問題の一般的な原因

- 輪廻、欠陥のあるスイッチ、シャットダウンされたインターフェース、ブロックされたファイアウォールなどのネットワークの問題。すべてのProxmox VEノードがcorosyncクラスタネットワークおよびCephパブリックネットワークとクラスタネットワークで確実に到達可能かどうかを確認します。
- ディスクまたは接続部品：
  - 不完全
  - 取り付け不可
- 高負荷時のI/O性能不足（HDD、消費者向けハードウェア、不適切なRAIDコントローラを使用した場合など）
- 健全なCephクラスタの推奨事項を満たしていません。

## CEPHの一般的な問題

### OSDダウン/クラッシュ

故障したOSDは、10分後にダウンと報告され、ほとんどが（自動）停止します。次の場合は

をクリックします。Webインターフェイスを使用して手動でアクティベーションを試すには、任意のノード→ Ceph→ OSDにアクセスし、OSDを選択して、[Start]、[In]、[Reload]をクリックします。シェルを使用する場合は、影響を受けるノードで次のコマンドを実行します：

```
ceph-volume lvm activate --all
```

故障したOSDを有効にするには、各ノードを安全に再起動するか、最後のOSDを再作成または交換する必要があります。

<sup>15</sup> Cephログとデバッグ <https://docs.ceph.com/en/quincy/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レプリケートし、共有ストレージを使用せずにすべてのデータを利用できるようにします。レプリケーションでは、スナップショットを使用してネットワーク経由で送信されるトラフィックを最小限に抑えます。そのため、新しいデータは、最初の完全な同期後に増分的にのみ送信されます。ノードに障害が発生した場合でも、ゲストデータはレプリケートされた利用可能です。

レプリケーションは設定可能な間隔で自動的に行われます。最小レプリケーション間隔は1分で、最大間隔は週に1回です。これらの間隔を指定するために使われるフォーマットは `systemd` カレンダーイベントのサブセットです：

ゲストを複数のターゲットノードにレプリケートすることは可能ですが、同じターゲットノードに2回レプリケートすることはできません。各レプリケーションの帯域幅は、ストレージやサーバーの過負荷を避けるために制限することができます。

ゲストがすでにレプリケートされているノードに移行する場合は、最後のレプリケーション以降の変更（いわゆるデルタ）のみを転送する必要があります。これにより、必要な時間が大幅に短縮されます。ゲストをレプリケーションターゲットノードに移行すると、レプリケーションの方向が自動的に切り替わります。

例えばVM現在nodeA上にあり、nodeBにレプリケートされています。VM100をnodeBに移行すると、自動的にnodeBからnodeAにレプリケートされます。

ゲストがレプリケートされていないノードに移行する場合は、ディスクデータ全体を送信する必要があります。移行後、レプリケーションジョブはこのゲストを設定されたノードにレプリケートし続けます。



### 重要

高可用性はストレージ・レプリケーションとの組み合わせで許可されますが、最後に同期された時刻とノードが故障した時刻の間に何らかのデータ損失が発生する可能性があります。

## 9.1 対応ストレージタイプ

表9.1：ストレージの種類

説明	プラグインタイプ	スナップ写真	安定
ZFS（ローカル）	ゼットスプール	はい	はい

## 9.2 スケジュール形式

レプリケーションでは、スケジュールの設定に[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・ジョブに問題が発生すると、エラー状態になります。この状態では、設定されたレプリケーション間隔が一時的に中断されます。失敗したレプリケーションは、30分間隔で繰り返し再試行されます。これが成功すると、元のスケジュールが再び有効になります。

### 9.3.1 考えられる問題

最も一般的な問題は、以下のリストのとおりです。セットアップによっては別の原因があるかもしれません。

- ネットワークが機能していません。
- レプリケーション先のストレージに空き容量がありません。
- 同じストレージIDを持つストレージがターゲットノードにありません。

#### 備考

レプリケーション・ログを使用すれば、いつでも問題の原因を突き止めることができます。

### 9.3.2 エラー時のゲストの移行

重大なエラーが発生した場合、仮想ゲストは障害が発生したノード上でスタッカする可能性があります。その場合、手動で再び動作するノードに移動する必要があります。

### 9.3.3 例

ノードAで2つのゲスト（VM 100とCT 200）を実行し、ノードBにレプリケートしているとします。そこで、ゲストを手動でノードBに移行する必要があります。

- sshでノードBに接続するか、ウェブUIでシェルを開きます。
- クラスタがクオーレートかどうかをチェックします。

```
# pvecm ステータス
```

- クオーラムがない場合は、まずこれを修正し、ノードを再び操作できるようにすることを強くお勧めします。現時点ではこれが不可能な場合のみ、以下のコマンドを使用して現在のノードにクオーラムを強制することができます：

```
# pvecm 期待値 1
```



## 警告

予想される投票が設定されている場合にクラスタに影響を与えるような変更（ノード、ストレージ、仮想ゲストの追加/削除など）はください。重要なゲストを再び稼働させるか、クオーラムの問題そのものを解決する場合にのみ使用してください。

- 両方のゲスト設定ファイルを元のノードAからノードBに移動します:

```
# mv /etc/pve/nodes/A/qemu-server/100.conf /etc/pve/nodes/B/qemu-server!
# mv /etc/pve/nodes/A/lxc/200.conf /etc/pve/nodes/B/lxc/200.conf
```

- これでゲストをお迎えすることができます:

```
# qm start 100 # pct
start 200
```

VMIDとノード名をそれぞれの置き換えることを忘れないでください。

## 9.4 求人管理

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・インターフェース（CLI）ツールのpvesrを使用することもできます。

レプリケーション・パネルは、Web GUIのすべてのレベル（データセンター、ノード、仮想ゲスト）で見つけることができます。どのジョブが表示されるかは、すべてのジョブ、ノード固有のジョブ、ゲスト固有のジョブで異なります。

新しいジョブを追加する際、まだ選択されていない場合はゲストとターゲットノードを指定する必要があります。レプリケーションスケジュールは、デフォルトのすべて15分が不要場合に設定できます。レプリケーションジョブにレート制限を課すことができます。レート制限は、ストレージの負荷を許容範囲に保つのに役立ちます。

レプリケーション・ジョブはクラスタ全体で一意のIDによって識別されます。このIDは、ジョブ番号に加えてVMIDで構成されます。CLIツールを使用する場合のみ、このIDを手動で指定する必要があります。

## 9.5 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、Web インターフェイスの Datacenter -> Options -> Migration Settings または datacenter.cfg で [移行ネットワーク](#) を設定します。詳細については、[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、帯域幅を10 Mbps（メガバイト/秒）に制限して5分ごとに実行するレプリケーションジョブを作成します。

```
# pvesr create-local-job 100-0 pvel --schedule "*/5" --rate 10  
ID 100-0 のアクティブなジョブを無効にします。
```

```
# pvesr disable 100-0
```

IDが100-0の非アクティブ化されたジョブを有効にします。

```
# pvesr enable 100-0
```

ID 100-0のジョブのスケジュール間隔を時間に1回に変更します。

```
# pvesr update 100-0 --schedule '*/00'
```

## 第10章

### QEMU/KVM仮想マシン

QEMU (Quick Emulator の略) は、物理コンピュータをエミュレートするオープンソースのハイパーバイザーです。QEMU が実行されているホストシステムから見ると、QEMU はユーザー プログラムであり、パーティション、ファイル、ネットワーク カードなどの多数のローカル リソースにアクセスできます。

エミュレートされたコンピュータで実行されてい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、実際のハードウェア上で実行されているかのように動作します。たとえば、ISO イメージを QEMU にパラメータとして渡すと、エミュレートされたコンピュータで実行されている OS には、CD ドライブに挿入された本物の CD-ROM が表示されます。

QEMU は ARM から Sparc まで多種多様なハードウェアをエミュレートできますが、Proxmox VE は 32 ビットおよび 64 ビットの PC クローンのエミュレーションのみに対応しています。PC クローンのエミュレーションは、エミュレートするアーキテクチャがホストアーキテクチャと同じ場合に QEMU を大幅に高速化するプロセッサ拡張機能を利用できるため、最も高速なエミュレーションの 1 つでもあります。

---

#### 備考

KVM (Kernel-based Virtual Machine) という用語に遭遇することがあります。これは、QEMU が Linux KVM モジュールを介して仮想化プロセッサ拡張のサポートを受けて実行されていることを意味します。Proxmox VE の QEMU は常に KVM モジュールをロードしようとするため、Proxmox VE のコンテキストでは QEMU と KVM は同じ意味で使用できます。

---

Proxmox VE 内の QEMU は、ブロックデバイスと PCI デバイスにアクセスするために必要なルートプロセスとして実行されます。

#### 10.1 エミュレートされたデバイスと準仮想化デバイス

QEMU によってエミュレートされる PC ハードウェアには、マザーボード、ネットワークコントローラ、SCSI、IDE、SATA コントローラ、シリアルポート (完全なリストは `kvm(1)` man ページを参照してください) が含まれます。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まったく同等のソフトウェアデバイスであり、ゲストで実行されている OS に適切なドライバがあれば、実際のハードウェア上で実行されているかのようにデバイスを使用できます。これにより、QEMU は変更されていないオペレーティングシステムを実行できます。

しかし、ハードウェアで実行する予定だったものをソフトウェアで実行すると、ホスト CPU に多くの余分な作業が発生するため、これにはパフォーマンス上のコストが伴います。これを軽減するために、QEMU はゲスト OS に準仮想化デバイスを提示することができます。この場合、ゲスト OS は QEMU 内で実行されることを認識し、ハイパーバイザーと協調します。

QEMU は virtio 仮想化標準に依存しているため、準仮想化された汎用ディスク コントローラー、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 コントローラーなど、準仮想化された virtio デバイスを表示。

### チップ

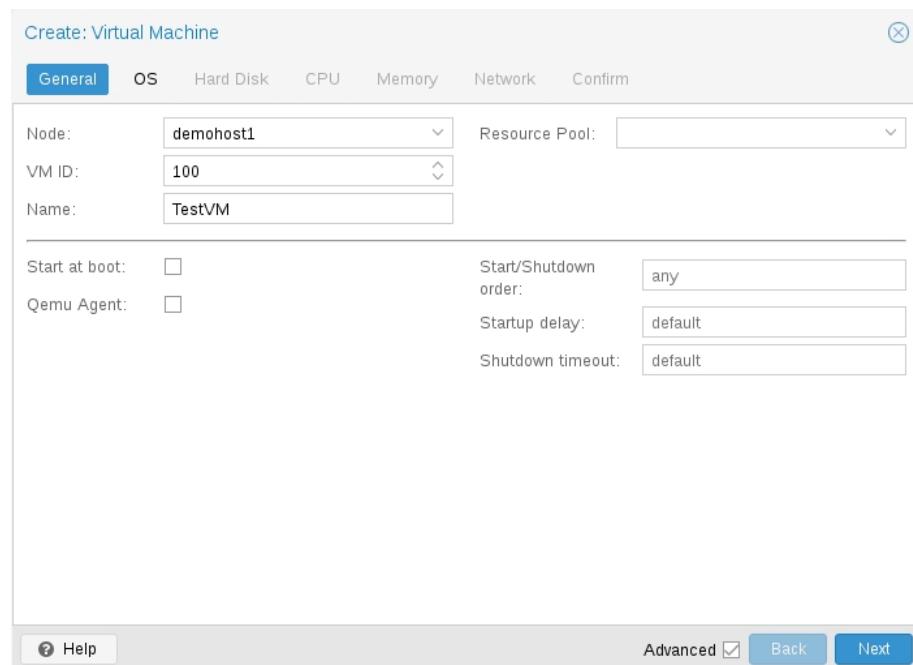
可能な限り、virtio デバイスを使用することを強くお勧めします。エミュレートされた IDE コントローラに対して virtio 汎用ディスクコントローラを使用すると、bonnie++(8) で測定されたシーケンシャル書き込みスループットが 2 倍になります。virtio ネットワークインターフェースを使用すると、iperf(1) で測定されたように、エミュレートされた Intel E1000 ネットワークカードのスループットの最大 3 倍を提供できます。<sup>a</sup>

<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を選択しようとします。パフォーマンスを低下させたりデータを危険にさらす可能性があるため、変更する設定の意味を理解してください。

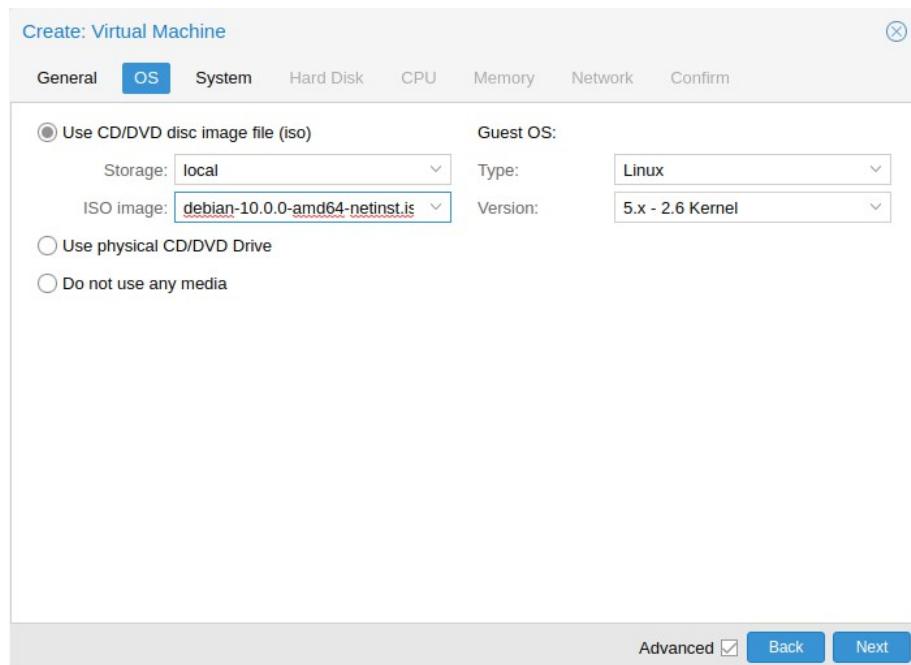
### 10.2.1 一般設定



VMの一般的な設定は次のとおりです。

- **ノード**: VM が実行される物理サーバー
- **VM ID**: VMを識別するためのProxmox VEインストール内で一意の番号です。
- **Name**: VM を説明するために使用できる自由形式のテキスト文字列です。
- **リソースプール**: VMの論理グループ

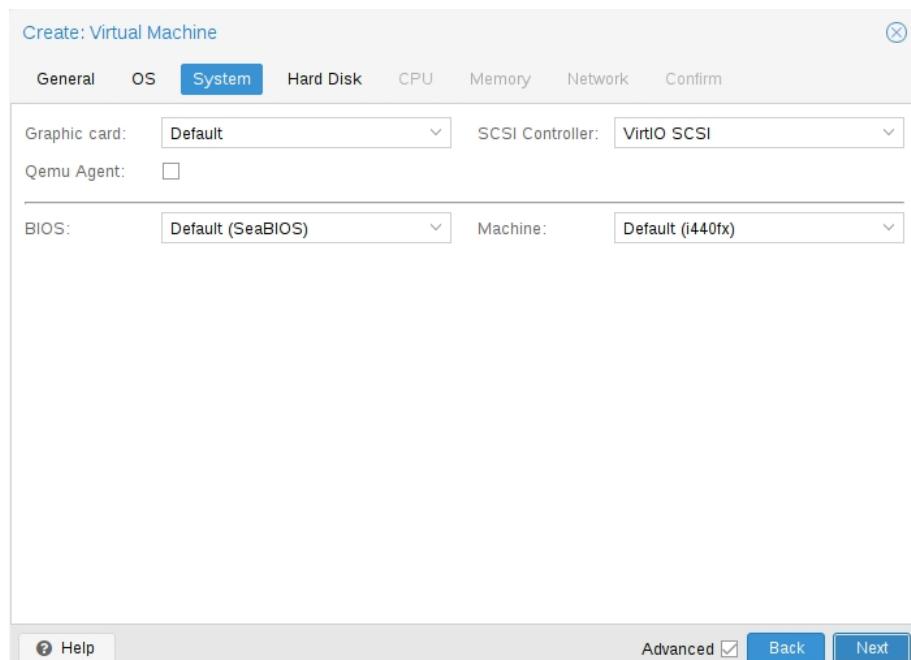
## 10.2.2 OS設定



仮想マシン(VM)を作成する際、適切なオペレーティングシステム(OS)を設定することで、Proxmox VEはいくつかの低レベルパラメータを最適化することができます。例えば、Windows OSは、BIOSクロックがローカル時間を使用することを期待しますが、UnixベースのOSは、BIOSクロックがUTC時間を使用することを期待します。

## 10.2.3 システム設定

VMの作成時に、新しいVMの基本的なシステムコンポーネントの一部を変更することができます。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](#)も変更できます。QEMU ゲストエージェントをインストールする予定がある場合、または選択した ISO イメージがすでに出荷されており、自動的にインストールされる場合は、[QEMU](#) チェックを入れるとよいでしょう。

これはProxmox VEに、より多くの情報を表示し、いくつかのアクション（シャットダウンやスナップショットなど）をよりインテリジェントに完了するために、その機能を使用できることを知らせます。

Proxmox VEでは、[SeaBIOSとOVMF](#)という異なるファームウェアとマシンタイプでVMをブートすることができます。ほとんどの場合、[PCIe パススルー](#)を使用する場合のみ、デフォルトの SeaBIOS から OVMF に切り替えます。

## マシンタイプ

VMのマシン・タイプは、VMの仮想マザーボードのハードウェア・レイアウトを定義します。デフォルトの[Intel 440FX](#)か、仮想PCIeバスも提供する[Q35チップセット](#)のいずれかを選択できます。さらに、[vIOMMU の実装](#)を選択することもできます。

## 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、ある QEMU バイナリは多くのマシンバージョンをサポートしています。新しいバージョンは、新機能のサポート、修正、一般的な改良をもたらすかもしれません。ただし、仮想ハードウェアのプロパティも変更されます。ゲストの観点からの突然の変更を回避し、VM状態の互換性を確保するために、RAMを使用したライブマイグレーションとスナップショットは、新しいQEMUインスタンスで同じマシンバージョンを使用し続けます。

Windowsゲストの場合、マシンバージョンは作成時に固定されます。Windowsは仮想ハードウェアの変更（コールドブート間であっても）に敏感だからです。例えば、ネットワークデバイスの例は、異なるマシンバージョンで異なるかもしれません。Linuxのような他のOSは、通常このような変更にうまく対応できます。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。これは、再スタート後、QEMUバイナリがサポートする最新のマシンバージョンが使用されることを意味します（例えば、QEMU 8.1がサポートする最新のマシンバージョンは、各マシンタイプのバージョン8.1です）。

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ような新機能や修正を実装する際のセーフガードとしても使用され、後方互換性を確保します。ライブマイグレーションなどの実行中のVMに対する操作では、QEMU仮想化の観点からだけでなく、Proxmox VEがQEMU仮想マシンインスタンスをどのように作成するかという観点からも、VMを正確にリカバリできるように、実行中のマシンバージョンが保存されます。

## PVEマシン改訂

Proxmox VEでは、新しいQEMUリリースを待たずにハードウェアレイアウトの変更やオプションの変更が必要になることがあります。このため、Proxmox VEは+pveXという形で追加のダウNSTリームリビジョンを追加しています。これらのリビジョンでは、X は新しい QEMU マシン バージョンごとに 0 になり、この場合は省略されます。たとえば、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またはデフォルトオプションを変更したい場合、リビジョンはインクリメントされ、新しく作成されたゲスト、または常に最新のマシンバージョンを使用するVMの再起動時に使用されます。

## QEMU マシン バージョンの非推奨

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、メジャーリリースは約2年ごとに行われるため、Proxmox VEのメジャーリリースでは、約2つ前のProxmox VEのメジャーリリースのマシンバージョンがサポートされます。

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リリースで削除されるすべてのマシンバージョンを避けるようにVM設定を更新する必要があります。これにより、そのリリース中もゲストを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の削除ポリシーはまだ有効になっていません。

2.4.Proxmox VE 9用にリリースされるQEMUバイナリの最終バージョンはQEMU 11.2になる予定です。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、Proxmox VE 9リリースのライフサイクルでは6.0が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メジャーバージョンずつ増加する予定です（たとえば、Proxmox VE 10は8.0）。

#### 新しいマシンバージョンへのアップデート

非推奨の警告が表示されたら、マシンのバージョンを新しい変更する必要があります。最初にバックアップを取り、ゲストがハードウェアを認識する方法の変更に備えるようにしてください。シナリオによっては、特定のドライバの再インストールが必要になるかもしれません。また、これらのマシンバージョンで取得されたRAMのスナップショット（つまり `runningmachine` 設定エントリ）をチェックする必要があります。残念ながら、スナップショットのマシンバージョンを変更する方法はないので、スナップショットからデータを救い出すにはスナップショットをロードする必要があります。

### 10.2.4 ハードディスク

#### バス/コントローラ

QEMUは、多数のストレージコントローラをエミュレートできます：

##### チップ

**VirtIO SCSI** または **VirtIO Block** コントローラを使用することを強くお勧めします。

- 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です。このコントローラが最近の設計に取って代わられたとしても、思いつく限りのすべてのOSがこのコントローラをサポートしているため、2003年以前にリリースされたOSを実行したい場合に最適です。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 (Serial ATA) コントローラは、2003年に開発されたもので、より近代的な設計となっており、より高いスループットとより多くのデバイスを接続することができます。このコントローラには最大6台のデバイスを接続できます。
  - 1985年に設計されたSCSIコントローラは、サーバグレードのハードウェアで一般的で、最大14台のストレージデバイスを接続できます。Proxmox VEはデフォルトでLSI 53C895Aコントローラをエミュレートします。
- パフォーマンスを重視する場合は、*VirtIO SCSI* シングルタイプのSCSIコントローラを使用し、接続ディスクの**IOスレッド**設定を有効にすることをお勧めします。これは、Proxmox VE 7.3 以降で新しく作成される Linux VM のデフォルトです。各ディスクは独自の *VirtIO SCSI* コントローラーを持ち、QEMU は専用スレッドでディスクの IO を処理します。Linuxディストリビューションは2012年から、FreeBSDは2014年からこのコントローラをサポートしています。Windows OSの場合は、インストール時にドライバを含む追加のISOを提供する必要があります。
- VirtIO または `virtio-blk` と呼ばれることが多い VirtIO Block コントローラーは、古いタイプの準仮想化コントローラーです。機能面では、VirtIO SCSI コントローラーに取って代わられました。

## 画像フォーマット

各コントローラには多数のエミュレートされたハードディスクが接続され、設定されたストレージに存在するファイルまたはブロックデバイスによってバックアップされます。ストレージタイプの選択により、ハードディスクイメージのフォーマットが決まります。ブロックデバイスを使用するストレージ (LVM、ZFS、Ceph) では **raw ディスクイメージフォーマット**が必要ですが、ファイルベースのストレージ (Ext4、NFS、CIFS、GlusterFS) では **raw ディスクイメージフォーマット**または **QEMU イメージフォーマット**のいずれかを選択できます。

- **QEMU イメージフォーマット**は、ディスクイメージのシンプロビジョニングを可能にするコピーオンライトフォーマットです。
- **rawディスクイメージ**は、ハードディスクのビット単位のイメージで、Linuxでブロックデバイスに対して dd コマンドを実行したときに得られるものと似ています。このフォーマットは、それ自体で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ません。ただし、**QEMU イメージ形式**よりも最大 10% 高速になる可能性があります。<sup>1</sup>
- **VMwareイメージフォーマット**は、ディスクイメージを他のハイパーバイザーにインポート/エクスポートする場合にのみ意味があります。

## 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を設定すると、ホストシステムがゲストシステムにブロック書き込み完了を通知する方法に影響します。デフォルトのキャッシングなしは、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ときに書き込みが完了したことをゲストシステムに通知し、ホストのページキャッシングを無視することを意味します。これにより、安全性と速度のバランスがとれます。

Proxmox の VE バックアップマネージャが VM のバックアップを実行する際にディスクをスキップするように設定するには  
そのディスクにはバックアップオプションはありません。

Proxmox VE のストレージレプリケーション機構でレプリケーションジョブの開始時にディスクをスキップさせたい場合は、そのディスクにレプリケーションをスキップするオプションを設定します。Proxmox VE 5.0 では、レプリケーションにはディスクイメージが zfspool タイプのストレージ上にある必要があるため、VM にレプリケーションが設定されている場合にディスクイメージを他のストレージに追加すると、このディスクイメージのレプリケーションをスキップする必要があります。

## トリム/廃棄

ストレージがシンプロビジョニングをサポートしている場合 (Proxmox VE ガイドのストレージの章を参照)、ドライブの **Discard** オプションを有効にすることができます。Discard が設定され、TRIM が有効なゲスト OS<sup>(2)</sup> では、VM のファイルシステムがファイルを削除した後にブロックを未使用としてマークすると、コントローラはこの情報をストレージに伝え、ストレージはそれに応じてディスクイメージを縮小します。ゲストが TRIM コマンドを発行できるようにするには、ドライブの **Discard** オプションを有効にする必要があります。ゲストオペレーティングシステムによっては、**SSD Emulation** フラグを設定する必要があります。VirtIO Block ドライブの Discard は、Linux Kernel 5.0 以降を使用するゲストでのみサポートされていることに注意してください。

ドライブを回転ハードディスクではなくソリッドステートドライブとしてゲストに表示したい場合は、そのドライブに **SSD エミュレーション** オプションを設定できます。基盤となるストレージが実際に SSD でバックアップされている必要はありません、この機能はあらゆるタイプの物理メディアで使用できます。SSD エミュレーションは、VirtIO Block ドライブではサポートされていません。

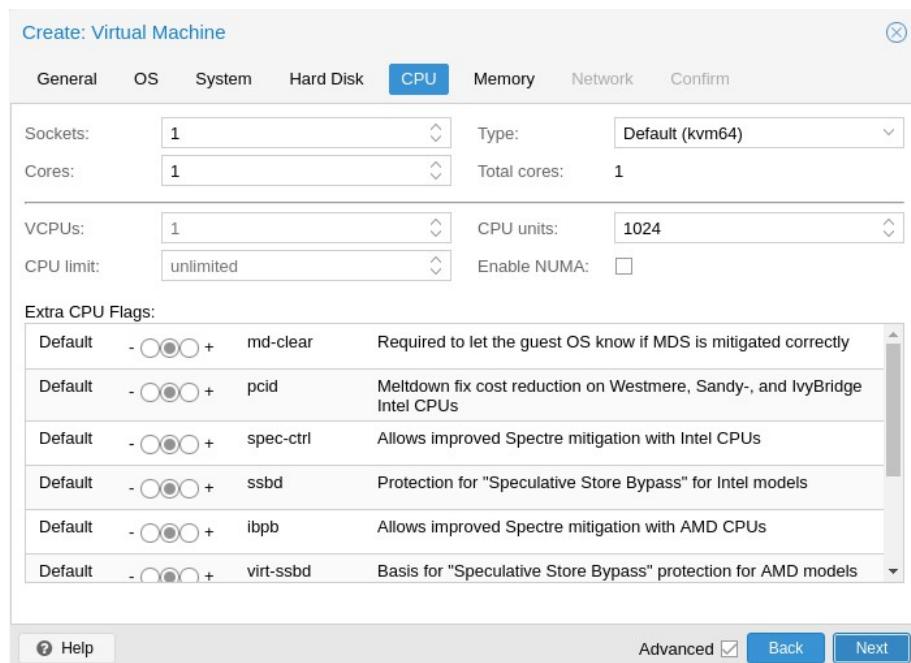
<sup>1</sup> 参照 この ベンチマーク をご覧ください。 詳細 [https://events.static.linuxfound.org/sites/events/files/slides/CloudOpen2013\\_Khoa\\_Huynh\\_v3.pdf](https://events.static.linuxfound.org/sites/events/files/slides/CloudOpen2013_Khoa_Huynh_v3.pdf)

<sup>2</sup> TRIM, UNMAP, and discard [https://en.wikipedia.org/wiki/Trim\\_%28computing%29](https://en.wikipedia.org/wiki/Trim_%28computing%29)

## IOスレッド

**IO Thread** オプションは、**VirtIO** コントローラでディスクを使用する場合、またはエミュレートされるコントローラのタイプが **VirtIO SCSI シングル** である場合に **SCSI** コントローラでディスクを使用する場合にのみ使用できます。**IO Thread** を有効にすると、QEMU は、メインイベントループまたは vCPU スレッドですべての I/O を処理するのではなく、ストレージコントローラごとに 1 つの I/O スレッドを作成します。利点の 1 つは、基礎となるストレージの作業分散と利用が向上することです。もう 1 つの利点は、メインスレッドも vCPU スレッドもディスク I/O によってブロックされないため、非常に I/O 負荷の高いホストワークロードのゲストでの待ち時間(ハング)が短縮されることです。

### 10.2.5 中央演算処理装置



CPUソケットは、PCマザーボード上の物理的なスロットで、CPUを差し込むことができます。このCPUには、独立した処理ユニットであるコアを1つ、または複数含めることができます。CPUソケットが1つで4コアであろうと、CPUソケットが2つで2コアであろうと、性能の観点からはほとんど関係ありません。しかし、ソフトウェアのライセンスによっては、マシンのソケット数に依存するものがあります。

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の用途に大きく依存します。仮想CPUを追加するごとに、QEMUはホスト・システム上に新しい実行スレッドを作成するからです。VMのワークロードについて確信が持てない場合は、通常、合計コア数を2に設定するのが安全です。

#### 備考

すべてのVMの全体のコア数がサーバーのコア数よりも大きい場合(例えば、8コアしかないマシンにそれぞれ4コア(=合計16コア)のVMを4つ配置する場合)は、まったく問題ありません。この場合、ホスト・システムは、標準的なマルチスレッド・アプリケーションを実行する場合と同様に、QEMUの実行スレッドをサーバー・コア間でバランスさせます。しかし、Proxmox VEは、物理的に利用可能な数よりも多くの仮想CPUコアでVMを起動することを防ぎます。

## リソース制限 cpulimit

仮想コア数に加えて、VMで使用可能な「ホストCPU時間」の合計もcpulimitオプションで設定できます。これはCPU時間をパーセントで表した浮動小数点値で、

1.0は100%に相当します。

2.5%から250。1つのプロセスが1つのコアをフルに使用する場合、CPU時間は100%になります。

の使用率です。4つのコアを持つVMがすべてのコアをフルに使用した場合、理論的には400%の使用率になります。QEMUはvCPUコア以外にVMペリフェラル用の追加スレッドを持つことができるため、実際の使用率はもう少し高くなる可能性があります。

この設定は、VMがいくつかのプロセスを並行して実行しているために複数のvCPUを持つ必要があるが、VM全体としてはすべてのvCPUを同時に100%で実行することはできない場合に役立ちます。

例えば、8つの仮想CPUを持つことで恩恵を受けられる仮想マシンがあるとして、その仮想マシンが8つのコアをフル稼働させることはできません。これを解決するには、cpulimitを4.0 (=400%) に設定します。これは、VMが8つのプロセスを同時に実行して8つすべての仮想CPUを完全に利用する場合、各vCPUは物理コアから最大50%のCPU時間を受け取ることを意味します。ただし、VMのワークロードが4つの仮想CPUしかフルに使用しない場合でも、物理コアから最大100%のCPU時間を受け取ることができ、合計で400%になります。

### 備考

VMは、その構成によっては、ネットワーキングやIOオペレーション、ライブマイグレーションなど、追加のスレッドを使用することができます。そのため、VMは仮想CPUが使用できる以上のCPU時間を使用するように表示されることがあります。VMが割り当てられたvCPUよりも多くのCPU時間を使用しないようになるには、cpulimitを総コア数と同じ値に設定します。

## CPU

cpuunits オプション（最近では CPU shares または CPU weight と呼ばれることがあります）を使用すると、VM が他の実行中の VM と比較してどれだけの CPU 時間を取得するかを制御できます。これは相対的な重みで、デフォルトは100（またはホストがレガシー cgroup v1 を使用している場合は1024）です。これを増やすと、VMはスケジューラによって、より低いウェイトの他のVMよりも優先されます。

たとえば、VM 100がデフォルトの100に設定され、VM 200が200に変更された場合、後者のVM 200は最初のVM 100の2倍のCPU帯域幅を受け取ることになります。

より詳しい情報は `man systemd.resource-control` を参照してください。

と CPUWeight を cpuunits 設定に変更します。参考文献や実装の詳細については、Notesセクションを参照してください。

### 親和性

アフィニティ・オプションを使用すると、VMのvCPUの実行に使用する物理CPUコアを指定できます。I/OなどのペリフェラルVMプロセスは、この設定の影響を受けません。CPUアフィニティはセキュリティ機能ではないことに注意してください。

CPUアフィニティを強制することは、場合によっては意味がありますが、複雑さとメンテナンスの手間の増加を伴います。例えば、後からVMを追加したい場合や、VMをCPUコアの少ないノードに移行したい場合などです。また、一部のCPUがフルに使用され、他のCPUがほとんどアイドル状態である場合、非同期でシステムのパフォーマンスが制限されやすくなります。

アフィニティはtaskset CLIツールで設定します。これは `man cpuset の List Format` にあるホスト CPU 番号 (`lscpu` を参照) を受け入れます。このASCII 10進数リストには、数値だけでなく数値範囲も含めることができます。例えば、アフィニティ0-1,8-11（拡張0,1,8,9,10,11）は、VMがこれら6つの特定のホストコアのみで実行されるようにします。

## CPUタイプ

QEMUは、486から最新のXeonプロセッサまで、さまざまな種類のCPUをエミュレートできます。プロセッサの世代が新しくなるごとに、ハードウェアアシスト3dレンダリング、乱数生成、メモリ保護などの新機能が追加されます。また、現在の世代は、バグやセキュリティの修正を伴うマイクロコードアップデートによってアップグレードできます。

通常、ホスト・システムのCPUに近いプロセッサ・タイプをVMに選択する必要があります。これは、ホストのCPU機能（CPUフラグとも呼ばれます）がVMで利用できることを意味します。完全に一致させたい場合は、CPUタイプをホストに設定すると、VMはホスト・システムと全く同じCPUフラグを持つことになります。

しかし、これには欠点があります。異なるホスト間でVMのライブマイグレーションを行いたい場合、VMは異なるCPUタイプや異なるマイクロコードバージョンを持つ新しいシステム上で終了するかもしれません。ゲストに渡されたCPUフラグが欠落していると、QEMUプロセスは停止します。これを解決するために、QEMUには独自の仮想CPUタイプがあり、Proxmox VEはデフォルトでこれを使用します。

バックエンドのデフォルトはkvm64で、基本的にすべてのx86\_64ホストCPUで動作します。新しいVMを作成する際のUIのデフォルトはx86-64-v2-AESで、Intelの場合はWestmereから始まるホストCPU、AMDの場合は少なくとも第4世代のOpteronが必要です。

要するに：

ライブマイグレーションにこだわらない場合や、すべてのノードが同じCPUと同じマイクロコードバージョンを持つ均質なクラスタの場合は、CPUタイプをホストに設定してください。

ライブマイグレーションとセキュリティを重視し、Intel CPUのみ、またはAMD CPUのみを使用している場合は、クラスタのCPUの世代が最も低いモデルを選択してください。

セキュリティなしのライブマイグレーションを重視する場合や、Intel/AMD混在のクラスタを使用する場合は、互換性の最も低い仮想QEMU CPUタイプを選択してください。

---

### 備考

IntelとAMDのホストCPU間のライブマイグレーションは、動作保証がありません。

---

QEMUで定義されているAMDとIntelのCPUタイプの一覧」も参照してください。

## QEMU CPUの種類

QEMUは、IntelとAMDの両方のホストCPUと互換性のある仮想CPUタイプも提供しています。

---

### 備考

仮想CPUタイプのSpectre脆弱性を緩和するには、関連するCPUフラグを追加する必要があります。

---

歴史的に、Proxmox VEのCPUモデルはkvm64で、Pentium 4レベルのCPUフラグが有効になっていました。

2020年夏、AMD、Intel、Red Hat、SUSEは共同で、x86-64ベースラインの上に3つのx86-64マイクロアーキテクチャ・レベルを定義し、モダン・フラグを有効にしました。詳細については、[x86-64-ABI仕様](#)を参照してください。

## 備考

CentOS 9のような新しいディストリビューションは、x86-64-v2フラグを最低要件としてビルトされています。

- *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 CPU フラグを追加: +cx16, +lahf-lm, +popcnt, +pni, +sse4.1, +sse4.2, +ssse3.
- *x86-64-v2-AES*: Intel CPU>= Westmere, AMD CPU>= Opteron\_G4 に対応。x86-64-v2 と比較して CPU フラグを追加: +aes。
- *x86-64-v3*: Intel CPU>= Broadwell、AMD CPU>= EPYC に対応。と比較したCPUフラグを追加しました。  
*x86-64-v2-AES*: +avx、+avx2、+bmi1、+bmi2、+f16c、+fma、+movbe、+xsave。
- *x86-64-v4*: Intel CPU>= Skylake、AMD CPU>= EPYC v4 Genoa に対応。x86-64-v3 と比較して CPU フラグを追加: +avx512f, +avx512bw, +avx512cd, +avx512dq, +avx512vl.

## カスタムCPUタイプ

設定可能な持つカスタムCPUタイプを指定できます。これらは

ファイル /etc/pve/virtual-guest/cpu-models.conf を参照してください。man cpu-models を参照してください。  
をご覧ください。

指定されたカスタムタイプは、/nodesのSys.Audit権限を持つユーザであれば誰でも選択できます。CLIまたはAPIを介してVMにカスタムCPUタイプを設定する場合、名前の前にcustom-を付ける必要があります。

## メルトダウン／スペクター関連のCPUフラグ

MeltdownとSpectreの脆弱性<sup>3</sup>に関連するCPUフラグがいくつかあり、VMのCPUタイプがデフォルトで有効になっている場合を除き、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、2つの条件を満たす必要があります：

- ホスト CPU がその機能をサポートし、それをゲストの仮想 CPU に伝える必要があります。
- ゲストオペレーティングシステムは、攻撃を軽減し、CPU機能を利用できるバージョンにアップデートする必要があります。

そうでない場合は、Web UIでCPUオプションを編集するか、VM設定ファイルで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用のいわゆる「マイクロコードアップデート」を提供する必要があります。影響を受けるすべての CPU が spec-ctrl をサポートするようにアップデートできるわけではないことに注意してください。

Proxmox VE ホストが脆弱かどうかを確認するには、root で以下のコマンドを実行します：

```
for f in /sys/devices/system/cpu/vulnerabilities/*; do echo "${f##*/}.-" `'  
cat "$f"); done
```

ホストがまだ脆弱かどうかを検出するコミュニティスクリプトも用意されています。<sup>4</sup>

<sup>3</sup> メルトダウン攻撃 <https://meltdownattack.com/>

<sup>4</sup> spectre-meltdown-checker <https://meltdown.ovh/>

## インテルプロセッサー

- **ピピッド**

これにより、ユーザー空間からカーネルメモリを効果的に隠すKPTI (*Kernel Page- Table Isolation*) と呼ばれるMeltdown (CVE-2017-5754) 緩和策のパフォーマンスへの影響が軽減されます。PCIDがなければ、KPTIはかなり高価なメカニズムです<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で次のコマンドを実行します：

```
# grep ' pcid ' /proc/cpuinfo
```

これがemptyを返さない場合、ホストのCPUはpcidをサポートしています。

- **スペックctrl**

Spectre v1 (CVE-2017-5753) および Spectre v2 (CVE-2017-5715) の修正を有効にするために必要です。サフィックスが -IBRS のインテル CPU モデルにはデフォルトで含まれています。サフィックスが -IBRS でないインテル CPU モデルでは、明示的にオンにする必要があります。更新されたホスト CPU マイクロコード (intel- microcode >= 20180425) が必要です。

- **エスエスピーーディー**

Spectre v4 (CVE-2018-3639) 修正を有効にするために必要です。どのインテル CPU モデルにもデフォルトでは含まれていません。すべてのインテル CPU モデルで明示的にオンにする必要があります。更新されたホスト CPU マイクロコード (intel- microcode >= 20180703) が必要です。

## AMDプロセッサー

- **イブビ**

Spectre v1 (CVE-2017-5753) および Spectre v2 (CVE-2017-5715) の修正を有効にするために必要です。サフィックスが -IBPB の AMD CPU モデルにはデフォルトで含まれています。IBPBサフィックスがないAMD CPUモデルでは、明示的にオンにする必要があります。ゲスト CPU で使用する前に、ホスト CPU のマイクロコードがこの機能をサポートする必要があります。

- **ヴィルトエスピーーディー**

Spectre v4 (CVE-2018-3639) 修正を有効にするために必要です。どのAMD CPUモデルにもデフォルトでは含まれていません。すべての AMD CPU モデルで明示的にオンにする必要があります。ゲストの互換性を最大化するため、amd-ssbd も提供されている場合でも、ゲストに提供する必要があります。これは物理 CPU には存在しない仮想機能であるため、"ホスト" CPU モデルを使用する場合は明示的に有効にする必要があることに注意してください。

- **amd-ssbd**

Spectre v4 (CVE-2018-3639) 修正を有効にするために必要です。どのAMD CPUモデルにもデフォルトでは含まれていません。すべての AMD CPU モデルで明示的にオンにする必要があります。これは、virt- よりも高いパフォーマンスを提供するため、これをサポートするホストは、可能であれば常にこれをゲストに公開する必要があります。

- **amd-no-ssb**

ホストが Spectre V4 (CVE-2018-3639) に対して脆弱ではないことを示すために推奨されます。どの AMD CPU モデルにもデフォルトで含まれていません。将来的なハードウェア世代の CPU は CVE-2018- 3639 に対して脆弱ではないため、amd-no-ssb を公開することで、その緩和策を有効にしないようにゲストに伝える必要があります。これは virt-ssbd と amd-ssbd とは相互に排他的です。

<sup>5</sup> PCID は現在、x86 のパフォーマンス/セキュリティ上重要な機能です <https://groups.google.com/forum/m/#topic/mechanical-sympathy/L9mHTbeQLNU>

## エヌユーエムエー

オプションで、VMでNUMA<sup>6</sup>アーキテクチャをエミュレートすることもできます。NUMAアーキテクチャの基本は、すべてのコアで利用可能なグローバル・メモリ・プールを持つ代わりに、メモリを各ソケットに近いローカル・バンクに分散させることを意味します。これにより、メモリバスがボトルネックにならなくなり、速度が向上します。お使いのシステムがNUMAアーキテクチャ<sup>7</sup>の場合、ホストシステム上でVMリソースを適切に分配できるようになるため、このオプションを有効にすることをお勧めします。このオプションは、VMのコアやRAMをホットプラグする場合にも必要です。

NUMAオプションを使用する場合は、ソケット数をホストシステムのノード数に設定することをお勧めします。

## vCPUホットプラグ

最近のオペレーティング・システムでは、実行中のシステムでCPUをホットプラグしたり、ある程度までホットアンプラグしたりする機能が導入されています。仮想化によって、このようなシナリオで実際のハードウェアが引き起こす（物理的な）問題の多くを回避することができます。それでも、これはかなり新しく複雑な機能なので、その使用は絶対に必要な場合に限られるべきです。ほとんどの機能は、他の、よくテストされた、より複雑でない機能で再現できます。

Proxmox VEでは、プラグインされたCPUの最大数は常にコア\*ソケットです。このコア数未満のCPUでVMを起動するには、**vcpus**設定を使用します。

現在、この機能はLinuxでのみサポートされており、3.10より新しいカーネルが必要で、4.7より新しいカーネルが推奨されています。

以下のように udev ルールを使用して、新しい CPU をゲストのオンラインとして自動的に設定できます：

```
SUBSYSTEM=="cpu", ACTION=="add", TEST=="online", ATTR{online}=="0", ATTR{!online}="1"
```

これを /etc/udev/rules.d/ の下に .rules で終わるファイルとして保存します。

注意: CPU ホットリムーブはマシン依存であり、ゲストの協力が必要です。削除コマンドは CPU の削除が実際に行われることを保証するものではなく、通常は x86/amd64 の ACPI のようなターゲットに依存するメカニズムを使用してゲスト OS に転送されるリクエストです。

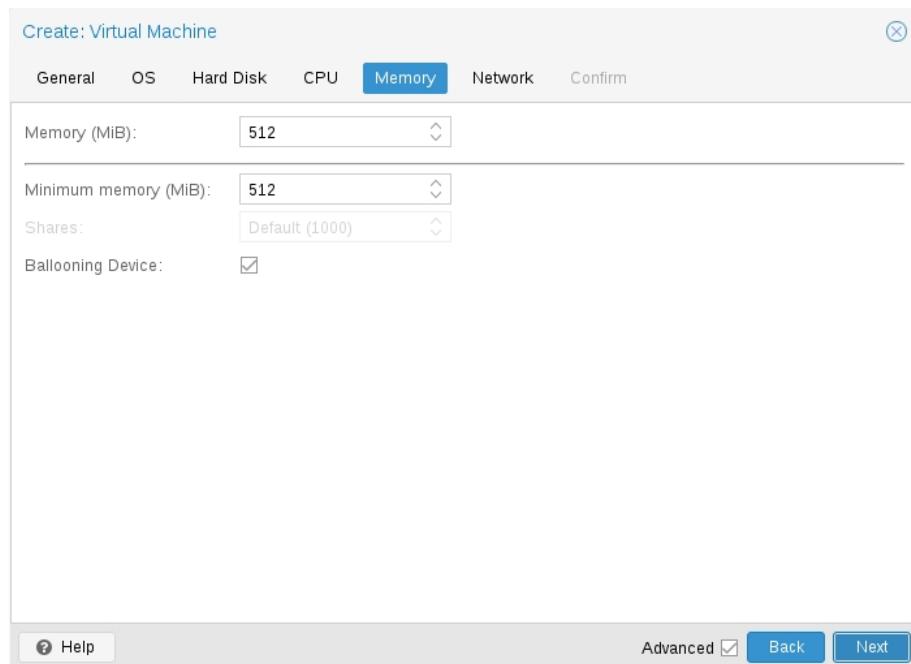
## 10.2.6 メモリー

各VMには、固定サイズのメモリを設定するか、ホストの現在のRAM使用量に基づいて動的にメモリを割り当てるようにProxmox VEに依頼するオプションがあります。

<sup>6</sup> [https://en.wikipedia.org/wiki/Non-uniform\\_memory\\_access](https://en.wikipedia.org/wiki/Non-uniform_memory_access)

<sup>7</sup> numactl --hardware| grep availableコマンドが複数のノードを返す場合、ホストNUMAアーキテクチャです。

## 固定メモリ割り当て



メモリと最小メモリを同じ量に設定した場合、Proxmox VEは単純に指定した量をVMに割り当てます。

固定メモリサイズを使用している場合でも、バルーニングデバイスはVMに追加されます。一般的に、バルーンは有効のままにしておく必要がありますが、(デバッガ目的などで)無効にしたい場合は、単に **Ballooning Device** のチェックを外すか、または

バルーン: 0

を設定します。

## 自動メモリ割り当て

最小メモリをメモリより低く設定すると、Proxmox VEは指定した最小量が常にVMで利用可能であることを確認し、ホスト上のRAM使用率が特定のターゲットパーセンテージを下回る場合、指定した最大メモリまで動的にゲストにメモリを追加します。目標パーセンテージのデフォルトは80%で、[ノードオプション](#)で設定できます。

ホストのRAMが不足すると、VMはホストにメモリを解放し、必要に応じて実行中のプロセスをスワップし、最後の手段としてoom killerを起動します。ホストとゲスト間のメモリの受け渡しは、ゲスト内部で動作する特別なバルーンカーネルドライバを介して行われ、ホストからメモリページを取得または解放します。<sup>8</sup>

複数のVMが自動割り当て機能を使用する場合、各VMが占有すべき空きホスト・メモリの相対量を示すシェア係数を設定することができます。例えば、4つのVMがあり、そのうち3つはHTTPサーバ、最後の1つはデータベースサーバを実行しているとします。ホストは80%のRAM使用率を目標に設定されています。より多くのデータベース・ブロックをデータベース・サーバのRAMにキャッシュするために、予備のRAMが利用可能な場合にデータベースVMを優先したいとします。このため、データベースVMに3000のSharesプロパティを割り当て、他のVMはデフォルト設定の1000にします。ホスト・サーバには32GBの

<sup>8</sup> バルーンドライバーの内部構造については、<https://rwmj.wordpress.com/2010/07/17/-virtio-balloon/>に説明があります。

現在 16GB を使用しており、設定された最小メモリ量に加え、 $32 * 80/100 - 16 = 9\text{GB}$  の RAM が VM に割り当てられます。データベース VM には  $9 * 3000 / (3000 + 1000 + 1000) = 4.5\text{ GB}$  の RAM が、HTTP サーバにはそれぞれ 1.5 GB の RAM が追加されます。

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、バルーンカーネルドライバーが含まれています。Windows OS の場合、バルーンドライバは手動で追加する必要があり、ゲストの速度を低下させる可能性があるため、重要なシステムでの使用はお勧めしません。

VMにRAMを割り当てる場合、ホストが使用できるRAMを常に1GB残しておくのが良い経験則です。

## 10.2.7 メモリの暗号化

### AMD SEV

SEV（セキュア暗号化仮想化）は、AES-128暗号化とAMDセキュア・プロセッサを使用して、VMごとのメモリ暗号化を可能にします。

また、SEV-ES（Secure Encrypted Virtualization - Encrypted State）は、CPUレジスタの内容をすべて暗号化し、ハイパーテーパー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・ベースの整合性攻撃を防止する試みも行っています。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。

#### ホストの条件

- 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ています。
- SEV-SNP にはホストカーネルのバージョン 6.11 以上が必要です。
- ホスト・マシンのBIOS設定でAMDメモリー暗号化を設定します。
- デフォルトで有効になっていない場合は、カーネルパラメータに「kvm\_amd.sev=1」を追加します。
- ホスト(SME)上のメモリーを暗号化したい場合は、カーネル・パラメーターに "mem\_encrypt=on" を追加 <https://www.kernel.org/doc/Documentation/x86/amd-memory-encryption.txt> を参照。
- SWIOTLBは <https://github.com/AMDESE/AMDSEV#faq-4> を参照。

ホスト上でSEVが有効になっているかどうかを確認するには、dmesgでsevを検索し、kvm\_amdのSEVカーネルパラメータを出力します：

```
# dmesg | grep -i sev
[...] ccp 0000:45:00.1: sev 有効
[...] ccp 0000:45:00.1: SEV API: <buildversion> [...] SEV supported:
<number> ASIDs
[...] SEV-ES supported: <number> ASIDs # cat
/sys/module/kvm_amd/parameters/sev Y
```

#### ゲストの条件

- edk2-OVMF
- Q35を使用することをお勧めします。
- ゲストOSにSEVサポートが含まれている必要があります。

#### 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間違っています。
- スナップショットやライブマイグレーションのようなメモリの保存や復元を伴う操作はまだ機能していないか、**攻撃可能**です。
- PCIバススルーパスはサポートされていません。
- SEV-ESとSEV-SNPは非常に実験的なものです。
- EFIディスクはSEV-SNPではサポートされていません。
- SEV-SNPでは、VM内のrebootコマンドは単にVMをシャットダウンするだけです。

#### 構成例（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="は必要ありません）。使用可能なオプションは std、es、snp です。

QEMUポリシーパラメータは、**no-debug**パラメータと**no-key-sharing**パラメータで計算されます。typeが**es**の場合、ポリシービット2は1に設定され、SEV-ESが有効になります。policy-bit 3 (nosend) はマイグレーション攻撃を防ぐために常に 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、以下を参照してください：[AMD SEV API仕様 第3章](#)

**kernel-hashes**オプションは、古いOVMFイメージやカーネル/initrdを測定しないゲストとの後方互換性のため、デフォルトではオフになっています。  
<https://lists.gnu.org/archive/html/qemu-devel/2021-11/msg02598.html>を参照してください。

#### SEVがVMで動作しているかの確認

方法1 - dmesg:

出力はこのようになります。

```
# dmesg | grep -i sev
AMDメモリー暗号化機能がアクティブに: SEV
```

方法2 - MSR 0xc0010131 (MSR\_AMD64\_SEV):

出力は1でなければなりません。

```
# apt install msr-tools #
modprobe msr
# rdmsr -a 0xc0010131 1
```

#### 設定例（SEV-SNP）：

```
# qm set <vmid> -amd-sev type=snp,allow-smt=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 に設定してこれを無効にすると、VM を実行するには、ホスト上で SMT を無効にする必要があります。

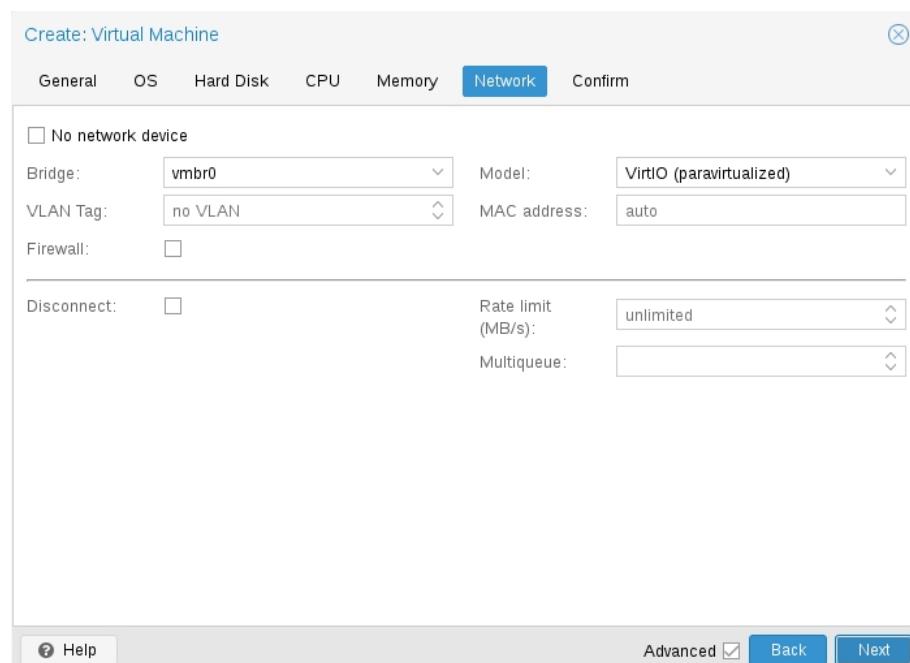
VM 内で SEV-SNP が動作しているか確認します。

```
# dmesg | grep -i snp
メモリ暗号化機能が有効です: AMD SEV SEV-ES SEV-SNP SEV: SNP CPUIDテーブルを使用しています。
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 Secure Nested Paging Firmware ABI 仕様

## 10.2.8 ネットワーク機器



各VMは、4つの異なる多数のネットワーク・インターフェイス・コントローラ (NIC) を持つことができます:

- Intel E1000がデフォルトで、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。
- の準仮想化 NIC を使用する必要があります。すべての VirtIO デバイスと同様に、ゲスト OS には適切なドライバがインストールされている必要があります。
- Realtek 8139は旧式の100MB/sネットワークカードをエミュレートしているため、旧式のオペレーティングシステム（2002年以前にリリースされたもの）をエミュレートする場合にのみ使用してください。

- **vmxnet3** は別の準仮想化デバイスで、別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します。

Proxmox VEは各NICにランダムな**MACアドレス**を生成し、VMがイーサネットネットワーク上でアドレス指定できるようにします。

VMに追加したNICは、2つの異なるモデルのいずれかに従います：

- デフォルトの**ブリッジモード**では、各仮想NICはタップデバイス(イーサネットNICをシミュレートするソフトウェアループバックデバイス)によってホスト上でバックアップされます。このタップデバイスは、Proxmox VEではデフォルトでvmbr0というブリッジに追加されます。このモードでは、VMはホストがあるイーサネットLANに直接アクセスできます。
- 代替**NATモード**では、各仮想NICはQEMUユーザーネットワーキングスタックとのみ通信し、内蔵ルーターとDHCPサーバーがネットワークアクセスを提供します。この内蔵DHCPは、プライベート10.0.2.0/24範囲のアドレスを提供します。NATモードはブリッジモードよりもはるかに遅いので、テストにのみ使用してください。このモードはCLIまたはAPI経由でのみ利用可能で、Web UI経由では利用できません。

また、**ネットワークデバイスなし**を選択することで、VMの作成時にネットワークデバイスの追加を省略することもできます。

各 VM ネットワーク・デバイスの **MTU** 設定を上書きできます。mtu=1 オプションは、MTU 値が基礎となるブリッジから継承される特殊なケースを表します。このオプションは、**VirtIO** ネットワークデバイスでのみ使用できます。

## マルチキュー

VirtIO ドライバを使用している場合、オプションで **Multiqueue** オプションを有効にできます。このオプションにより、ゲスト OS は複数の仮想 CPU を使用してネットワー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ネットワークキューはホストカーネルに渡され、キューはvhostドライバによって生成されたカーネルスレッドによって処理されます。このオプションを有効にすると、各 NIC に対して複数のネットワーク・キューをホスト・カーネルに渡すことができます。

**Multiqueue** を使用する場合は、ゲストの vCPU 数と同じ値に設定することをお勧めします。vCPU 数は、ソケット数×VM に設定されたコア数であることを忘れないでください。また、この ethtool コマンドを使用して、VM の各 VirtIO NIC の多目的チャネル数を設定する必要があります：

```
ethtool -L ens1 結合 X
```

ここで、XはVMのvCPU数です。

WindowsゲストをMultiqueue用に設定するには、Redhat VirtIO Ethernetアダプタドライバをインストールし、次のようにNICの設定を変更します。デバイスマネージャーを開き、"Network adapters" で NIC を右クリックし、"Properties" を選択します。次に「詳細設定」タブを開き、左側のリストから「受信側スケーリング」を選択します。有効」になっていることを確認してください。次に、リストの「Maximum number of RSS Queues」に移動し、VMのvCPU数に設定します。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。

Multiqueue パラメータを 1 より大きな値に設定すると、トラフィックが増加するにつれてホスト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 ガルーティー、リバースプロキシ、または長時間のポーリングを行うビギーな HTTP サーバーとして実行されている場合など、VM が多数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お勧めします。

## 10.2.9 ディスプレイ

QEMU は、数種類の VGA ハードウェアを仮想化できます。いくつか例を：

- デフォルトの **std** は、Bochs VBE 拡張を持つカードをエミュレートします。
- cirrus**、これはかつてデフォルトでした、それはすべての問題を持つ非常に古いハードウェアモジュールをエミュレートします。このディスプレイタイプは、本当に必要な場合<sup>9</sup>、例えばWindows XP以前を使用している場合にのみ使用してください。
- vmware**は、VMWare SVGA-II互換アダプタです。
- qxl** は QXL 準仮想化グラフィックカードです。これを選択すると、VMの**SPICE**（リモート・ビューワ・プロトコル）も有効になります。
- Virtio-g1**（しばしばVirGLと呼ば）は、VM内で使用するための仮想3D GPUで、特別な（高価な）モデルやドライバを必要とせず、ホストGPUを完全にバインドすることなく、ワークロードをホストGPUにオフロードすることができます。

### 備考

VirGLのサポートにはいくつかの追加ライブラリが必要ですが、比較的大きいためデフォルトではインストールされず、またすべてのGPUモデル/ベンダーでオープンソースとして利用できるわけではありません。ほとんどのセットアップでは、次のようにするだけです: `apt install libgl1-mesa-glx`

メモリオプションを設定することで、仮想 GPU に与えるメモリ量を編集できます。これにより、特に SPICE/QXL で、VM 内でより高い解像度を実現できます。

メモリは表示デバイスによって予約されるため、SPICE でマルチモニタ・モード（デュアルモニタの場合は `qxl2` など）を選択すると、いくつかの意味があります：

- Windowsは各モニタにデバイスを必要とするため、ostypeがWindowsの場合、Proxmox VEはモニタごとに追加のデバイスをVMに与えます。各デバイスは、指定された量のメモリを取得します。
- Linux VMでは、常に多くの仮想モニタを有効にすることができますが、マルチモニタ・モードを選択すると、デバイスに与えられるメモリがモニタの数に乗算されます。

ディスプレイタイプとしてserialXを選択すると、VGA出力が無効になり、Webコンソールは選択したシリアルポートにリダイレクトされます。この場合、ディスプレイメモリの設定は無視されます。

### VNCクリップボード

```
# 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することができます。

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。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 のゲストツールをインストールしたら、VNC のクリップボード機能を使うことができます (noVNC コンソールパネルなど)。ただし、SPICE、virtio、virglを使用している場合は、使用するクリップボードを選択する必要があります。これは、`clipboard 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 の使用は有害と考えられます。

## 10.2.10 USBバススルーパススルーデバイス

USBバススルーパススルーデバイスには2種類あります:

- ホストUSBバススルーパススルーデバイス
- SPICE USBバススルーパススルーデバイス

ホストUSBバススルーパススルーデバイスは、VMにホストのUSBデバイスを与えることで機能します。これは、ベンダーIDやプロダクトIDを経由するか、ホスト・バスやポートを経由して行われます。

vendor/product-idは次のようにになります: **0123:abcd**、ここで0123はベンダーのID、abcdは製品のIDです。

バス/ポートは次のようにになります: **1-2.3.4**で、1がバス、2.3.4がポートパスです。これはホストの物理ポートを表します（usbコントローラの内部順序に依存します）。

VMの起動時にVMのコンフィグレーションにデバイスが存在しても、そのデバイスがホストに存在しない場合、VMは問題なく起動できます。ホストでデバイス/ポートが利用可能になると、すぐにそのデバイスが通過します。



### 警告

このようなUSBバススルーパススルーデバイスを使用することは、VMを別のホストにオンラインで移動できないことを意味します。

バススルーパススルーデバイスの2つ目のタイプは、SPICE USBバススルーパ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VMにローカルのUSBデバイスを動的に渡すことができます。これは、入力デバイスやハードウェアドングルを一時的にリダイレクトするのに便利です。

また、クラスタレベルでデバイスをマッピングすることも可能で、HAで適切に使用できるようにしたり、ハードウェアの変更を検出したり、非rootユーザができるようにしたりできます。[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ために、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として知られるファームウェアは、VMの起動時に最初のステップとして実行されます。BIOSは、基本的なハードウェアの初期化を行い、オペレーティング・システムにファームウェアとハードウェアへのインタフェースを提供します。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セットアップに適しています。

オペレーティングシステムによっては(Windows 11など)、UEFI互換の実装を使う必要があるかもしれません。そのような場合、代わりにオープンソースのUEFI実装である[OVMF](#)を使う必要があります。<sup>10</sup>

例えば、VGAバススルーパススルーデバイスを行いたい場合などです。<sup>11</sup>

OVMFを使いたい場合、考慮すべき点がいくつかあります:

ブート順などを保存するには、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ます。

このようなディスクは以下のコマンドで作成できます:

<sup>10</sup> OVMFプロジェクト参照 <https://github.com/tianocore/tianocore.github.io/wiki/OVMF>

<sup>11</sup> アレックス・ウィリアムソンのブログ <https://vfio.blogspot.co.at/2014/08/primary-graphics-assignment-without-vga.html>。

```
# qm set <vmid> -efidisk0 <storage>:1,format=<format>,efitype=4m,pre-'  
enrolled-keys=1
```

ここで、<storage> はディスクを置きたいストレージ、<format> はストレージがサポートするフォーマットです。または、Web インターフェースから、VM のハードウェア・セクションにある *Add → EFI Disk* を使用して、このような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。新しい VM の場合、これは常に 4m であるべきです。なぜなら、これはセキュア・ブートをサポートしており、将来の開発をサポートするために割り当てられる容量が多いからです（GUI ではこれがデフォルトです）。

**pre-enroll-keys** は、efidisk にディストリビューション固有のセキュアブートキーと Microsoft 標準セキュアブートキーを事前にロードするかどうかを指定します。また、デフォルトでセキュアブートを有効にします（ただし、VM 内の OVMF メニューで無効にすることもできます）。

---

#### 備考

既存の VM (2m の efidisk を使用している) でセキュアブートの使用を開始する場合は、efidisk を再作成する必要があります。ためには、古い efidisk を削除し (`qm set <vmid> -delete efidisk0`)、前述の方法で新しい efidisk を追加します。これにより、OVMFメニューで行ったカスタム設定がリセットされます！

---

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（ブート中にESCボタンを押すと表示されます）でクライアントの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 ト拉斯ティッド・プラットフォーム・モジュール (TPM)

ト拉斯ティッド・プラットフォーム・モジュール (Trusted Platform Module) は、暗号化キーなどの秘密データを安全に保存し、システムの起動を検証するための耐タンパ一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、このようなデバイスをマシン（物理または仮想）に接続する必要があります。

TPMは、`tpmstate`ボリュームを指定して追加します。これはefidiskに似た働きをしますが、一度作成すると変更（削除のみ）。以下のコマンドで追加できます：

```
# qm set <vmid> -tpmstate0 <storage>:1,version=<version>.
```

ここで、<storage> は状態を置きたいストレージ、<version> は v1.2 または v2.0 のいずれかです。ウェブ・インターフェイスから、VMのハードウェア・セクションで *Add → TPM State* を選択して追加することもできます。

v2.0 TPM仕様の方が新しく、サポートが充実しているため、v1.2 TPMを必要とする特定の実装がない限り、そちらを優先すべきです。

---

#### 備考

物理的なTPMと比較すると、エミュレートされたTPMにはセキュリティ上の利点はありません。TPMのポイントは、TPM仕様の一部として指定されたコマンドを除いて、その上のデータを簡単に変更できることです。エミュレートされたデバイスの場合、データの保存は通常のボリューム上で行われるため、そのボリュームにアクセスできる人なら誰でも編集できる可能性があります。

---

## 10.2.13 VM間共有メモリ

VM間共有メモリーデバイス(ivshmem)を追加すること、ホストとゲスト間、または複数のゲスト間でメモリーを共有することができます。

このようなデバイスを追加するには、qm:

```
# qm set <vmid> -ivshmem size=32,name=foo
```

サイズの単位はMiBです。ファイルは /dev/shm/pve-shm-\$name (デフォルト名は vmid) 配下に配置されます。

### 備考

現在のところ、デバイスを使用しているVMがシャットダウンまたは停止すると、デバイスはすぐに削除されます。オープン・コネクションは維持されますが、全く同じデバイスへの新規接続はできなくなります。

このようなデバイスのユースケースは、ホストとゲスト間で高性能、低遅延のディスプレイミラーリングを可能にするLooking Glass<sup>12</sup>プロジェクトです。

## 10.2.14 オーディオ機器

オーディオデバイスを追加するには、以下のコマンドを実行します:

```
qm set <vmid> -audio0 device=<デバイス>.
```

対応オーディオデバイスは以下の通りです:

- ich9-intel-hda: Intel HD オーディオコントローラ、ICH9 をエミュレート
- intel-hda: Intel HD オーディオコントローラ、ICH6 をエミュレート
- AC97: Audio Codec '97、Windows XPのような古いOSに便利です:
- スパイズ
- 皆無

spiceバックエンドはSPICEと組み合わせて使用することができ、noneバックエンドはVM内のオーディオデバイスがソフトウェアの動作に必要な場合に便利です。ホストの物理オーディオ・デバイスを使用するには、デバイス・バススルーを使用します (PCIバススルーとUSBバススルーを参照)。MicrosoftのRDPのようないリモートプロトコルには、サウンドを再生するオプションがあります。

<sup>12</sup> Looking Glass: <https://looking-glass.io/>

## 10.2.15 VirtIO RNG

RNG (Random Number Generator) は、システムにエントロピー（ランダム性）を提供するデバイスです。仮想ハードウェア RNG を使用して、ホストシステムからゲスト VM にエントロピーを提供することができます。これは、特にゲストのブートプロセス中に、ゲスト内のエントロピー飢餓問題（十分なエントロピーが利用できず、システムが遅くなったり問題が発生したりする状況）を回避するのに役立ちます。

VirtIOベースのエミュレートされたRNGを追加するには、次のコマンドを実行します：

```
qm set <vmid> -rng0 source=<ソース>[,max_bytes=X,period=Y]。
```

source はエントロピーがホスト上のどこから読み込まれるかを指定：

- /dev/urandom: ノンブロッキングカーネルエントロビープール（推奨）
- /dev/random: カーネルプールをブロック（推奨されません。ホストシステムでエントロピーが枯渇する可能性があります）
- /dev/hwrng: ホストに接続されたハードウェア RNG を通過させます（複数の RNG が利用可能な場合、/sys/devices/virtual/misc/hw\_random/rng\_current で選択されたものが使用されます）。

max\_bytes と period パラメータで制限を指定することができ、これらはミリ秒単位で1周期あたりの max\_bytes として読み込まれます。しかし、これは線形関係を表すものではありません：1024B/1000ms は、1 秒のタイマーで最大 1 KiB のデータが利用可能になることを意味し、1 秒の間に 1 KiB がゲストにストリーミングされることを意味しません。そのため、周期を短くすることで、より速い速度でゲストにエントロピーを注入することができます。

デフォルトでは、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。ゲストがホストのリソースを使いように、常にリミッターを使用することをお勧めします。必要であれば、max\_bytes に 0 を指定することで、すべての制限を無効にすることができます。

## 10.2.16 ヴァーチョフ

Virtiofs は仮想環境用に設計された共有ファイルシステムです。VM 内にマウントすることで、ホスト上で利用可能なディレクトリツリーを共有することができます。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様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、[virtiofsd](#) デーモンをバックグラウンドで実行する必要があります。これは、virtiofs マウントを使用して VM を起動する際に Proxmox VE で自動的に行われます。

カーネル >=5.4 の Linux VM はデフォルトで virtiofs をサポートしますが（[virtiofs カーネルモジュール](#)）、いくつかの機能はより新しいカーネルを必要とします。

virtiofs を使用するには、virtiofsd が Proxmox VE ホストにインストールされていることを確認します：

```
apt install virtiofsd
```

Windows VM で virtiofs を利用する方法については、[ガイド](#)があります。

## 既知の制限

- virtiofsd がクラッシュすると、そのマウントポイントは VM が完全に停止するまで VM 内でハングします。
- virtiofsd が応答しない場合、到達不能な NFS と同様に、VM のマウントがハングすることがあります。
- メモリホットプラグは virtiofs との組み合わせでは動作しません (アクセスもハングします)。
- ライブマイグレーション、スナップショット、ハイバネートなどのメモリー関連の機能は virtiofs デバイスでは使用できません。
- Windows は virtiofs のコンテキストで ACL を理解できません。そのため、Windows VM に対して ACL を公開しないでください。

## 共有ディレクトリのマッピングの追加

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](#)のセクションで説明されているように、`pvesh`で直接APIを使用することができます：

```
pvesh create /cluster/mapping/dir --id dir1 \
--map node=node1,path=/path/to/share1 \
--map node=node2,path=/path/to/share2 \
```

## VMへのvirtiofsの追加

virtiofs を使用してディレクトリを共有するには、パラメータ `virtiofs<N>` (`N` は 0 から 9 の間) を VM config に追加し、リソースマッピングで設定されたディレクトリ 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`、`auto` (デフォルト: `auto`) のいずれかに設定できます。異なるキャッシュモードがどのように動作するかについては、「[キャッシュモード](#)」のセクションを参照してください。

`virtiofsd` は ACL と xattr パススルーをサポートします (`expose-acl` と `expose-xat` で有効にできます)。オプション) がサポートされている場合、ゲストが ACL と xattrs にアクセスできるようにしますが、ゲストのファイルシステムと互換性がなければなりません (たとえば、ほとんどの Linux ファイルシステムは ACL をサポートしていますが、Windows ファイルシステムはサポートしていません)。

`expose-acl` オプションは自動的に `expose-xattr` を意味します。

`expose-acl` が 1 に設定されている場合、`expose-xattr` は 0 になります。

`virtiofs` に `O_DIRECT` フラグを尊重させたい場合は、`direct-io` パラメータを 1 に設定します (デフォルト: 0)。これはパフォーマンスを低下させますが、アプリケーションが独自のキャッシュを行う場合に便利です。

```
qm set <vmid> -virtiofs0 dirid=<dirid>,cache=always,direct-io=1 qm set <vmid> -virtiofs1 <dirid>,cache=never,expose-xattr=1
qm set <vmid> -virtiofs2 <dirid>,expose-acl=1
```

Linux カーネルの `virtiofs` ドライバーでゲスト VM に `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：

```
マウント -t virtiofs <ディレクトリID> <マウントポイント
```

`virtiofs` を持続的にマウントするには、`fstab` エントリーを作成します：

```
<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ために使用される名前）としても使用されます。

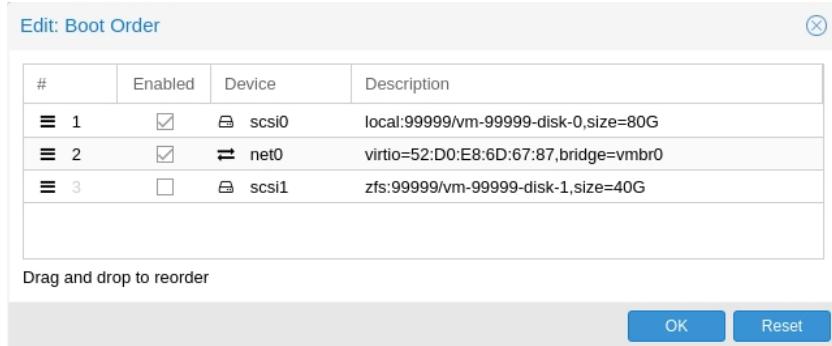
利用可能な virtiofsd パラメーターの詳細については、[GitLab virtiofsd プロジェクトページ](#)を参照してください。

## 10.2.17 デバイスの起動順序

QEMU は、どのデバイスからどの順序でブートすべきかをゲストに指示できます。これは、例えば、boot プロパティ経由で config で指定できます：

```
boot: order=scsi0;net0;hostpci0
```

この方法では、ゲストはまずディスク scsi0 からのブートを試み、失敗した場合は net0 からのネットワークブートを試み、それも失敗した場合は、最後に



通過した PCIe デバイス (NVMe の場合はディスクとみなされ、そうでない場合はオプション ROM に起動しようとしています) からのブートを試みます。

GUI 上では、ドラッグアンドドロップエディタを使ってブート順序を指定したり、チェックボックスを使って特定のデバイスのブートを有効または無効にすることができます。

### 備考

ゲストが OS を起動したりブートローダをロードするために複数のディスクを使用する場合、ゲストが起動できるようにするために、それら全てがブータブルとしてマークされていなければなりません (つまり、チェックボックスが有効になっているか、設定内のリストに表示されていなければなりません)。これは、最近の SeaBIOS と OVMF のバージョンは、ディスクがブータブルとマークされている場合にのみディスクを初期化するためです。

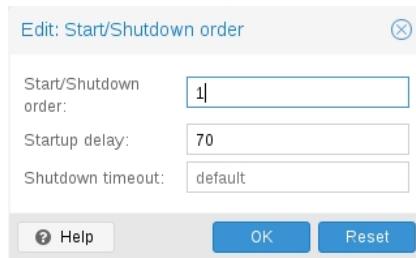
いずれにせよ、リストに表示されないデバイスやチェックマークが無効になっているデバイスでも、ゲストのオペレーティングシステムが起動して初期化されれば、ゲストはまだ利用可能です。ブータブルフラグはゲスト BIOS とブートローダーにのみ影響します。

## 10.2.18 仮想マシンの自動起動とシャットダウン

VMを作成した後、ホスト・システムのブート時にVMを自動的に起動させたいと思うでしょう。そのためには、ウェブ・インターフェイスのVMのオプション・タブからStart at bootオプションを選択するか、以下のコマンドで設定必要があります：

```
# qm set <vmid> -onboot 1
```

## スタートとシャットダウンの順序



例えば、あるVMが他のゲストシステムにファイアウォールやDHCPを提供している場合など、VMの起動順序を微調整したい場合があります。この場合、以下のパラメータを使用できます：

- スタート/シャットダウン順：**開始順序の優先順位を定義します。例えば、VMを最初に起動させたい場合は1に設定します。(シャットダウンには逆の起動順序を使用するため、起動順序が1のマシンは最後にシャットダウンされます)。ホスト上で複数のVMが同じ順序で定義されている場合、それらはさらにVMIDの昇順で並べられます。
- 起動遅延：**このVMが起動してから後続のVMが起動するまでの間隔を定義します。例えば、240秒待ってから他のVMを起動する場合は240に設定します。
- シャットダウンタイムアウト：**Proxmox VEがシャットダウンコマンドを発行した後、VMがオフラインになるまで待機する時間を秒単位で定義します。デフォルトでは、この値は180に設定されています。これは、Proxmox VEがシャットダウン要求を発行し、マシンがオフラインになるまで180秒待つことを意味します。タイムアウト後もマシンがオンラインの場合、強制的に停止されます。

### 備考

HAスタックによって管理されるVMは、現在のところ起動時開始と起動順序オプションに従いません。これらのVMは、HAマネージャ自身がVMの起動と停止を確実に行うため、起動とシャットダウンのアルゴリズムによってスキップされます。

開始/シャットダウン順序パラメータが設定されていないマシンは、常にパラメータが設定されているマシンの後に開始することに注意してください。さらに、このパラメータは同じホスト上で実行されている仮想マシン間でのみ適用され、クラスタ全体では適用されません。

ホストのブートと最初のVMのブートの間に遅延が必要な場合は、[Proxmox VEノード](#)管理のセクションを参照してください。

## 10.2.19 QEMUゲストエージェント

QEMU ゲストエージェントは VM 内で実行されるサービスで、ホストとゲスト間の通信チャネルを提供します。情報を交換するために使用され、ホストがゲストにコマンドを発行できるようにします。

例えば、VM サマリーパネルの IP アドレスはゲストエージェント経由で取得されます。

または、バックアップを開始する際に、ゲストエージェントを介して、`fs-freeze`を介して未処理の書き込みを同期するようにゲストに指示します。`fs-thaw`コマンド。

ゲストエージェントが正しく動作するためには、以下の手順をする必要があります：

- ゲストにエージェントをインストールし、実行されていることを確認します。
- Proxmox VEでエージェント経由の通信を有効にします。

## 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。このパッケージは通常 `qemu-guest-agent` という名前です。Windows では、[Fedora VirtIO ドライバー 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にできます。変更を有効にするには、VM の再スタートが必要です。

### QGAによる自動TRIM

*Run guest-trim* オプションを有効にすることができます。これを有効にすると、Proxmox VE は、ストレージにゼロを書き出す可能性のある以下の操作の後に、ゲストにトリムコマンドを発行します：

- ディスクを別のストレージに移動
- 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、未使用領域を解放するのに役立ちます。

#### 備考

Linux の ext4 では、重複した TRIM リクエストの発行を避けるためにメモリ内の最適化を使用するため、注意点があります。ゲストは基礎となるストレージの変更について知らないので、最初のゲストトリムだけが期待通りに実行されます。それ以降の TRIM は、次のリブートまで、それ以降に変更されたファイルシステムの一部のみを考慮します。

## バックアップ時のファイルシステムのフリーズと解凍

デフォルトでは、バックアップが実行されると、`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、一貫性が提供されます。

Windowsゲストでは、Windows VSS (Volume Shadow Copy Service) レイヤーをフックすることで一貫性のあるバックアップを処理するアプリケーションもあるため `fs-freeze` がそれを妨害する可能性があります。たとえば、一部の SQL Server で `fs-freeze` を呼び出すと、VSS が SQL Writer VSS モジュールを呼び出す引き金となり、差分バックアップの SQL Server バックアップチェーンが切断されることが確認されています。

このような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設定して、他の VSS ユーザーと干渉しない別の VSS バリアントを使用します。[Proxmox VE wiki](#)に詳細があります。
2. また、QGAオプションの `freeze-fs-on-backup` を 0 に設定することで、バックアップ時にフリーズと解凍のサイクルを発行しないように Proxmox VEを設定することもできます。これは、GUIから `Freeze/thaw guest filesystems on backup for consistency` オプションを使用して実行することもできます。



#### 重要

このオプションを無効にすると、一貫性のないファイルシステムでのバックアップにつながる可能性があります。したがって、ゲストで QEMU ゲストエージェントの設定を適応させることが望ましいオプションです。

## トラブルシューティング

### VMがシャットダウンしない

ゲストエージェントがインストールされ、実行されていることを確認します。

ゲストエージェントが有効になると、Proxmox VE はゲストエージェント経由でシャットダウンなどの電源コマンドを送信します。ゲストエージェントが実行されていない場合、コマンドは正しく実行できず、シャットダウンコマンドはタイムアウトになります。

## 10.2.20 SPICEの強化

SPICE Enhancements は、リモート・ビューワの性を向上させるオプション機能です。

GUIで有効にするには、仮想マシンの【オプション】パネルを開きます。CLIで有効にするには、次のコマンドを実行します：

```
qm-set <vmid> --spice_enhancements foldersharing=1,videostreaming=すべて
```

#### 備考

これらの機能を使用するには、仮想マシンのDisplayをSPICE (qxl) に設定する必要があります。

### フォルダ共有

ローカルフォルダをゲストと共有します。spice-webdavd デーモンをゲストにインストールする必要があります。これは、<http://localhost:9843> にあるローカル WebDAV サーバーを通じて共有フォルダーを利用可能にします。

Windowsゲストの場合、Spice WebDAVデーモンのインストーラは[SPICE公式ウェブサイト](#)からダウンロードできます。

ほとんどのLinuxディストリビューションには、spice-webdavdというパッケージがインストールされています。

Virt-Viewer (Remote Viewer)でフォルダを共有するには、[ファイル]→[環境設定]に進みます。共有するフォルダを選択し、チェックボックスを有効にします。

#### 備考

フォルダ共有は現在、Linux 版 Virt-Viewer でのみ機能します。



#### 注意

実験的です！ 現在、この機能は安定して動作していません。

### ビデオストリーミング

高速リフレッシュエリアはビデオストリームにエンコードされます。2つのオプションが：

- **すべて**：高速リフレッシュ領域はすべてビデオストリームにエンコードされます。

- **フィルタを使用します:** ビデオストリーミングを使用するかどうかを決定するために、追加のフィルタが使用されます（現在、小さなウィンドウ面のみがスキップされます）。

ビデオストリーミングを有効にすべきかどうか、また、どのオプションを選択すべきかについては、一般的な推奨はできません。特定の状況によって異なります。

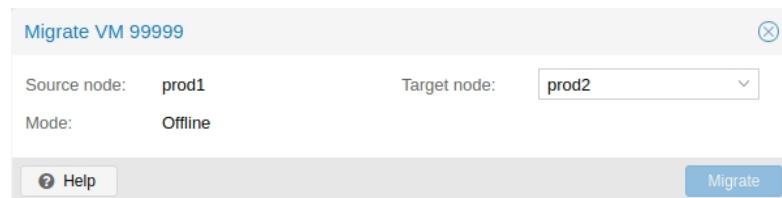
## トラブルシューティング

### 共有フォルダが表示されない

ゲストで WebDAV サービスが有効で実行されていることを確認します。Windows では *Spice webdav proxy* と呼ばれます。Linux では *spice-webdavd* という名前ですが、ディストリビューションによって異なります。

サービスが実行されている場合は、ゲストでブラウザで <http://localhost:9843> を開いて、WebDAV サーバを確認してください。SPICE セッションを再起動するのに役立ちます。

## 10.3 移住



クラスタがある場合、VMを別のホストに移行するには

```
# qm migrate <vmid> <target>
```

これには一般的に2つのメカニズムがあります。

- オンラインマイグレーション（別名ライブマイグレーション）
- オフライン移行

### 10.3.1 オンライン移行

VMが実行中で、ローカルにバインドされたリソースが設定されていない場合（パススルーされるデバイスなど）、`qm migration`コマンドエポケーションの`--online`フラグを使用してライブマイグレーションを開始できます。VMが実行中の場合、Webインターフェイスのデフォルトはライブマイグレーションです。

#### 仕組み

オンラインマイグレーションではまず、ターゲットホスト上で着信フラグを持つ新しいQEMUプロセスを起動し、ゲストvCPUを一時停止したまま基本的な初期化のみを実行し、ソース仮想マシンのゲストメモリとデバイス状態のデータストリームを待ちます。ディスクなどの他のリソースはすべて、VMの実行時状態移行が始まる前に共有されるか、すでに送信されているため、転送が必要なのはメモリコンテンツとデバイス状態のみです。

この接続が確立されると、ソースは非同期でメモリコンテンツをターゲットに送信し始めます。ソース上のゲストメモリが変更されると、それらのセクションはデータとマークされ、ゲストメモリデータを送信するために別のパスが作成されます。このループは、実行中のソースVMと着信中のターゲットVMとのデータ差が数ミリ秒で送信できるほど小さくなるまで繰り返されます。その後、ソースVMはユーザーやプログラムに気づかれることなく完全に一時停止し、残りのデータをターゲットに送信することができます。

#### 必要条件

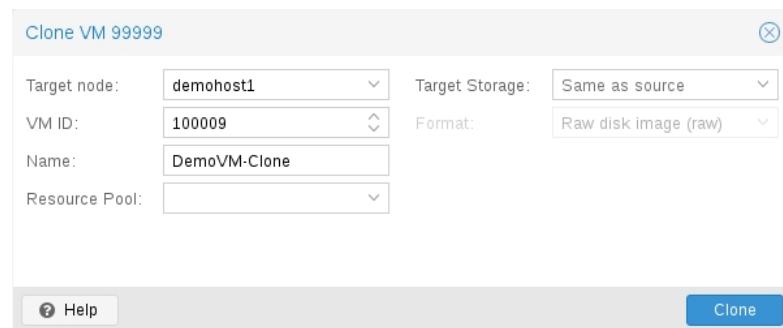
ライブマイグレーションが機能するためには、いくつかのことが必要です：

- VMには、移行できないローカルリソースがありません。例えば、PCIやUSBデバイスは、現在ライブマイグレーションをブロックしています。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できます。
- ホストは同じProxmox VEクラスタに配置されています。
- ホスト正常に（信頼できる）ネットワーク接続されています。
- ターゲットホストには、Proxmox VEパッケージと同じかそれ以上のバージョンが必要です。逆に動作する場合もありますが、保証できません。
- ホストは同じベンダーの同じようなCPUを使用しています。異なるベンダーであっても、設定されている実際のモデルとVMのCPUタイプによっては動作するかもしれません、保証はできません。

### 10.3.2 オフライン移行

ローカルリソースがある場合でも、すべてのディスクが両方のホストで定義されたストレージ上にある限り、VMをオフラインで移行することができます。マイグレーションは、オンラインマイグレーションと同様に、ネットワーク経由でターゲットホストにディスクをコピーします。ハードウェアパススルーの設定は、ターゲットホスト上のデバイスの場所に合わせる必要があるかもしれないことに注意してください。

## 10.4 コピーとクローン



VMのインストールは通常、OSベンダーのインストール・メディア（CD-ROM）を使って行われます。OSによっては、これは避けたい時間のかかる作業です。

同じタイプのVMを多数デプロイする簡単な方法は、既存のVMをコピーすることです。このようなコピーにはクローンという用語を使用し、リンククローンとフルクローンを区別します。

## フルクローン

このようなコピーの結果は独立したVMです。新しいVMは元のVMとストレージ・リソースを共有しません。

**ターゲット・ストレージ**を選択することができるので、VMを全く別のストレージに移行することができます。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は、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。

---

### 備考

フル・クローンは、すべてのVMイメージ・データを読み込んでコピーする必要があります。これは通常、リンクされたクローンを作成するよりもはるかに遅いです。

---

ストレージの種類によっては、特定のスナップショットをコピーすることができます。これはまた、最終的なコピーに元のVMからの追加スナップショットが含まれないことを意味します。

## リンクド・クローン

最近のストレージ・ドライバは、リンクされたクローンを高速に生成する方法をサポートしています。このようなクローンは書き込み可能なコピーで、初期内容は元のデータと同じです。リンクされたクローンの作成はほぼ瞬時に行え、最初は追加のスペースを消費しません。

新しい画像はまだ元の画像を参照しているため、リンクされていると呼ばれます。変更されていないデータブロックは元の画像から読み込まれますが、変更は新しい場所から書き込まれます（その後、読み込まれます）。この手法は`コピー・オン・ライト`と呼ばれます。

これには、元のボリュームが読み取り専用であることが必要です。Proxmox VEを使用すると、任意のVMを読み取り専用のテンプレートに変換できます。）このようなテンプレートは、後でリンクされたクローンを効率的に作成するために使用できます。

---

### 備考

リンクされたクローンが存在する間は、元のテンプレートを削除することはできません。

---

これはストレージの内部機能であるため、リンクされたクローンのターゲットストレージを変更することはできません。

**ターゲット・ノード・オプション**を使用すると、新しいVMを別のノード上に作成できます。唯一の制限は、VMが共有ストレージ上にあり、そのストレージがターゲット・ノードでも利用可能であることです。

リソースの競合を避けるため、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

VMをテンプレートに変換することができます。このようなテンプレートは読み取り専用で、リンクされたクローンを作成するために使用できます。

---

### 備考

ディスクイメージを変更することになるため、テンプレートを起動することはできません。テンプレートを変更したい場合は、リンクされたクローンを作成し、それを変更してください。

---

## 10.6 VMジェネレーションID

Proxmox VEは、仮想マシンのVirtual Machine Generation ID (*vmgenid*)<sup>(13)</sup>をサポートしています。これは、ゲストオペレーティングシステムが、バックアップの復元やスナップショットのロールバックなどのタイムシフトイベントを検出するために使用できます。

新しいVMを作成すると、*vmgenid*が自動的に生成され、設定保存されます。

既存のVMに*vmgenid*を作成して追加するには、特別な値'1'を渡してProxmox VEに自動生成させるか、UUID<sup>(14)</sup>を値として使用して手動で設定します：

```
# qm set VMID -vmgenid 1  
# qm set VMID -vmgenid 00000000-0000-0000-0000-000000000000
```

### 備考

既存のVMに*vmgenid*デバイスを最初に追加すると、スナップショットのロールバックやバックアップのリストアなどで、VMがこれを世代の変更と解釈するため、変更と同じ効果が生じる可能性があります。

*vmgenid*メカニズムが不要な場合は、VM作成時にその値に'0'を渡すか、またはコンフィギュレーションでそのプロパティを遡及的に削除することができます：

```
# qm set VMID -delete vmgenid
```

*vmgenid*の最も顕著なユースケースは、新しいMicrosoft Windowsオペレーティングシステムで、スナップショットのロールバック、バックアップのリストア、またはVM全体のクローン操作において、時間的制約のあるサービスやレプリケートサービス（データベースやドメインコントローラ<sup>(15)</sup>など）の問題を回避するために使用されます。

## 10.7 仮想マシンのインポート

海外のハイパーバイザーや他のProxmox VEクラスタから既存の仮想マシンをインポートするには、さまざまな方法があります：

- ESXiの特殊ストレージが提供するインポートコンテンツタイプを利用するネイティブインポートウィザードを使用します。
- ソースでバックアップを実行し、ターゲットでリストアします。この方法は、別のProxmox VEインスタンスから移行する場合に最適です。
- `qm` コマンドラインツールの OVF 専用インポートコマンドを使用します。

他のハイパーバイザーからProxmox VEにVMをインポートする場合は、[Proxmox VEの概念](#)に慣れることをお勧めします。

<sup>(13)</sup> 公式 *vmgenid* 仕様 [https://docs.microsoft.com/en-us/windows/desktop/hyperv\\_v2/virtual-machine-generation- 識別子](https://docs.microsoft.com/en-us/windows/desktop/hyperv_v2/virtual-machine-generation- 識別子)

<sup>(14)</sup> オンライン GUID ジェネレータ <http://guid.one/>

<sup>(15)</sup><https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/get-started/virtual-dc/virtualized-domain-controller->

アーキテクチャ

## 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 ×

**General**   **Advanced**   **Resulting Config**

VM ID:	100	Name:	deb-mediawiki
Sockets:	1	CPU Type:	x86-64-v3 <span style="float: right;">×</span>
Cores:	2	Total cores:	2
Memory (MiB):	1024	OS Type:	Linux <span style="float: right;">▼</span>
		Version:	6.x - 2.6 Kernel <span style="float: right;">▼</span>
Default Storage:	cp	Default Bridge:	vnet1 <span style="float: right;">▼</span>
Format:	Raw disk image (raw) <span style="float: right;">▼</span>		
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>		
<b>Import</b>			

Proxmox VEはAPIとWebベースのユーザインターフェースにネイティブに統合するために、ストレージプラグインシステムを使用した統合VMインポーターを提供します。これを使用してVMを全体としてインポートし、そのコンフィグのほとんどをProxmox VEのコンフィグモデルにマッピングし、ダウントIMEを削減することができます。

### 備考

インポートウィザードはProxmox VE 8.2の開発サイクルで追加され、技術プレビューの状態です。すでに有望で安定して動作していますが、まだアクティブな開発中です。

インポートウィザードを使用するには、まず新しいストレージをインポートソースに設定する必要があります。設定するには、ウェブインターフェイスの *Datacenter* → *Storage* → *Add* で行います。

次に、リソースツリーで新しいストレージを選択し、*Virtual Guests* コンテンツタブを使用して、インポート可能なすべてのゲストを確認できます。

**Import Guest - esxi-7.0:ha-datacenter/tom-nasi/deb-mediawiki/deb-mediawiki.vmx** ×

**General** **Advanced** **Resulting Config**

**Disks:**

Use	Disk ↑	Source	Size	Storage	Format
<input checked="" type="checkbox"/>	scsi0	deb-mediawiki....	32.00 GiB	From Default	Raw disk image

Prepare for VirtIO-SCSI      SCSI Controller: **VirtIO SCSI single**

**CD/DVD Drives:**

Use	Slot ↑	Storage	ISO
<input checked="" type="checkbox"/>	sata0	none	none

**Network Interfaces:**

Use	ID ↑	MAC address	Model	Bridge
<input checked="" type="checkbox"/>	net0	auto	VirtIO (paravirtualize)	From Default

Unique MAC addresses

**Import**

1つを選択し、インポートボタン（またはダブルクリック）を使ってインポートウィザードを開きます。ここで利用可能なオプションのサブセットを修正し、インポートを開始することができます。インポートが完了した後に、さらに詳細な修正を行うことができます。

### チップ

ESXi インポートウィザードは、ESXi バージョン 6.5 から 8.0 でテストされています。vSAN ストレージを使用しているゲストは直接インポートできないことに注意してください。インポート元として vCenter を使用することは可能ですが、パフォーマンス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方法のステップバイステップのガイドとヒントについては、[Proxmox VE wiki 記事をへのマイグレーション](#)の参照してください。

### OVA/OVF輸入

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のファイルベースの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あり、OVAファイルまたはOVFファイルと対応する画像をフラットな構造で置くことができます。また、Web UIを使用してOVAファイルを直接アップロードまたはダウンロードすることもできます。その後、Web UIを使用してそれらを選択し、インポートウィザードを使用してゲストをインポートすることができます。

OVAファイルの場合、一時的にイメージを展開するために必要な領域が追加されます。これには、イメージのコンテンツタイプが設定されたファイルベースのストレージが必要です。デフォルトでは、このために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ができます。

## 備考

OVA/OVF ファイルの構造やコンテンツは常に維持・定義されているわけではないので、いくつかのゲスト設定を手動で適応させる必要があるかもしれません。例えば、SCSI コントローラのタイプは OVA/OVF ファイルではほとんど定義されていませんが、デフォルトでは OVMF (UEFI) で起動できません。

## 10.7.2 CLIによるOVF/OVAのインポート

海外のハイパーバイザーからのVMエクスポートは通常、VMの設定（RAM、コア数）を記述したコンフィギュレーション・ファイルとともに、1つまたは複数のディスク・イメージの形で行われます。

ディスクイメージは、ディスクがVMwareやVirtualBoxから提供されている場合はvmdkフォーマット、ディスクがKVMハイパーバイザーから提供されている場合はqcow2となります。VMエクスポート用の最も一般的な設定フォーマットはOVF標準ですが、多くの設定が標準自体で実装されおらず、ハイパーバイザーが非標準の拡張機能で補足情報をエクスポートするため、実際には相互運用が制限されます。

フォーマットの問題だけでなく、エミュレートされるハードウェアがハイパーバイザー大きく変わると、他のハイパーバイザーからのディスクイメージのインポートに失敗することがあります。WindowsのVMは、OSがハードウェアの変更に非常にうるさいため、この問題を特に懸念しています。この問題は、エクスポートする前にインターネットから入手可能な MergeDE.zip ユーティリティをインストールし、インポートした Windows VM を起動する前にハードディスクのタイプを **IDE** に選択することで解決できます。

最後に、エミュレートされたシステムの速度を向上させ、ハイパーバイザに特化した準仮想化ドライバの問題があります。GNU/Linuxやその他のフリーのUnix OSには必要なドライバがデフォルトでインストールされており、VMをインポートした直後に準仮想化ドライバに切り替えることができます。Windows VMの場合は、Windows準仮想化ドライバ自分でインストールする必要があります。

GNU/Linuxやその他のフリーUnixは通常、問題なくインポートできます。上記の問題のため、Windows VMのインポート/エクスポートがすべてのケースで成功することは保証できません。

### Windows OVFインポートのステップバイステップ例

マイクロソフトは、Windows開発を始めるための[仮想マシンのダウンロード](#)を提供しています。私たちは、OVFインポート機能のデモを行うために、これらのうちの1つを使用するつもりです。

#### 仮想マシンのzipファイルをダウンロード

ユーザー同意書について説明を受けた後、VMwareプラットフォーム用の*Windows 10 Enterprise (Evaluation - Build)*を選択し、zipをダウンロードします。

#### zipからディスクイメージを解凍します。

unzipユーティリティまたは任意のアーカイバを使用してzipを解凍し、ssh/scp経由でovfファイルとvmdkファイルをProxmox VEホストにコピーします。

#### 仮想マシンのインポート

これにより、OVFマニフェストから読み取ったコア、メモリ、VM名を使用して新しい仮想マシンが作成され、ディスクがlocal-lvmストレージにインポートされます。ネットワークは手動で設定する必要があります。

```
# qm importovf 999 WinDev1709Eval.ovf local-lvm
```

VMを起動する準備ができました。

#### 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VMに追加することもできます。

`vmdebootstrap`ツールでDebian/Ubuntuディスクイメージを作成したとします:

```
vmdebootstrap --verbose \
--サイズ 10GiB -シリアルコンソール
--grub --no-extlinux \
-パッケージ openssh-server
-パッケージ avahi-daemon \
-パッケージ qemu-guest-agent \
--hostname vm600 --enable-dhcp \
--customize=./copy_pub_ssh.sh \
--sparse --image vm600.raw
```

これで新しいターゲットVMを作成し、イメージをストレージ`pvedir`にインポートしてVMのSCSIコントローラにアタッチすることができます:

```
# qm create 600 --net0 virtio,bridge=vmbr0 --name vm600 --serial0 socket \
--boot order=scsi0 --scsihw virtio-scsi-pci --ostype 126 \
--scsi0 pvedir:0,import-from=/path/to/dir/vm600.raw
```

VMを起動する準備ができました。

## 10.8 クラウドインイットのサポート

[Cloud-Init](#)は、仮想マシンのインスタンスの初期化を処理する、事実上の複数配布パッケージです。Cloud-Initを使用すると、ハイパーバイザ側で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が起動すると、VM内部のCloud-Initソフトウェアがこれらの設定を適用します。

多くのLinuxディストリビューションは、すぐに使えるCloud-Initイメージを提供しています。これらのイメージはProxmox VEでも動作します。このようなすぐに使えるイメージ入手するのは便利かもしれません、通常は自分でイメージを準備することをお勧めします。その利点は、何をインストールしたかを正確に知ることができ、後で自分のニーズに合わせてイメージを簡単にカスタマイズできることです。

このようなCloud-Initイメージを作成したら、それをVMテンプレートに変換することをお勧めします。VMテンプレートからリンクされたクローンを素早く作成できるので、これは新しいVMインスタンスをロールアウトする迅速な方法です。新しいVMを起動する前にネットワーク（と多分sshキー）を設定するだけです。

Cloud-InitでプロビジョニングされたVMにログインするには、SSHキーベースの認証を使用することをお勧めします。パスワードを設定することも可能ですが、Proxmox VEはCloud-Initのデータ内に暗号化されたパスワードを保存する必要があるため、SSHキーベースの認証ほど安全ではありません。

Proxmox VEはCloud-InitデータをVMに渡すためにISOイメージを生成します。そのためには、すべてのCloud-Init VMにCD-ROMドライブが割り当てられている必要があります。通常は、シリアルコンソールを追加して

ディスプレイ。多くのCloud-Initイメージはこれに依存しており、これはOpenStackの要件です。しかし、他のイメージではこの設定に問題があるかもしれません。シリアルコンソールを使ってうまくいかない場合は、デフォルトのディスプレイ構成に戻してください。

## 10.8.1 Cloud-Initテンプレートの準備

最初のステップはVMの準備です。基本的にはどんなVMでも使えます。Cloud-Initパッケージをインストールするだけです。

を実行します。Debian/Ubuntuベースのシステムでは、これは以下のように簡単です：

```
apt-get install cloud-init
```



### 警告

このコマンドはProxmox VEホスト上では実行されず、VM内部でのみ実行されます。

すでに多くのディストリビューションがすぐに使えるCloud-Initイメージ（.qcow2ファイルとして提供）を提供しているので、そのようなイメージをダウンロードしてインポートするだけでもかまいません。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。

```
# 画像をダウンロード
wget https://cloud-images.ubuntu.com/bionic/current/bionic-server-cloudimg-
      amd64.img
```

```
# VirtIO SCSI コントローラーで新しい VM を作成します。
qm create 9000 --memory 2048 --net0 virtio,bridge=vmbr0 --scsihw virtio-
      scsi-pci
```

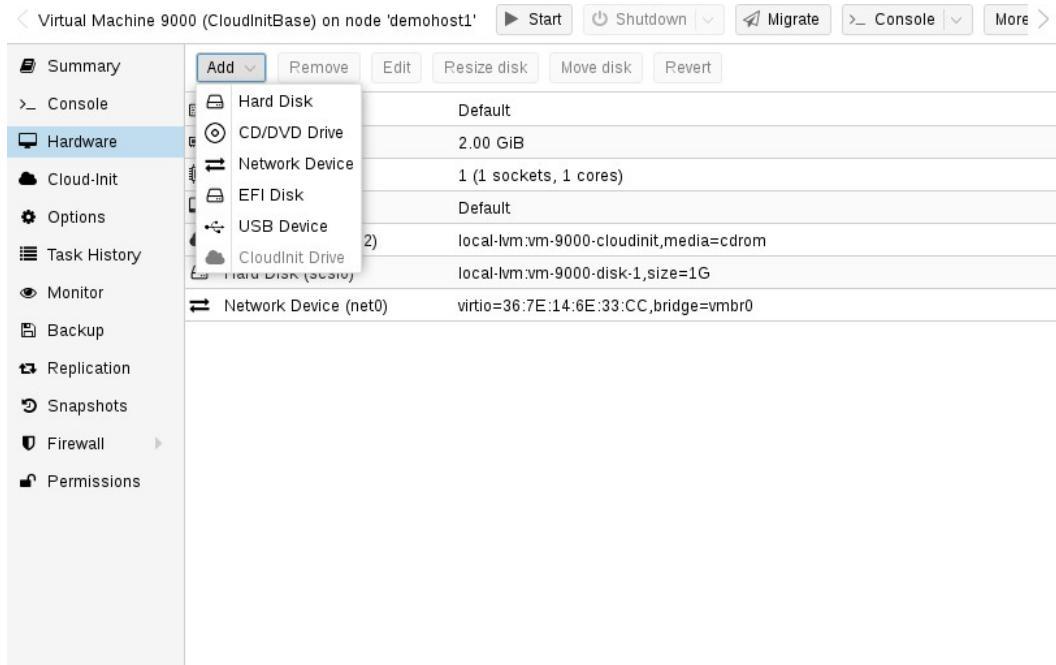
```
# ダウンロードしたディスクをlocal-lvmストレージにインポートし、SCSI ドライブとしてアタッチします。
qm set 9000 --scsi0 local-lvm:0,import-from=/path/to/bionic-server-cloudimg'
      -amd64.img
```

←

### 備考

Ubuntu Cloud-Init イメージには、SCSI ドライブ用に `virtio-scsi-pci` コントローラタイプが必要です。

### Cloud-InitのCD-ROMドライブを追加



次のステップはCD-ROMドライブを設定することです。CD-ROMドライブはCloud-InitデータをVMに渡すために使用されます。

```
qm set 9000 --ide2 local-lvm:cloudinit
```

Cloud-Init イメージから直接ブートするには、ブートパラメータを `order=scsi0` に設定して、BIOS がこのディスクからのみブートするように制限します。これにより、VM BIOS はブート可能な CD-ROM のテストをスキップするため、ブートが高速化されます。

```
qm set 9000 --boot order=scsi0
```

多くの Cloud-Init イメージでは、シリアルコンソールを設定してディスプレイとして使用する必要があります。しかし、その特定のイメージで機能しない場合は

```
qm set 9000 --serial0 socket --vga serial0
```

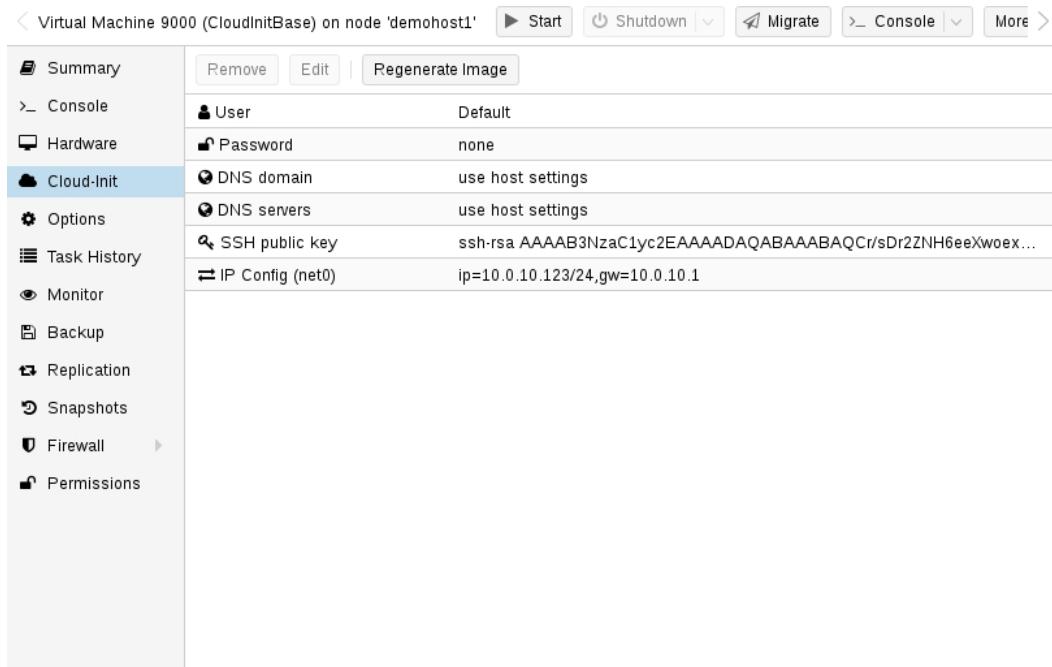
代わりにデフォルトのディスプレイに切り替えてください。

最後のステップでは、VMをテンプレートに変換すると便利です。このテンプレートからリンクされたクローンを素早く作成できます。VMテンプレートからのデ

QMテンプレート 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テンプレートのデプロイ



このようなテンプレートはクローンすることで簡単に導入できます:

```
qm clone 9000 123 --name ubuntu2
```

次に、認証に使用するSSH公開鍵を設定し、IP行います:

```
qm set 123 --sshkey ~/.ssh/id_rsa.pub
qm set 123 --ipconfig0 ip=10.0.10.123/24,gw=10.0.10.1
```

1つのコマンドだけですべてのCloud-Initオプションを設定することもできます。上記の例では、行数を減らすためにコマンドを単純に分割しています。また、特定の環境のIPセットアップを採用するようにしてください。

## 10.8.3 カスタムクラウドインイット構成

Cloud-Init の統合では、自動生成された設定の代わりにカスタム設定ファイルを使うこともできます。これはコマンドラインの `cicustom` オプションで行います:

```
qm set 9000 --cicustom "user=<volume>,network=<volume>,meta=<volume>"
```

カスタム設定ファイルは、スニペットをサポートするストレージ上にあり、VMを移行するすべてのノードで利用可能でなければなりません。そうでないと、VM

```
qm set 9000 --cicustom "user=local:snippets/userconfig.yaml"
```

は起動できません。例えば

Cloud-Initには3種類の設定があります。一つ目は上の例にあるようにユーザー設定です。2つ目はネットワーク設定、3つ目はメタ設定です。これらはすべて一緒に指定することもできますし、必要に応じて組み合わせて指定することもできます。カスタムコンフィグファイルが指定されていない場合は、自動生成されたコンフィグが使用されます。

生成されたコンフィグはカスタムコンフィグのベースとしてダンプすることができます:

```
qm cloudinit dump 9000 ユーザー
```

ネットワークとメタにも同じコマンドがあります。

## 10.8.4 Windows上のクラウドインイット

Cloudbase-initというWindows用のCloud-Initの再実装があります。Cloudbase-InitではCloud-Initのすべての機能が利用できるわけではなく、Cloud-Initとは異なる機能もあります。

Cloudbase-Initでは、ostypeをWindowsバージョンに設定し、citypeをconfigdrive2に設定する必要があります。

Windows用の既製のクラウドイメージは無料で入手できません。Cloudbase-Initを使うにはWindowsゲストを手動でインストールして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。Cloudbase-Initをダウンロードしてゲストにインストールします。ベータ版をインストールする必要があるかもしれません。インストールの最後にSysprepを実行しないでください。代わりにCloudbase-Initを最初に設定します。

一般的なオプションは以下の通りです：

- *username*: 管理者のユーザー名を設定します。
- グループに追加します：これにより、ユーザーをAdministratorsグループに追加できます。
- *inject\_user\_password* : VMコンフィグでパスワードを設定できるようにするには、これをtrueに設定します。
- *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に設定します。
- *rename\_admin\_user* : これをtrueに設定すると、デフォルトの管理者ユーザーをusernameで指定したユーザー名に変更できます。
- *metadata\_services* に設定します： Cloudbase-Initが最初にこのサービスをチェックするように、  
cloudbaseinit.metadata.services.configdrive.ConfigDriに設定します。そうしないと、Cloudbase-Initがブート後にシステムをコンフィグするのに数分かかるかもしれません。

プラグインの中には、例えば SetHostnamePlugin のように再起動を必要とし、自動的に再起動するものもあります。Cloudbase-Initによる自動再起動を無効にするには、allow\_reboot を false に設定します。

設定オプションの完全なセットは [cloudbase-init の公式ドキュメント](#)にあります。

設定後にスナップショットを作成することは、設定の一部にまだ調整が必要な場合に役立ちます。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にします。

```
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
```

その後、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=10.0.10.1 qm set 123 --sshkey  
~/.ssh/id_rsa.pub
```

パスワードを設定する前に、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。そうしないとパスワードは暗号化され、Cloudbase-Initは暗号化されたパスワードを平文のパスワードとして使用します。

すべてが設定されたら、クローンしたゲストを起動します。最初の起動ではログインは機能せず、変更されたホスト名で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て新しいシステムを提供する機能です。Cloudbase-Initと併用することで、クリーンなテンプレートを作成することができます。

Sysprepを使用する際には、2つの設定ファイルを適合させる必要があります。1つ目は通常の設定ファイル、2つ目は-unattend.confで終わる設定ファイルです。

Cloudbase-Initは2つのステップで実行されます。最初に-unattend.confを使用するSysprepステップ、次にプライマリ設定ファイルを使用する通常のステップです。

提供されているUnattend.xmlファイルを使用してSysprepを実行しているWindows Serverでは、すぐに動作するはずです。しかし、通常のWindowsバージョンでは、追加の手順が必要です：

1. PowerShellインスタンスを開きます。

2. Administratorユーザーを有効にします：

```
net user Administrator /active:yes
```

3. 管理者ユーザを使ってCloudbase-Initをインストールします。

4. Unattend.xmlを変更し、sysprepping後の最初のブートでAdministratorユーザーを有効にするコマンドを含めます：

```
<RunSynchronousCommand wcm:action="add">を実行します。  
<パス>ネットユーザー管理者/active:yes</パス  
<オーダー>1</オーダー

- 他の同期コマンドと競合しないことを確認してください。<オーダー>の変更



の後に実行するCloudbase-Initコマンドの大きさします：



```
<オーダー>2</オーダー
```


```

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  
-すべてのユーザー
```

6. Cloudbase-Init configディレクトリにcdします：

```
cd 'C:\Program Files\Cloudbase Solutions\Cloudbase-Init\conf'
```

7. (オプション) Sysprep前にVMのスナップショットを作成し、設定ミスに備えます。

8. Sysprepを実行します：

```
C:\Windows\System32\Sysprep\Sysprep.exe /generalize /oobe /unattend: ←'  
出席しない.xml
```

上記の手順を踏むと、SysprepによってVMはシャットダウン状態になるはずです。これでVMをテンプレート化し、クローンして必要に応じて設定することができます。

## 10.8.7 クラウドイニット固有のオプション

**カスタム [meta=<volume>] [,network=<volume>] [,user=<volume>] [,vendor=<volume>]** です。

開始時に自動生成されたファイルを置き換えるカスタムファイルを指定します。

### **meta= <ボリューム>**

cloud-init経由でVMに渡されるすべてのメタデータを含むカスタムファイルを指定します。これはプロバイダ固有で、configdrive2とnocloudは異なります。

### **ネットワーク= <ボリューム>**

すべてのネットワークデータを含むカスタムファイルをcloud-init経由でVMに渡す場合。

### **ユーザー= <ボリューム>**

すべてのユーザーデータを含むカスタムファイルをcloud-init経由でVMに渡す場合。

### **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に渡すには。

### **cipassword:<文字列>**

ユーザーに割り当てるパスワード。これを使用することは一般的に推奨されません。代わりに ssh 鍵を使ってください。また、cloud-init の古いバージョンではハッシュ化されたパスワードをサポートしていないことに注意しましょう。

### **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、設定されているオペレーティングシステムの種類 (ostype.Linuxではnocloud形式、Windowsではconfigdrive2を使用します)。

### **ciupgrade:<boolean> (デフォルト= 1)**

最初のブート後にパッケージの自動アップグレードを行います。

### **ciuser:<文字列>**

イメージに設定されているデフォルトユーザの代わりに、ssh キーとパスワードを変更するユーザ名。

### **ipconfig[n]: [gw=<GatewayIPv4>] [,gw6=<GatewayIPv6>] [,ip=<IPv4Format/CIDR>] [,ip6=<IPv6Format/CIDR>]** となります。

対応するインターフェイスのIPアドレスとゲートウェイを指定します。

IPアドレスはCIDR表記を使用し、ゲートウェイはオプションですが、同じタイプのIPを指定する必要があります。

特別な文字列 `dhcp` は、DHCPを使用するIPアドレスに使用でき、その場合、明示的な ゲートウェイは提供されるべきではありません。IPv6では、ステートレス自動設定を使うために特別な文字列 `auto` を使うことができます。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= <GatewayIPv4>**

IPv4トラフィックのデフォルトゲートウェイ。

**備考**

必要なオプション: `ip`

**gw6= <GatewayIPv6>**

IPv6トラフィックのデフォルトゲートウェイ。

**備考**

必要なオプション: `ip6`

**ip= <IPv4Format/CIDR> (デフォルト= dhcp)**

CIDR IPv4アドレス。

**ip6= <IPv6Format/CIDR> (デフォルト= dhcp)**

CIDR IPv6アドレス。

**ネームサーバー: <文字列**

コンテナの DNS サーバー IP アドレスを設定します。searchdomain も nameserver も設定されていない場合、Create はホストからの設定を自動的に使用します。

**searchdomain: <文字列**

コンテナの DNS 検索ドメインを設定します。searchdomain も nameserver も設定されていない場合、Create はホストからの設定を自動的に使用します。

**sshkeys: <文字列**

公開 SSH 鍵を設定します（1行に1つの鍵、OpenSSH形式）。

## 10.9 PCI(e)バススルー

PCIバススルーは、ホストから仮想マシンにPCIデバイスを制御させるメカニズムです。これには、仮想化ハードウェアを使用する場合よりも、低レイテンシ、高パフォーマンス、より多くの機能（オフロードなど）などの利点があります。

しかし、デバイスを仮想マシンに渡すと、そのデバイスはホスト上でも他のVMでも使用できなくなります。

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、PCIe対応デバイスがPCIデバイスとしてバススルーされることを意味するものではありません。

はPCI速度でのみ動作します。PCIeとしてデバイスを通過させることは、デバイスが"本当に高速なレガシーPCIデバイス"ではなくPCIeデバイスであることをゲストに伝えるためのフラグを設定するだけです。一部のゲストアプリケーションは恩恵を受けます。

## 10.9.1 一般要件

バススルーパススルーハードウェアは実際のハードウェア上で実行されるため、いくつかの要件を満たす必要があります。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については、[PCIバススルーパススルーハードウェアの例](#)を参照してください。

### ハードウェア

ハードウェアがIOMMU(I/O Memory Management Unit)割り込みリマッピングをサポートしている必要があります。

一般的に、VT-dを搭載したIntelシステムとAMD-Viを搭載したAMDシステムはこれをサポートしています。しかし、ハードウェアの実装が悪かったり、ドライバが不足していたり、品質が低かったりするため、箱から出してすぐにすべてが動作する保証はありません。

さらに、サーバーグレードのハードウェアは、コンシューマーグレードのハードウェアよりもサポートが優れていることが多いのですが、それでも最近のシステムの多くはこれをサポートしています。

この機能がLinuxでサポートされているかどうかは、ハードウェアベンダーにお問い合わせください。

### PCIカードアドレスの決定

最も簡単な方法は、GUIを使用してVMのハードウェア・タブに「Host PCI」タイプのデバイスを追加することです。また、コマンドラインを使用することもできます。

カードを探すには

エルエスピーシー

### 構成

ハードウェアがバススルーハードウェアをサポートしていることを確認したら、PCI(e)バススルーハードウェアを有効にするための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ですが、正確なオプション名はマザーボードのマニュアルに記載されています。

AMD CPUでは、IOMMUはデフォルトで有効になっています。最近のカーネル(6.8以降)ではIntel CPUでも有効です。それ以前のカーネルでは、[カーネルコマンド](#)

```
intel_iommu=on  
ラインに以下を追加して、Intel CPUで有効に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 がハイパーバイザによって通常実行される（デフォルトの）DMA 変換をバイパスし、代わりに DMA 要求をハードウェア IOMMU に直接渡すためです。これらのオプションを有効にするには、以下を追加します：

を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ロードされていることを確認する必要があります。etc/modulesに追加してください。

### 媒介機器のバススルーモード

仲介されたデバイス（vGPUなど）を通過する場合、以下は必要ありません。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
```

モジュール関連の何かを変更したら、initramfsをリフレッシュする必要があります。Proxmox VEでは、これを実行することで行えます：

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

モジュールがロードされているかどうかをチェックするには

```
# lsmod | grep vfio
```

4つのモジュールが含まれている必要があります。

## 仕上げの構成

最後に再起動して変更を有効にし、実際に有効になっていることを確認してください。

```
# dmesg | grep -e DMAR -e IOMMU -e AMD-Vi
```

は、IOMMU、Directed I/O、またはInterrupt Remappingが有効であることを表示するはずですが、正確なメッセージはハードウェアとカーネルによって異なります。

IOMMUが意図したとおりに動作しているかどうかを確認する方法については、wikiの「[IOMMU検証](#)」セクションを参照してください。

また、通過させたいデバイスが別の IOMMU グループに属していることも重要です。これは Proxmox VE API を呼び出すことで確認できます：

```
# pvesh get /nodes/{nodename}/hardware/pci --pci-class-blacklist ""
```

デバイスが、その機能（例えば、HDMIオーディオデバイスを持つGPU）と一緒にIOMMUグループにあるか、そのルートポートまたはPCI(e)ブリッジと一緒にある場合は問題ありません。

### PCI(e)スロット

プラットフォームによっては、PCI(e)スロットの物理的な扱いが。そのため、希望のIOMMUグループ分離が得られない場合、カードを別のPCI(e)スロットに置くことが助けになることがあります。

### 安全でない割り込み

プラットフォームによっては、安全でない割り込みを許可する必要があるかもしれません。この場合、*/etc/modprobe.d/* の '.conf' で終わるファイルに以下の

```
オプション vfio_iommu_type1 allow_unsafe_interrupts=1  
行を追加してください:
```

このオプションはシステムを不安定にする可能性がありますのでご注意ください。

### GPUバススルーバーに関する注意事項

Proxmox VEのウェブインターフェースで、NoVNCやSPICEを介してGPUのフレームバッファを表示することはできません。

GPU全体またはvGPUを通してグラフィック出力が必要な場合、物理的にモニターをカードに接続するか、ゲスト内でリモートデスクトップソフトウェア(例えばVNCやRDP)を設定する必要があります。

GPUをハードウェア・アクセラレータとして使用したい場合、たとえばOpenCLやCUDAを使用するプログラムでは、これは必要ありません。

## 10.9.2 ホスト・デバイス・バススルーバー

PCI(e)/バススルーバーの最も一般的な方法は、PCI(e)カード全体(GPUやネットワークカードなど)を通過させる方法です。

### ホスト構成

Proxmox VEは自動的にPCI(e)デバイスをホストから使用できないようにしようとします。しかし、これがうまくいかない場合、2つの方法があります：

- を追加することで、*vfio-pci*モジュールのオプションにデバイスIDを渡します。

```
オプション vfio-pci:5678,4321:8765
```

ここで1234:5678と4321:8765は、*/etc/modprobe.d/*の.confファイルで取得したベンダーとデバイスのIDです：

```
# lspci -nn
```

- ホスト上のドライバを完全にブラックリストに入れ、バススルーバー用にバインドできるようにします。

```
ブラックリスト DRIVERNAME
```

を*/etc/modprobe.d/*内の.confファイルに追加しま

す。ドライバ名を見つけるには

```
# lspci -k
```

例えば

```
# lspci -k | grep -A 3 "VGA"
```

のようなものが出力されます。

```
01:00.0 VGA互換コントローラ: NVIDIA Corporation GP108 [GeForce GT']  
    1030] (rev a1)  
        サブシステムマイクロスターインターナショナル株式会社[msi] gp108 ['  
        GeForce GT 1030]  
    使用中のカーネルドライバ: <some-module> カーネルモジュール:  
    <some-module>
```

ドライバを.conf記述することで、ドライバをブラックリスト化することができます:

```
echo "blacklist <some-module>">> /etc/modprobe.d/blacklist.conf
```

どちらの方法でも、[initramfs](#)を更新再度し、再起動する必要があります。

これがうまくいかない場合は、*vfio-pci*をロードする前にgpuモジュールをロードするよう、ソフト依存を設定する必要があるかもしれません。これは *softdep* フラグで行えます。詳しくは *modprobe.d* の man ページも参照してください。

例えば、<some-module>という名前のドライバを使っている場合です:

```
# echo "softdep <some-module> pre: vfio-pci">> /etc/modprobe.d/<some>  
モジュール>.conf
```

## 設定の確認

変更が成功したかどうかを確認するには

```
# lspci -nnk
```

をクリックし、デバイスのエントリを確認してください。もし

```
使用中のカーネルドライバ: vfio-pci
```

または [使用中の行](#)が全くない場合、デバイスはバススルーに使用する準備ができています。

## 媒介機器

[mediated](#)デバイスの場合、デバイスは*vfio-pci*ではなく直接ホストドライバとして所有されるため、この行は異なります。

## VMの構成

GPUをバススルーする場合、q35をマシンタイプとして使い、SeaBIOSの代わりにOVMF(VM用UEFI)、そしてPCIの代わりにPCIeを使うと最高の互換性が得られます。GPUバススルーにOVMFを使いたい場合、GPUがUEFI対応のROMを持っている必要があることに注意してください。ROMがUEFIに対応しているかチェックするには[PCI Passthrough Examples](#) wikiを見て下さい。

さらに、OVMFを使用することで、vgaのアビトレーションを無効にすことができ、ブート中に実行する必要があるレガシーコードの量を減らすことができます。vgaアビトレーションを無効にするには

```
echo "options vfio-pci,<device-id> disable_vga=1"> /etc/modprobe.d/vfio.conf
```

を<vendor-id>と<device-id>ください:

```
# lspci -nn
```

PCIデバイスは、VMのハードウェア・セクションにあるウェブ・インターフェイスで追加できます。VMコンフィギュレーションで**hostpciX**オプションを設定します:

```
# qm set VMID -hostpci0 00:02.0
```

またはVM行を追加します:

```
ホストPCI0: 00:02.0
```

デバイスに複数のファンクション（例えば'00:02.0'と'00:02.1'）がある場合、「00:02」という短縮構文でまとめて渡すことができます。これはウェブインターフェイスで「All Functions」チェックボックスをチェックするのと同じです。

デバイスとゲストOSによっては、必要なオプションがいくつかあります:

- **x-vga=on|off** は、PCI(e) デバイスを VM のプライマリ GPU としてマークします。これを有効にすると、**vga** 設定オプションは無視されます。
- **pcie=on|off** は PCIe または PCI ポートを使用するように Proxmox VE に指示します。ゲスト/デバイスの組み合わせによっては PCI ではなく PCIe を必要とします。PCIe は q35 マシンタイプでのみ使用可能です。
- **rombar=on|off** は、ファームウェア ROM をゲストから見えるようにします。デフォルトは on です。いくつかの PCI(e) デバイスは、これを無効にする必要があります。
- **romfile=<path>** には、デバイスが使用する ROM ファイルへのパスを任意で指定します。これは /usr/share/kvm/。

## 例

GPUをプライマリに設定したPCIeバススルーの例:

```
# qm set VMID -hostpci0 02:00,pcie=on,x-vga=on.
```

## PCI IDオーバーライド

ゲストによって認識される PCI ベンダー ID、デバイス ID、サブシステム ID を上書きすることができます。これは、デバイスがゲストのドライバが認識しない ID を持つバリエントであるにもかかわらず、それらのドライバを強制的にロードしたい場合に便利です（例えば、デバイスがサポートされるバリエントと同じチップセットを共有していることが分かっている場合など）。

使用可能なオプションは **vendor-id**、**device-id**、**sub-vendor-id**、**sub-device-id** です。デバイスのデフォルト ID を上書きするために、これらのいずれかまたはすべてを設定できます。

例えば

```
# qm set VMID -hostpci0 02:00,device-id=0x10f6,sub-vendor-id=0x0000.
```

## 10.9.3 SR-IOV

PCI(e)デバイスを通過させるもう一つの方法は、もし利用可能であれば、デバイスのハードウェア仮想化機能を使うことです。

### SR-IOVの実現

SR-IOVを使用するには、プラットフォームのサポートが特に重要です。最初に BIOS/UEFI でこの機能を有効にするか、特定の PCI(e) ポートを使用する必要があるかもしれません。不明な点は、プラットフォームのマニュアルを参照するか、ベンダーにお問い合わせください。

SR-IOV (Single-Root Input/Output Virtualization) は、1つのデバイスで複数の VF (仮想機能) をシステムに提供することができます。これらの VF はそれぞれ異なる VM で使用でき、完全なハードウェア機能を備え、ソフトウェア仮想化デバイスよりも優れたパフォーマンスと低レイテンシを実現します。

現在、最も一般的なユースケースは SR-IOV をサポートする NIC (ネットワーク・インターフェイス・カード) で、物理ポートごとに複数の VF を提供することができます。これにより、チェックサム・オフロードなどの機能を VM 内部で使用できるようになり、(ホストの) CPU オーバーヘッドを削減できます。

### ホスト構成

一般的に、上で仮想機能を有効にするには2つの方法があります。

- ドライバモジュールにオプションがある場合もあります。

```
max_vfs=4
```

これは、`/etc/modprobe.d/` の下に `.conf` で終わるファイルを置くことができます。(その後、`initramfs` を更新することを忘れないでください)。

正確なパラメータとオプションについては、ドライバ・モジュールのマニュアルを参照してください。

- つ目の、より一般的な方法は、`sysfs` を使う方法です。デバイスとドライバがこれをサポートしていれば、VF の数をその場で変更することができます。例えば、デバイス `0000:01:00.0` に4つの VF をセットアップするには、次のように実行します:

```
# echo 4 > /sys/bus/pci/devices/0000:01:00.0/sriov_numvfs
```

この変更を持続させるには、`sysfsutils` Debian パッケージを使用します。インストール後、`/etc/sysfs.conf` または `/etc/sysfs.d/` 内の `FILE.conf` で設定します。

### VMの構成

VF を作成した後、それらを `lspci` で出力すると、別々の PCI(e) デバイスとして見えるはずです。それらの ID を取得し、[通常の PCI\(e\) デバイス](#) のように通してください。

## 10.9.4 媒介デバイス (vGPU、GVT-g)

媒介デバイスは、物理ハードウェアの機能と性能を仮想化ハードウェアに再利用するもう1つの方法です。これらは、IntelのGVT-gやNVIDIAのGRIDテクノロジーで使用されているvGPUのような仮想化GPUセットアップで最もよく見られます。

これにより、物理カードは SR-IOV と同様に仮想カードを作成することができます。違いは、mediatedデバイスはホスト上で PCI(e) デバイスとして表示されず、仮想マシンでの使用にのみ適していることです。

## ホスト構成

一般的に、カードのドライバがその機能をサポートしていなければ、動作しません。そのため、互換性のあるドライバとその設定方法については、各ベンダーにお問い合わせください。

インテルのGVT-g用ドライバはカーネルに統合されており、第5、第6、第7世代のインテルCoreプロセッサ、およびE3 v4、E3 v5、E3 v6のXeonプロセッサで動作するはずです。

Intel Graphicsで有効にするには、モジュール*kvmgt*を（例えば/etc/modules経由で）ロードし、[カーネルコマンドライン](#)で有効にして、以下のパラメータを追加する必要があります：

```
i915.enable_gvt=1
```

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の構成

mediatedデバイスを使うには、hostpciX VMのコンフィギュレーション・オプションでmdevプロパティを指定するだけです。サポートされているデバイスは、sysfs経由で取得できます。例えば、デバイスのサポートされているタイプをリストするには、次のようにします。

0000:00:02.0単に実行するだけです：

```
# ls /sys/bus/pci/devices/0000:00:02.0/mdev_supported_types
```

各エントリーは、以下の重要なファイルを含むディレクトリです：

- available\_instancesには、このタイプのまだ利用可能なインスタンスの量が含まれます。
- 説明には、その型の能力に関する短い説明が含まれます。
- createは、このようなデバイスを作成するためのエンドポイントです。  
オプションが設定されています。

```
# qm set VMID -hostpci0 00:02.0,mdev=i915-GVTg_V5_4o
```

Intel GVT-g vGPU (Intel Skylake 6700k) を使用した構成例：

この設定により、Proxmox VEはVM起動時にこのようなデバイスを自動的に作成し、VM停止時に再びクリーンアップします。

### 10.9.5 クラスターでの使用

クラスタレベルでデバイスをマッピングすることも可能で、HAで適切に使用でき、ハードウェアの変更が検出され、ルートユーザー以外でも設定できるようになります。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、仮想マシン内のハードウェアIOMMUのエミュレーションで、仮想化I/Oデバイスのメモリアクセス制御とセキュリティを向上させます。vIOMMU オプションを使用すると、ネストされた仮想化によって、レベル 1 VM 内のレベル 2 VM に PCI(e) デバイスを渡すこともできます。物理PCI(e) デバイスをホストからネストされたVMに渡すには、PCI(e) パススルーの説明に従ってください。

現在、2つのvIOMMU実装が利用可能です：IntelとVirtIOです。

## インテル vIOMMU

Intel vIOMMU固有のVM要件:

- ホストでIntelまたはAMDのどちらのCPUを使用している場合でも、VMのカーネル・パラメータで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れば、PCIデバイスを通過できるはずのVMのコンフィギュレーションのマシン・パラメータにviommu=intelを追加できます。

```
# qm set VMID -machine q35,viommu=intel
```

## VT-d用 QEMU ドキュメント

### ヴィルティオ・ヴィオム

このvIOMMUの実装はより新しく、Intel vIOMMUなどの制限はありませんが、現在のところ実運用での使用は少なく、文書化もされていません。

VirtIO vIOMMUでは、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を使用する場合は使用することをお勧めします。

[virtio-iommuを説明するマイケル・ザオのブログ記事](#)

## 10.10 フックスクリプト

設定プロパティhookscriptを使用すると、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
```

このスクリプトはゲストが生きている間の様々な局面で呼び出されます。例とドキュメントは /usr/share/pve-docs/examples/guest-example-hookscript.pl にあるサンプルスクリプトを参照してください。

## 10.11 冬眠

VMをディスクにサスペンドするには、GUIオプションのHibernateまたは

```
# qm suspend ID --todisk
```

つまり、メモリの現在の内容がディスクに保存され、VMは停止します。次の起動時には、メモリの内容がロードされ、VMは中断したところから続行することができます。

### ステートストレージの選択

メモリーのターゲット・ストレージが指定されていない場合は、自動的に選択さ：

1. VMコンフィグのストレージvmstatestorage。
2. 任意のVMディスクからの最初の共有ストレージ。
3. どのVMディスクからも最初の非共有ストレージ。
4. フォールバックとしてのストレージローカル。

## 10.12 リソースマッピング

The screenshot shows the Proxmox VE Datacenter interface with the 'Resource Mappings' section selected in the sidebar. It displays two tables: 'PCI Devices' and 'USB Devices'. Both tables show mappings between host resources and guest devices.

ID/Node/Path ↑	Actions	Vendor/De...	Subsystem...	IOMMU gr...	Status	Comment
NIC						
pve-ceph-01						
0000:06:12.0		1af4:1000	1af4:0001	9	Mapping matches host data	
pve-ceph-02						
0000:06:12.0		1af4:1000	1af4:0001	9	Mapping matches host data	

ID/Node/Vendor&Device ↑	Actions	Path	Status	Comment
STICK				
pve-ceph-01				
0627:0001			Mapping matches host data	
pve-ceph-02				
0627:0001		1-1	Mapping matches host data	

ローカルリソース（例えば、pciデバイスのアドレス）を使用したり参照したりする場合、生のアドレスやidを使用することは、時々問題があります：

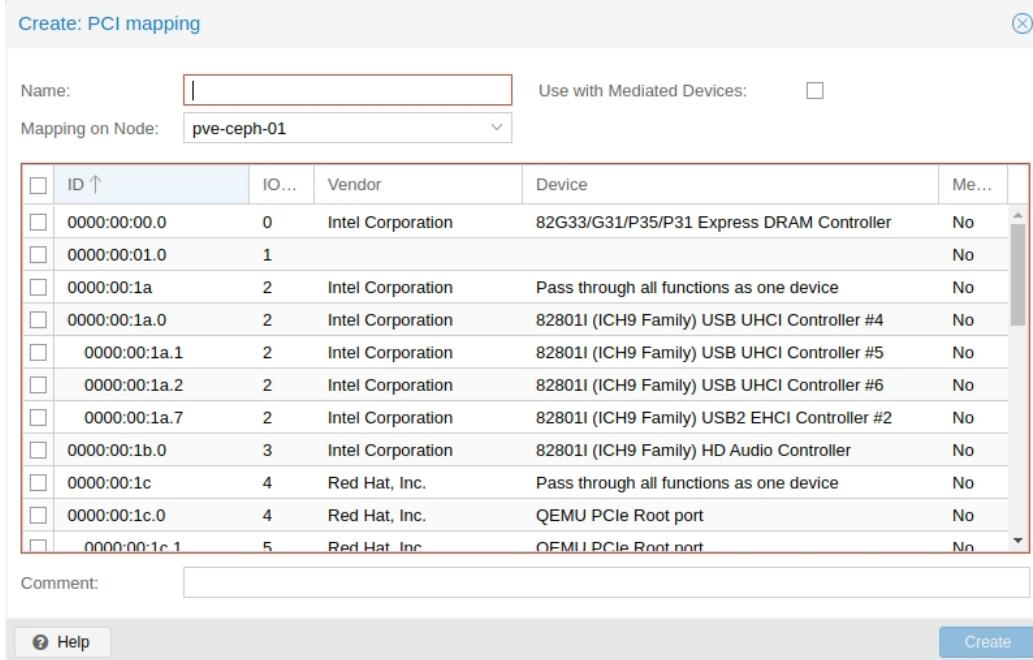
- HAを使用する場合、ターゲットノード上に同じIDやパスを持つ別のデバイスが存在する可能性があり、そのようなゲストをHAグループに割り当てる際に注意しなければ、間違ったデバイスが使用され、設定が壊れる可能性があります。
- ハードウェアを変更するとIDやパスが変わることがあるので、割り当てられたすべてのデバイスをチェックして、パスやIDが正しいかどうかを確認する必要があります。

これをうまく処理するには、クラスタ全体のリソースマッピングを定義して、リソースがクラスタ固有の、ユーザが選択した識別子を持ち、異なるホスト上の異なるデバイスに対応できるようにします。これにより、HAが間違ったデバイスでゲストを起動することはなくなり、ハードウェアの変更も検出できるようになります。

このようなマッピングの作成は、Proxmox VE ウェブ GUI のリソースマッピングカテゴリの関連タブにあるデータセンターで行うことができます。

```
# pvesh create /cluster/mapping/<type> <options>
```

<type>はハードウェアのタイプ（現在はpci、usb、dirのいずれか）、<options>はデバイスマッピングとその他の設定パラメータです。



オプションには、ハードウェアが変更されておらず、正しいデバイスが通過していることを確認できるように、そのハードウェアのすべての識別プロパティを持つマッププロパティを含める必要があることに注意してください。

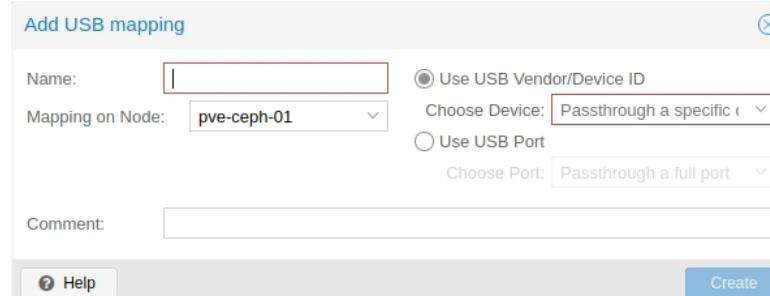
例えば、デバイスID 0001を持つパス0000:01:00.0のPCIデバイスをdevice1として追加するには、次のようにします。

、ノード node1 にベンダ ID 0002、ノード node2 に 0000:02:00.0 を追加します:

```
# pvesh create /cluster/mapping/pci --id device1 \
--map node=node1,path=0000:01:00.0,id=0002:0001 \
--map node=node2,path=0000:02:00.0,id=0002:0001
```

デバイスがマッピングされるべき各ノードに対して、mapパラメータを繰り返す必要があります（現在、1つのマッピングにつき1つのノードにつき1つのUSBデバイスしかマッピングできないことに注意してください）。

GUIを使用すると、正しいプロパティが自動的にピックアップされ、APIに送信されるため、これが非常に簡単になります。



PCI デバイスでは、ノードごとに複数のマッププロパティを持つ複数のデバイスを提供することも可能です。このようなデバイスがゲストに割り当てられた場合、ゲストの起動時に最初に空いているものが使用されます。与えられたバスの順番は試行される順番でもあるため、任意の割り当てポリシーを実装できます。

これは、SR-IOVを持つデバイスにとって便利です。

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 <名前>.
```

PCIデバイスの場合

```
# qm set <vmid> -usb0 <name>
```

USB用。

ここで、<vmid> はゲスト ID で、<name> は作成されるマッピングの名前です。mdev のような、デバイスを通過するための通常のオプションはすべて許可されています。

マッピングを作成するには、/mapping/<type>/<name>のMapping.Modifyが必要です。  
はデバイスタイプ、<name>はマッピング名）。

これらのマッピングを使用するには、/mapping/<type>/<name>上のMapping.Useが必要です（設定を編集する通常のゲスト権限に加えて）。

#### 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、さらにオプションがあります。現在、以下のオプションがあります：

- mdev (PCI)：これは、PCI デバイスを仲介デバイスを提供できるようにマークします。これが有効な場合、ゲスト上で設定する際にタイプを選択できます。マッピングに複数の PCI デバイスが選択された場合、仲介デバイスは、選択されたタイプの利用可能なインスタンスがある最初のデバイス上に作成されます。
- live-migration-capable (PCI) : これは、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。最近のカーネルを持つNVIDIA GPUのみが、これをサポートすることが知られています。バススルーされた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、動作しないか、問題が発生する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 上の QEMU/KVM 仮想マシンを管理するツールです。仮想マシンの作成と破棄、実行の制御(スタート/ストップ/サスペンド/レジューム)が可能です。さらに、qm を使用して、関連する設定ファイルにパラメータを設定できます。仮想ディスクの作成と削除も可能です。

## 10.13.1 CLIの使用例

ローカル・ストレージにアップロードされたisoファイルを使って、4GBのIDEディスクを持つVMをlocal-lvmストレージに作成します。

```
# qm create 300 -ide0 local-lvm:4 -net0 e1000 -cdrom local:iso/proxmox-'  
メールゲートウェイ_2.1.iso
```

新しいVMの起動

```
# qm start 300
```

シャットダウン要求を送信し、VMが停止するまで待ちます。

```
# qm shutdown 300 && qm wait 300
```

上記と同じですが、40待つだけです。

```
# qm shutdown 300 && qm wait 300 -timeout 40
```

VMがシャットダウンしない場合は、VMを強制停止し、実行中のシャットダウン・タスクを上書きします。VMを停止するとデータが失われる可能性があるため、

```
# qm stop 300 -overrule-shutdown 1
```

使用には注意が必要です。

VMを破棄すると、そのVMは常にアクセス制御リストから削除され、そのVMのファイアウォール構成も常に削除されます。レプリケーション・ジョブ、バックアップ・ジョブ、およびHAリソース・コンフィギュレーションからもVMを削除する場合は、-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、ディスク scsil がソースVMから削除され、scsi3 としてターゲットVMにアタッチされます。バックグラウンドでは、ディスク・イメージの名前が新しい所有者と一致するように変更されます。

```
# qm move-disk 300 scsil --target-vmid 400 --target-disk scsi3
```

## 10.14 構成

VM設定ファイルはProxmoxクラスタファイルシステム内に保存され、/etc/pve/qemuにアクセスできます。

etc/pve/ 内に保存されている他のファイルと同様に、他のすべてのクラスタ・ノードに自動的に複製されます。

### 備考

VMID<100は内部用に予約されており、VMIDはクラスタ全体で一意である必要があります。

## VMの構成例

```
boot: order=virtio0;net0 コア: 1
ソケット1
メモリ512 名前: webmail
ostype: 126
net0: e1000=EE:D2:28:5F:B6:3E,bridge=vmbr0 virtio0: local:vm-100-disk-
1,size=32G。
```

これらの設定ファイルは単純なテキストファイルであり、通常のテキストエディタ（vi、nano、...）これは小さな修正を行うのに便利ですが、そのような変更を適用するにはVMを再起動する必要があることに注意してください。

そのため、通常はqmコマンドを使用してファイルを生成および変更するか、GUIを使用してすべてを行なう方がよいでしょう。私たちのツールキットは十分に賢く、実行中のVMにほとんどの変更を瞬時に適用します。この機能は「ホットプラグ」と呼ばれ、この場合VMを再起動する必要はありません。

### 10.14.1 ファイル形式

VM設定ファイルは、コロンで区切られた単純なキー/値形式を使用します。各行の書式は以下のとおりです：

```
# これはコメントです。
```

これらのファイルの空白行は無視され、#文字で始まる行はコメントとして扱われ、これも無視されます。

### 10.14.2 スナップ写真

スナップショットを作成すると、qm はスナップショット時の設定を同じ設定ファイル内の別のスナップショット秒に保存します。例えば、"testsnapshot" というスナップショットを作成した後、設定ファイルは次のようにになります：

#### スナップショットによるVM構成

```
メモリ512
スワップ512
親: テストナファト
...
[testsnapshot] メモリ
: 512
スワップ512
スナップタイム: 1457170803
...
```

parent や snaptime のようなスナップショット関連のプロパティがいくつかあります。parent プロパティはスナップショット間の親子関係を保存するため使用されます。snaptime はスナップショットの作成タイムスタンプ (Unix epoch) です。

オプションの `vmstate` を使用すると、実行中のVMのメモリを保存できます。VMの状態に対してターゲット・ストレージがどのように選択されるかの詳細については、[ハイバネーションの章の状態ストレージの選択](#)を参照してください。

### 10.14.3 オプション

#### `acpi: <boolean>` (デフォルト= 1)

ACPI を有効/無効にします。

#### 親和性: <文字列>

ゲストプロセスの実行に使用されるホストコアのリスト（例：0,5,8-11

#### `エージェント: [enabled=<1|0> [,freeze-fs-on-backup=<1|0>] [,fstrim_cloned_disks=<1|0>] [,type=<virtio|isa>]` ]。

QEMU ゲストエージェントとそのプロパティとの通信を有効/無効にします。

##### `enabled= <boolean>` (デフォルト= 0)

VM 内で実行されている QEMU ゲストエージェント (QGA) との通信を有効/無効にします。

##### `freeze-fs-on-backup= <boolean>` (デフォルト= 1)

一貫性を保つために、バックアップ時にゲストファイルシステムをフリーズ/解凍します。

##### `fstrim_cloned_disks= <boolean>` (デフォルト= 0)

ディスクの移動またはVMの移行後にfstrimを実行します。

##### `タイプ= <isa| virtio>` (デフォルト= `virtio`)

エージェントタイプの選択

#### `amd-sev: [type=<sev-type>[,allow-smt=<1|0>] [,kernel-hashes=<1|0>] [,no-debug=<1|0>] [,no-key-sharing=<1|0>]]。`

AMD CPUによるセキュア暗号化仮想化 (SEV) 機能

##### `allow-smt= <boolean>` (デフォルト= 1)

同時マルチスレッド (SMT) を許可するポリシービットを設定します (SEV-SNP の場合を除き無視されます)。

##### `kernel-hashes= <boolean>` (デフォルト= 0)

カーネル・ハッシュをゲスト・ファームウェアに追加して、Linuxカーネルの起動を測定します。

##### `no-debug= <boolean>` (デフォルト= 0)

ゲストのデバッグを禁止するポリシービットを設定します。

##### `no-key-sharing= <boolean>` (デフォルト= 0)

他のゲストとの鍵共有を禁止するポリシービットを設定します (SEV-SNP では無視)

##### `タイプ= <sev-type>`

`type=std`で標準的なSEVを有効にするか、`es`オプションで実験的なSEV-ESを有効にするか、`snp`オプションで実験的なSEV-SNPを有効にします。

**arch: <aarch64 | x86\_64>**  
仮想プロセッサ・アーキテクチャ。デフォルトはホスト。

#### 引数: <文字列>

kvm に渡される任意の引数、例: args: -no-reboot -smbios

type=0, vendor=FOO

#### 備考

このオプションはエキスパートです。

**audio0: device=<ich9-intel-hda | intel-hda | AC97>**  
[, driver=<spice | none>].

QXL/Spiceと組み合わせると便利です。

**device= <AC97 | ich9-intel-hda | intel-hda>**

オーディオデバイスを設定します。

**driver= <none | spice> (デフォルト= spice)**

オーディオドライババックエンド。

**autostart: <boolean> (デフォルト= 0)**

クラッシュ後の自動再起動（現在は無視されています）。

#### バルーン: <整数> (0 - N)

VM のターゲット RAM の量 (MiB 単位)。ゼロを使用すると、バロンドライバが無効になります。

**bios: <ovmf | seabios> (デフォルト= seabios)**

BIOSの実装を選択します。

**を起動します: [[legacy=]<[acdn]{1,4}>] ブート: [,order=<デバイス[;デバイス...]>]]。**

ゲストの起動順序を指定します。order= サブプロパティを使用してください。

**レガシー= <[acdn]{1,4}> (デフォルト= cdn)**

フロッピー(a)、ハードディスク(c)、CD-ROM(d)、ネットワーク(n)で起動。非推奨。代わりに order= を使ってください。

**オーダー= <デバイス[;デバイス...]>**

ゲストは、ここに表示されている順番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・ストレージ USBデバイスは直接起動し、NICはPXEをロードし、PCIeデバイスはディスクのように動作するか (NVMeなど)、オプションROMをロードします (RAIDコントローラ、ハードウェアNICなど)。

このリストにあるデバイスだけがブート可能としてマークされ、ゲストファームウェア (BIOS/UEFI) によってロードされることに注意してください。ブートに複数のディスクが必要な場合 (例: ソフトウェアレイド)、ここで全てのディスクを指定する必要があります。

指定された場合、非推奨の legacy=[acdn]\* 値を上書きします。

**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 `-ide2` オプションのエイリアスです。

**カスタム [meta=<volume>] [,network=<volume>] [,user=<volume>] [,vendor=<volume>] です。**

`cloud-init`: 開始時に自動生成されるファイルを置き換えるカスタムファイルを指定します。

**meta= <ボリューム>**

`cloud-init` 経由で VM に渡されるすべてのメタデータを含むカスタムファイルを指定します。これはプロバイダ固有で、`configdrive2` と `nocloud` は異なります。

**ネットワーク= <ボリューム>**

すべてのネットワークデータを含むカスタムファイルを `cloud-init` 経由で VM に渡す場合。

**ユーザー= <ボリューム>**

すべてのユーザーデータを含むカスタムファイルを `cloud-init` 経由で VM に渡す場合。

**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 に渡すには。

**cipassword: <文字列>**

`cloud-init`: ユーザーに割り当てるパスワード。一般的に、これを使うことは推奨されません。代わりに `ssh` 鍵を使います。また、古いバージョンの `cloud-init` はハッシュ化されたパスワードをサポートしていないことに注意しましょう。

**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、設定されているオペレーティングシステムの種類 (`ostype.Linux` では `nocloud` 形式、Windows では `configdrive2` を使用します)。

**ciupgrade: <boolean> (デフォルト= 1)**

`cloud-init`: 最初の起動後にパッケージの自動アップグレードを行います。

**ciuser: <文字列>**

`cloud-init`: イメージに設定されているデフォルトユーザーの代わりに、`ssh` キーとパスワードを変更するユーザー名。

**core: <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>] [,hidden=<1|0>] [,hv-vendor-id=<vendor-id>] を指定します。

[,phys-bits=<8-64|host>] [,reported-model=<enum> ]。

エミュレートされた CPU タイプ。

**cputype= <string> (デフォルト= kvm64)**

エミュレートされたCPUタイプ。デフォルトまたはカスタムの名前を指定できます（カスタムのモデル名の前に*custom-*を付ける必要があります）。

**flags= <+FLAG[ ; -FLAG... ]>**

で区切られた追加CPUフラグのリスト。フラグを有効にするには *+FLAG* を、無効にするには *-FLAG* を使用します。カスタム CPU モデルは、QEMU/KVM でサポートされる任意のフラグを指定できます。VM 固有のフラグは、セキュリティ上の理由から以下のセットから選択する必要があります: pcid、spec-ctrl、ibpb、ssbd、virt-ssbd、amd-ssbd、amd-no-ssb、pdpe1gb、md-clear、hv-tlbflush、hv-evmcs、aes

**hidden= <boolean> (デフォルト= 0)**

KVM仮想マシンとして識別しないでください。

**hv-vendor-id= <vendor-id>**

Hyper-V のベンダーID。Windows ゲスト内的一部のドライバーまたはプログラムには、特定の ID が必要です。

**phys-bits= <8-64|host>**

ゲスト OS に報告される物理メモアドレスビット。ホストの値より小さいか等しくなければなりません。ホスト CPU の値を使用するにはホストに設定しますが、そうすると他の値を持つ CPU へのライブマイグレーションが壊れることに注意してください。

**reported-model= <486| Broadwell||**

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 | EPYC-Genoa | EPYC-IBPB | EPYC-Milan | EPYC-Milan-v2 | EPYC-Rome | EPYC-Rome-v2

EPYC-Rome-v3 | EPYC-Rome-v4 | EPYC-v3 | EPYC-v4 | GraniteRapids

| Haswell | Haswell-IBRS | Haswell-noTSX | Haswell-noTSX-IBRS | Icelake-Client | Icelake-

Client-noTSX | Icelake-Server | Icelake-Server-noTSX | Icelake-Server-v3 | Icelake-Server-v4 |

Icelake-Server-v5 | Icelake-Server-v6 | IvyBridge | (アイビーブリッジ

IvyBridge-IBRS | KnightsMill | Nehalem | Nehalem-IBRS | Opteron\_G1 | Opteron\_G2 | Opteron\_G3 | Opteron\_G4 | Opteron\_G5

| Penryn | SandyBridge | SandyBridge-IBRS | SapphireRapids | SapphireRapids-v2 | Skylake-Client | Skylake-Client-IBRS | Skylake-Client-noTSX-IBRS | Skylake-Client-v4 | Skylake-Server

| Skylake-Server-IBRS | Skylake-Server-noTSX-IBRS | Skylake-Server-v4 |

Skylake-Server-v5 | Westmere | ウエストミア

Westmere-IBRS | athlon | core2duo | coreduo | host | kvm32 | kvm64 | max | pentium |

pentium2 | pentium3 | phenom | qemu32 | qemu64> (デフォルト=kvm64)

ゲストに報告する CPU モデルとベンダー。QEMU/KVM がサポートするモデルである必要があります。カスタム CPU モデル定義に対してのみ有效で、デフォルトモデルは常にゲスト OS に自分自身を報告します。

**cpulimit: <数値> (0 - 128) (デフォルト= 0)**

CPU 使用量の上限。

**備考**

コンピュータに2つのCPUがある場合、合計2つのCPU時間を持ちます。値0はCPU制限なしを示します。

**cpuunits: <integer> (1 - 262144) (デフォルト= cgroup v1: 1024, cgroup v2: 100)**

VMのCPUウェイト。引数はカーネルのフェア・スケジューラで使用されます。数値が大きいほど、このVMはより多くのCPU時間を得ます。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。

**説明: <文字列>**

VMの説明。WebインターフェイスのVMのサマリーに表示されます。コンフィギュレーション・ファイルのコメントとして保存されます。

**efidisk0: [file=<volume> [,efitype=<2m|4m>] [,format=<enum>] [,pre-enrolled-keys=<1|0>] [,size=<DiskSize>].**

EFIバーを保存するディスクを設定します。

**efitype= <2m| 4m> (デフォルト= 2m)**

OVMF EFIバーのサイズとタイプ。4mが新しく推奨され、セキュアブートには必須です。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。

**ファイル= <ボリューム>**

ドライブのバックアップ・ボリューム。

**フォーマット= <cloop| qcow| qcow2| qed| raw| vmdk>**

ドライブのバックアップ・ファイルのデータ形式。

**事前登録キー= <ブール値> (デフォルト= 0)**

efitype=4mとともに使用する場合は、ディストリビューション固有および Microsoft Standard キーが登録された am EFI vars テンプレートを使用します。この場合、デフォルトでセキュアブートが有効になりますが、VM 内からオフにすることも可能です。

**size= <ディスクサイズ>**

ディスクサイズ。これは単なる情報提供であり、何の効果もありません。

**freeze: <ブール値>**

起動時にCPUをフリーズ (c monitorコマンドで実行開始)。

**フックスクリプト: <文字列>**

vmsのライフタイムの様々なステップで実行されるスクリプト。

**hostpci[n]: [[host=<HOSTPCIID[,HOSTPCIID2...]>] [,deviceid=<hex id>] [,legacy-igd=<1|0>] [,mapping=<mapping-id>,device-id=<hex id>] [,legacy-igd=<1|0>] [,mapping=<mapping-id>] [,mdev=<string>] [,pcie=<1|0>] [,rombar=<1|0>] [,romfile=<string>]. [,sub-device-id=<hex id>] [,sub-vendor-id=<hex id>] [,vendor-id=<hex id>] [,x-vga=<1|0>] .**

ホストのPCIデバイスをゲストにマッピングします。

**備考**

このオプションは、ホストハードウェアへの直接アクセスを可能にします。そのため、このようなマシンのマイグレーションはもはや不可能です。

**注意**

実験的！このオプションに関する問題が報告されました。

**device-id= <hex id>**

ゲストから見えるPCIデバイスIDを上書き

**ホスト= <HOSTPCIID[,HOSTPCIID2...]>.**

ホストPCIデバイス・バススルー。ホストのPCIデバイスのPCI ID、またはホストのPCI仮想ファンクションのリスト。HOSTPCIID の構文は次のとおりです：

*bus:dev.func* (16進数)

*lspci* コマンドを使用すると、既存の PCI デバイスをリストアップできます。これかマッピング

キーのどちらかが設定されている必要があります。

**legacy-igd= <boolean> (デフォルト= 0)**

このデバイスをレガシーIGDモードで渡すと、VMのプライマリで排他的なグラフィックデバイスになります。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。

**マッピング= <マッピングID>**

クラスタ全体のマッピングのID。このホストかdefault-keyホストのどちらかを設定する必要があります。

**mdev= <文字列>**

使用する仲介デバイスのタイプ。このタイプのインスタンスは、VM の起動時に作成され、VM の停止時にクリーンアップされます。

**pcie= <ブール値> (デフォルト= 0)**

PCI-expressバスを選択してください (q35マシンモデルが必要です)。

**rombar= <boolean> (デフォルト= 1)**

デバイスの ROM をゲストのメモリーマップに表示するかどうかを指定します。

**romfile= <文字列>**

カスタム pci デバイス rom ファイル名 /usr/share/kvm/ にある必要があります)。

**sub-device-id= <hex id>**

ゲストから見えるPCIサブシステムのデバイスIDを上書きします。

**sub-vendor-id= <hex id>**

ゲストから見えるPCIサブシステムのベンダーIDを上書きします。

**vendor-id= <hex id>**

ゲストに見えるPCIベンダーIDの上書き

**x-vga= <boolean> (デフォルト = 0)**  
vfio-vgaデバイスのサポートを有効にします。

#### **hotplug: <文字列> (デフォルト = ネットワーク, ディスク, USB)**

ホットプラグ機能を選択的に有効にします。これはカンマ区切りのホットプラグ機能のリストです： ネットワーク、ディスク、CPU、メモリ、USB、cloudinit。ホットプラグを完全に無効にするには0を使用します。値として1を使用すると、デフォルトの network、disk、usb のエイリアスになります。USB ホットプラグは、マシン>= 7.1 および ostype l26 または windows> 7.

#### **hugepages: <1024 | 2 | any>**

hugepagesメモリの有効/無効。

**ide[n]: [ファイル=<ボリューム> [,aio=<native|threads|io\_uring>] [,backup=<1|0>] [,bps=<bps>]  
[,bps\_max\_length=<seconds>] [,bps\_rd=<bps>] [,bps\_rd\_max\_length=<seconds>] [,bps\_wr=<bps>]  
[,bps\_wr\_max\_length=<seconds>] [,cache=<enum>] [,cyls=<integer>] [,detect\_zeroes=<1|0>]  
[,discard=<ignore|on>] [,format=<enum>] [,heads=<integer>] [,iops=<iops>] [,iops\_max=<iops>]  
[,iops\_max\_length=<seconds>] [,iops\_rd=<iops>] [,iops\_rd\_max=<iops>]  
[,iops\_rd\_max\_length=<seconds>] [,iops\_wr=<iops>] [,iops\_wr\_max=<iops>]  
[,iops\_wr\_max\_length=<seconds>] [,mbps=<mbps>] [,mbps\_max=<mbps>] [,mbps\_rd=<mbps>]  
[,mbps\_rd\_max=<mbps>] [,mbps\_wr=<mbps>] [,mbps\_wr\_max=<mbps>] [,media=<cdrom|disk>]  
[,model=<model>] [,replicate=<1|0>] [,rerror=<ignore|report|stop>] [,secs=<integer>]  
[,serial=<serial>] [,shared=<1|0>] [,size=<Disk>] [,size=<DiskSize>] [,snapshot=<1|0>]  
[,ssd=<1|0>] [,trans=<none|lba|auto>] [,werror=<enum>] [,wwn=<wwn> ]。**

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0~3)。

#### **aio= <io\_uring| ネイティブ| スレッド>**

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r/w速度 (バイト/秒)。

#### **bps\_max\_length= <秒>。**

I/Oバーストの最大長 (秒)。

#### **bps\_rd= <bps>**

最大読み取り速度 (バイト/秒)。

#### **bps\_rd\_max\_length= <秒>。**

読み取りI/Oバーストの最大長 (秒)。

**bps\_wr= <bps>**

最大書き込み速度（バイト/秒）。

**bps\_wr\_max\_length= <秒>**

書き込みI/Oバーストの最大長（秒）。

**キャッシュ= <directsync| none| unsafe| writeback| writethrough>**

ドライブのキャッシング・モード

**cyls= <整数>**

ドライブの物理ジオメトリを特定のシリンダ強制します。

**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= <ignore| on>**

破棄/トリム要求を基礎となるストレージに渡すかどうかを制御します。

**ファイル= <ボリューム>**

ドライブのバックアップ・ボリューム。

**フォーマット= <cloop| qcow| qcow2| qed| raw| vmdk>**

ドライブのバックアップ・ファイルのデータ形式。

**heads= <整数>**

ドライブの物理ジオメトリが特定のヘッド数を持つように強制します。

**iops= <iops>**

最大r/w I/O（オペレーション毎秒）。

**iops\_max= <iops>**

スロットルされていないr/w I/Oプールの最大オペレーション/秒。

**iops\_max\_length= <秒>**

I/Oバーストの最大長（秒）。

**iops\_rd= <アイオバス>**

1秒あたりの最大読み取りI/O。

**iops\_rd\_max= <iops>**

スロットルされていない最大読み取りI/Oプール（1秒あたりの処理数）。

**iops\_rd\_max\_length= <秒>**

読み取りI/Oバーストの最大長（秒）。

**iops\_wr= <アイオバス>**

最大書き込みI/O（オペレーション/秒）。

**iops\_wr\_max= <iops>**

スロットルなしの最大書き込みI/Oプール（1秒あたりの操作数）。

**iops\_wr\_max\_length= <秒>**

書き込みI/Oバーストの最大長（秒）。

**mbps= <mbps>**

最大R/W速度（メガバイト/秒）。

**mbps\_max= <mbps>**

最大スロットルなしR/Wプール（メガバイト/秒）。

**mbps\_rd= <mbps>**

最大読み取り速度（メガバイト/秒）。

**mbps\_rd\_max= <mbps>**

スロットルのない最大読み取りプール（メガバイト/秒）。

**mbps\_wr= <mbps>**

最大書き込み速度（メガバイト/秒）。

**mbps\_wr\_max= <mbps>**

スロットルなしの最大書き込みプール（メガバイト/秒）。

**media= <cdrom| disk> (default=disk)**

ドライブのメディア・タイプ。

**model= <モデル>**

ドライブのモデル名（URLエンコード最大40バイト）。

**replicate= <boolean> (デフォルト= 1)**

ドライブをレプリケーション・ジョブの対象とするかどうか。

**rerror= <ignore| report| stop>**

読み取りエラー。

**秒= <整数>**

ドライブの物理ジオメトリを特定のセクタ強制します。

**シリアル= <シリアル>**

報告されたドライブのシリアル・ナンバー（URLエンコード、最大バイト）。

**shared= <boolean> (デフォルト= 0)**

このローカル管理ボリュームをすべてのノードで使用可能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しません！

**size= <ディスクサイズ>**

ディスクサイズ。これは単なる情報提供であり、何の効果もありません。

**snapshot= <boolean>**

qemu のスナップショットモード機能を制御します。有効な場合、ディスクに加えられた変更は一時的なもので、VM がシャットダウンされると破棄されます。

**ssd= <ブール値>**

このドライブを回転式ハードディスクではなくSSDとして公開するかどうか。

**trans= <auto| lba| none>**

強制ディスクジオメトリバイオス変換モード。

**werror= <enospc| ignore| report| stop>**

書き込みエラーアクション。

**wwn= <wwn>**

16バイトの16進文字列としてエンコードされたドライブのワールドワイド名で、先頭に0xが付きます。

**ipconfig[n]: [gw=<GatewayIPv4>] [,gw6=<GatewayIPv6>] [,ip=<IPv4Format/CIDR>]**

**[,ip6=<IPv6Format/CIDR>]** となります。

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。

特別な文字列`dhcp`は、DHCPを使用するIPアドレスに使用でき、その場合、明示的な ゲートウェイは提供されるべきではありません。IPv6では、ステートレス自動設定を使うために特別な文字列`auto`を使うことができます。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= <GatewayIPv4>**

IPv4トラフィックのデフォルトゲートウェイ。

---

**備考**

必要なオプション: ip

---

**gw6= <GatewayIPv6>**

IPv6トラフィックのデフォルトゲートウェイ。

---

**備考**

必要なオプション: ip6

---

**ip= <IPv4Format/CIDR> (デフォルト= dhcp)**

CIDRIPv4アドレス。

**ip6= <IPv6Format/CIDR> (デフォルト= dhcp)**

CIDRIPv6アドレス。

**ivshmem: size=<整数> [,name=<文字列>]** です。

VM間共有メモリ。VM間やホストとの直接通信に便利。

**name= <文字列>**

ファイル名。先頭に`pve-shm-`が付きます。デフォルトはVMIDです。VMの停止時に削除されます。

**サイズ= <整数> (1 - N)**

単位のファイルサイズ。

**keephugepages: <boolean> (デフォルト= 0)**

hugepagesと一緒に使用します。有効にすると、hugepagesはVMシャットダウン後も削除されず、その後の起動に使用できます。

**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。このオプションは一般的には必要なく、ゲストOS内で処理した方が良い場合が多いです。

**kvm: <boolean> (デフォルト= 1)**

KVMハードウェア仮想化の有効/無効を設定します。

**localtime: <ブール値**

リアルタイムクロック (RTC) をローカルタイムに設定します。これは、ostypeが以下を示す場合、デフォルトで有効になります。

Microsoft Windows OS。

**lock: <backup| clone| create| migrate| rollback| snapshot | snapshot-delete | suspended | suspending>.**

VMをロック/アンロックします。

**マシン[タイプ=]<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 電源状態を有効にします。マシンタイプ 9.2+pve1 以降は false、それ以前は true。

**enable-s4= <boolean>**

S4 電源状態を有効にします。デフォルトは、マシンタイプ 9.2+pve1 以降は false、それ以前は true。

**タイプ= <マシン・タイプ**

QEMUのマシンタイプを指定します。

**viommu= <intel| virtio>**

ゲストの vIOMMU バリアントを有効にして設定します (Intel vIOMMU はマシンタイプとして q35 を設定する必要があります)。

**メモリを使用します: [カレント=]<整数>**

メモリ特性。

**current= <integer> (16 - N) (デフォルト= 512)**

VM の現在のオンライン RAM 量 (MiB 単位)。これは、バルーン デバイスを使用する際に使用可能な最大メモリです。

**migrate\_downtime: <number> (0 - N) (デフォルト:= 0.1)**

マイグレーションの最大許容ダウンタイム（秒）を設定します。新しく汚れたRAMを転送する必要があるため、マイグレーションが最後まで収束しない場合、マイグレーションが収束するまで、この上限は自動的に段階的に増加します。

**migrate\_speed: <integer> (0 - N) (デフォルト:= 0)**

マイグレーションの最大速度（MB/s）を設定します。値 0 は制限なしです。

**name: <文字列>**

VM の名前を設定します。コンフィギュレーション Web インターフェイスでのみ使用します。

**ネームサーバー: <文字列>**

cloud-init: コンテナの DNS サーバー IP アドレスを設定します。searchdomain も nameserver も設定されていない場合、Create は自動的にホストからの設定を使用します。

```
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>] [<model>=<macaddr>] [<model>=<macaddr>
ネットワーク機器を指定します。
```

**bridge= <ブリッジ>**

ネットワークデバイスを接続するブリッジ。Proxmox VE 標準ブリッジは `vmbr0` と呼ばれます。

ブリッジを指定しない場合は、DHCP と DNS サービスを提供する kvm ユーザー（NAT）ネットワークデバイスを作成します。以下のアドレスが使用されます：

- |          |          |
|----------|----------|
| 10.0.2.2 | ゲートウェイ   |
| 10.0.2.3 | DNS サーバー |
| 10.0.2.4 | SMB サーバー |

DHCP サーバーは、10.0.2.15 から始まるアドレスをゲストに割り当てます。

**firewall= <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>**

このインターフェイスを切断するかどうか（プラグを抜くように）。

**macaddr= <XX:XX:XX:XX:XX:XX> です。**

I/G (Individual/Group) ビットが設定されていない共通の MAC アドレス。

**モデル= <e1000| e1000-82540em| e1000-82544gc| e1000-82545em| e1000e | i82551 | i82557b |****i82559er | ne2k\_isa | ne2k\_pci | pcnet | rtl18139 | virtio | vmxnet3>**

ネットワークカードモデル。`virtio` モデルは、非常に低い CPU オーバーヘッドで最高のパフォーマンスを提供します。ゲストがこのドライバをサポートしていない場合、通常は `e1000` を使用するのが最善です。

**mtu= <整数> (1 - 65520)**

MTU を強制します。ブリッジ MTU を使用するには 1 を設定します。

**キュー= <整数> (0 - 64)**

デバイスで使用するパケットキューの数。

**レート= <数値> (0 - N)**

Mbps (メガバイト/秒) 単位のレート制限を浮動小数点数で指定します。

**タグ= <整数> (1 - 4094)**

このインターフェイスのパケットに適用するVLANタグ。

**trunks= <vlanid[;vlanid...]>**

このインターフェイスを通過する VLAN トランク。

 **numa: <ブール値> (デフォルト= 0)**

NUMAの有効/無効。

numa [n]: cpus=<id[-id] ; ...> [, hostnodes=<id[-id] ; ...>] [, policy=<優先|バインド|インターリーブ>] [, policy=<優先|バインド|インターリーブ[, メモリ=<数>]] [, ポリシー=<preferred|bind|interleave>]。

NUMAトポロジー。

**cpus= <id[-id] ; ...>**

このNUMAノードにアクセスするCPU。

**hostnodes= <id[-id] ; ...>** です。

するホストNUMAノード。

**メモリ= <番号**

このNUMAノードが提供するメモリ量。

**ポリシー= <バインド| インターリーブ| 優先>**

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: <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。これは、特定のオペレーティングシステム用の特別な最適化/機能を有効にするために使用されます：

その他 不特定OS

Windows XP マイクロソフト Windows XP

w2k Microsoft Windows 2000

w2k3 Microsoft Windows 2003

w2k8 マイクロソフト Windows 2008

ビスタ	マイクロソフト Windows Vista
ワインナセブン	Microsoft Windows 7
WINエイト	Microsoft Windows 8/2012/2012r2
WINテン	マイクロソフト Windows 10/2016/2019
WIN11	Microsoft Windows 11/2022/2025
I24	Linux 2.4 カーネル
I26	Linux 2.6 - 6.X カーネル
ソラリス	Solaris/OpenSolaris/OpenIndianaカーネル

**parallel[n]: /dev/parportd+|/dev/usb/lpd+**

ホスト・パラレル・デバイスをマップします（nは0～2）。

**備考**

このオプションは、ホストハードウェアへの直接アクセスを可能にします。そのため、このようなマシンの移行はもはや不可能です。

**注意**

実験的！このオプションに関する問題が報告されました。

**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 の削除とディスクの削除操作が無効になります。

**reboot: <boolean> (デフォルト= 1)**

再起動を許可。0 に設定すると、VM は時に終了します。

**rng0: [source=</dev/urandom|/dev/random|/dev/hwrng> [,max\_bytes=<integer>] [,period=<integer>]** です。

VirtIO ベースの乱数ジェネレータを設定します。

**max\_bytes= <整数> (デフォルト= 1024)**

ミリ秒ごとにゲストに注入されるエントロピーの最大バイト数。使用

0 で制限を無効にします（潜在的に危険です！）。

**期間= <整数> (デフォルト= 1000)**

ミリ秒ごとにエントロピーの注入クォータがリセットされ、ゲストは他の *max\_bytes* のエントロピーを取得できるようになります。

**ソース=** </dev/hwrng| /dev/random| /dev/urandom>。

エントロピーを収集するホスト上のファイル。urandomを使用しても、実際のエントロピーからシードされることに変わりはなく、提供されるバイトはゲスト上の実際の混合される可能性が高いので、意味のある方法でセキュリティを低下させることはできません。/dev/hwrngはホストからハードウェアRNGを渡すために使用できます。

**sata[n]: [ファイル=]〈ボリューム〉 [,aio=<native|threads|io\_uring>] [,backup=<1|0>] [,bps=<bps>]  
[,bps\_max\_length=<seconds>] [,bps\_rd=<bps>] [,bps\_rd\_max\_length=<seconds>] [,bps\_wr=<bps>]  
[,bps\_wr\_max\_length=<seconds>] [,cache=<enum>] [,cyls=<integer>] [,detect\_zeroes=<1|0>]  
[,discard=<ignore|on>] [,format=<enum>] [,heads=<integer>] [,iops=<iops>] [,iops\_max=<iops>]  
[,iops\_max\_length=<seconds>] [,iops\_rd=<iops>] [,iops\_rd\_max=<iops>] [,iops\_rd\_max\_length=<秒>]  
[,iops\_wr=<iops>] [,iops\_wr\_max=<iops>] [,iops\_wr\_max\_length=<seconds>] [,mbps=<mbps>]  
[,mbps\_max=<mbps>] [,mbps\_rd=<mbps>] [,mbps\_rd\_max=<mbps>] [,mbps\_wr=<mbps>]  
[,mbps\_wr\_max=<mbps>] [,media=<cdrom|disk>] [,replicate=<1|0>] [,rerror=<ignore|report|stop>]  
[,secs=<integer>] [,serial=<serial>] [,shared=<1|0>] [,size=<DiskSize>] [,snapshot=<1|0>]  
[,ssd=<1|0>] [,trans=<none|lba|auto>] [,werror=<enum>] [,wwn=<wwn> ]。**

ボリュームをSATA/ハードディスクまたはCD-ROMとして使用します (nは0~5)。

**aio= <io\_uring| ネイティブ| スレッド>**

使用するAIOタイプ。

**backup= <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r/w速度 (バイト/秒)。

**bps\_max\_length= <秒>。**

I/Oバーストの最大長 (秒)。

**bps\_rd= <bps>**

最大読み取り速度 (バイト/秒)。

**bps\_rd\_max\_length= <秒>。**

読み取りI/Oバーストの最大長 (秒)。

**bps\_wr= <bps>**

最大書き込み速度 (バイト/秒)。

**bps\_wr\_max\_length= <秒>。**

書き込みI/Oバーストの最大長 (秒)。

**キャッシング= <directsync| none| unsafe| writeback| writethrough>**

ドライブのキャッシング・モード

**cyts= <整数>**

ドライブの物理ジオメトリを特定のシリンド強制します。

**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= <ignore| on>**

破棄/トリム要求を基礎となるストレージに渡すかどうかを制御します。

**ファイル= <ボリューム>**

ドライブのバックアップ・ボリューム。

**フォーマット= <cloop| qcow| qcow2| qed| raw| vmdk>**

ドライブのバックアップ・ファイルのデータ形式。

**heads= <整数>**

ドライブの物理ジオメトリが特定のヘッド数を持つように強制します。

**iops= <iops>**

最大r/w I/O（オペレーション毎秒）。

**iops\_max= <iops>**

スロットルされていないr/w I/Oプールの最大オペレーション/秒。

**iops\_max\_length= <秒>**

I/Oバーストの最大長（秒）。

**iops\_rd= <アイオバス>**

1秒あたりの最大読み取りI/O。

**iops\_rd\_max= <iops>**

スロットルなしの最大読み取りI/Oプール（1秒あたりの処理数）。

**iops\_rd\_max\_length= <秒>**

読み取りI/Oバーストの最大長（秒）。

**iops\_wr= <アイオバス>**

最大書き込みI/O（オペレーション/秒）。

**iops\_wr\_max= <iops>**

スロットルなしの最大書き込みI/Oプール（1秒あたりの操作数）。

**iops\_wr\_max\_length= <秒>**

書き込みI/Oバーストの最大長（秒）。

**mbps= <mbps>**

最大R/W速度（メガバイト/秒）。

**mbps\_max= <mbps>**

最大スロットルなしR/Wプール（メガバイト/秒）。

**mbps\_rd= <mbps>**

最大読み取り速度（メガバイト/秒）。

**mbps\_rd\_max= <mbps>**

スロットルのない最大読み取りプール（メガバイト/秒）。

**mbps\_wr= <mbps>**

最大書き込み速度（メガバイト/秒）。

**mbps\_wr\_max= <mbps>**

スロットルなしの最大書き込みプール（メガバイト/秒）。

**media= <cdrom| disk> (default=disk)**

ドライブのメディア・タイプ。

**replicate= <boolean> (デフォルト= 1)**

ドライブをレプリケーション・ジョブの対象とするかどうか。

**rerror= <ignore| report| stop>**

読み取りエラー。

**秒= <整数>**

ドライブの物理ジオメトリを特定のセクタ強制します。

**シリアル= <シリアル>**

報告されたドライブのシリアル・ナンバー（URLエンコード、最大バイト）。

**shared= <boolean> (デフォルト= 0)**

このローカル管理ボリュームをすべてのノードで使用可能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しません！

**size= <ディスクサイズ>**

ディスクサイズ。これは単なる情報提供であり、何の効果もありません。

**snapshot= <boolean>**

qemu のスナップショットモード機能を制御します。有効な場合、ディスクに加えられた変更は一時的なもので、VM がシャットダウンされると破棄されます。

**ssd= <プール値>**

このドライブを回転式ハードディスクではなくSSDとして公開するかどうか。

**trans= <auto| lba| none>**

強制ディスクジオメトリバイオス変換モード。

**werror= <enospc| ignore| report| stop>**

書き込みエラーアクション。

**wwn= <wwn>**

16バイトの16進文字列としてエンコードされたドライブのワールドワイド名で、先頭に0xが付きます。

```
scsi[n]: [ファイル=<ボリューム> [,aio=<native|threads|io_uring>] [,backup=<1|0>] [,bps=<bps>]
[,bps_max_length=<seconds>] [,bps_rd=<bps>] [,bps_rd_max_length=<seconds>] [,bps_wr=<bps>]
[,bps_wr_max_length=<seconds>] [,cache=<enum>] [,cyls=<integer>] [,detect_zeroes=<1|0>]
[,discard=<ignore|on>] [,format=<enum>] [,heads=<integer>] [,iops=<iops>]
[,iops_max=<iops>] [,iops_max_length=<秒>] [,iops_rd=<iops>] [,iops_rd_max=<iops>]
[,iops_rd_max_length=<seconds>] [,iops_wr=<iops>] [,iops_wr_max=<iops>]
[,iops_wr_max_length=<seconds>] [,iothread=<1|0>] [,mbps=<mbps>] [,mbps_max=<mbps>]
[,mbps_rd=<mbps>] [,mbps_rd_max=<mbps>] [,mbps_wr=<mbps>] [,mbps_wr_max=<mbps>]
[,media=<cdrom|disk>] [,product=<product>] [,queues=<integer>] [,replicate=<1|0>]
[,rerror=<ignore|report|stop>] [,ro=<1|0>] [,scsiblock=<1|0>] [,secs=<integer>]
[,serial=<serial>] [,shared=<1|0>] [,size=<DiskSize>] [,snapshot=<1|0>] [,ssd=<1|0>]
[,trans=<none|lba|auto>] [,vendor=<vendor>] [,werror=<enum>] [,wwn=<wwn>] となります。
```

ボリュームをSCSIハードディスクまたはCD-ROMとして使用します (nは0~30)。

**aio= <io\_uring| ネイティブ| スレッド>**

使用するAIOタイプ。

**backup= <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r/w速度 (バイト/秒)。

**bps\_max\_length= <秒>。**

I/Oバーストの最大長 (秒)。

**bps\_rd= <bps>**

最大読み取り速度 (バイト/秒)。

**bps\_rd\_max\_length= <秒>。**

読み取りI/Oバーストの最大長 (秒)。

**bps\_wr= <bps>**

最大書き込み速度 (バイト/秒)。

**bps\_wr\_max\_length= <秒>。**

書き込みI/Oバーストの最大長 (秒)。

**キャッシング= <directsync| none| unsafe| writeback| writethrough>**

ドライブのキャッシング・モード

**cyls= <整数>**

ドライブの物理ジオメトリを特定のシリンド強制します。

**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= <ignore| on>**  
破棄/トリム要求を基礎となるストレージに渡すかどうかを制御します。

**ファイル= <ボリューム>**  
ドライブのバックアップ・ボリューム。

**フォーマット= <cloop| qcow| qcown| qed| raw| vmdk>**  
ドライブのバックアップ・ファイルのデータ形式。

**heads= <整数>**  
ドライブの物理ジオメトリが特定のヘッド数を持つように強制します。

**iops= <iops>**  
最大r/w I/O（オペレーション毎秒）。

**iops\_max= <iops>**  
スロットルされていないr/w I/Oプールの最大オペレーション/秒。

**iops\_max\_length= <秒>。**  
I/Oバーストの最大長（秒）。

**iops\_rd= <アイオバス>**  
1秒あたりの最大読み取りI/O。

**iops\_rd\_max= <iops>**  
スロットルされていない最大読み取りI/Oプール（1秒あたりの処理数）。

**iops\_rd\_max\_length= <秒>。**  
読み取りI/Oバーストの最大長（秒）。

**iops\_wr= <アイオバス>**  
最大書き込みI/O（オペレーション/秒）。

**iops\_wr\_max= <iops>**  
スロットルなしの最大書き込みI/Oプール（1秒あたりの操作数）。

**iops\_wr\_max\_length= <秒>。**  
書き込みI/Oバーストの最大長（秒）。

**iothread= <boolean>**  
このドライブにiothreadsを使用するかどうか

**mbps= <mbps>**  
最大R/W速度（メガバイト/秒）。

**mbps\_max= <mbps>**  
最大スロットルなしR/Wプール（メガバイト/秒）。

**mbps\_rd= <mbps>**  
最大読み取り速度（メガバイト/秒）。

**mbps\_rd\_max= <mbps>**

スロットルのない最大読み取りプール（メガバイト/秒）。

**mbps\_wr= <mbps>**

最大書き込み速度（メガバイト/秒）。

**mbps\_wr\_max= <mbps>**

スロットルなしの最大書き込みプール（メガバイト/秒）。

**media= <cdrom| disk> (default = disk)**

ドライブのメディア・タイプ。

**製品= <製品**

ドライブの製品名（最大16バイト）。

**キュー= <整数> (2 - N)**

キューの数。

**replicate= <boolean> (デフォルト= 1)**

ドライブをレプリケーション・ジョブの対象とするかどうか。

**rerror= <ignore| report| stop>**

読み取りエラー。

**ro= <ブール値**

ドライブが読み取り専用かどうか。

**scsiblock= <boolean> (デフォルト= 0)**

ホストブロックデバイスのフルパススルーに scsi-block を使用するかどうか

**警告**

ホストのメモリ不足やメモリの断片化が進むと、I/O エラーが発生する可能性があります。

**秒= <整数**

ドライブの物理ジオメトリを特定のセクタ強制します。

**シリアル= <シリアル**

報告されたドライブのシリアル・ナンバー（URLエンコード、最大バイト）。

**shared= <boolean> (デフォルト= 0)**

このローカル管理ボリュームをすべてのノードで使用可能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しません！

**size= <ディスクサイズ**

ディスクサイズ。これは単なる情報提供であり、何の効果もありません。

**snapshot= <boolean>**

qemu のスナップショットモード機能を制御します。有効な場合、ディスクに加えられた変更は一時的なもので、VM がシャットダウンされると破棄されます。

**ssd= <ブール値>**

このドライブを回転式ハードディスクではなくSSDとして公開するかどうか。

**trans= <auto| lba| none>**

強制ディスクジオメトリバイオス変換モード。

**ベンダー= <ベンダー>**

ドライブのベンダー名（最大8バイト）。

**werror= <enospc| ignore| report| stop>**

書き込みエラーアクション。

**wwn= <wwn>**

16バイトの16進文字列としてエンコードされたドライブのワールドワイド名で、先頭に0xが付きます。

**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: <文字列>**

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はホストからの設定を自動的に使用します。

**serial[n]: (/dev/.+|socket)**

VM内部でシリアル・デバイスを作成し（nは0~3）、ホストのシリアル・デバイス（例：/dev/ttyS0）を通すか、ホスト側でunixソケットを作成します（qm terminalを使ってターミナル接続を開きます）。

**備考**

ホストシリアルデバイスを経由すると、そのようなマシンの移行はできなくなります。

**注意**

実験的！ このオプションに関する問題が報告されました。

**shares: <整数> (0 - 50000) (デフォルト=1000)**

オート・バルーニングのメモリ・シェア量。この数値が大きいほど、このVMはより多くのメモリを取得します。数値は、他のすべての実行中のVMの重みに対する相対値です。ゼロを使用すると、オートバルーニングは無効になります。オートバルーニングはpvestatdによって実行されます。

**smbios1:** [base64=<1|0>] [,family=<Base64 エンコードされた文字列>] [,manufacturer=<Base64 エンコードされた文字列>] [,product=<Base64 エンコードされた文字列>] [,serial=<Base64 エンコードされた文字列>] [,sku=<Base64 エンコードされた文字列>] [,uuid=<UUID>] [,version=<Base64 エンコードされた文字列>] .

SMBIOSタイプ1のフィールドを指定します。

**base64= <boolean>**  
SMBIOSの値がbase64エンコードされていることを示すフラグ

**ファミリー= <Base64エンコード文字列**

SMBIOS1 ファミリ文字列を設定します。

**メーカー= <Base64エンコード文字列**

SMBIOS1 メーカーを設定します。

**製品= <Base64エンコード文字列**

SMBIOS1のプロダクトIDを設定します。

**シリアル= <Base64エンコード文字列**

SMBIOS1のシリアル設定します。

**sku= <Base64エンコードされた文字列>**

SMBIOS1 SKU文字列を設定します。

**uuid= <UUID>**

SMBIOS1 の UUID を設定します。

**バージョン= <Base64エンコード文字列**

SMBIOS1のバージョンを設定します。

**smp:<整数> (1 - N) (デフォルト= 1)**

CPU数。代わりに -sockets オプションを使用してください。

**sockets:<整数> (1 - N) (デフォルト= 1)**

CPUソケットの数。

**spice\_enhancements: [foldersharing=<1|0>]**

[,videostreaming=<off|all|filter>]。

SPICE の追加機能拡張を設定します。

**foldersharing= <boolean> (デフォルト= 0)**

SPICE 経由でのフォルダ共有を有効にします。VMにSpice-WebDAVデーモンがインストールされている必要があります。

**videostreaming= <all| filter| off> (デフォルト= off)**

ビデオストリーミングを有効にします。検出されたビデオストリームに圧縮を使用します。

**sshkeys:<文字列**

cloud-init: 公開 SSH 鍵を設定します (1 行に 1 つの鍵、OpenSSH 形式)。

**startdate: (now | YYYY-MM-DD | YYYY-MM-DDTHH:MM:SS) (デフォルト= now)**

リアルタイムクロックの初期日付を設定します。有効な日付の書式は、'now' または 2006-06-17T16:01:21 または 2006-06-17.

**を起動します: [[order=]† [,up=d+] [,down=d+]**

スタートアップとシャットダウンの動作。Orderは一般的な起動順序を定義する非負の数値です。シャットダウンは逆の順序で行われます。さらに、次のVMが起動または停止するまでの待ち時間を秒単位で設定できます。

**tablet: <ブール値> (デフォルト = 1)**

USBタブレットデバイスを有効/無効にします。このデバイスは通常、VNCで絶対的なマウス位置決めを可能にするために必要です。そうしないと、マウスは通常のVNCクライアントと同期しません。1つのホスト上で多くのコンソール専用ゲストを実行している場合、コンテキストの切り替えを節約するために、これを無効にすることを検討してください。spice (qm set <vmid> --vga qxl) を使用すると、これはデフォルトでオフになります。

**タグ: <string>**

VMのタグ。これは単なるメタ情報です。

**tdf: <boolean> (デフォルト = 0)**

時間ドリフト有効/無効。

**テンプレート: <boolean> (デフォルト = 0)**

テンプレートの有効/無効。

**tpmstate0: [file=]<volume> [,size=<DiskSize>] [,version=<v1.2|v2.0>].**

TPMの状態を保存するDiskを設定します。フォーマットはraw固定。

**ファイル= <ボリューム>**

ドライブのバックアップ・ボリューム。

**size= <ディスクサイズ>**

ディスクサイズ。これは単なる情報提供であり、何の効果もありません。

**バージョン= <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の方が新しいので、そちらを優先してください。これは後で変更ごとに注意してください。

**unused[n]です: [ファイル=]<ボリューム>**

未使用ボリュームへの参照。これは内部的に使用されるので、手動で変更しないでください。

**ファイル= <ボリューム>**

ドライブのバックアップ・ボリューム。

**usb[n]: [[host=]<HOSTUSBDEVICE|spice>] [,mapping=<mappingid>] [,usb3=<1|0[,mapping=<mapping-id>]] [,usb3=<1|0>] となります。**

USBデバイスを設定します (nは0~4、マシンのバージョンが7.1以上、かつostype l26またはwindows > 7の場合、nは最大14まで)。

**ホスト= <HOSTUSBDEVICE|spice>**

ホスト USB デバイスまたはポート、または値のスパイス。HOSTUSBDEVICE 構文は次のとおりです:

'bus-port(.port)\*' (10進数) または'vendor\_id:product\_id' (16進数) または  
'spice'

*lsub -t* コマンドを使えば、既存の usb デバイスをリストアップできます。

**備考**

このオプションは、ホストハードウェアへの直接アクセスを可能にします。そのため、このようなマシンのマイグレーションはもはや不可能です。

値*spice*は、spice用のusbリダイレクトデバイスを追加するために使用できます。このキーか マッピングキーのどちらかが設定されている必要があります。

**マッピング= <マッピングID>**

クラスタ全体のマッピングのID。このホストかdefault-keyホストのどちらかを設定する必要があります。

**usb3= <boolean> (デフォルト= 0)**

与えられたホストオプションが USB3 デバイスかポートかを指定します。最近のゲスト (マシンのバージョン >= 7.1 および ostype l26 と Windows > 7) では、このフラグは無関係です (すべてのデバイスは xhci コントローラに接続されています)。

**vcpus: <整数> (1 - N) (デフォルト= 0)**

ホットプラグされたvcpusの数。

**vga: [[type=]<enum>] [[クリップボード=<vnc[クリップボード=<vnc>] [,メモリ=<整数>]**

VGAハードウェアを設定します。高解像度モード (>= 1280x1024x16) を使用したい場合は、VGAメモリオプションを増やす必要があるかもしれません。QEMU 2.9以降、デフォルトのVGAディスプレイタイプは、*cirrus*を使用する一部のWindowsバージョン (XPおよびそれ以前) 以外のすべてのOSタイプで標準となっています。*qxl* オプションは SPICE ディスプレイサーバーを有効にします。Win\* OS では、いくつの独立したディスプレイが欲しいかを選択できます。また、シリアルデバイスを端末として使用し、グラフィックカードなしで実行することもできます。

**クリップボード= <vnc>**

特定のクリップボードを有効にします。設定されていない場合、ディスプレイの種類に応じて SPICE のものが追加されます。VNCクリップボードとの移行はまだサポートされていません!

**メモリ= <整数> (4 - 512)**

VGA メモリ (MiB) を設定します。シリアル表示には影響しません。

**type= <cirrus| none| qxl| qxl2| qxl3| qxl4| serial0 | serial1 | serial2 | serial3 | std | virtio | virtio-gl | vmware> (default = std)**

VGA タイプを選択します。*cirrus*タイプの使用はお勧めしません。

`virtio[n] です: [file=]<volume> [,aio=<native|threads|io_uring>] [,backup=<1|0>] [,bps=<bps>]  
[,bps_max_length=<seconds>] [,bps_rd=<bps>] [,bps_rd_max_length=<seconds>] [,bps_wr=<bps>]  
[,bps_wr_max_length=<seconds>] [,cache=<enum>] [,cyls=<integer>] [,detect_zeroes=<1|0>]  
[,discard=<ignore|on>] [,format=<enum>] [,heads=<integer>] [,iops=<iops>] [,iops_max=<iops>]  
[,iops_max_length=<seconds>] [,iops_rd=<iops>] [,iops_rd_max=<iops>] [,iops_rd_max_length=<  
秒>] [,iops_wr=<iops>] [,iops_wr_max=<iops>] [,iops_wr_max_length=<seconds>]  
[,iothread=<1|0>] [,mbps=<mbps>] [,mbps_max=<mbps>] [,mbps_rd=<mbps>] [,mbps_rd_max=<mbps>]  
[,mbps_wr=<mbps>] [,mbps_wr_max=<mbps>]。] [,media=<cdrom|disk>] [,replicate=<1|0>]  
[,rerror=<ignore|report|stop>] [,ro=<1|0>] [,secs=<integer>] [,serial=<serial>]  
[,shared=<1|0>] [,size=<DiskSize>] [,snapshot=<1|0>] [,trans=<none|lba|auto>]  
[,werror=<enum>] となります。`

ボリュームを VIRTIO ハード ディスクとして使用します (n は 0 ~ 15)。

#### `aio= <io_uring| ネイティブ| スレッド>`

使用するAIOタイプ。

#### `backup= <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r/w速度 (バイト/秒)。

#### `bps_max_length= <秒>。`

I/O/バーストの最大長 (秒)。

#### `bps_rd= <bps>`

最大読み取り速度 (バイト/秒)。

#### `bps_rd_max_length= <秒>。`

読み取りI/Oバーストの最大長 (秒)。

#### `bps_wr= <bps>`

最大書き込み速度 (バイト/秒)。

#### `bps_wr_max_length= <秒>。`

書き込みI/Oバーストの最大長 (秒)。

#### `キャッシング= <directsync| none| unsafe| writeback| writethrough>`

ドライブのキャッシング・モード

#### `cyls= <整数>`

ドライブの物理ジオメトリを特定のシリンド強制します。

#### `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= <ignore| on>**

破棄/トリム要求を基礎となるストレージに渡すかどうかを制御します。

**ファイル= <ボリューム>**

ドライブのバックアップ・ボリューム。

**フォーマット= <cloop| qcow| qcow2| qed| raw| vmdk>**

ドライブのバックアップ・ファイルのデータ形式。

**heads= <整数>**

ドライブの物理ジオメトリが特定のヘッド数を持つように強制します。

**iops= <iops>**

最大r/w I/O（オペレーション毎秒）。

**iops\_max= <iops>**

スロットルされていないr/w I/Oプールの最大オペレーション/秒。

**iops\_max\_length= <秒>**

I/Oバーストの最大長（秒）。

**iops\_rd= <アイオバス>**

1秒あたりの最大読み取りI/O。

**iops\_rd\_max= <iops>**

スロットルされていない最大読み取りI/Oプール（1秒あたりの処理数）。

**iops\_rd\_max\_length= <秒>**

読み取りI/Oバーストの最大長（秒）。

**iops\_wr= <アイオバス>**

最大書き込みI/O（オペレーション/秒）。

**iops\_wr\_max= <iops>**

スロットルなしの最大書き込みI/Oプール（1秒あたりの操作数）。

**iops\_wr\_max\_length= <秒>**

書き込みI/Oバーストの最大長（秒）。

**iothread= <boolean>**

このドライブにiothreadsを使用するかどうか

**mbps= <mbps>**

最大R/W速度（メガバイト/秒）。

**mbps\_max= <mbps>**

最大スロットルなしR/Wプール（メガバイト/秒）。

**mbps\_rd= <mbps>**

最大読み取り速度（メガバイト/秒）。

**mbps\_rd\_max= <mbps>**

スロットルのない最大読み取りプール（メガバイト/秒）。

**mbps\_wr= <mbps>**

最大書き込み速度（メガバイト/秒）。

**mbps\_wr\_max= <mbps>**

スロットルなしの最大書き込みプール（メガバイト/秒）。

**media= <cdrom| disk> (default = disk)**

ドライブのメディア・タイプ。

**replicate= <boolean> (デフォルト= 1)**

ドライブをレプリケーション・ジョブの対象とするかどうか。

**werror= <ignore| report| stop>**

読み取りエラー。

**ro= <ブール値**

ドライブが読み取り専用かどうか。

**秒= <整数>**

ドライブの物理ジオメトリを特定のセクタ強制します。

**シリアル= <シリアル>**

報告されたドライブのシリアル・ナンバー（URLエンコード、最大バイト）。

**shared= <boolean> (デフォルト= 0)**

このローカル管理ボリュームをすべてのノードで使用可能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しません！

**size= <ディスクサイズ>**

ディスクサイズ。これは単なる情報提供であり、何の効果もありません。

**snapshot= <boolean>**

qemu のスナップショットモード機能を制御します。有効な場合、ディスクに加えられた変更は一時的なもので、VM がシャットダウンされると破棄されます。

**trans= <auto| lba| none>**

強制ディスクジオメトリバイオス変換モード。

**werror= <enospc| ignore| report| stop>**

書き込みエラーアクション。

**virtiofs[n]: [dirid=>] <mapping-id> [,cache=<enum>] [,direct-io=<1|0>] [,expose-acl=<1|0>] [,expose-xattr=<1|0>].**

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。

**cache= <always| auto| metadata| never> (default= 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= <boolean> (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 フラグを尊重します。

**dirid= <マッピング-id>**  
ゲストと共有するディレクトリマッピングのマッピング識別子。VM 内のマウントタグとしても使用されます。

**expose-acl= <ブール値> (デフォルト= 0)**  
このマウントで POSIX ACL のサポートを有効にします (有効な ACL は xattr を意味します)。

**expose-xattr= <boolean> (デフォルト= 0)**  
このマウントの拡張属性のサポートを有効にします。

#### **vmgenid: <UUID> (デフォルト= 1 (自動生成))**

VM generation ID (vmgenid) デバイスは、128 ビットの整数値識別子をゲスト OS に公開します。これにより、仮想マシンが異なる構成 (スナップショット実行やテンプレートからの作成など) で実行された場合に、ゲストOSに通知することができます。ゲストOSはこの変更に気づき、分散データベースのコピーをダーティとしてマークしたり、乱数ジェネレーターを再初期化したりするなど、適切に対応することができます。自動作成はAPI/CLIのcreate またはupdateメソッドで実行された場合のみ機能し、設定ファイルを手動で編集した場合は機能しないことに注意してください。

#### **vmstatestorage: <ストレージID>**

VM 状態ボリューム/ファイルのデフォルトストレージ。

#### **watchdog: [&model=<i6300esb|ib700>] [アクション]**

仮想ハードウェアウォッチドッグデバイスを作成します。いったん (ゲストのアクションによって) 有効になると、ウォッチドッグはゲスト内部のエージェントによって定期的にポーリングされる必要があります。

**action= <debug| none| pause| poweroff| reset| shutdown>**  
アクティブ化後、ゲストが時間内にウォッチドッグのポーリングに失敗した場合に実行するアクション。

**model= <i6300esb| ib700> (デフォルト= i6300esb)**  
エミュレートするウォッチドッグ・タイプ。

## 10.15 錠前

オンライン・マイグレーション、スナップショット、およびバックアップ (vzdump) は、影響を受けるVMに対する互換性のない同時アクションを防止するためロックを設定します。時々、このようなロックを手動で削除する必要があります (電源障害後など)。

```
# qm unlock <vmid>
```

**注意**

ロックを設定したアクションがもう実行されていないことを確認した場合のみ、この操作を行ってください。

## 第11章

# Proxmoxコンテナツールキット

コンテナは、完全に仮想化されたマシン（VM）に代わる軽量なものです。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、実行するホストシステムのカーネルを使用します。つまり、コンテナはホストシステム上のリソースに直接アクセスできます。

コンテナのランタイム・コストは低く、通常は無視できます。しかし、考慮すべき欠点もあります：

- 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。例えば、FreeBSDやMicrosoft Windowsのような他のオペレーティングシステムをコンテナ内で実行することはできません。
- セキュリティ上の理由から、ホストリソースへのアクセスを制限する必要があります。そのため、コンテナはそれぞれ別の名前空間で実行されます。さらに、一部のシステムコール（Linuxカーネルに対するユーザー空間の要求）はコンテナ内で許可されていません。

Proxmox VEは、基盤となるコンテナ技術として[Linux Containers \(LXC\)](#)を使用しています。Proxmox Container Toolkit（pct）は、複雑なタスクを抽象化するインターフェイスを提供することで、LXCの使用と管理を簡素化します。

コンテナはProxmox VEと緊密に統合されています。つまり、コンテナはクラスタのセットアップを認識し、仮想マシンと同じネットワークおよびストレージリソースを使用できます。Proxmox VEのファイアウォールを使用したり、HAフレームワークを使用してコンテナを管理することもできます。

私たちの主な目標は、VMを使用する利点を提供しながら、追加のオーバーヘッドがない環境を提供することです。つまり、Proxmox Containersは "アプリケーションコンテナ"ではなく、"システムコンテナ"に分類されます。

### 備考

アプリケーションコンテナ、例えばDockerイメージを実行する場合は、Proxmox QEMU VM内で実行することをお勧めします。これにより、アプリケーション

- コンテナ化のすべての利点が得られるだけでなく、ホストからの強力な分離や、コンテナでは不可能なライブ・マイグレーション機能など、VMが提供する利点も得られます。

## 11.1 技術概要

- LXC (<https://linuxcontainers.org/>)

- Proxmox VE グラフィカル・ウェブ・ユーザー・インターフェイス(GUI)に統合
- 使いやすいコマンドラインツール pct
- Proxmox VE REST API経由でのアクセス
- コンテナ化された /proc ファイルシステムを提供する lxcfs
- リソースの分離と制限のためのコントロールグループ (cgroups)
- セキュリティを向上させる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](#)を使用
- ホストからのコンテナ設定 (ネットワーク、DNS、ストレージなど)

## 11.2 対応ディストリビューション

公式にサポートされているディストリビューションのリストは以下にあります。

以下のディストリビューション用のテンプレートは、私たちのリポジトリから入手できます。[pveam tool](#) または Graphical User Interface を使ってダウンロードできます。

### 11.2.1 アルパインリナックス

Alpine Linux は、musl libc と busybox をベースにしたセキュリティ重視の軽量 Linux ディストリビューションです。

- <https://alpinelinux.org/>

現在サポートされているリリースについては、こちらをご覧ください:

<https://alpinelinux.org/releases/>

### 11.2.2 アーチリナックス

Arch Linux は軽量で柔軟な Linux® ディストリビューションです。

- <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。詳しくは wiki を見て下さい: [https://wiki.archlinux.org/title/Arch\\_Linux](https://wiki.archlinux.org/title/Arch_Linux)

### 11.2.3 CentOS, AlmaLinux, Rocky Linux

#### CentOS / CentOS Stream

CentOS Linux ディストリビューションは、Red Hat Enterprise Linux (RHEL) のソースから派生した、安定した、予測可能な、管理可能な、再現可能なプラットフォームです。

- <https://centos.org>

現在サポートされているリリースについては、[https://en.wikipedia.org/wiki/CentOS#End-of-support\\_schedule](https://en.wikipedia.org/wiki/CentOS#End-of-support_schedule) を。参照してください

#### アルマリナックス

オープンソース、コミュニティが所有し管理する、永久無料のエンタープライズLinuxディストリビューションで、長期的な安定性に重点を置き、堅牢なプロダクショングレードのプラットフォームを提供します。AlmaLinux OSはRHEL®とプレストリームCentOSと1:1のバイナリ互換性があります。

- <https://almalinux.org>

現在サポートされているリリースについては、

<https://en.wikipedia.org/wiki/AlmaLinux#Releases> を。参照してください

#### ロッキーリナックス

Rocky Linuxは、下流のパートナーが方向転換した現在、アメリカのトップエンタープライズLinuxディストリビューションと100%バグ互換性を持つように設計されたコミュニティ・エンタープライズ・オペレーティング・システムです。

- <https://rockylinux.org>

現在サポートされているリリースについては、[https://en.wikipedia.org/wiki/Rocky\\_Linux#Releases](https://en.wikipedia.org/wiki/Rocky_Linux#Releases) を。参照してください

### 11.2.4 デビアン

Debian はフリーなオペレーティングシステムであり、Debian プロジェクトによって開発・保守されています。フリーな Linux ディストリビューションで、ユーザのニーズを満たす数千のアプリケーションがあります。

- <https://www.debian.org/intro/index#software>

現在サポートされているリリースについては、<https://www.debian.org/releases/stable/releasenotes> を。参照してください

## 11.2.5 デヴァン

Devuan GNU+Linux は systemd を使わない Debian のフォークで、不要な絡みを避け Init Freedom を確保することで、ユーザがシステムの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、

<https://www.devuan.org/os/releases> を。参照してください

## 11.2.6 フェドラ

Fedoraは、ハードウェア、クラウド、コンテナのための革新的で無償のオープンソースプラットフォームを作成し、ソフトウェア開発者とコミュニティメンバーがユーザーのためにカスタマイズされたソリューションを構築することを可能にします。

- <https://getfedora.org>

現在サポートされているリリースについては、

<https://fedoraproject.org/wiki/Releases> を。参照してください

## 11.2.7 ジェンツー

柔軟性の高いソースベースのLinuxディストリビューションです。

- <https://www.gentoo.org>

Gentooはローリングリリースモデルを使用しています。

## 11.2.8 オープンソース

システム管理者、開発者、デスクトップ・ユーザーのためのメーカー選択です。

- <https://www.opensuse.org>

現在サポートされているリリースについては、こちらをご覧ください:

<https://get.opensuse.org/leap/>

## 11.2.9 ウブントゥ

Ubuntuは、エンタープライズ・サーバー、デスクトップ、クラウド、IoT向けのLinuxの最新オープンソース・オペレーティング・システムです。

- <https://ubuntu.com/>

現在サポートされているリリースについては、

<https://wiki.ubuntu.com/Releases> を。参照してください

## 11.3 コンテナ画像

コンテナイメージは、「テンプレート」または「アプライアンス」とも呼ばれることがあります。コンテナを実行するためのすべてを含む tar アーカイブです。

Proxmox VE自体は、最も一般的なLinuxディストリビューション用のさまざまな基本テンプレートを提供しています。これらのテンプレートは、GUIまたはpveam (Proxmox VE Appliance Managerの略) コマンドラインユーティリティを使用してダウンロードできます。さらに、TurnKey Linux コンテナテンプレートもダウンロードできます。

利用可能なテンプレートのリストはpve-daily-updateタイマーによって毎日更新されます。手動で更新することもできます：

```
#pveam アップデート
```

利用可能な画像のリストを表示するには

```
#pveam available
```

例えば、基本システムなど、興味のセクションを指定することで、この大きなリストを制限することができます。

画像をご覧ください：

### 利用可能なシステムイメージのリスト

```
# pveam available --section system
システム          alpine-3.12-default_20200823_amd64.tar.xz system alpine-
3.13-default_20210419_amd64.tar.xz system      alpine-3.14-
default_20210623_amd64.tar.xz system      archlinux-base_20210420-1_amd64.tar.gz
system      centos-7-default_20190926_amd64.tar.xz system      centos-8-
default_20201210_amd64.tar.xz system      debian-9.0-standard_9.7-1_amd64.tar.gz
system      debian-10-standard_10.7-1_amd64.tar.gz system      devuan-3.0-
standard_3.0_amd64.tar.gz system fedora-33-default_20201115_amd64.tar.xz system
fedora-34-default_20210427_amd64.tar.xz
システム          gentoo-current-default_20200310_amd64.tar.xz system
                  opensuse-15.2-default_20200824_amd64.tar.xz system      ubuntu-
16.04-standard_16.04.5-1_amd64.tar.gz system      ubuntu-18.04-standard_18.04.1-
1_amd64.tar.gz system      ubuntu-20.04-standard_20.04-1_amd64.tar.gz system ubuntu-20.10-
standard_20.10-1_amd64.tar.gz system      ubuntu-21.04-standard_21.04-1_amd64.tar.gz
```

このようなテンプレートを使用する前に、いずれかのストレージにダウンロードする必要があります。ストレージがわからない、ローカルの名前付きストレージを使用することができます。クラスタ化されたインストールでは、すべてのノードがこれらのイメージにアクセスできるように共有ストレージを使用することが推奨されます。

```
# pveam download local debian-10.0-standard_10.0-1_amd64.tar.gz
```

これで、そのイメージを使ってコンテナを作成する準備ができました。

```
# pveam list local
local:vztmpl/debian-10.0-standard_10.0-1_amd64.tar.gz 219.95MB
```

ローカル・ウィズ

## チップ

また、Proxmox VEウェブインターフェースGUIを使用して、コンテナテンプレートのダウンロード、一覧表示、削除を行うこともできます。

pctは、例えば新しいコンテナを作成するためにこれらを使用します：

```
# pct create 999 local:vztmp1/debian-10.0-standard_10.0-1_amd64.tar.gz
```

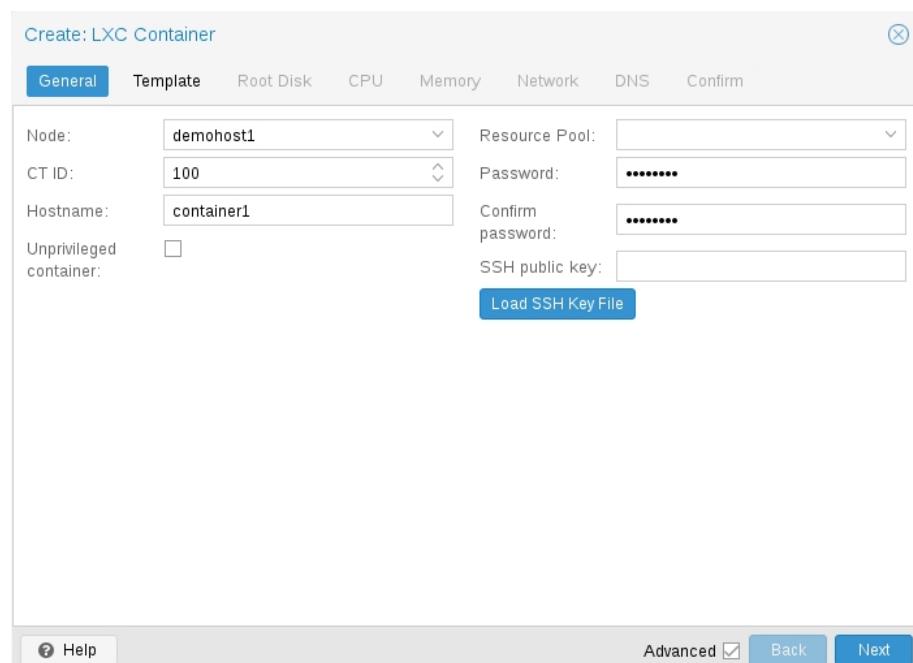
上記のコマンドは、完全なProxmox VEボリューム識別子を表示します。この識別子にはストレージ名が含まれており、他のほとんどのProxmox VEコマンドはこ

```
# pveam remove local:vztmp1/debian-10.0-standard_10.0-1_amd64.tar.gz
```

の識別子を使用できます。例えば、後でそのイメージを削除するには、次のようにします：

## 11.4 コンテナ設定

### 11.4.1 一般設定



コンテナの一般的な設定には以下が含まれます。

- **ノード**: コンテナが実行される物理サーバー
- **CT ID**: コンテナを識別するために使用されるProxmox VEのインストールで一意の番号です。
- **ホスト名**: コンテナのホスト名
- **リソースプール**: コンテナとVMの論理グループ
- **パスワード**: コンテナのルートパスワード
- **SSH公開鍵**: SSH経由でルート・アカウントに接続するための公開鍵。
- **非特権コンテナ**: このオプションでは、特権コンテナまたは非特権コンテナのどちらを作成するかを作成時に選択できます。

## 非特権コンテナ

非特権コンテナは、ユーザー・ネームスペースと呼ばれる新しいカーネル機能を使用します。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（コンテナのエスケープ、リソースの乱用など）が、ランダムな非特権ユーザーに影響することを意味。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、新しいコンテナを作成するときのデフォルトのオプションです。

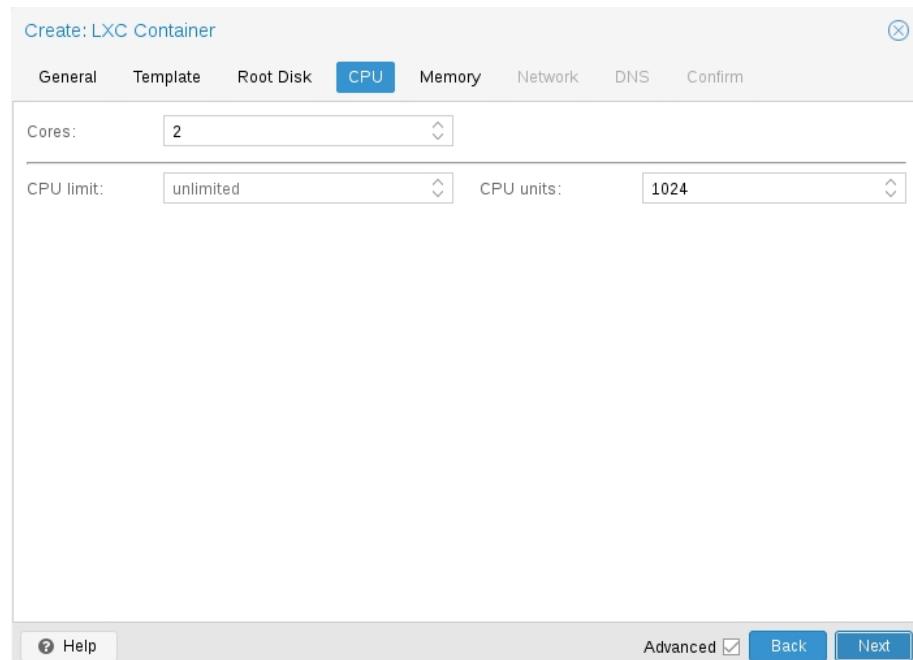
### 備考

コンテナが init システムとして systemd を使用して場合、コンテナ内で実行されている systemd のバージョンは 220 以上でなければならぬことに注意してください。

## 特権コンテナ

コンテナのセキュリティは、強制アクセス制御 AppArmor 制限、*seccomp* フィルタ、Linux カーネル名前空間を使用することで実現されます。LXCチームは、この種のコンテナを安全でないと見なしており、新しいコンテナ脱出エクスプロイトを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る必要があります。

### 11.4.2 中央演算処理装置



`cores` オプションを使用すると、コンテナ内の可視 CPU 数を制限できます。これは、Linuxの`cpuset cgroup`（制御グループ）を使用して実装されます。`pvestatd` 内部の特別なタスクは、実行中のコンテナを利用可能な CPU に定期的に分散しようとします。割り当てられたCPUを表示するには、以下のコマンドを実行します：

```
# 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 (Completely Fair Scheduler)スケジューラを使用します。

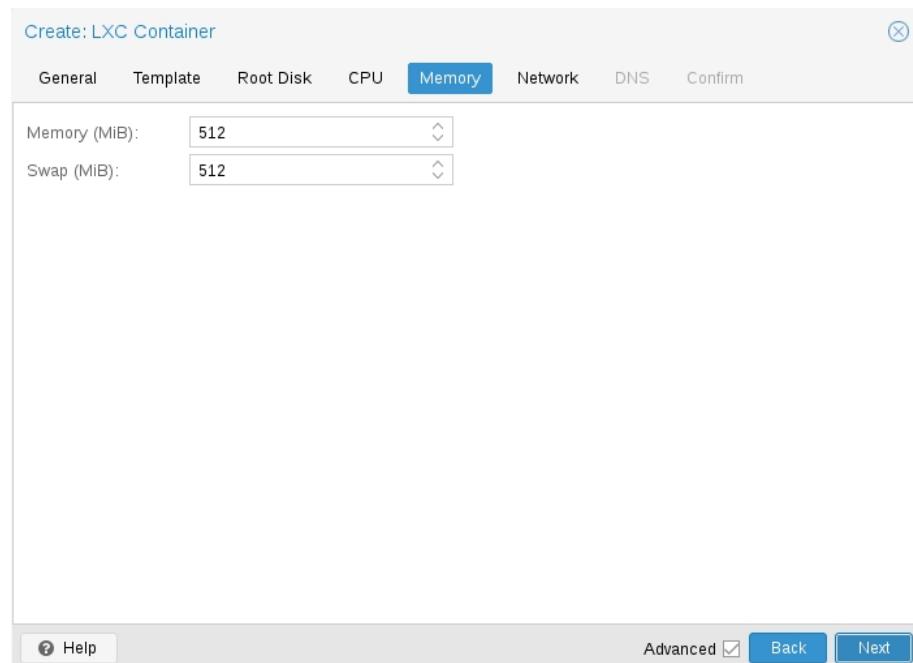
cpulimit: このオプションを使用すると、割り当てられたCPU時間をさらに制限できます。これは浮動小数点数なので、コンテナに2つのコアを割り当ても、全体のCPU消費を半分のコアに制限することはまったく問題ありません。

コア数: 2

cpulimit: 0.5パーセント

cputunits: 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。この数値が大きいほど、このコンテナのCPU時間は長くなります。数値は、実行中の他のすべてのコンテナの重みに対する相対値です。デフォルトは100（ホストがレガシーアルゴリズムを使用している場合は1024）です。この設定を使用して、一部のコンテナに優先順位を付けることができます。

### 11.4.3 メモリー

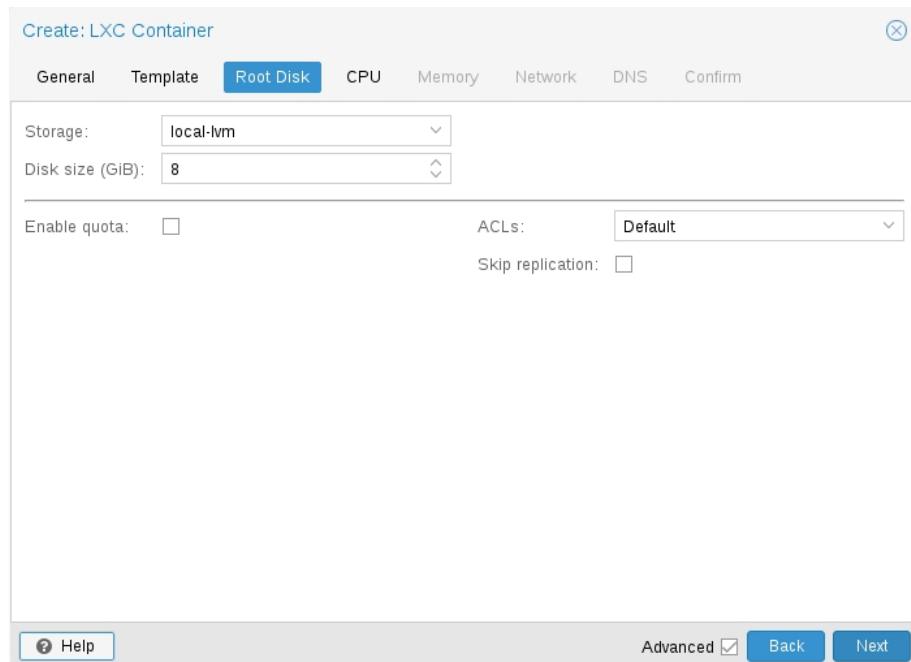


コンテナ・メモリは、cgroupメモリ・コントローラを使用して制御されます。

メモリの使用を制限します: 全体のメモリ使用量を制限します。これは `memory.limit_in_bytes` に対応します。  
cgroupの設定。

**swap:** コンテナがホストのスワップ領域から追加のスワップメモリを使用できるようにします。これは memory.memsw.limit\_in\_bytes cgroup 設定に対応し、両方の値（メモリ + スワップ）の合計に設定されます。

#### 11.4.4 マウントポイント



ルート・マウント・ポイントは、rootfs プロパティで設定します。さらに最大256個のマウントポイントを設定できます。対応するオプションは mp0 から mp255 と呼ばれます。これらのオプションには、以下の設定を含めることができます：

```
rootfs: ボリューム=>[ボリューム] [,acl=<1|0>] [,mountoptions=<opt[;opt...]>] [,quota=<1|0>]
[,replicate=<1|0>] [,ro=<1|0>] [,shared=<1|0>]
[,size=<1|0>[quota=<1|0>][,replicate=<1|0>][,ro=<1|0>][,shared=<1|0>][,size=<DiskSize>]]。
```

ボリュームをコンテナ・ルートとして使用します。すべてのオプションの詳細については、以下を参照してください。

```
mp[n]: ボリューム=>[ボリューム] ,mp=<Path> [,acl=<1|0>] [,backup=<1|0>] [,mountoptions=<opt[;opt...]>]
[,quota=<1|0>] [,replicate=<1|0>] [,ro=<1|0>] [,shared=<1|0>] [,size=<1|0>] [,マウントオプション
=<opt[;opt...][quota=<1|0>][,replicate=<1|0>][,ro=<1|0>][,shared=<1|0>][,size=<DiskSize>]]。
```

ボリュームをコンテナ・マウント・ポイントとして使用します。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= <ブール値>**

ACL サポートを明示的に有効または無効にします。

**backup= <boolean>**

マウント・ポイントをバックアップに含めるかどうか（ボリューム・マウント・ポイントにのみ使用）。

**mountoptions= <opt[;opt...]>を指定します。**

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。

**備考**

セキュリティ上の、シンボリックリンクを含んではいけません。

**quota= <boolean>**

コンテナ内でユーザークオータを有効化 (zfsサブボリュームではサポートされていません。)

**replicate= <boolean> (デフォルト= 1)**

このボリュームをストレージ・レプリカ・ジョブに含めます。

**ro= <ブール値**

読み取り専用のマウントポイント

**shared= <boolean> (デフォルト= 0)**

この非ボリュームマウントポイントをすべてのノードで使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しません!

**size= <ディスクサイズ**

ボリュームサイズ (読み取り専用値)。

**ボリューム= <ボリューム**

コンテナにマウントするボリューム、デバイス、またはディレクトリ。

現在、マウントポイントには、ポイント、バインドマウント、デバイスマウントの3種類があります。

**典型的なコンテナ rootfs 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、3つの異なる種類があります：

- イメージベース： 単一のext4フォーマットされたファイル含むrawイメージです。
- ZFSサブボリューム： これらは技術的にはバインドマウントですが、管理されたストレージを持つため、サイズ変更とスナップショットが可能です。
- ディレクトリ： size=0を渡すと、生イメージの代わりにディレクトリが作成される特殊なケースが発生します。

**備考**

ストレージバックマウントポイントボリューム用の特別なオプション構文STORAGE\_ID:SIZE\_IN\_GBは、指定されたストレージ上に指定されたサイズのボリュームを自動的に割り当てます。例えば

```
pct set 100 -mp0 thin1:10,mp=/path/in/container
```

は、ストレージthin1上に10GBのボリュームを割り当て、ボリュームIDのプレースホルダー10を割り当てられたボリュームIDに置き換え、コンテナ内の/path/in/containerにマウントポイントを設定します。

**マウントポイントのバインド**

バインドマウントを使用すると、コンテナ内のProxmox VEホストから任意のディレクトリにアクセスできます。以下のような使用例が考えられます：

- ゲスト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
- ゲストのホストからNFSマウントへのアクセス

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、コンテナ内からスナップショットを作成したりクォータを処理したりすることはできません。非特権コンテナでは、ユーザーマッピングに起因するパーミッションの問題に遭遇する可能性があり、ACLを使用できません。

**備考**

vzdumpを使用すると、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、例えば/mnt/bindmounts下のディレクトリ階層を使用してのみ確立されるべきです。マウント・システム・ディレクトリ（/、/var、/etcなど）をコンテナにバインドしてはいけません。

**備考**

バインドマウントソースパスにはシンボリックリンクを含んではいけません。

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、次のようにします。

の下に、次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 に追加してく

ださい。あるいはpctツールを使うこともできます

```
pct set 100 -mp0 /mnt/bindmounts/shared,mp=/shared
```

同じ結果を得るために。

## デバイスマウントポイント

デバイスマウントポイントを使用すると、ホストのブロックデバイスをコンテナに直接マウントできます。バインドマウントと同様に、デバイスマウントは Proxmox VE のストレージサブシステムによって管理されませんが、クォータと acl オプションは尊重されます。

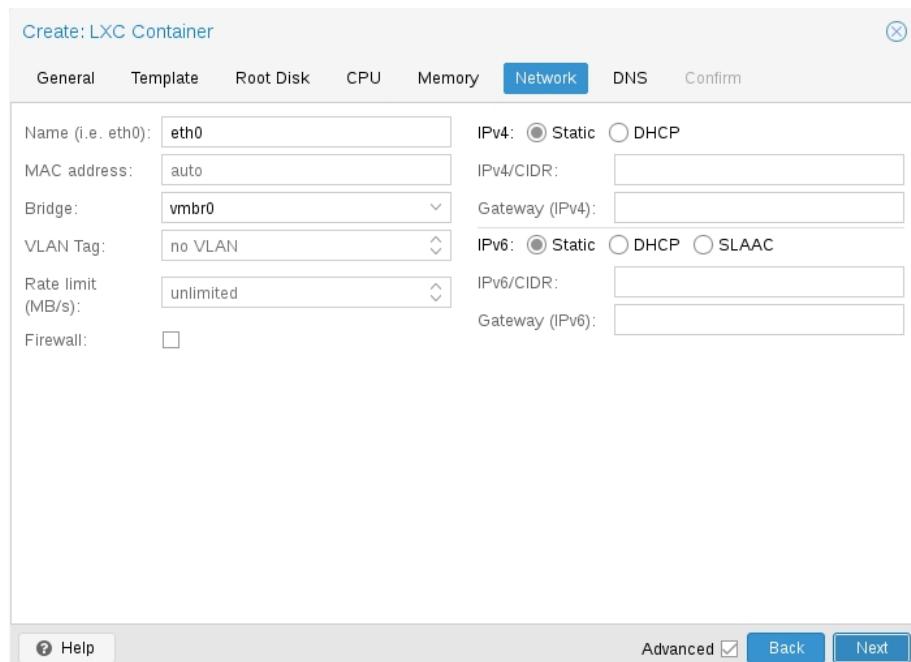
### 備考

デバイスマウント・ポイントは、特別な状況下でのみ使用すべきです。ほとんどの場合、ストレージバックされたマウントポイントは、同じパフォーマンスとより多くの機能を提供します。

### 備考

vzdumpを使用すると、デバイスマウント・ポイントの内容はバックアップされません。

## 11.4.5 ネットワーク



1つのコンテナに最大10個のネットワーク・インターフェースを設定できます。対応するオプションは

net0からnet9までで、以下の設定が可能です:

```
net[n]:name=<string> [,bridge=<bridge>] [,firewall=<1|0>] [,gw=<GatewayIPv4>]
[,gw6=<GatewayIPv6>] [,hwaddr=<XX:XX:XX:XX:XX:XX>] [,ip=<(IPv4/CIDR|dhcp|manual)>]
[,ip6=<(IPv6/CIDR|auto|dhcp|manual)>] [,link_down=<1|0>] [,mtu=<integer>] [,rate=<mbps>]
[,tag=<integer>] [,trunks=<vlanid[;vlanid...]>] [,type=<veth>]
```

コンテナのネットワーク・インターフェースを指定します。

### bridge= <ブリッジ>

ネットワークデバイスを接続ブリッジ。

**firewall= <boolean>**  
このインターフェイスのファイアウォールルールを使用するかどうかを制御します。

**gw= <GatewayIPv4>**  
IPv4トラフィックのデフォルトゲートウェイ。

**gw6= <GatewayIPv6>**  
IPv6トラフィックのデフォルトゲートウェイ。

**hwaddr= <xx:xx:xx:xx:xx:xx>。**  
I/G (Individual/Group) ビットが設定されていない共通のMACアドレス。

**ip= <(IPv4/CIDR|dhcp|manual)>**  
CIDRIPv4アドレス。

**ip6= <(IPv6/CIDR|auto|dhcp|manual)>**  
CIDRIPv6アドレス。

**link\_down= <boolean>**  
このインターフェイスを切断するかどうか（プラグを抜くように）。

**mtu= <整数> (64 - 65535)**  
インターフェースの最大転送単位。(lxc.network.mtu)

**name= <文字列>**  
コンテナ内部から見たネットワークデバイスの名前。(lxc.network.name)

**レート= <mbps>**  
インターフェイスにレート制限を適用

**タグ= <整数> (1 - 4094)**  
このインターフェイスのVLANタグ。

**trunks= <vlanid[;vlanid...]>。**  
インターフェイスを通過するVLAN ID

**タイプ= <ベース>**  
ネットワークインターフェースのタイプ。

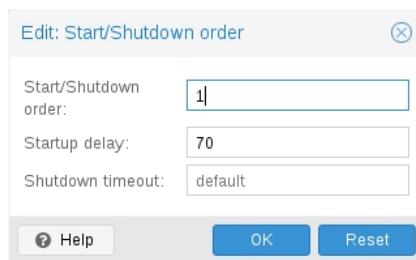
## 11.4.6 コンテナの自動スタートとシャットダウン

ホスト・システムのブート時にコンテナを自動的に起動するには、*Options* で *Start at boot* オプションを選択します。

パネルを開くか、次のコマンドを実行します：

```
# pct set CTID -onboot 1
```

### スタートとシャットダウンの順序



コンテナの起動順序を微調整したい場合は、以下のパラメータを使用できます：

- スタート/シャットダウン順：**開始順序の優先順位を定義します。例えば、CTを最初に起動させたい場合は設定します。(シャットダウンには逆の起動順序を使用するため、起動順序が1のコンテナは最後にシャットダウンされます)
- 起動遅延：**このコンテナの起動と後続のコンテナの起動の間隔を定義します。たとえば、他のコンテナを起動する前に 240 秒待つ場合は 240 に設定します。
- シャットダウンタイムアウト：**Proxmox VEがシャットダウンコマンドを発行した後、コンテナがオフラインになるまで待機する時間を秒単位で定義します。デフォルトでは、この値は60に設定されています。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秒待ちます。

Start/Shutdown順序パラメータが設定されていないコンテナは、常にパラメータが設定されているコンテナの後に開始することに注意してください。

ホストのブートと最初のコンテナのブートの間に遅延が必要な場合は、[Proxmox VE Node Management](#)のセクションを参照してください。

### 11.4.7 フックスクリプト

設定プロパティ `hookscript` で CT にフックスクリプトを追加できます。

```
# pct set 100 -hookscript local:snippets/hookscript.pl
```

このスクリプトはゲストが生きている間の様々な局面で呼び出されます。例とドキュメントは `/usr/share/pve-docs/examples/guest-example-hookscript.pl` にあるサンプルスクリプトを参照してください。

## 11.5 セキュリティ

コンテナはホストシステムのカーネルを使用します。そのため、悪意のあるユーザーにとって攻撃対象が露出することになります。一般的に、完全な仮想マシンの方がより優れた分離を提供します。コンテナを未知の人や信頼できない人に提供する場合は、この点を考慮する必要があります。

攻撃対象領域を減らすために、LXCはAppArmor、CGroups、kernel namespacesといった多くのセキュリティ機能を使用しています。

## 11.5.1 AppArmor

AppArmorプロファイルは、危険な可能性のあるアクションへのアクセスを制限するために使用されます。一部のシステムコール（マウントなど）は実行が禁止されています。

AppArmorのアクティビティをトレース、以下を使用します：

```
# dmesg | grep apparmor
```

推奨されませんが、AppArmorはコンテナに対して無効にすることができます。これにはセキュリティ・リスクが伴います。システムが誤って設定されている場合や、LXCまたはLinuxカーネルに脆弱性が存在する場合、コンテナ内で一部のシステムコールを実行すると、特権の昇格につながる可能性があります。

コンテナのAppArmorを無効にするには、以下の行を、以下の場所にあるコンテナ構成ファイルに追加します。

/etc/pve/lxc/CTID.conf:

```
lxc.apparmor.profile= アンコンファインド
```



### 警告

本番用にはお勧めできませんのでご注意ください。

## 11.5.2 対照群 (*cgroup*)

*cgroup* は、プロセスを階層的に整理し、システムリソースを分配するために使用されるカーネルのメカニズムです。

*cgroups*によって制御される主なリソースは、CPU時間、メモリとスワップの制限、デバイスノードへのアクセスです。*cgroups*は、スナップショットを取得する前にコンテナを「フリーズ」するためにも使用されます。

*cgroups* 現在、*legacy* 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」セットアップが使用され、リソース制御は主に *cgroupv1* で行われ、*cgroup\_no\_v1* カーネルコマンドラインパラメータで一部のサブシステムを引き継ぐことができる *cgroupv2* コントローラーが追加されました。（詳細は[カーネルパラメータのドキュメント](#)を参照してください）。

### CGroup バージョン互換性

Proxmox VE に関する純粋な *cgroupv2* と従来のハイブリッド環境の主な違いは、*cgroupv2* ではメモリとスワップが独立して制御されるようになったことです。以前はメモリの上限とメモリとスワップの合計の上限しか制限できませんでしたが、コンテナのメモリとスワップの設定はこれらの値に直接マッピングできます。

もう1つの重要な違いは、デバイスコントローラが全く異なる方法で設定されることです。このため、純粋な*cgroupv2*環境ではファイルシステムクォータは現在サポートされていません。

純粋な *cgroupv2* 環境で実行するには、コンテナの OS による *cgroupv2* サポートが必要です。*systemd* バージョン 231 以降を実行しているコンテナは *cgroupv2* をサポートしており<sup>1</sup>、*systemd* を init システムとして使用していないコンテナもサポートしています<sup>2</sup>。

<sup>1</sup> Proxmox VEから出荷されたコンテナテンプレートの最新メジャーバージョンをすべて含みます。

<sup>2</sup> 例えばアルペイン・リナックス

## 備考

CentOS 7とUbuntu 16.10は、*cgroupv2*環境で実行するには古すぎる*systemd*を持つ2つの著名なLinuxディストリビューションのリリースです。

- 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や20.04、CentOS 8（またはAlmaLinuxやRocky LinuxのようなRHEL/CentOSの派生版）などが考えられます。これには、最新のバグやセキュリティの修正、多くの場合新機能の追加、そしてEOLの時期を未来にずらすという利点があります。
- Containers *systemd* のバージョンをアップグレードします。ディストリビューションが *backports* リポジトリを提供している場合、これは簡単で迅速な応急処置になります。
- コンテナまたはそのサービスを仮想マシンに移動します。仮想マシンはホストとのインタラクションが非常に少ないため、何十年も前のバージョンのOSをインストールしても問題ありません。
- レガシーの*cgroup*コントローラに戻します。これは有効な解決策ですが、永久的なものではないことに注意してください。Proxmox VE 9.0以降、レガシーコントローラはサポートされなくなります。

## CGグループのバージョン変更

### チップ

ファイルシステムのクオータが必要なく、すべてのコンテナが*cgroupv2*をサポートしている場合は、新しいデフォルトに固執することをお勧めします。

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ーを使用します：

```
systemd.unified_cgroup_hierarchy=0
```

パラメータを追加する場所については、[このセクション](#)をカーネルポートコマンドラインの編集に関する参照してください。

## 11.6 ゲストオペレーティングシステムの構成

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ようとし、いくつかのファイルを変更します。以下は、コンテナ起動時に実行されることの簡単なリストです：

### etc/hostname を設定します。

コンテナ名の設定

### etc/hosts を修正

ローカルホスト名の検索を許可するには

### ネットワーク設定

完全なネットワーク・セットアップをコンテナに渡します。

### DNSの設定

DNSサーバーに関する情報を渡します。

### initシステムの適応

例えば、`getty`プロセスの生成数を修正します。

### ルートパスワードの設定

新規コンテナ作成時

### `ssh_host_keys` を書き換えます。

各コンテナが一意のキーを持つように

### crontabのランダム化

全てのコンテナで`cron`が同時に起動しないようにします。

Proxmox VEによる変更はコメントマーカーで囲まれています：

```
# --- pve開始 ---
<データ
# --- pve 終了 ---
```

これらのマーカーはファイル内の適切な位置に挿入されます。そのようなセクションがすでに存在する場合、そのセクションはそのまま更新され、移動されることはありません。

ファイルの変更は`.pve-ignore`.ファイルを追加することで防ぐことができます。例えば

`/etc/.pve-ignore.hosts`が存在する場合、`/etc/hosts`ファイルは変更されません。このファイルは単純な空のファイルでも構いません：

```
# touch /etc/.pve-ignore.hosts
```

ほとんどの修正はOSに依存するため、ディストリビューションやバージョンによって異なります。手動で`ostype`を`unmanaged`に設定することで、修正を完全に無効にすることができます。

OS タイプの検出は、コンテナ内の特定のファイルをテストすることで行われます。 Proxmox VE は、最初に

`/etc/os-release`ファイル<sup>3</sup>。このファイルがしないか、明確に認識できるディストリビューション識別子が含まれていない場合、以下のディストリビューション固有のリリースファイルがチェックされます。

### ウブントウ

`etc/lsb-release` (`DISTRIB_ID=Ubuntu`) を検査します。

### デビアン

`test /etc/debian_version`

### フェドラ

`test /etc/fedora-release`

### RedHat または CentOS

`test /etc/redhat-release`

### ArchLinux

`test /etc/arch-release`

<sup>3</sup> `/etc/os-release` は、ディストリビューションごとの多数のリリースファイル <https://manpages.debian.org/stable/systemd/os-release.5.en.html> を置き換えます。

### アルパイン

テスト /etc/alpine-release

### ジェンツー

test /etc/gentoo-release

#### 備考

設定された `ostype` が自動検出された異なる場合、コンテナの起動に失敗します。

## 11.7 コンテナ保管

Proxmox VE LXCコンテナストレージモデルは、従来のコンテナストレージモデルよりも柔軟です。コンテナは複数のマウントポイントを持つことができます。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ます。

例えば、コンテナのルートファイルシステムは低速で安価なストレージに置き、データベースは2つ目のマウント経由して高速で分散ストレージに置くことができます。詳細については、「[マウントポイント](#)」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ストレージタイプであれば何でも使用できます。つまり、コンテナはローカル (`lvm`、`zfs`、ディレクトリなど)、共有外部 (`iSCSI`、`NFS`など)、あるいはCephのような分散ストレージシステムにも保存できます。基礎となるストレージがサポートしていれば、スナップショットやクローンのような高度なストレージ機能も使用できます。`vzdump`バックアップツールは、スナップショットを使用して一貫性のあるコンテナバックアップを提供できます。

さらに、バインドマウントを使ってローカルデバイスやローカルディレクトリを直接マウントすることもできます。これにより、実質的にオーバーヘッドゼロでコンテナ内のローカルリソースにアクセスできます。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使用できます。

### 11.7.1 ヒューズマウント

#### 警告



Linuxカーネルのフリーザーサブシステムには既存の問題があるため、コンテナ内でFUSEマウントを使用することは強くお勧めできません。

FUSEマウントを他のマウント機構やストレージ技術で置き換えることができない場合は、Proxmoxホスト上でFUSEマウントを確立し、バインドマウントポイントを使用してコンテナ内でアクセスできるようにすることができます。

### 11.7.2 コンテナ内でのクォータの使用

クォータは、コンテナ内で各ユーザが使用できるディスク容量の制限を設定することができます。

#### 備考

これには現在、レガシーの `cgroup` を使用する必要があります。

**備考**

これはext4イメージベースのストレージタイプでのみ動作し、現在のところ特権コンテナでのみ動作します。

クオータ・オプションを有効にすると、マウント・ポイントに次のマウント・オプションが使用されます: `usrjquota=aquo`

これにより、他の同様にクオータを使用することができます。 `/aquota.user`を初期化することができます。

`/aquota.group`ファイルを実行します:

```
# quotacheck -cmug / #
quotao /
```

次に、`edquota` コマンドを使用してクオータを編集します。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。

**備考**

上記のコマンドを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、/だけでなくマウントポイントのパスを渡す必要があります。

### 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もコンテナ内で利用できます。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 コンテナマウントポイントのバックアップ

マウントポイントをバックアップに含めるには、コンテナ構成でそのマウントポイントのバックアップ有効にします。既存のマウントポイント `mp0` の場合

```
mp0: ゲスト:subvol-100-disk-1,mp=/root/files,size=8G
```

`backup=1` 追加して有効にします。

```
mp0: guests:subvol-100-disk-1,mp=/root/files,size=8G,backup=1.
```

**備考**

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・バックアップを無効にするには、上記の方法で `backup=0` を追加するか、**Backup** チェックします。

### 11.7.5 コンテナのマウントポイントのレプリケーション

デフォルトでは、ルートディスクがレプリケートされると、追加のマウントポイントがレプリケートされます。Proxmox VEストレージレプリケーションメカニズムでマウントポイントをスキップしたい場合は、そのマウントポイントに **【レプリケーションをスキップ】** オプションを設定します。Proxmox VE 5.0では、レプリケーションには`zfspool`タイプのストレージが必要です。コンテナでレプリケーションが設定されているときに別のタイプのストレージにマウントポイントを追加するには、そのマウントポイントで**レプリケーションをスキップする**オプションを有効にする必要があります。

## 11.8 バックアップと復元

### 11.8.1 コンテナのバックアップ

コンテナのバックアップにvzdump使用することは可能です。詳細はvzdumpのマニュアルページをご参照ください。

### 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のバックアップをリストアするには、`pct restore` コマンドを使用します。デフォルトでは、`pct restore` はバックアップされたコンテナ構成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で設定することで、バックアップされた構成を上書きすることができます（詳細については`pct` のマニュアルページを参照してください）。

---

#### 備考

`pvesm extractconfig` を使用すると、vzdump アーカイブに含まれるバックアップされた設定を表示できます。

---

基本的なリストアモードは2つあり、マウントポイントの扱いだけが異なります：

#### 「シンプル」リストアモード

`rootfs` パラメーターもオプションの `mpX` パラメーターも明示的に設定されていない場合、バックアップされた設定ファイルからマウントポイントの設定が以下の手順で復元されます：

1. バックアップからマウントポイントとそのオプションを抽出
2. ストレージパラメータ（デフォルト：ローカル）で指定されたストレージ上に、ストレージバックされたマウントポイント用のボリュームを作成します。
3. バックアップアーカイブからファイルを抽出
4. リストアされた設定にバインドポイントとデバイスマウントポイントを追加（rootユーザーに限定）

---

#### 備考

バインドとデバイスのマウントポイントは決してされないので、最後のステップではファイルはリストアされず、設定オプションだけがリストアされます。このようなマウントポイントは、別のメカニズム（多くのコンテナにバインドマウントされているNFSスペースなど）でバックアップされているか、まったくバックアップされることを意図していないかのいずれかであることが前提です。

---

このシンプル・モードは、ウェブ・インターフェイスのコンテナ・リストア操作でも使用されます。

## 「アドバンス」リストアモード

`rootfs` パラメータ (およびオプションで `mpX` パラメータの任意の組み合わせ) を設定することで、`pct restore` コマンドは自動的にアドバンスマードに切り替わります。このアドバンスマードは、バックアップアーカイブに含まれる `rootfs` と `mpX` の設定オプションを完全に無視し、代わりにパラメータとして明示的に提供されたオプションのみを使用します。

このモードでは、リストアマウントポイントの設定を柔軟に設定できます：

- 各マウントポイントのターゲットストレージ、ボリュームサイズ、その他のオプションを個別に設定します。
- 新しいマウントポイントスキームに従ってバックアップされたファイルを再配布します。
- デバイスおよび/またはバインドマウントポイントへのリストア (rootユーザーに限定)

## 11.9 pctによるコンテナの管理

Proxmox Container Toolkit" (`pct`)はProxmox VEコンテナを管理するコマンドラインツールです。コンテナの作成や破棄、コンテナ実行の制御（開始、停止、再起動、マイグレーションなど）が可能です。また、コンテナの設定ファイルにパラメータを設定することもできます。

### 11.9.1 CLIの使用例

Debian テンプレートに基づいてコンテナを作成します (ウェブインターフェースからテンプレートをダウンロード済みの場合)。

```
# pct create 100 /var/lib/vz/template/cache/debian-10.0-standard_10.0-1!
      _amd64.tar.gz
コンテナ 100
```

```
# pct start 100
```

gettyによるログインセッションの開始

```
# pct コンソール 100
```

LXCネームスペースに入り、rootユーザーとしてシェルを実行します。

```
# pct enter 100
```

設定の表示

```
# pct config 100
```

ホストブリッジ`vmbr0`にブリッジされた`eth0`というネットワークインターフェースを追加し、アドレスとゲートウェイを設定します。

```
# 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、そのコンテナは常にアクセス制御リストから削除され、そのコンテナのファイアウォール構成も常に削除されます。レプリケーション・ジョブ、バックアップ・ジョブ、およびHAリソース構成からもコンテナを削除する場合は、*-purge*を有効にする必要があります。

```
# 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、ソースCTからボリュームmp0が削除されターゲットCTにmp1としてアタッチされます。バックグラウンドでは、ボリュームの名前が新しい所有者と一致するように変更されます。

```
# pct move-volume 100 mp0 --target-vmid 200 --target-volume mp1
```

## 11.9.2 デバッグログの取得

pct startが特定のコンテナを起動できない場合、*--debug*フラグ（CTIDをコンテナのCTIDに置き換える）を渡してデバッグ出力を収集すると便利です：

```
# pct start CTID --debug
```

代わりに、以下のlxc-startコマンドを使用することもできます。このコマンドは、*-o output*オプションで指定されたデバッグ・ログを保存します：

```
# lxc-start -n CTID -F -l DEBUG -o /tmp/lxc-CTID.log
```

このコマンドは、フォアグラウンド・コンテナの起動を試みます。コンテナを停止するには、2番目のターミナルでpct shutdown CTID または pct stop CTID を実行します。

収集されたデバッグログは/*tmp/lxc-CTID.log*に書き込まれます。

### 備考

最後にpct startで起動を試みてからコンテナの構成を変更した場合は、少なくとも1回pct startを実行して、lxc-startで使用する構成も更新する必要があります。

## 11.10 移住

クラスタがある場合、コンテナのマイグレーションは

```
# pct migrate <ctid> <target>
```

これは、コンテナがオフラインである限り動作します。ローカルボリュームまたはマウントポイントが定義されている場合、同じストレージがターゲットホストに定義されていれば、移行はネットワーク経由でコンテンツをターゲットホストにコピーします。

実行中のコンテナは、技術的な制限によりライブマイグレーションできません。再起動マイグレーションは可能で、シャットダウンしてコンテナを移動し、ターゲットノード上でコンテナを再度起動します。コンテナは非常に軽量であるため、通常は数百ミリ秒のダウンタイムしか発生しません。

マイグレーションの再開は、ウェブ・インターフェイスから行うか、`pct migrate`コマンドで`--restart`フラグを使用して行うことができます。

再起動マイグレーションはコンテナをシャットダウンし、指定されたタイムアウト（デフォルトは180秒）の後にコンテナを強制終了します。その後、オフラインマイグレーションのようにコンテナをマイグレーションし、終了するとターゲットノード上でコンテナを起動します。

## 11.11 構成

`etc/pve/lxc/<CTID>.conf` ファイルにはコンテナ構成が格納されます。`<CTID>` は指定されたコンテナの数値 ID です。`etc/pve/` 内に格納されている他のすべてのファイルと同様に、他のすべてのクラスタノードに自動的に複製されます。

### 備考

`CTID< 100`は内部用に予約されており、`CTID`はクラスタ全体で一意である必要があります。

### コンテナの構成例

```
ostype: debian arch:  
amd64 hostname: www  
memory: 512  
スワップ512  
net0: bridge=vmbr0,hwaddr=66:64:66:64:64:36,ip=dhcp,name=eth0,type=veth rootfs: local:107/vm-107-disk-  
1.raw,size=7G。
```

設定ファイルはシンプルなテキストファイルです。通常のテキストエディタ、たとえば `vi` や `nano` を使って編集できます。ちょっとした修正に便利な場合もありますが、そのような変更を適用するにはコンテナを再起動する必要があることに注意してください。

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および変更するか、GUIを使用してすべてを行う方がよいでしょう。私たちのツールキットは、実行中のコンテナに対してほとんどの変更を即座に適用できるほど賢いです。この機能は「ホットプラグ」と呼ばれ、この、コンテナを再起動する必要はありません。

変更をホットプラグできない場合は保留中の変更として登録されます（GUI では赤色で表示されます）。変更が適用されるのは、コンテナを再起動した後です。

### 11.11.1 ファイル形式

コンテナ設定ファイルでは、コロンで区切られた単純なキー/値形式を使用します。各行の書式は以下のとおりです：

```
# これはコメントです。
```

これらのファイルの空白行は無視され、#文字で始まる行はコメントとして扱われ、これも無視されます。

例えば、低レベルのLXCスタイルのコンフィギュレーションを直接追加することも可能です：

```
lxc.init_cmd: /sbin/my_own_init
```

または

```
lxc.init_cmd= /sbin/my_own_init
```

この設定は、LXCの低レベルツールに直接渡されます。

## 11.11.2 スナップ写真

スナップショットを作成、`pct`はスナップショット時の設定と同じ設定ファイル内の別のスナップショット秒に保存します。例えば、"testsnapshot"というスナップショットを作成した後、設定ファイルは次のようにになります：

スナップショットによるコンテナ構成

```
メモリ512
スワップ512
親: テストナフォト
...
[testsnapshot] メモリ
: 512
スワップ512
スナップタイム: 1457170803
...
```

`parent` や `snaptime` のようなスナップショット関連のプロパティがいくつかあります。`parent` プロパティはスナップショット間の親子関係を保存するため使用されます。`snaptime` はスナップショットの作成タイムスタンプ (Unix epoch) です。

## 11.11.3 オプション

**arch: <amd64| arm64| armhf| i386| riscv32| riscv64> (デフォルト)= amd64)**

OSアーキテクチャの種類。

**cmode: <コンソール| シェル| tty> (デフォルト= tty)**

コンソールモード。デフォルトでは、`console`コマンドは利用可能なttyデバイスの1つに接続を開こうとします。`cmod`を`console`に設定すると、代わりに`/dev/console`にアタッチしようとします。`cmod`を`shell`に設定すると、コンテナ内で単にシェルを起動します（ログインはしません）。

**console: <論理値> (デフォルト= 1)**

コンテナにコンソールデバイス（`/dev/console`）をアタッチします。

**コア:<整数> (1 - 8192)**

コンテナに割り当てられたコアの数。コンテナは、デフォルトで使用可能なすべてのコアを使用できます。

**cpulimit:<数値> (0 - 8192) (デフォルト=0)**

CPU使用量の上限。

**備考**

コンピュータに2つのCPUがある場合、合計2つのCPU時間があります。値0はCPU制限なしを示します。

**cpuunits:<integer> (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、このコンテナはより多くのCPU時間を得ます。数値は、他のすべての実行中のゲストに対する相対値です。

**debug:<boolean> (デフォルト= 0)**

もっと冗長にしてみてください。今のところ、これは起動時にデバッグログレベルを有効にするだけです。

**説明:<文字列>**

コンテナの説明。WebインターフェイスCTのサマリーに表示されます。これは設定ファイルのコメントとして保存されます。

**dev[n]: [[パス=<パス>] [,gid=<整数>[,deny-write=<1|0>] [,gid=<整数>] [,mode=<オクタルのアクセスモード>] [,uid=<整数>]**

コンテナへの通過装置

**deny-write= <boolean> (デフォルト= 0)**

コンテナのデバイスへの書き込みを拒否

**gid= <整数> (0 - N)**

デバイスノードに割り当てるグループID

**mode= <オクタル・アクセス・モード>**

デバイス・ノードで設定するアクセス・モード

**パス= </パス>**

コンテナに通すデバイスへのパス

**uid= <整数> (0 - N)**

デバイスノードに割り当てるユーザーID

の機能があります: [force\_rw\_sys=<1|0>] [,fuse=<1|0>] [,keyctl=<1|0>] [,mknod=<1|0>] [,mount=<fstype;fstype;...>] [,nesting=<1|0>] です。

コンテナが高度なアクセスできるようにします。

**force\_rw\_sys= <boolean> (デフォルト= 0)**

非特権コンテナの /sys を mixed ではなく rw でマウントするようにしました。これは、新しい (>= v245) systemd-network の使用下でネットワークを壊す可能性があります。

**fuse= <boolean> (デフォルト= 0)**

コンテナ内での 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= <boolean> (デフォルト= 0)**

非特権コンテナ: keyctl() システムコールの使用を許可します。これは、コンテナ内で docker を使用するために必要です。デフォルトでは、非特権コンテナはこのシステムコールを存在しないと見なします。これは主に systemd-networkd の回避策で、keyctl() の操作がパーミッション不足でカーネルに拒否された場合に致命的なエラーとして扱われます。基本的には、systemd-networkd を走らせるか docker を走らせるかを選ぶことができます。

**mknod= <ブール値> (デフォルト = 0)**

非特権コンテナが mknod() を使用して特定のデバイスノードを追加できるようにしました。これには、ユーザ空間への seccomp トラップがサポートされたカーネルが必要です (5.3 以降)。これは実験的なものです。

**マウント= <fstype;fstype;...>**

特定のタイプのファイルシステムのマウントを許可します。これは、mount コマンドで使用するファイルシステムタイプのリストである必要があります。これは、コンテナのセキュリティに悪影響を及ぼす可能性があることに注意してください。ループデバイスへのアクセスでは、ファイルをマウントするとデバイス cgroup の mknod パーミッションを回避できます。

**nesting= <boolean> (デフォルト= 0)**

ネストを許可します。id マッピングを追加した非特権コンテナで使用するのが最適です。これは、ホストの procfs と sysfs の内容をゲストに公開することに注意してください。

**フックスクリプト:<文字列**

コンテナのライフタイムのさまざまなステップで実行されるスクリプト。

**ホスト名:<文字列**

コンテナのホスト名を設定します。

**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

**メモリ: <整数> (16 - N) (デフォルト = 512)**

コンテナのRAM容量 (MB)。

**mp[n]: ボリューム=<ボリューム> ,mp=<Path> [,acl=<1|0>] [,backup=<1|0>] [,mountoptions=<opt[;opt...]>] [,quota=<1|0>] [,replicate=<1|0>] [,ro=<1|0>] [,shared=<1|0>] [,size=<1|0>] [,マウントオプション=<opt[;opt...]>] [,quota=<1|0>] [,replicate=<1|0>] [,ro=<1|0>] [,shared=<1|0>] [,size=<DiskSize>]。**

ボリュームをコンテナ・マウント・ポイントとして使用します。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= <ブール値>**

ACL サポートを明示的に有効または無効にします。

**backup= <boolean>**

マウント・ポイントをバックアップに含めるかどうか（ボリューム・マウント・ポイントにのみ使用）。

**mountoptions= <opt[;opt...]>を指定します。**

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。

**備考**

セキュリティ上の、シンボリックリンクを含んではいけません。

**quota= <boolean>**

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= <boolean> (デフォルト= 1)**

このボリュームをストレージ・レプリカ・ジョブに含めます。

**ro= <ブール値>**

読み取り専用のマウントポイント

**shared= <boolean> (デフォルト= 0)**

この非ボリュームマウントポイントをすべてのノードで使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しません！

**size= <ディスクサイズ>**

ボリュームサイズ（読み取り専用値）。

**ボリューム= <ボリューム>**

コンテナにマウントするボリューム、デバイス、またはディレクトリ。

**ネームサーバー: <文字列>**

コンテナの DNS サーバー IP アドレスを設定します。searchdomainもnameserverも設定しない場合、Createは自動的にホストの設定を使用します。

```
net[n]:name=<string> [,bridge=<bridge>] [,firewall=<1|0>] [,gw=<GatewayIPv4>]
[,gw6=<GatewayIPv6>] [,hwaddr=<XX:XX:XX:XX:XX: XX>] [,ip=<(IPv4/CIDR|dhcp|manual)>]
[,ip6=<(IPv6/CIDR|auto|dhcp|manual)>] [,link_down=<1|0>] [,mtu=<integer>] [,rate=<mbps>]
[,tag=<integer>] [,trunks=<vlanid[,vlanid...]>] [,type=<veth>]
```

コンテナのネットワーク・インターフェースを指定します。

**bridge= <ブリッジ>**

ネットワークデバイスを接続ブリッジ。

**firewall= <boolean>**

このインターフェイスのファイアウォールルールを使用するかどうかを制御します。

**gw= <GatewayIPv4>**

IPv4トラフィックのデフォルトゲートウェイ。

**gw6= <GatewayIPv6>**

IPv6トラフィックのデフォルトゲートウェイ。

**hwaddr= <xx:xx:xx:xx:xx:xx>。**

I/G (Individual/Group) ビットが設定されていない共通のMACアドレス。

**ip= <(IPv4/CIDR|dhcp|manual)>**

CIDRIPv4アドレス。

**ip6= <(IPv6/CIDR|auto|dhcp|manual)>**

CIDRIPv6アドレス。

**link\_down= <boolean>**

このインターフェイスを切断するかどうか（プラグを抜くように）。

**mtu= <整数> (64 - 65535)**

インターフェースの最大転送単位。(lxc.network.mtu)

**name= <文字列>**

コンテナ内部から見たネットワークデバイスの名前。(lxc.network.name)

**レート= <Mbps>**

インターフェイスにレート制限を適用

**タグ= <整数> (1 - 4094)**

このインターフェイスのVLANタグ。

**trunks= <vlanid[,vlanid...]>。**

インターフェイスを通過するVLAN ID

**タイプ= <ベース>**

ネットワークインターフェースのタイプ。

**onboot: <boolean> (デフォルト = 0)**

システム起動時にコンテナを起動するかどうかを指定します。

**ostype:<alpine| archlinux| centos| debian| devuan| fedora | gentoo | nixos | opensuse |ubuntu | unmanaged>**

OSタイプ。これは、コンテナ内の設定を行うために使用され、/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* は、OS固有のセットアップをスキップするために使用できます。

**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/更新操作を防止します。

**rootfs: ボリューム=><ボリューム> [,acl=<1|0>] [,mountoptions=<opt[;opt...]>] [,quota=<1|0>]  
[,replicate=<1|0>] [,ro=<1|0>] [,shared=<1|0>]**

**[,size=<1|0>[quota=<1|0>][,replicate=<1|0>][,ro=<1|0>][,shared=<1|0>][,size=<DiskSize>]。**

ボリュームをコンテナのルートとして使用します。

**acl= <ブール値>**

ACL サポートを明示的に有効または無効にします。

**mountoptions= <opt[;opt...]>**

roots/mps 用の追加マウントオプション。

**quota= <boolean>**

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= <boolean> (デフォルト= 1)**

このボリュームをストレージ・レプリカ・ジョブに含めます。

**ro= <ブール値>**

読み取り専用のマウントポイント

**shared= <boolean> (デフォルト= 0)**

この非ボリュームマウントポイントをすべてのノードで使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しません!

**size= <ディスクサイズ>**

ボリュームサイズ (読み取り専用値)。

**ボリューム= <ボリューム>**

コンテナにマウントするボリューム、デバイス、またはディレクトリ。

**searchdomain: <文字列>**

コンテナのDNS検索ドメインを設定します。searchdomainもnameserverも設定しない場合、Createはホストからの設定を自動的に使用します。

**を起動します: [[order=]↑ [,up=d+] [,down=d+`**

スタートアップとシャットダウンの動作。Orderは一般的な起動順序を定義する非負の数値です。シャットダウンは逆の順序で行われます。さらに、次のVMが起動または停止するまでの待ち時間を秒単位で設定できます。

**swap: <整数> (0 - N) (デフォルト = 512)**

MB単位のコンテナのSWAP量。

**タグ: <string>**

コンテナのタグ。これは単なるメタ情報です。

**テンプレート: <boolean> (デフォルト= 0)**

テンプレートの有効/無効。

**timezone: <文字列>**

コンテナで使用するタイムゾーン。オプションが設定されていない場合は、何もれません。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* から任意のタイムゾーンオプションを設定することができます。

**tty: <整数> (0 - 6) (デフォルト= 2)**

コンテナで利用可能なttyの数を指定

**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。(手動で変更してはいけません)。

**unused[n] です: [ボリューム=]<ボリューム>**

未使用ボリュームへの参照。これは内部的に使用されるので、手動で変更しないでください。

**ボリューム= <ボリューム>**

現在使用されていないボリューム。

## 11.12 錠前

コンテナの移行、スナップショット、およびバックアップ (*vzdump*) は、影響を受けるコンテナに対する互換性のない同時実行を防止するためにロックを設定します。このようなロックを手動で削除する必要がある場合があります（電源障害後など）。

```
# pct アンロック <CTID>
```

**注意**

ロックを設定したアクションがもう実行されていないことを確認している場合のみ、この操作を行ってください。

## 第12章

# ソフトウェア定義ネットワーク

Proxmox VEのSDN(Software-Defined Network)機能は、仮想ゾーンとネットワーク(VNet)の作成を可能にします。この機能により、高度なネットワーク構成とマルチテナントの簡素化されます。

## 12.1 はじめに

Proxmox VE SDN は、柔軟でソフトウェア制御のコンフィギュレーションを使用して、仮想ゲストネットワークの分離ときめ細かな制御を可能にします。

分離はゾーン、仮想ネットワーク（VNets）、サブネットによって管理されます。ゾーンは仮想的に分離された独自のネットワーク領域です。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。

ゾーンのタイプによって、ネットワークの動作は異なり、特定の機能、利点、制限を提供します。

SDN のユースケースは、個々のノード上の孤立したプライベートネットワークから、異なる場所にある複数の PVE クラスタにまたがる複雑なオーバーレイネットワークまで多岐にわたります。

クラスタ全体のデータセンタ SDN 管理インターフェイスで VNet を設定した後、各ノードでローカルに共通の Linux ブリッジとして利用でき、VM やコンテナに割り当てることができます。

## 12.2 サポート状況

### 12.2.1 沿革

Proxmox VE SDN スタックは 2019 年から実験的な機能として提供されており、多くの開発者とユーザによって継続的に改善とテストが行われてきました。Proxmox VE 6.2 のウェブインターフェースへの統合により、より広範な統合に向けた重要なマイルストーンが達成されました。Proxmox VE 7 のリリースサイクルでは、数多くの改良と機能が追加されました。ユーザーからのフィードバックに基づき、基本的な設計の選択とその実装が非常に健全で安定していることが明らかになりました。従って、'実験的' というラベルは SDN スタックの状態を正当に評価するものではありませんでした。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 のアクセス制御スタックのコアコンポーネントに昇格させることで、SDN 機能の完全な統合のための基礎を築くという決定がなされました。Proxmox VE 8.1 では、二つの大きなマイルストーンが達成されました。第一に、IP アドレス管理 (IPAM) 機能に DHCP の統合が追加され、第二に、SDN の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

SDN インストールの様々なレイヤーの現在のサポート状況は以下の通りです:

- VNet 管理と Proxmox VE スタックとの統合を含むコア SDN は完全にサポートいます。
- 仮想ゲストのDHCP管理を含むIPAMは技術レビュー中です。
- FRRoutingとコントローラ統合による複雑なルーティングは技術レビュー中です。

## 12.3 インストール

### 12.3.1 SDNコア

Proxmox VE 8.1 以降、Software-Defined Network (SDN) のコアパッケージがデフォルトでインストールされます。

古いバージョンからアップグレードする場合は、各ノードに `libpve-network-perl` パッケージをインストールする必要があります:

アプトアップデート

```
apt install libpve-network-perl
```

#### 備考

Proxmox VEバージョン7.0以上では、デフォルトで`ifupdown2`パッケージがインストールされています。それ以前のバージョンでシステムをインストールした場合は、`ifupdown2`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、`/etc/network/interfaces`の最後に以下の行があることを確認してください。

コンフィギュレーションファイルをすべてのノードに置き、SDN コンフィギュレーションが含まれてアクティブになるようにします。

ソース `/etc/network/interfaces.d/*`

### 12.3.2 DHCP IPAM

PVE内蔵IPアドレス管理スタックへのDHCP統合は、現在DHCPリースの提供に`dnsmasq`を使用しています。これは現在オプトインです。

この機能を使うには、各ノードに`dnsmasq`パッケージをインストールする必要があります:

アプトアップデート

```
apt 実行 dnsmasq
# デフォルトのインスタンスを無効化 systemctl
disable --now dnsmasq
```

### 12.3.3 FRRルーティング

Proxmox VE SDN スタックは高度なセットアップのために [FRRouting](#) プロジェクトを使います。これは現在オプトインです。SDN ルーティングの統合を使うには、全てのノードに `frr-pythontools` パッケージをインストールする必要があります:

アプトアップデート

```
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・レベルのWeb UIで行われ、以下のセクションに分かれています:

- **SDN:** ここで現在のアクティブな SDN 状態の概要が得られ、保留中のすべての変更をクラスタ全体に適用できます。
- **ゾーン**:仮想的に分離されたネットワークゾーンの作成と管理
- **VNets VNets**: 仮想ネットワークブリッジの作成とサブネットの管理

Optionsカテゴリでは、SDNセットアップで使用する追加サービスを追加・管理できます。

- **コントローラー**: 複雑なセットアップでのレイヤー3ルーティング制御用
- **DHCP**: IPAM内のゲストにIPを自動的に割り当て、DHCP経由でゲストにリースするゾーンのDHCPサーバーを定義します。
- **IPAM**: ゲストの外部IPアドレス管理を有効にします。
- **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 サーバーの統合を定義します。

## 12.5 技術と構成

Proxmox VE Software-Defined Network の実装は可能な限り標準的な Linux ネットワークを使います。その理由は、最新の Linux ネットワーキングは SDN 実装に必要なほとんど全ての機能を提供し、外部依存の追加を避け、壊れる可能性のあるコンポーネントの総量を減らすことができるからです。

Proxmox VE SDN 設定は `/etc/pve/sdn` にあり、Proxmox VE [設定ファイルシステムを通して](#)他の全てのクラスタノードと共有されます。これらのコンフィギュレーションは、基礎となるネットワークスタックを管理するツールのそれぞれのコンフィギュレーションフォーマット（例えば `ifupdown2` や `frr`）に変換されます。

新しい変更はすぐに適用されるのではなく、まず保留として記録されます。その後、ウェブインターフェイスのメインの SDN 概要パネルで一連の異なる変更を一度に適用することができます。このシステムは様々な変更を単一のアトミックなものとしてロールアウトすることができます。

SDN は `/etc/pve/sdn` にある `.running-config` と `.version` ファイルを通してロールアウト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。ゾーンは特定のノードに制限され、ユーザーを特定のゾーンとそのゾーンに含まれるVネットに制限するために、パーミッションが割り当てられます。

分離にはさまざまな技術を用いることができます：

- シンプル：孤立ブリッジ。シンプルなレイヤ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方法です。
- QinQ：スタックドVLAN（正式名称：IEEE 802.1ad）
- VXLAN：UDPトンネル経由のレイヤー2 VXLANネットワーク
- evpn (bgp evpn)：レイヤ3ルーティングを確立するためのBGP付きVXLAN

### 12.6.1 共通オプション

以下のオプションはすべてのゾーンタイプで利用可能です：

#### ノード

ゾーンと関連するVNetsを配置ノード。

#### アイパム

ゾーン内のIPを管理するためにIPアドレス管理(IPAM)ツールを使用します。オプション、デフォルトは pve。

#### DNS

DNS APIサーバー。オプション。

#### リバースDNS

リバースDNS APIサーバー。オプション。

#### DNSゾーン

DNSドメイン名。<ホスト名>.<ドメイン名>のようにホスト名を登録するときに使用します。DNSゾーンはDNSサーバー上にすで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

これは最もシンプルなプラグインです。孤立したVNetブリッジを作成します。このブリッジは物理インターフェイスにリンクされておらず、VMトラフィックは各ノードのローカルのみになります。NATまたはルーティングされたセットアップで使用できます。

## 12.6.3 VLANゾーン

VLAN プラグインは、既存のローカル Linux または OVS ブリッジを使用してノードの物理インターフェイスに接続します。VLAN プラグインは、VNet で定義された VLAN タギングを使用してネットワークセグメントを分離します。これにより、異なるノード間での VM の接続が可能になります。

VLANゾーン設定オプション：

### ブリッジ

ノード間の接続を可能にする、各ノードに設定済みのローカルブリッジまたはOVSスイッチ。

## 12.6.4 QinQゾーン

QinQ は VLAN スタッキングとも呼ばれ、複数の VLAN タグを使用して分離します。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。

### 備考

この構成では、物理ネットワークスイッチがスタックドVLANをサポートしている必要があります。

QinQゾーン設定オプション：

### ブリッジ

各ローカルノードに設定済みのローカルVLAN対応ブリッジ

### サービスVLAN

このゾーンのメインVLANタグ

### サービスVLANプロトコル

802.1q (デフォルト) または 802.1ad サービス VLAN タイプを選択できます。

### エムティーユー

タグの二重スタックのため、QinQ VLAN にはさらに 4 バイトが必要です。たとえば、物理インターフェイスの MTU が 1500 の、MTU を 1496 に減らす必要があります。

## 12.6.5 VXLANゾーン

VXLAN プラグインは、既存のネットワーク（アンダーレイ）の上にトンネル（オーバーレイ）を確立します。これは、レイヤー2のイーサネットフレームを、デフォルトの宛先ポート 4789 を使用するレイヤー4の UDP データグラム内にカプセル化します。

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。

例えば、パブリック・インターネットの上に VXLAN オーバーレイ・ネットワークを作成し、あたかも同じローカル・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 は、それ自体では暗号化を提供しません。VXLAN 経由で複数のサイトに参加する場合は、サイト間 VPN を使用するなどして、サイト間で安全な接続を確立してください。

VXLANゾーン設定オプション:

#### ピアアドレスリスト

VXLANゾーン内の各ノードのIPアドレスのリスト。このアドレスで到達可能な外部ノードでもかまいません。クラスタ内のすべてのノードをここに記載する必要があります。

#### エムテ

イーウ VXLAN カプセル化は 50 バイトを使用するため、MTU は発信物理インターフェイスより 50 バイト低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を利用して実現されます。

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、仮想ゲストがゲートウェイとしてこのアドレスを使用できることを意味します。

ルーティングはVRF（Virtual Routing and Forwarding）インターフェイスを通じて、異なるゾーンのVNet間で機能します。

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。VNets の VXLAN-ID 異なる必要があります。

#### コントローラー

このゾーンで使用するEVPNコントローラ。(コントローラのプラグインのセクションを参照)。

#### VNet MACアドレス

このゾーン内のすべてのVネットに割り当てられるエニーキャストMACアドレス。定義されていない場合は自動生成されます。

#### 出口ノード

EVPNネットワークから実ネットワークを経由する出口ゲートウェイとして設定されるノード。設定されたノードはEVPNネットワークのデフォルトルートをアナウンスします。オプション。

#### 一次出口ノード

複数の出口ノードを使用する場合は、すべてのノードで負荷分散を行う代わりに、このプライマリ出口ノードを介してトラフィックを強制します。オプションですが、SNATを使用する場合、またはアップストリームルーターがECMPをサポートしていない場合は必要です。

#### 出口ノード ローカルルーティング

これは、出口ノードからVM/CTサービスに到達する必要がある場合の特別なオプションです。(デフォルトでは、出口ノードはリアルネットワークとEVPNネットワーク間のトラフィック転送のみを許可します)。オプションです。

### サブネットの広告

EVPNネットワーク内のフルサブネットをアナウンスします。サイレントVM/CTがある場合(例えば、複数のIPがあり、これらのIPからのトラフィックを見ない場合、IPアドレスはEVPNネットワーク内に到達できません)。オプション。

### ARP ND抑制の無効化

ARPやND (Neighbor Discovery) パケットを抑制しないでください。これは、VMでフローティングIPを使用している場合に必要です (IPアドレスとMACアドレスがシステム間で移動されます)。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るようにします。クロスDCまたは異なるEVPNネットワークの相互接続に使用します。オプションです。

### エムテ

**イーウ** VXLAN カプセル化は 50 バイトを使用するため、MTU は発信物理インターフェイスの最大 MTU よりも 50 バイト少なくする必要があります。オプションで、デフォルトは 1450 です。

## 12.7 ブイネット

SDN GUI で仮想ネットワーク (VNet) を作成すると、各ノードで同じ名前のローカルネットワークインターフェースが利用可能になります。ゲストを VNet に接続するには、インターフェイスをゲストに割り当て、それに応じて IP アドレスを設定します。

ゾーンによって、これらのオプションの意味は異なり、本書の各ゾーンのセクションで説明されています。



#### 警告

現状では、一部のオプションは効果がなかったり、特定のゾーンで機能しないことがあります。

VNet設定オプション:

#### 身分証明書

VNetを識別するための最大8文字のID

#### コメント

より説明的な識別子。インターフェースのエイリアスとして割り当てられます。オプション

#### ゾーン

このVNetの関連ゾーン

#### タグ

固有のVLANまたはVXLAN ID

#### VLAN対応

インターフェイスの vlan 対応オプションを有効にし、ゲストでの設定を有効にします。

### 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。つまり、ゲストは非絶縁ブリッジポート（ブリッジ自体）にのみトラ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。

### 備考

ポートの分離は各ホストに対してローカルに行われます。ノード間で VNET のトラフィックをさらに分離するには [VNET ファイアウォール](#)を使用します。例えば、デフォルトでDROPし、IPサブネットからゲートウェイへのトラ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

- 特定のVNetに定義できるIPアドレスの制限
- レイヤー3ゾーンのVNetにルート/ゲートウェイを割り当て
- レイヤー3ゾーンのVNetでSNATを有効化
- IPAMプラグインによる仮想ゲスト(VMまたはCT)へのIP自動割り当て
- DNSプラグインによるDNS登録

サブネットゾーンにIPAMサーバーが関連付けられている場合、サブネットプレフィックスは自動的にIPAMに登録されます。

サブネット構成オプション：

### 身分証明書

CIDRネットワークアドレス（例：10.0.0.0/8

### ゲートウェイ

ネットワークのデフォルトゲートウェイのIPアドレス。レイヤー3ゾーン（Simple/EVPNプラグイン）では、VNet上に配置されます。

### SNAT

ソースNATを有効にすると、パケットをノードの送信インターフェイスに転送することで、VNet内のVMが外部ネットワークに接続できるようになります。EVPNゾーンでは、転送はEVPNゲートウェイノードで行われます。オプション。

### DNSゾーンプレフィックス

<ホスト名>.prefix.<ドメイン>のように、ドメイン登録にプレフィックスを追加します。

## 12.9 コントローラー

ゾーンによっては、制御プレーンとデータプレーンを分離して実装しているため、VNetの制御プレーンを管理する外部コントローラが必要になります。

現在、外部コントローラを必要とするのは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、コントロールプレーンを管理するために外部コントローラが必要です。EVPN コントローラプラグインは、Free Range Routing (frr) ルータを設定します。

EVPNコントローラを有効にするには、各ノードでFRRを有効にする必要があります。EVPN コントローラの設定オプション

---

#### ASN #

一意のBGP ASN番号。プライベートな ASN 番号 (64512 - 65534, 4200000000 - 4294967294) を使用することを強くお勧めします。

#### ピアーズ

EVPNゾーンの全ノードのIPリスト。(外部ノードまたはルートリフレクタサーバも可能)

### 12.9.2 BGPコントローラ

BGP コントローラはゾーンによって直接使用されません。BGPピアを管理するためにFRRを設定するために使うことができます。

BGP-EVPNでは、ノードごとに異なるASNを定義してEBGPを行うことができます。また、外部のBGPピアにEVPNルートをエクスポートするためにも使用できます。

---

#### 備考

デフォルトでは、シンプルなフルメッシュEVPNの場合、BGPコントローラを定義する必要はありません。

---

BGPコントローラの設定オプション:

#### ノード

このBGPコントローラのノード

#### ASN #

一意のBGP ASN番号。プライベートASN番号は、(64512)、(64512)、(64512)、(64512)の範囲で使用することを強くお勧めします。  
- 65534) または (4200000000 - 4294967294) を指定します。

#### ピア

基礎となるBGPネットワークを使用して通信したいピアIPアドレスのリスト。

#### EBGP

ピアのリモートASが異なる場合、EBGPを有効にします。

### ループバックインターフェース

EVPNネットワークのソースとしてループバックまたはダミーインターフェイスを使用します（マルチパス用）。

### エップルミューティホップ

ピアが直接接続されていない場合やループバックを使用している場合は、ピアに到達するためのホップ数を増やします。

### bgp-multipath-as-path-relax

ピアのASNが異なる場合、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によって直接使用されません。ISIS ドメインにEVPNルートをエクスポートするためにFRRを設定するために使用できます。

ISIS コントローラ構成オプション:

### ノード

この ISIS コントローラのノード。

### ドメイン

ISIS独自のドメイン。

### ネットワークエンティティ名

このノードを識別する一意の ISIS ネットワーク・アドレス。

### インターフェース

ISIS が使用する物理インターフェースのリスト。

### ループバック

EVPNネットワークのソースとしてループバックまたはダミーインターフェイスを使用します（マルチパス用）。

## 12.10 アイパム

IPアドレス管理(IPAM)ツールはネットワーク上のクライアントのIPアドレスを管理します。Proxmox VE の SDN は IPAM を使用して、例えば新しいゲストのために空いている IP アドレスを見つけます。

1つのIPAMインスタンスは、1つまたは複数のゾーンに関連付けることができます。

### 12.10.1 PVE IPAM プラグイン

Proxmox VEクラスタのデフォルトの組み込みIPAMです。

PVE IPAM Plugin の現在のステータスは、データセンター設定の SDN セクションにある IPAM パネルで確認できます。この UI を使用して、IP マッピングの作成、更新、および削除を行うことができます。[DHCP機能と](#)併用すると特に便利です。

DHCP を使用している場合、IPAM パネルを使用して特定の VM に対してリースを作成または編集することができます。DHCP を使用している VM の IP を編集する場合は、ゲストに新しい DHCP リースを取得させる必要があります。これは通常、ゲストのネットワークスタックをリロードするか、ゲストを再起動することで可能です。

## 12.10.2 NetBox IPAM プラグイン

NetBoxは、オープンソースのIPアドレス管理（IPAM）およびデータセンター・インフラストラクチャ管理（DCIM）ツールです。

NetBox と Proxmox VE SDN を統合するには、以下の説明に従って NetBox で API トークンを[作成します](https://docs.netbox.d ja/stable/integrations/rest-api/#tokens): <https://docs.netbox.d ja/stable/integrations/rest-api/#tokens>

NetBoxの設定プロパティは以下の通りです：

### URL

NetBox REST API エンドポイント: `http://yournetbox.domain.com/api`

### トークン

APIアクセストークン

## 12.10.3 phplPAMプラグイン

phplPAM 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 admin 権限を持つ API トークンを追加する必要があります。phplPAM の設定プロパティ

は次のとおりです：

### URL

REST-API エンドポイント: `http://phpipam.domain.com/api/<appname>/`

### トークン

APIアクセストークン

### セクション

整数の ID。セクションは phplPAM のサブネットのグループです。デフォルトのインストールでは `sectionid=1` を使用します。  
お客様のために

## 12.11 DNS

Proxmox VE SDN の DNS プラグインは、ホスト名と IP アドレスを登録するための DNS API サーバを定義するために使用します。DNS 設定は 1 つ以上のゾーンに関連付けられ、ゾーンに設定されたすべてのサブネット IP に対して DNS 登録を提供します。

## 12.11.1 PowerDNSプラグイン

<https://doc.powerdns.com/authoritative/http-api/index.html>

PowerDNSの、ウェブサーバーとAPIを有効にする必要があります:

```
api=はい
api-key=arandomgeneratedstring webserver=yes
ウェブサーバーポート=8081
```

PowerDNSの設定オプションは以下の通りです:

### url

REST API エンドポイント : <http://yourpowerdnsdomain.com:8081/api/v1/servers/localhost>

### キー

APIアクセスキー

### トル

レコードのデフォルト TTL

## 12.12 ディーエイチシーピー

Proxmox VE SDN の DHCP プラグインを使用すると、ゾーンに DHCP サーバを自動的に配置できます。DHCP 範囲が設定されているゾーン内のすべてのサブネットに DHCP を提供します。現在、DHCP で使用可能なバックエンドプラグインは dnsmasq プラグインだけです。

DHCPプラグインは、VM/CTに新しいネットワークインターフェイスを追加する際に、ゾーンに設定されたIPAMプラグインでIPを割り当てることで動作します。IPAMの設定方法については、[ドキュメントの各セクション](#)を参照してください。

VMが起動すると、ゾーンのDHCPプラグインにMACアドレスとIPのマッピングが作成されます。ネットワークインターフェイスが削除されるか、VM/CTが破棄されると、IPAMとDHCPサーバのエントリも削除されます。

---

### 備考

現在、一部の機能（IPマッピングの追加/編集/削除）は、[PVE IPAMプラグイン](#)を使用した場合にのみ利用可能です。

---

## 12.12.1 構成

Web UIでゾーンの自動DHCPを有効にするには、「ゾーン」パネルを使用し、ゾーンの詳細オプションで有効にします。

---

### 備考

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が有効になったら、ゾーンサブネットにDHCP範囲を設定する必要があります。そのためには、[Vnets]パネルに移動して、DHCP範囲を設定したいサブネットを選択します。編集ダイアログでは、それぞれのタブでDHCP範囲を設定できます。または、以下のCLIコマンドを使用して、サブネットのDHCP範囲を設定することもできます：

```
pvsh set /cluster/sdn/vnets/<vnet>/subnets/<subnet>
-dhcp-range start-address=10.0.1.100,end-address=10.0.1.200
-dhcp-range start-address=10.0.2.100,end-address=10.0.2.200
```

また、サブネットにゲートウェイを設定する必要があります - そうしないと自動DHCPは機能しません。DHCPプラグインは、設定された範囲でのみIPAMにIPを割り当てます。

[dnsmasq DHCPプラグインのインストール](#)忘れないでください。

## 12.12.2 プラグイン

### Dnsmasqプラグイン

現在、これは唯一のDHCPプラグインであり、したがってゾーンでDHCPを有効にするときに使用されるプラグインです。

#### 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。

#### 構成

プラグインは dnsmasq がデプロイさゾーンごとに新しい systemd サービスを作成します。サービス名は `dnsmasq@<zone>` です。このサービスのライフサイクルは DHCP プラグインによって管理されます。

プラグインは、`/etc/dnsmasq.d/<zone>` フォルダに以下の設定ファイルを自動的に生成します：

#### 00-default.conf

dnsmasqインスタンスのデフォルトのグローバル構成が含まれます。

#### 10-<zone>-<subnet\_cidr>.conf

このファイルは、DHCP経由で設定されるべきDNSサーバーなど、サブネットの特定のオプションを設定します。

#### 10-<zone>-<subnet\_cidr>.ranges.conf

このファイルは、dnsmasqインスタンスのDHCP範囲を構成します。

#### エーテル

このファイルには、IPAMプラグインからのMACアドレスとIPマッピングが含まれています。これらのマッピングを上書きするには、dnsmasqプラグインによって上書きされるため、このファイルを編集するのではなく、それぞれのIPAMプラグインを使用してください。

上記のファイルはDHCPプラグインによって管理されるため、編集してはいけません。dnsmasqの設定をカスタマイズするには、設定フォルダに追加のファイル（90-custom.confなど）を作成します。

設定ファイルは順番に読み込まれるので、カスタム設定ファイルに適切な名前を付けることで、設定ディレクティブの順番を制御することができます。

DHCPリースは/var/lib/misc/dnsmasq.<zone>.leasesファイルに保存されます。

PVE IPAMプラグインを使用すると、DHCPリースの更新、作成、削除ができます。詳細については、[PVE IPAMプラグインのドキュメント](#)を参照してください。  
他のIPAMプラグインでは、DHCPリースの変更は現在サポートされていません。

## 12.13 ファイアウォールの統合

SDNは、ファイアウォールルールのソース/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VNetsとIPAMエントリに対して自動的に行われます。

### 12.13.1 vネットとサブネット

ファイアウォールはすべてのVNetに対して、SDNスコープ内に以下のIPSetを自動的に生成します：

#### ネットオール

VNet内のすべてのサブネットのCIDRを含みます。

#### ネットゲートウェイ

VNet内のすべてのサブネットのゲートウェイのIPを含みます。

#### vnet-no-gateway

VNet内のすべてのサブネットのCIDRを含みますが、ゲートウェイは含みません。

#### vnet-dhcp

VNetのサブネットに設定されたすべてのDHCPレンジを含みます。

設定を変更すると、IPSetsは自動的に更新されるため、サブネットの設定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。

#### シンプルゾーンの例

VNetとそのサブネットについて以下のコンフィギュレーションを仮定します：

```
# /etc/pve/sdn/vnets.cfg
vnet: vnet0
    ゾーンシンプル
# /etc/pve/sdn/subnets.cfg
```

```
サブネット: simple-192.0.2.0-24 vnet vnet0
    dhcp-range 開始アドレス=192.0.2.100,終了アドレス=192.0.2.199 ゲートウェイ 192.0.2.1
```

```
サブネット: simple-2001:db8::64 vnet vnet0
    dhcp-range 開始アドレス=2001:db8::1000,終了アドレス=2001:db8::1999 ゲートウェイ 2001:db8::1
```

この例では、VNet vnet0に192.0.2.0/24をIPレンジとするIPv4サブネットを設定しました、  
192.0.2.1がゲートウェイで、DHCPの範囲は192.0.2.100～192.0.2.199です。

さらに、2001:db8::64をIPレンジ、2001:db8::1をゲートウェイ、2001:db8::1000～2001:db8::1999をDHCPレンジとするIPv6サブネットを設定しました。

vnet0用に自動生成された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-ゲートウェイ**

- 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

## **12.13.2 アイパム**

内蔵のPVE IPAMを使用している場合、ファイアウォールはIPAMにエントリがあるゲストごとにIPセットを自動的に生成します。ID 100のゲスト用のそれぞれのIPセットはguest-ipam-100になります。これは、すべてのIPAMエントリからのすべてのIPアドレスを含みます。したがって、ゲスト100が複数のVNetsのメンバーである場合、IPsetにはすべてのVNetsからのIPが含まれます。

エントリーが追加/更新/削除されると、それに応じてそれぞれのIPSetが更新されます。

**警告**

ゲストのすべてのエントリを削除したときに、自動生成されたIPS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と、ファイアウォールは存在しないIPS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。

## 12.14 例

この節では、一般的な SDN のユースケースに合わせた複数の設定例を示します。利用可能な設定オプションの理解を深めるために追加的な詳細を提供し、具体的な実装を提供することを目的としています。

### 12.14.1 シンプルゾーンの例

シンプルゾーンネットワークは、単一のホスト上のゲストが接続するための隔離されたネットワークを作成します。

**チップ**

ゲスト間の接続は、すべてのゲストが同じホストに存在し、他のノードに到達できない場合に可能です。

- simpleという名前のシンプルなゾーンを作成します。
- VNet名vnet1を追加します。
- ゲートウェイとSNATオプションを有効にしてサブネットを作成します。
- 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するゲストに割り当て、IPアドレスを設定します。

2つのVMのネットワーク・インターフェイスのコンフィギュレーションは、10.0.1.0/24ネットワーク経由で通信できるように、次のようにになります。

```
allow-hotplug ens19 iface ens19
inet static
    アドレス 10.0.1.14/24
```

```
allow-hotplug ens19 iface ens19
inet static
    アドレス 10.0.1.15/24
```

### 12.14.2 ソースNATの例

シンプルネットワークゾーンでゲストの発信接続を許可する場合、シンプルゾーンにはソースNAT (SNAT) オプションがあります。

[上記の](#)設定から、VNet vnet1にサブネットを追加し、ゲートウェイIPを設定し、SNATオプションを有効にします。

```
サブネット: 172.16.0.0/24 ゲート  
ウェイ: 172.16.0.1 SNAT: チェック  
済み
```

ゲストでは、サブネットのIPレンジ内で静的IPアドレスを設定します。

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のNATゲートウェイとして機能します。

### 12.14.3 VLAN設定例

異なるノード上のVMが分離されたネットワークを介して通信する必要がある場合、VLANゾーンはVLANタグを使用してネットワークレベルの分離を可能にします。

myvlanzone という名前の VLAN ゾーンを作成します:

```
ID: myvlanzone ブリッジ:  
vmbr0
```

VLANタグ10と先に作成したmyVlanzoneでmyVnet1というVNetを作成します。

```
ID: myvnet1 Zone:  
myvlanzone Tag: 10
```

メイン SDN パネルから設定を適用し、各ノードにローカルに VNets を作成します。node1 上に Debian ベースの仮想マシン (vm1) を作成し、vNIC を myvnet1 にします。

このVMには、以下のネットワーク構成を使用します:

```
auto eth0  
iface eth0 inet static  
    アドレス 10.0.3.100/24
```

ノード2上に、vm1と同じVNet myvnet1上にvNICを持つ2台目の仮想マシン (vm2) を作成します。この仮想マシンには以下のネットワーク構成を使用します:

```
auto eth0  
iface eth0 inet static  
    アドレス 10.0.3.101/24
```

これに従って、そのネットワークを使用して両方のVM間でpingを実行できるはずです。

### 12.14.4 QinQセットアップの例

この例では、2つのQinQゾーンを設定し、各ゾーンに2つのVMを追加して、より分離されたVLANの設定を可能にするVLANタグの追加レイヤを示します。

この構成の典型的なユースケースは、ホスティング・プロバイダーがVM通信のために顧客に隔離されたネットワークを提供し、VMを他の顧客から隔離する場合です。

サービスVLAN 20でqinqzone1という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でqinqzone2という別のQinQゾーンを作成します。

```
ID: qinqzone2 ブリッジ:  
vmbr0 サービス VLAN: 30
```

先に作成したqinqzone1ゾーンに、VLAN-ID 100のmyVnet1というVNetを作成します。

```
ID: qinqvnet1 Zone:  
qinqzone1 Tag: 100
```

qinqzone2ゾーンにVLAN-ID 100のmyVnet2を作成します。

```
ID: qinqvnet2 Zone:  
qinqzone2 Tag: 100
```

メインの SDN ウェブインターフェースパネルで設定を適用して、各ノードでローカルに VNets を作成します。

Debian ベースの仮想マシンを 4 台 (vm1、vm2、vm3、vm4) 作成し、vm1 と vm2 にブリッジ qinqvnet1 を、vm3 と vm4 にブリッジ qinqvnet2 を使用してネットワークインターフェイスを追加します。

VM内部では、/etc/network/interfacesなどでインターフェースのIPアドレスを設定します：

```
auto eth0  
iface eth0 inet static  
    アドレス 10.0.3.101/24
```

4つのVMすべてに10.0.3.101～10.0.3.104のIPアドレスを設定します。

これで、VM vm1とvm2の間、およびvm3とvm4の間でpingが送れるようになります。しかし、VM vm1、vm2のどちらもVM vm3、vm4に対してpingを打つことができません。VM vm3、vm4はサービスVLANが異なるゾーンにあるためです。

## 12.14.5 VXLANの設定例

この例では、ノードIPアドレスが192.168.0.1、192.168.0.2、192.168.0.3の3つのノードを持つクラスタを想定しています。

myvxlanzone という名前の VXLAN ゾーンを作成し、ノードからのすべての IP をピアアドレスリストに追加します。デフォルトの MTU 1450 を使用するか、適宜設定します。

```
ID: myvxlanzone  
ピアのアドレスリスト 192.168.0.1, 192.168.0.2, 192.168.0.3
```

```
ID: vxvnet1 Zone:  
myvxlanzone Tag: 100000
```

先に作成したVXLANゾーンmyvxlanzoneを使用して、vxvnet1というVNetを作成します。

メイン SDN ウェブインターフェースパネルで設定を適用して、各ノードにローカルに VNets を作成します。node1 上に Debian ベースの仮想マシン (vm1) を作成し、vxvnet1 上に vNIC を作成します。

このVMには、以下のネットワーク構成を使用します (MTUが低いことに注意してください)。

```
auto eth0
iface eth0 inet static
    アドレス 10.0.3.100/24 mtu 1450
```

ノード3上に、vm1と同じVNet vxvnet1上にvNICを持つ2台目の仮想マシン (vm2) を作成します。この仮想マシンには以下のネットワーク構成を使用します:

```
auto eth0
iface eth0 inet static
    アドレス 10.0.3.101/24 mtu 1450
```

すると、vm1とvm2の間でpingが送れるようになるはずです。

## 12.14.6 EVPNの設定例

この例では、IPアドレスが192.168.0.1、192.168.0.2、192.168.0.3の3つのノード (node1、node2、node3) を持つクラスタを想定しています。

プライベートASN番号と上記のノードアドレスをピアとして使用して、EVPNコントローラを作成します。

```
ID: myevpnctl ASN#:
65000
ピア192.168.0.1,192.168.0.2,192.168.0.3
```

myevpnzoneという名前のEVPNゾーンを作成し、先に作成したEVPN-コントローラーを割り当て、次のように定義します。

node1とnode2を終了ノードとします。

```
ID: myevpnzone
VRF VXLAN タグ10000
コントローラ: myevpnctl MTU:
1450
VNet MACアドレス: 32:f4:05:fe:6c:0a
終了ノード: node1,node2
```

EVPNゾーンmyevpnzoneを使用して、myvnet1という最初のVNetを作成します。

```
ID: myvnet1 Zone:
myevpnzone Tag: 11000
```

myvnet1にサブネットを作成します:

```
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番目のVNetを作成します。

```
ID: myvnet2 Zone:  
myevpnzone Tag: 12000
```

サブネット: 10.0.2.0/24 ゲート

ウェイ: 10.0.2.1

myvnet2`に別のサブネットを作成します:

メイン SDN ウェブインターフェースパネルからコンフィグレーションを適用して、各ノードにローカルに VNets を作成し、FRR コンフィグレーションを生成します。

node1 上に Debian ベースの仮想マシン (vm1) を myvnet1 上の vNIC で作成します。vm1 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 10.0.1.100/24 ゲートウェ  
    イ 10.0.1.1  
    ムチュ1450
```

もう1つのVNet myvnet2上にvNICを持つ2つ目の仮想マシン (vm2) をnode2上に作成します。vm2には以下のネットワーク構成を使用します:

```
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  
    ムチュ1450
```

これでvm1からvm2へ、vm1からvm2へpingが打てるようになります。

ゲートウェイでないnode3のvm2から外部IPにpingを打つと、パケットは設定されたmyvnet2ゲートウェイに行き、その後、出口ノード (node1またはnode2) にルーティングされ、そこからnode1またはnode2に設定されたデフォルトゲートウェイを経由してそれらのノードから離脱します。

#### 備考

外部ゲートウェイのnode1とnode2に10.0.1.0/24と10.0.2.0/24ネットワークのリバースルートを追加して、パブリックネットワークが返信できるようにする必要があります。

外部BGPルータを設定した場合、BGP-EVPNルート（この例では10.0.1.0/24と10.0.2.0/24）は動的にアナウンスされます。

## 12.15 備考

### 12.15.1 複数のEVPN出口ノード

複数のゲートウェイノードがある場合、`rp_filter` (Strict Reverse Path Filter) オプションを無効に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
```

## 12.15.2 VXLAN IPSEC暗号化

VXLAN の上に IPSEC 暗号化を追加するために、この例では strongswan を使用する方法を示します。

暗号化を処理するために、IPv4の場合は60バイト、IPv6の場合は80バイト、MTUを減らす必要があります。

従って、デフォルトのリアル1500 MTUでは、1370のMTUを使う必要があります ( $1370 + 80 (\text{IPSEC}) + 50 (\text{VXLAN}) == 1500$ )。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だけを暗号化する必要があります。

```
conn %default
    ike=aes256-sha1-modp1024!           # 最も速いがそこそく安全な暗号
    esp=aes256-sha1!
    leftfirewall=yes。
    ファイアウォールルール             # これはProxmox VEを使用する際に必要です。←'
```

### コネクション出力

```
rightsubnet=%dynamic[udp/4789] right=%any
type=transport
authby=psk
auto=route
```

### コネクション入力

```
leftsubnet=%dynamic[udp/4789] type=transport
authby=psk
auto=ルート
```

で事前共有キーを生成します:

```
openssl rand -base64 128
```

そして、その鍵を`/etc/ipsec.secrets`に追加して、ファイルの内容が次のようにします:

```
:PSK <generatedbase64key>.
```

VXLANネットワークに参加しているすべてのノードにPSKとコンフィグレーションをコピーします。

## 第13章

# Proxmox VE ファイアウォール

Proxmox VE Firewallは、ITインフラストラクチャを保護する簡単な方法を提供します。クラスタ内のすべてのホストのファイアウォールルールを設定したり、仮想マシンやコンテナのルールを定義したりできます。ファイアウォールマクロ、セキュリティグループ、IPセット、エイリアスなどの機能が、このタスクを容易にします。

すべての設定はクラスタ・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、仮想マシン間で完全に分離されます。また、このシステムは分散型であるため、中央のファイアウォール・ソリューションよりもはるかに高い帯域幅を提供します。

ファイアウォールはIPv4とIPv6をフルサポートしています。IPv6のサポートは完全に透過的で、デフォルトで両方のプロトコルのトラフィックをフィルタリングします。そのため、IPv6用に別のルールセットを維持する必要はありません。

## 13.1 道順とゾーン

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。各ゾーンのルールは個別に定義できます。ゾーンに応じて、受信、送信、転送トラフィックのルールを定義できます。

### 13.1.1 道順

ゾーンのルールを定義する際、3つの方向から選ぶことができます：

で

ゾーンに到着しているトラフィック。

アウト

ゾーンを出るトラフィック。

フォワード

ゾーンを通過するトラフィック。ホストゾーンでは、ルーティングされたトラフィックになります（ホストがゲートウェイとして動作している場合やNATを実行している場合）。VNetレベルでは、ブリッジ接続されたネットワークインターフェイスのトラフィックを含め、VNetを通過するすべてのトラフィックに影響します。

**重要**

転送されたトラフィックのルールを作成することは、現在のところ、新しい [nftables ベースの proxmox-firewall](#) を使用している場合にのみ可能です。いかなる転送ルールも純正の `pve-firewall` では無視され、何の効果もありません！

### 13.1.2 ゾーン

ファイアウォールルールを定義ゾーンは3種類あります：

**ホスト**

ホストから/ホストへ向かうトラフィック、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。

**かそうけいさんき**

VMまたはCTを発着するトラフィック。転送されるトラフィックに対してルールを定義することはできません。

**ブイネット**

SDN VNet を通過するトラフィックで、ゲストからゲストへ、あるいはホストからゲストへ、あるいはその逆。このトラフィックは常に転送されるトラフィックなので、方向が `forward` のルールしか作成できません。

**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](#) を使っている場合にのみ可能です。VNet レベルのルールは純正の `pve-firewall` では無視され、何の効果もありません！

## 13.2 設定ファイル

ファイアウォール関連の設定はすべて proxmox クラスタファイルシステムに保存されます。そのため、これらのファイルは自動的にすべてのクラスタノードに配布され、`pve-firewall` 変更時に自動的に基礎となる `iptables` 更新します。

GUI（Datacenter→Firewall や Node→Firewall など）を使用して設定することもできますし、お好みのエディタを使用して設定ファイルを直接編集することもできます。

ファイアウォール設定ファイルには、キーと値のペアのセクションがあります。で始まる行と空白行はコメントとみなされます。セクションは [ ] で囲まれたセクション名を含むヘッダー行で始まります。

### 13.2.1 クラスタ全体の設定

クラスタ全体のファイアウォール設定は、次の場所に保存されます：

`/etc/pve/firewall/cluster.fw`

コンフィギュレーションには以下のセクションがあります:

### [オプション]

クラスタ全体のファイアウォールオプションを設定します。

**ebtables: <boolean> (デフォルト= 1)**

クラスタ全体でebtablesルールを有効にします。

**enable: <整数> (0 - N)**

ファイアウォールクラスター全体を有効または無効にします。

**log\_ratelimit: [enable=<1|0> [,burst=<integer>] [,rate=<rate>] です。**

ログ・レイトリミットの設定

**burst= <整数> (0 - N) (デフォルト= 5)**

レートが適用される前に常に記録されるパッケージの初期バースト

**enable= <boolean> (デフォルト= 1)**

ログレート制限の有効化または無効化

**rate= <rate> (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: <ACCEPT | DROP>**

前向きな方針。

**policy\_in: <ACCEPT | DROP | REJECT>**

入力ポリシー。

**policy\_out: <ACCEPT | DROP | REJECT>**

出力ポリシー。

### [ルール]

このセクションには、全ノードのクラスタ全体のファイアウォールルールが含まれます。

**[IPSET <名前>]。**

クラスタ全体のIPセット定義。

**[GROUP <名前>]。**

クラスタ全体のセキュリティグループ定義。

### [ALIASES]

クラスタ全体のエイリアス定義。

## ファイアウォールの有効化

ファイアウォールはデフォルトでは完全に無効になっていますので、ここで有効オプションを設定する必要があります:

### [オプション]

```
# enable firewall クラスタ全体の設定、デフォルトは無効) enable: 1
```



### 重要

ファイアウォールを有効にすると、デフォルトですべてのホストへのトラフィックがブロックされます。唯一の例外は、ローカルネットワークからのWebGUI(8006)とssh(22)です。

リモートから Proxmox VE ホストを管理する場合は、リモート IP から Web GUI (ポート 8006) へのトラフィックを許可するルールを作成する必要があります。また、ssh(ポート22)やSPICE(ポート3128)も許可する必要があります。

### チップ

ファイアウォールを有効にする前に、Proxmox VEホストの1つにSSH接続を開いてください。そうすれば、何か問題が発生してもホストにアクセスできます。

この作業を単純化するには、代わりに「管理」という名前のIPSセットを作成し、すべてのリモートIPをそこに追加します。これにより、リモートからGUIにアクセスするために必要なファイアウォールルールがすべて作成されます。

## 13.2.2 ホスト固有の設定

ホスト関連の設定を:

```
/etc/pve/nodes/<nodename>/host.fw
```

cluster.fw設定からルールを上書きしたい場合に便利です。また、ログの冗長性を上げたり、ネットフィルタ関連のオプションを設定することもできます。コンフィギュレーションには以下のセクションを含めることができます:

### [オプション]

ホスト関連のファイアウォールオプションを設定します。

#### enable: <ブール値>

ホストファイアウォールルールを有効にします。

**log\_level\_forward: <alert| crit| debug| emerg| err| info | nolog | notice | warning>** です。

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>** です。

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>** です。

発信トラフィックのログレベル。

**log\_nf\_conntrack:<boolean> (デフォルト= 0)**

conntrack 情報のロギングを有効にします。

**ndp:<boolean> (デフォルト= 0)**

NDP (Neighbor Discovery Protocol) を有効にします。

**nf\_conntrack\_allow\_invalid:<boolean> (デフォルト= 0)**

接続追跡時に無効なパケットを許可します。

**nf\_conntrack\_helpers:<string> (デフォルト='')**

特定のconntrack 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., netbios-ns, pptp, sane, sip, snmp, tftp

**nf\_conntrack\_max:<整数> (32768 - N) (デフォルト= 262144)**

追跡接続の最大数。

**nf\_conntrack\_tcp\_timeout\_established:<整数> (7875 - N)(= 432000)**

CONNTRACK がタイムアウトしました。

**nf\_conntrack\_tcp\_timeout\_syn\_recv:<整数> (30 - 60) (デフォルト= 60)**

Conntrack syn recv タイムアウト。

**nftables:<ブール値> (デフォルト= 0)**

nftablesベースのファイアウォールの有効化 (技術プレビュー)

**nosmurfs:<ブール値>**

SMURFS有効にします。

**protection\_synflood:<boolean> (デフォルト= 0)**

シンフラッド 防御の有効化

**protection\_synflood\_burst:<整数> (デフォルト= 1000)**

ip srcによるシンフラッドプロテクションレートバースト。

**protection\_synflood\_rate:<整数> (デフォルト= 200)**

ip srcによるシン・フラッド・プロテクション・レート syn/sec。

**smurf\_log\_level:<alert| crit| debug| emerg| err| info| nolog**

|お知らせ| 警告

SMURFSログレベル。

**tcp\_flags\_log\_level:** <alert| crit| debug| emerg| err| info | nolog | notice | warning>。

不正なtcpフラグログレベル。

**tcpflags:** <boolean> (デフォルト= 0)

TCP フラグの不正な組み合わせをフィルタリングします。

#### [ルール]

このセクションにはホスト固有のファイアウォールルールが含まれます。

### 13.2.3 VM/コンテナの構成

VM のファイアウォール設定を：

/etc/pve/ファイアウォール/<VMID>.fw

そして以下のデータを含んでいます：

#### [オプション]

VM/Container 関連のファイアウォールオプションを設定します。

**dhcp:** <boolean> (デフォルト= 0)

DHCPを有効にします。

**enable:** <ブール値> (デフォルト= 0)

ファイアウォールルールを有効/無効にします。

**ipfilter:** <ブール値>

デフォルトIP フィルタを有効にします。これは各インターフェイスに空の ipfilter-net<id> ipset を追加するのと同じです。このような ipset は、IPv6 リンクのローカルアドレスをインターフェイスの MAC アドレスに由来するものに制限するような、まともなデフォルトの制限を暗黙的に含んでいます。コンテナに対しては、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> です。

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> です。

発信トラフィックのログレベル。

**macfilter:** <boolean> (デフォルト= 1)

MAC アドレスフィルターの有効/無効。

**ndp:** <boolean> (デフォルト= 0)

NDP (Neighbor Discovery Protocol) を有効にします。

**policy\_in: <ACCEPT | DROP | REJECT>**

入力ポリシー。

**policy\_out: <ACCEPT | DROP | REJECT>**

出力ポリシー。

**radv: <ブール値**

ルーター広告の送信を許可します。

#### [ルール]

このセクションには、VM/コンテナファイアウォールルールが含まれます。

#### [IPSET <名前>]。

IPセットの定義。

#### [ALIASES]

IPエイリアスの定義。

### VMとコンテナのファイアウォールの有効化

各仮想ネットワーク・デバイスは、それぞれ独自のファイアウォール有効フラグを持っています。そのため、インターフェイスごとにファイアウォールを選択的に有効にすることができます。これは、一般的なファイアウォール有効オプションに加えて必要です。

## 13.2.4 VNetコンフィギュレーション

VNet関連の読み込みます:

```
/etc/pve/sdn/firewall/<vnet_name>.fw
```

VNet内の各VMに対して個別にファイアウォール・ルールを設定することなく、VNetレベルでグローバルにファイアウォール設定を行うことができます。受信トラフィックと送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・インターフェースを含め、ブリッジ・ポートから別のポートに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チェインを通過するトラフィックは双方向なので、トラフィックを双方向に通過させたい場合は、双方向のルールを作成する必要があります。例えば、特定のホストに対するHTTPトラフィックを許可する場合、以下のルールを作成する必要があります:

```
FORWARD ACCEPT -dest 10.0.0.1 -dport 80  
FORWARD ACCEPT -source 10.0.0.1 -sport 80
```

## [オプション]

VNet 関連のファイアウォールオプションを設定します。

**enable: <ブール値> (デフォルト= 0)**

ファイアウォールルールを有効/無効にします。

**log\_level\_forward: <alert| crit| debug| emerg| err| info | nolog | notice | warning>** です。

転送トラフィックのログレベル。

**policy\_forward: <ACCEPT | DROP>**

前向きな方針。

## [ルール]

このセクションには、VNet 固有のファイアウォール ルールが含まれています。

### 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、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できます。マクロには、定義済みのルールとオプションのセットが含まれます。ルールの前に | を付けると、ルールを無効にできます。

ファイアウォールルールの構文

[ルール]

方向アクション [オプション]

| DIRECTION ACTION [OPTIONS] # 無効ルール

DIRECTION MACRO(ACTION) [OPTIONS] # 定義済みのマクロを使用

以下のオプションを使用して、ルールの一致を絞り込むことができます。

#### --dest <文字列>

パケットの宛先制限します。これは、単一のIPアドレス、IPセット(+ipsetname)、またはIPエイリアスの参照できます。また、20.34.101.207-201.3.9.99のようなアドレス範囲や、IPアドレスとネットワークのリスト（エントリはカンマで区切られます）を指定することもできます。このようなリストの中にIPv4アドレスとIPv6アドレスを混在させないでください。

#### --dport <文字列>

TCP/UDP宛先ポートを制限します。etc/services で定義されているように、サービス名または単純な番号 (0 ~ 65535) を使用できます。ポート範囲は、80:85 のように @@+:@@+ で指定でき、カンマ区切りのリストを使って複数のポートまたはポート範囲に一致させることができます。

#### --icmp-type <文字列>

icmp-type を指定します。proto が icmp または icmpv6/ipv6-icmp の場合のみ有効。

**--iface <文字列>**

ネットワークインターフェース名。VMとコンテナにはネットワーク構成キー名を使用する必要があります (*netd+*)。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 <alert| crit| debug| emerg| err| info| nolog| notice| warning>**

ファイアウォールルールのログレベル。

**-プロト <文字列>**

IPプロトコル。*etc/protocols*で定義されているプロトコル名 (*tcp/udp*) や単純な数字を使うことができます。

**--ソース <文字列>**

パケットの送信元アドレスを制限します。これは、単一のIPアドレス、IPセット (+ipsetname)、またはIPエイリアスの参照できます。また、*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（エントリはカンマで区切られます）を指定することもできます。このようなリストの中にIPv4アドレスとIPv6アドレスを混在させないでください。

**--スポーツ <文字列>**

TCP/UDPソース・ポートを制限します。*etc/services*で定義されているように、サービス名または単純な番号 (0 ~ 65535) を使用できます。ポート範囲は、*80:85*のように *端:端* で指定することができます。

いくつか例を挙げましょう：

```
[ルール]
IN SSH(ACCEPT) -i net0
IN SSH(ACCEPT) -i net0 # コメント
IN SSH(ACCEPT) -i net0 -source 192.168.2.192 # 'からの SSH だけを許可します。
192.168.2.192
IN SSH(ACCEPT) -i net0 -source 10.0.0.1-10.0.0.10 # IPレンジのSSHを受け入れる IN SSH(ACCEPT) -i net0 -source
10.0.0.1,10.0.0.2,10.0.0.3 # のSSHを受け入れる '
IPリスト
IN SSH(ACCEPT) -i net0 -source +mynetgroup # ipsetssh を受け入れる'
マイネットグループ
IN SSH(ACCEPT) -i net0 -source myserveralias # エイリアスの ssh を受け入れる'
サーバーエイリアス

| IN SSH(ACCEPT) -i net0 # 無効ルール

IN DROP # 全ての受信パッケージを廃棄 OUT ACCEPT # 全ての送信パッケ
ージを受理
```

## 13.4 セキュリティグループ

セキュリティ・グループとは、クラスタ・レベルで定義されるルールの集まりで、すべてのVMのルールで使用できます。例えば、*http*と*https*のポートを開くルールを持つ "webserver" という名前のグループを定義することができます。

```
# /etc/pve/firewall/cluster.fw

[グループ・ウェブサーバー]
IN ACCEPT -p tcp -dport 80 IN ACCEPT -p
tcp -dport 443
```

次に、このグループをVMのファイアウォールに追加します。

```
# /etc/pve/firewall/<VMID>.fw

[ルール]
GROUP ウェブサーバー
```

## 13.5 IPエイリアス

IPエイリアスを使用すると、ネットワークのIPアドレスを名前に関連付けることができます。そして、その名前を参照することができます：

- インサイドIPセット定義
- ファイアウォールルールの送信元と送信先のプロパティで

### 13.5.1 標準IPエイリアス local\_network

このエイリアスは自動的に定義されます。割り当てられた値を確認するには、以下のコマンドを使用してください：

```
# pve-firewall localnet ローカルホ
スト名: example
ローカルIPアドレス192.168.2.100 ネットワーク自動検出:
192.168.0.0/20
検出されたlocal_networkを使用しています: 192.168.0.0/20
```

ファイアウォールは、このエイリアスを使用してクラスタ通信（corosync、API、SSH）に必要なすべてのものを許可するルールを自動的に設定します。

ユーザはcluster.fw aliasセクションでこれらの値を上書きできます。パブリックネットワーク上で単一のホストを使用する場合は、明示的にローカルIPアドレス

```
# /etc/pve/firewall/cluster.fw [ALIASES].
local_network 1.2.3.4 # シングルIPアドレスを使用
```

## 13.6 IPセット

IPセットは、ネットワークとホストのグループを定義するために使用できます。ファイアウォールルールのsourceとdestプロパティで'+name'を使って参照できます。

以下の例では、管理IPセットからのHTTPトラフィックを許可しています。

```
IN HTTP(ACCEPT) -ソース +管理
```

## 13.6.1 標準IPセット管理

このIPセットはホストファイアウォールのみに適用されます(VM ファイアウォールではありません)。これらのIPは通常の管理タスク(Proxmox VE GUI、VNC、SPICE、SSH)を実行できます。

ローカル・クラスタ・ネットワークは自動的にこのIPセット(alias cluster\_network)に追加され、ホスト間のクラスタ通信が可能になります。(マルチキヤスト、ssh、……)

```
# /etc/pve/firewall/cluster.fw
```

```
[IPSET管理] 192.168.2.10  
192.168.2.10/24
```

## 13.6.2 標準IPセットブラックリスト

これらのIPからのトライフィックは、各ホストとVMのファイアウォールによってドロップされます。

```
# /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\*

これらのフィルターはVMのネットワーク・インターフェースに属し、主にIPスプーフィングを防ぐために使用されます。このようなセットがインターフェイスに存在する場合、そのインターフェイスの対応するipfilterセットに一致しないソースIPを持つ発信トライフィックはすべてドロップされます。

IPアドレスが設定されているコンテナでは、これらのセットが存在する場合（またはVMのファイアウォールのオプションタブで一般的なIPフィルターオプションを使用して有効になっている場合）、関連するIPアドレスが暗黙的に含まれます。

仮想マシンとコンテナの両方で、近隣発見プロトコルを動作させるために、標準のMAC由来のIPv6リンクローカルアドレスも暗黙的に含まれています。

```
/etc/pve/ファイアウォール/<VMID>.fw
```

```
[IPSET ipfilter-net0] # net0 192.168.2.10上の指定されたIPのみを許可します。
```

## 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 の置き換え)。
- pve-firewall: iptables ルールの更新

また、`pve-firewall`というCLIコマンドもあり、これを使用してファイアウォールサービスを開始および停止することができます：

```
# pve-firewall start # pve-
firewall stop
```

ステータスを取得するには

```
# pve-firewall status
```

上記のコマンドはすべてのファイアウォールルールを読み込んでコンパイルするので、ファイアウォール設定にエラーがある場合は警告が表示されます。

生成されたiptablesのルールを見たい場合は、次のようにします：

```
# 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 データセンター着信/発信 DROP/REJECT

ファイアウォールの入力または出力ポリシーがDROPまたはREJECTに設定されている場合でも、クラスタ内のすべてのProxmox VEホストで次のトラフィックが許可されます：

- ループバックインターフェース上のトラフィック
- すでに確立された接続
- IGMPプロトコルを使用したトラフィック
- ウェブインターフェースへのアクセスを許可するため、管理ホストからのTCPトラフィックをポート8006へ
- 管理ホストからのTCPトラフィックは、VNCウェブコンソールのトラフィックを許可するポート範囲5900～5999へ。
- 管理ホストからのTCP トラフィックは、SPICE プロキシへの接続用にポート 3128 に送られます。
- 管理ホストからのTCP トラフィックをポート22に送り、sshアクセスを許可します。
- クラスタネットワークのUDPトラフィックをポート5405-5412に送ってcorosyncします。
- クラスタネットワークのUDPマルチキャストトラフィック
- ICMP トラフィック タイプ 3 (Destination Unreachable)、4 (congestion control)、または 11 (Time Exceeded) 以下のトラフィックはドロップされます。ログには記録されません。

- 無効な接続状態のTCPコネクション
- corosyncに関連しないブロードキャスト、マルチキャスト、エニーキャストのトラフィック、すなわちポート5405-5412を経由しないトラフィック

- ポート43へのTCPトラフィック
- ポート135と445へのUDPトラフィック
- UDPトラフィックはポート範囲137から139へ
- 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
- 135、139、445番ポートへのTCPトラフィック
- 送信元ポート53からのUDPトラフィック

残りのトラフィックは、それぞれドロップまたは拒否され、ログにも記録されます。これは、NDP、SMURFS、TCP フラグフィルタリングなど、**ファイアウォール→オプション**で有効になっている追加オプションによって異なる場合があります。

の出力を確認してください。

```
# iptables-save
```

システム・コマンドを使用して、システムでアクティブなファイアウォール・チェーンとルールを確認できます。この出力はシステム・レポートにも含まれ、Web GUI のノードのサブスクリプション・タブ、または pverereport コマンドライン・ツールからアクセスできます。

### 13.8.2 VM/CT着信/発信 DROP/REJECT

設定されたコンフィグレーションに応じて、DHCP、NDP、ルーター広告、MAC、IP フィルタリングなどの例外を除き、VM へのすべてのトラフィックをドロップまたは拒否します。パケットをドロップ/拒否するルールはデータセンターから引き継がれ、ホストの送受信トラフィックの例外は適用されません。

繰り返しになりますが、[iptables-save \(上記参照\)](#)を使って、適用されたすべてのルールとチェインを検査することができます。

## 13.9 ファイアウォールルールのログ

デフォルトでは、ファイアウォールルールによってフィルタされたトラフィックのログはすべて無効になっています。ロギングを有効にするには、受信および/または送信トラフィックのログレベルを**ファイアウォール→オプション**で設定する必要があります。これは、VM/CT ファイアウォールだけでなく、ホストに対しても個別に行うことができます。これにより、Proxmox VEの標準ファイアウォール ルールのロギングが有効になり、**ファイアウォール→ログ**に出力さ。さらに、標準ルール ([デフォルトのファイアウォールルールを参照](#)) では、一部のドロップまたは拒否されたパケットのみがログに記録されます。

`loglevel` は、フィルタされたトラフィックがどの程度ログに記録されるかに影響しません。これは、フィルタリングと後処理を容易にするために、ログ出力に接頭辞として付加されるLOGIDを変更します。

`loglevel` は以下のフラグのいずれかです:

ログレベル	ロジッド
ノログ	-
エマージェンシ	0
ー	
アラート	1
クリティック	2

ログレベル	ロジッド
誤る	3
警告	4
お知らせ	5
インフォーメーション	6
デバッグ	7

典型的なファイアウォールのログ出力は次のようにになります:

```
vmid logid chain timestamp policy: PACKET_DETAILS
```

ホストファイアウォールの場合、VMIDは0になります。

### 13.9.1 ユーザー定義のファイアウォールルールのログ

ユーザー定義のファイアウォールルールによってフィルタされたパケットをログに記録するために、各ルールに対して個別にログレベルパラメータを設定することができます。これにより、**ファイアウォール→オプション**で標準ルールに定義されたログレベルとは独立して、きめ細かくログを記録できます。

個々のルールのログレベルは、ルールの作成中または変更中に Web UI で簡単に定義または変更できますが、対応する `pvsh` API 呼び出しによっても設定できます。

さらに、`-log <loglevel>`を追加することで、ファイアウォール設定ファイルを通してログレベルを設定することもできます。

を選択したルールに追加します ([可能なログレベル](#)を参照)。例え

ば、以下の2つは同じです:

```
IN REJECT -p icmp -log nolog IN REJECT -p
icmp
```

一方

```
IN REJECT -p icmp -log debug
```

は、デバッグ・レベルのフラグが付いたログ出力を生成します。

## 13.10 ヒントとコツ

### 13.10.1 FTPの許可方法

FTPは、ポート21と他のいくつかの動的ポートを使用する古いスタイルのプロトコルです。そのためポート21を受け入れるルールが必要です。さらに、`ip_conntrack_ftp`ロードする必要があります。ですから、実行してください:

```
modprobe ip_conntrack_ftp
```

を追加し、`/etc/modules` に `ip_conntrack_ftp` を追加します（再起動後も動作するように）。

## 13.10.2 スリカータIPS統合

Suricata IPS (Intrusion Prevention System)を使用することも可能です。パケットはファイアウォールがACCEPTした後にのみIPSに転送されます。

リジェクト/ドロップされたファイアウォールのパケットがIPSに届きません。proxmox

ホストにsuricataをインストールしてください:

```
# apt-get install suricata # modprobe  
nfnetlink_queue
```

次回のリブートのために、/etc/modules に nfnetlink\_queue を追加することを忘れないでください。次に、特定のVMに対してIPSを有効にします:

```
# /etc/pve/firewall/<VMID>.fw  
  
[オプション]  
ips1  
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IPv6特有のオプションがいくつかあります。IPv6はARPプロトコルを使用せず、代わりにIPレベルで動作するNDP (Neighbor Discovery Protocol) を使用するため、成功するにはIPアドレスが必要です。このため、インターフェイスのMACアドレスに由来するリンクローカルアドレスが使用されます。デフォルトでは、ホストとVMの両方のレベルでNDPオプションが有効になっており、近隣探索 (NDP) パケットの送受信が可能になっています。

NDPはネイバーディスカバリーの他にも、自動コンフィギュレーションやルーターの広告など、いくつかの用途に使われます。

デフォルトでは、VMはルーター勧誘メッセージの送信（ルーターへの問い合わせ）とルーター広告パケットの受信を許可されています。これにより、ステートレス自動コンフィグレーションを使用できます。一方、"Allow Router Advertisement" (raadv: 1) オプションが設定されていない限り、VMは自分自身をルーターとしてアドバタイズできません。

NDPに必要なリンク・ローカル・アドレスに関しては、"IP Filter" (ipfilter: 1) オプションもあり、これを有効にすると、対応するリンク・ローカル・アドレスを含むVMの各ネットワーク・インターフェースにipfilter-net\*追加したのと同じ効果が得られます。(詳細については、[標準 IP セット ipfilter-net\\*](#) のセクションを参照してください)。

## 13.12 Proxmox VEが使用するポート

- ウェブインターフェース8006 (TCP、HTTP/1.1 over TLS)

- VNCウェブコンソール5900-5999 (TCP、WebSocket)
- SPICE プロキシ: 3128 (TCP)
- sshd クラスタアクションに使用): 22 (TCP)
- rpcbind: 111 (UDP)
- sendmail: 25 (TCP、送信)
- corosyncクラスタのトラフィック: 5405-5412 UDP
- ライブマイグレーション (VMメモリとローカルディスクのデータ) : 60000-60050 (tcp)

## 13.13 エヌエフティーブルズ

pve-firewallの代替として、私たちはproxmox-firewallを提供しています。これはiptablesではなく、より新しいnftablesをベースにしたProxmox VEファイアウォールの実装です。



### 警告

proxmox-firewall 現在技術プレビュー中です。バグやオリジナルのファイアウォールとの非互換性があるかもしれません。現在のところ本番環境での使用には適していません。

この実装では、同じ設定ファイルと設定フォーマットを使用します。いくつかの例外を除いて、まったく同じ機能を提供します：

- REJECTは現在、ゲストトラフィックにはできません（代わりにトラフィックはドロップされます）。
- NDP、ルーター広告、またはDHCPオプションを使用すると、デフォルトのポリシーに関係なく、常にファイアウォールルールが作成されます。
- ゲスト用のファイアウォールルールは、conntrackテーブルエントリを持つ接続でも評価されます。

### 13.13.1 インストールと使用方法

proxmox-firewall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/<ノード名>/host.fw) で nftables パックエンドを有効にします：

[オプション]

```
nftables: 1
```

### 備考

proxmox-firewall を有効/無効にした後、古い/新しいファイアウォールを正しく動作させるために、実行中のすべての VM とコンテナを再起動する必要があります。

nftables 設定キーを設定すると、新しい proxmox-firewall サービスが引き継がれます。新しいサービスが動作しているかどうかは、proxmox-

```
systemctl status proxmox-firewall
```

firewall の systemctl status をチェックすることで確認できます：

生成されたルールセットを調べることもできます。これについては「[役に立つコマンド](#)」のセクションに詳細があります。pve-firewall iptables ルールを生成しなくなったかどうかを確認する必要があります。

古いファイアウォールに戻すには、設定値を 0/No に戻すだけです。

### 13.13.2 使用方法

proxmox-proxmox-firewall サービスによって管理される 2 つのテーブル: proxmox-firewall と proxmox-firewall-guests を作成します。Proxmox VE ファイアウォール設定の外側にあるカスタムルールを作成したい場合は、カスタムファイアウォールルールを管理するために独自のテーブルを作成できます。proxmox-firewall 生成したテーブルにしか触れないで、独自のテーブルを追加する簡単に proxmox-firewall の動作を拡張・変更できます。

pve-firewall コマンドを使う代わりに、nftables ベースのファイアウォールは proxmox-firewall を使います。これは systemd サービスなので、

```
systemctl start proxmox-firewall systemctl stop proxmox-firewall
```

systemctl で起動と停止ができます：

ファイアウォールサービスを停止すると、生成されたルールがすべて削除されます。

ファイアウォールのステータスを照会するには、systemctl ステータスを照会します：

```
systemctl status proxmox-firewall
```

### 13.13.3 役立つコマンド

生成されたルールセットは、以下のコマンドで確認できます：

```
nftリストルールセット
```

proxmox-firewall をデバッグしたい場合は、RUST\_LOG で デーモンを フォアグラウンドで 実行するだけです。

```
RUST_LOG=trace /usr/libexec/proxmox/proxmox-firewall
```

環境変数を trace に設定してください。これで 詳細な デバッグ出力が 得られるはずです：

ファイアウォール デーモンの 詳細な 出力を 得たい 場合は、systemctl サービスを 編集する ことも できます：

```
systemctl proxmox-firewall を編集します。
```

次に、RUST\_LOG 環境オーバーライドを 追加する 必要があります：

```
[サービス] Environment="RUST_LOG=trace"
```

これは 大量の ログを 素早く 生成するので、デバッグ目的でのみ 使用して ください。他の、より冗長でない ログレベルは info と debug です。

フォアグラウンドで 実行すると、ログ出力が STDERR に 書き込まれるので、以下の コマンドで リダイレクト できます（コミュニティ フォーラムに ログを 提出する 場合など）：

```
RUST_LOG=trace /usr/libexec/proxmox/proxmox-firewall 2> firewall_log_$('  
ホスト名).txt
```

ファイアウォールルールをデバッグするために、異なるチェーンを経由するパケットフローをトレースすると便利です。これは、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お勧めします。

```
#!/usr/sbin/nft -f table bridge  
tracebridge  
削除テーブルブリッジトレースプリッジ  
  
table bridge tracebridge { chain trace {  
    meta l4proto icmp meta nftrace set 1  
}  
  
    チェーン・プリルート  
    type filter hook prerouting priority -350; policy accept; jump trace  
}  
  
    チェーン・ポストルーティング  
    type filter hook postrouting priority -350; policy accept; jump trace  
}
```

このファイルを保存し、実行可能にしてから 1 回実行すると、それぞれのトレースチェーンが作成されます。トレース出力は、Proxmox VE Web UI (Firewall > Log) または `nft monitor trace` で確認できます。

上記の例では、すべてのブリッジのトラフィックをトレースしていますが、これは通常、ゲストのトラフィックが流れる場所です。ホストのトラフィックを調べたい場合は、ブリッジテーブルの代わりに `inet` テーブルにこれらのチェーンを作成します。

---

#### 備考

これは大量のログスパムを生成し、ネットワークスタックのパフォーマンスを著しく低下させる可能性があることに注意してください。

---

以下のコマンドを実行することで、トレースルールを削除することができます:

```
nft テーブル削除ブリッジトレース
```

## 第14章

### ユーザー管理

Proxmox VEは、Linux PAM、統合Proxmox VE認証サーバ、LDAP、Microsoft Active Directory、OpenID Connectなど、複数の認証ソースをサポートしています。

すべてのオブジェクト（VM、ストレージ、ノードなど）に対してロールベースのユーザーと権限管理を使用することで、きめ細かなアクセスを定義できます。

#### 14.1 ユーザー

Proxmox VE はユーザ属性を `/etc/pve/user.cfg` に保存します。パスワードはここには保存されません。代わりに、ユーザは後述の認証レルムに関連付けられます。そのため、ユーザは内部的にはユーザ名とレルムで `<userid>@<realm>` という形で識別されることが多いです。

このファイルの各ユーザー・エントリには、以下の含まれています：

- 名前
- 姓
- メールアドレス
- 団体会員
- 任意の有効期限
- このユーザーに関するコメント
- このユーザーが有効か無効か
- オプションの二要素認証キー

---

#### 注意

ユーザーを無効化または削除した場合、または設定された有効期限が場合、このユーザーは新しいセッションにログインしたり、新しいタスクを開始したりできなくなります。このユーザーによってすでに開始されているすべてのタスク（ターミナル・セッションなど）は、このようなイベントによって自動的に終了することはありません。

---

## 14.1.1 システム管理者

システムのルート・ユーザーは、Linux PAMレルムを介して常にログインでき、制約のない管理者です。このユーザは削除できませんが、属性は変更できます。システムメールはこのユーザに割り当てられたメールアドレスに送信されます。

## 14.2 グループ

各ユーザーは複数のグループのメンバーになることができます。グループは、アクセス・パーミッションを整理するのに適した方法です。個々のユーザーではなく、常にグループにアクセス許可を与えるべきです。そうすることで、より管理しやすいアクセス制御リストを得ることができます。

## 14.3 APIトークン

APIトークンを使用すると、別のシステム、ソフトウェア、またはAPIクライアントからREST APIのほとんどの部分にステートレスでアクセスできます。トークンは個々のユーザーのために生成することができ、アクセスの範囲と期間を制限するために個別の権限と有効期限を与えることができます。APIトークンが漏洩した場合、ユーザー自身を無効にすることなく、トークンを失効させることができます。

APIトークンには基本的に2つのタイプがあります:

- 権限の分離: トークンにはACLで明示的なアクセス権を与える必要があります。有効な権限は、ユーザー権限とトークン権限を交差させて計算されます。
- 完全な権限: トークンの権限は、関連付けられたユーザーの権限と同じです。



### 注意

トークンの値は、トークンが生成されたときに一度だけ表示/返されます。後日、APIを介して再度取得することはできません!

APIトークンを使用するには、HTTPヘッダーのAuthorizationをPVEAPIToken=USER@の形式で表示される値に設定します。

APIリクエストを行う際には、APIクライアントのドキュメントを参照してください。

## 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のセットです。リソースごとに管理する必要がなく、単一のパーミッションを一連の要素に適用することができるため、特定のユーザーが特定のリソース・セットへのアクセスを制御する必要がある場合のパーミッション処理に便利です。リソース・プールはしばしばグループと同時に使用され、グループのメンバーが一連のマシンとストレージに対するパーミッションを持つようにします。

## 14.5 認証領域

Proxmox VEのユーザは、外部レルムに存在するユーザと対になるため、`/etc/pve/domains.cfg`でレルムを設定する必要があります。以下のレルム(認証方法)が利用可能です：

### Linux PAM 標準認証

Linux PAMは、システム全体のユーザー認証のためのフレームワークです。これらのユーザは、ホストシステム上で `adduser` などのコマンドを使用して作成します。PAM ユーザーが Proxmox VE ホストシステムに存在する場合、対応するエントリを Proxmox VE に追加して、これらのユーザーがシステムのユーザー名とパスワードでログインできるようにすることができます。

### Proxmox VE 認証サーバー

これは Unix ライクなパスワードストアで、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を使ってハッシュされます。これは、ユーザが Proxmox VE の外部にアクセスする必要がない小規模な（あるいは中規模の）インストールに最も便利な領域です。この場合、ユーザは Proxmox VE によって完全に管理され、GUI を介して自分のパスワードを変更することができます。

### ライトウェイトディレクトリアクセスプロトコル

LDAP (Lightweight Directory Access Protocol) は、ディレクトリサービスを使った認証のための、オープンでクロスプラットフォームなプロトコルです。OpenLDAP は、LDAP プロトコルの一般的な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用のディレクトリサービスで、Proxmox VE の認証レルムとしてサポートされています。認証プロトコルとして LDAP をサポートしています。

### OpenID Connect

OpenID Connect は、OAuth 2.0 プロトコルの上に ID レイヤーとして実装されています。これにより、クライアントは、外部の認証サーバーによって実行された認証に基づいて、ユーザーの身元を確認することができます。

### 14.5.1 Linux PAM 標準認証

Linux PAM はホスト・システム・ユーザーに対応するため、ユーザーがログインを許可される各ノードにシステム・ユーザーが存在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。このレルムはデフォルトで追加され、削除することはできません。

GUI または同等の `/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、クラスタ全体には適用されません。Proxmox VE がマルチマスター設計であっても、ノードごとに異なるパスワードを使用することで、セキュリティ上のメリットが得られます。

設定可能性という点では、管理者はレルムからのログインで二要素認証を要求したり、レルムをデフォルトの認証レルムとして設定したりすることができます。

### 14.5.2 Proxmox VE 認証サーバー

Proxmox VE 認証サーバのレルムは、単純な Unix ライクなパスワードストアです。レルムはデフォルトで作成され、Linux PAM と同様に、利用可能な設定項目は、レルムのユーザに二要素認証を要求する機能と、ログインのデフォルトレルムとして設定する機能だけです。

他のProxmox VEレルムタイプとは異なり、ユーザは他のシステムに対して認証するのではなく、Proxmox VEを通じて作成および認証されます。したがって、このタイプのユーザーは作成時にパスワードを設定する必要があります。

### 14.5.3 ライトウェイトディレクトリーアクセスプロトコル

ユーザ認証に外部LDAPサーバを使用することもできます(OpenLDAPなど)。このレルム・タイプでは、ユーザはユーザ属性名(user\_attr)フィールドで指定されたユーザ名属性を使用して、ベース・ドメイン名(base\_dn)の下で検索されます。

サーバーとオプションのフォールバック・サーバーを設定し、SSLで接続を暗号化することができます。さらに、ディレクトリとグループに対してフィルタを設定することができます。フィルタにより、レルムの範囲をさらに制限することができます。

例えば、あるユーザーが次のようなLDIFデータセットで表現されているとします:

```
# ldap-test.comのPeopleのuser1
dn: uid=user1,ou=People,dc=ldap-test,dc=com objectClass: top
オブジェクトクラス: person
objectClass: organizationalPerson objectClass: inetOrgPerson
uid: user1
cn: テストユーザー1 sn: テ
スター

の説明を参照してください: これは最初のテストユーザーです。
```

ベース・ドメイン名はou=People,dc=ldap-test,dc=comで、ユーザ属性はuidです。

Proxmox VEがユーザを照会および認証する前にLDAPサーバに認証(バインド)する必要がある場合は、/etc/pve/domains.cfg の bind\_dn プロパティでバインド・ドメイン名を設定できます。

そのパスワードは、/etc/pve/priv/realm/<realmname>.pw に保存する必要があります。  
このファイルには、生の1行だけ記述してください。

証明書を検証するには、capath を設定する必要があります。LDAPサーバのCA証明書を直接設定するか、すべての信頼されたCA証明書を含むシステムパス(/etc/ssl/certs)を設定することができます。さらに、verify オプションを設定する必要があります。

LDAPサーバ・レルムの主な設定オプションは以下のとおりです:

- レルム (realm): Proxmox VEユーザのレルム識別子
- ベース・ドメイン名 (base\_dn): ユーザを検索するディレクトリ
- ユーザ属性名 (user\_attr): ユーザがログインするユーザ名を含むLDAP属性
- サーバー(server1): LDAPディレクトリをホストするサーバー
- 予備サーバー (server2) : プライマリサーバが到達不能な場合のフォールバックサーバアドレス (オプション)
- Port (ポート) : LDAPサーバーがリッスンするポート

**備考**

特定のユーザにLDAPサーバを使用した認証を許可するには、Proxmox VEサーバからそのレルムのユーザとして追加する必要があります。これは[同期](#)によって自動的に実行できます。

#### 14.5.4 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と同じプロパティのほとんどをサポートしています。さらに、ユーザは設定後、[同期](#)操作により自動的にProxmox VEに追加することができます。

LDAPと同様に、Proxmox VEがADサーバーにバインドする前に認証が必要な場合は、以下のように設定する必要があります。

**バインド・ユーザー (bind\_dn)** プロパティ。このプロパティは通常、Microsoft ADではデフォルトで必要です。Microsoft Active Directory

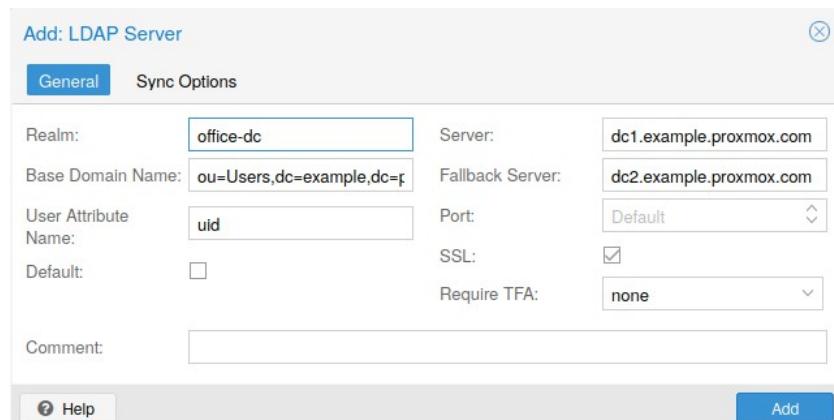
の主な構成設定は以下のとおりです：

- **レルム (realm)**: Proxmox VEユーザのレルム識別子
- **ドメイン (domain)** : サーバーのADドメイン
- **サーバー (server1)** : サーバーのFQDNまたはIPアドレス
- **予備サーバー (server2)** : プライマリサーバが到達不能な場合のフォールバックサーバアドレス (オプション)
- **Port (ポート)** : Microsoft ADサーバーがリッスンするポート。

**備考**

Microsoft ADは通常、大文字と小文字を区別せずにユーザー名のような値をチェックします。Proxmox VEでも同じようにするには、Web UIでrealmを編集するか、CLIを使用して(realm IDでIDを変更して)、デフォルトの大文字小文字を区別する無効にします: `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 & Microsoft Active Directory)のユーザーとグループを Proxmox VE に手動で追加するのではなく、自動的に同期することができます。同期オプションには、Web インターフェイスの認証パネルの [広告/編集] ウィンドウから、または `pveum realm add/modify` からアクセスできます。

コマンドを使用します。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 <realm>
```

ユーザーとグループはクラスタ全体の構成ファイル /etc/pve/user.cfg に同期されます。

#### 属性からプロパティへ

同期レスポンスにユーザー属性が含まれている場合、その属性は

user.cfgで指定します。例：firstnameまたはlastname。

属性の名前がProxmox VEのプロパティと一致しない場合は、sync\_attributesオプションを使用して、コンフィグでカスタムフィールド間マップを設定できます。

このようなプロパティが消えた場合にどのように処理されるかは、後述の同期オプションで制御できます。

#### 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) : ユーザとグループの問い合わせに使用する LDAP アカウントを指します。このアカウントは、必要なエントリすべてにアクセスできる必要があります。これが設定されている場合、検索はバインド経由で行われ、そうでない場合は匿名で行われます。ユーザは、例えば、cn=admin,dc=example,dc=comのように、完全なLDAP形式の識別名(DN)でなければなりません。
- グループ名attr (group\_name\_attr) : ユーザのグループを表します。user.cfgの通常の文字制限に従ったエントリのみが同期されます。グループは、名前の衝突を避けるために、名前に-\$realmが付加された状態で同期されます。同期によって手動で作成したグループが上書きされないようにしてください。
- ユーザー・クラス (user\_classes) : ユーザに関連するオブジェクト・クラス。
- グループ・クラス (group\_classes) : グループに関連するオブジェクト・クラス。
- Eメール属性: LDAPベースのサーバでユーザの電子メールアドレスが指定されている場合、ここで関連する属性を設定することで、これらの電子メールアドレスも同期することができます。これはコマンドラインから --sync\_attributes パラメータを指定することで実現できます。
- ユーザーフィルター (filter) : 特定のユーザーをターゲットとするフィルタオプションです。
- グループ・フィルター (group\_filter) : 特定のグループをターゲットとするフィルタオプション。

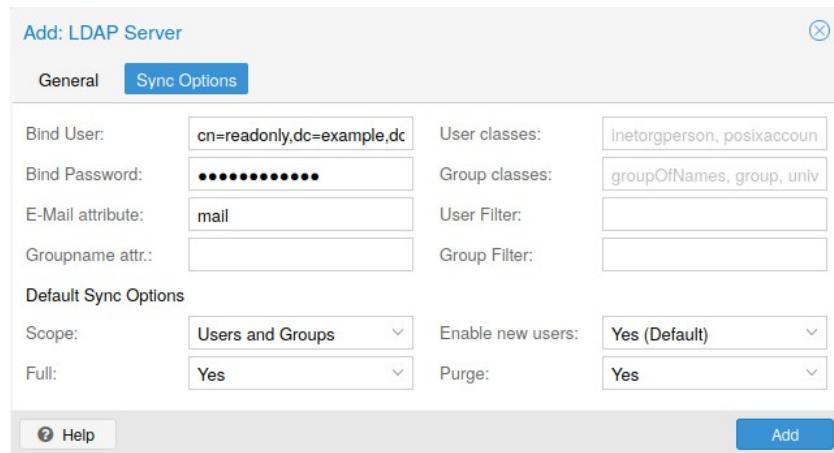
---

#### 備考

フィルタを使用すると、追加の一一致基準を作成して、同期の範囲を絞り込むことができます。使用可能な LDAP フィルタの種類とその使用方法については、[ldap.com](#) を参照してください。

---

## 同期オプション



前のセクションで指定したオプションに加えて、シンク操作の動作を記述するさらなるオプションを設定することもできます。

これらのオプションは、同期の前にパラメータとして設定するか、レルムオプションsync-defaults-でデフォルトとして設定します。

同期の主なオプションは以下の通りです：

- スコープ（範囲）：同期するものの範囲。ユーザー、グループ、または両方を指定できます。
- 新規を有効にする（enable-new）：設定すると、新しく同期されたユーザが有効になり、ログインできるようになります。デフォルトは本當です。
- Remove Vanished（remove-vanished）：これはオプションのリストで、有効にすると、同期レスポンスから返されなかったときに削除されるかどうかを決定します。オプションは次のとおりです：
  - ACL（acl）：同期応答で返されなかったユーザーとグループの ACL を削除します。  
これはエントリーとともに意味を持つことがほとんどです。
  - エントリ（entry）：エントリ（entry）：同期応答で返されなかったエントリ（ユーザーやグループなど）を削除します。
  - Properties（プロパティ）：プロパティ（properties）：シンク応答のユーザーがそれらの属性を含んでいないエントリのプロパティを削除します。  
これはすべてのプロパティを含み、一度も同期によって設定されていないものも含みます。これらはこのオプションを有効にしても保持されます。
- プレビュー（ドライラン）：データはコンフィグに書き込まれません。どのユーザとグループがuser.cfgに同期されるかを確認したい場合に便利です。

## 予約文字

特定の文字は予約されており（[RFC2253を参照](#)）、適切にエスケープしないとDNの属性値で簡単に使用できません。

以下の文字はエスケープが必要です：

- 先頭または末尾にスペース（）
- 先頭に数字記号（#）
- カンマ（,）

- プラス記号 (+)
- ダブルクオート ("")
- スラッシュ (/)
- 角括弧 ()<>
- セミコロン (;)
- 等号 (=)

このような文字をDNで使用するには、属性値を二重引用符で囲みます。例えば、ユーザ CN(共通名) Example, User の場合、CN="Example, User",OU=people,DC=example を使用します。を bind\_dn の値として指定します。

これは base\_dn 属性、bind\_dn 属性、group\_dn 属性に適用されます。

---

#### 備考

コロンとスラッシュはユーザー名の予約文字であるため、同期できません。

---

## 14.5.6 OpenID Connect

OpenID Connectの主な設定オプションは以下のとおりです：

- 発行者URL (issuer-url) : 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、さらに詳細を自動的に設定します。  
暗号化されていないhttp:// URLを使用することも可能ですが、暗号化された  
https:// 接続。
- レルム (realm): Proxmox VEユーザのレルム識別子
- クライアントID (client-id) : OpenIDクライアントID。
- クライアント・キー (client-key): オプションの OpenID クライアント鍵。
- ユーザーの自動作成 (autocreate) : ユーザが存在しない場合、自動的にユーザを作成します。認証はOpenIDサーバで行われますが、Proxmox VEのユーザ設定にはすべてのユーザのエントリが必要です。手動で追加するか、自動作成オプションを使用して新しいユーザを自動的に追加します。
- ユーザ名クレーム (username-claim): 一意なユーザ名を生成するために使用されるOpenIDクレーム(subject、  
ユーザー名または電子メール)。
- グループの自動作成(groups-autocreate): PVEグループを使用する代わりに、クレーム内のすべてのグループを作成します（デフォルトの動作）。
- Groups Claim (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。
- グループの上書き (groups-overwrite) : 既存のグループを保留する代わりに、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）。

## ユーザー名のマッピング

OpenID Connectの仕様では、`subject`という一意な属性（OpenID用語では`claim`）が定義されています。デフォルトでは、この属性の値を使ってProxmox VEのユーザ名を生成します。

残念なことに、ほとんどのOpenIDサーバはDGH76OKH34BNG3245SBのようなランダムな文字列を件名に使っているので、典型的なユーザ名はDGH76OKH34BNG3245SB@yourrealmのようになります。ユニークではありますが、人間がこのようなランダムな文字列を覚えておくのは難しいので実際のユーザと関連付けるのはかなり不可能です。

`username-claim`設定では、ユーザー名のマッピングに他の属性を使用することができます。これを`username`は、OpenID Connect サーバがその属性を提供し、その一意性が保証されている場合に使用されます。

もう1つのオプションは`email`を使うことで、こちらも人間が読めるユーザ名が得られます。繰り返しますが、サーバがこの属性の一意性を保証している場合にのみ、この設定を使用してください。

## 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`で指定するデータは、Proxmox VEでユーザが所属すべきグループに対応する文字列のリストでなければなりません。衝突を防ぐため、OpenID claimのグループ名のサフィックスは-`<realm name>`となります(例えば、realm oidcのOpenIDグループ名my-openid-groupの場合、Proxmox VEのグループ名はmy-openid-group-oidcとなります)。

OpenIDプロバイダから報告されたグループのうち、Proxmox VEに存在しないものはデフォルトで無視されます。OpenIDプロバイダから報告されたすべてのグループがProxmox VEに存在する必要がある場合は、`groups-autocreate`オプションを使用して、ユーザのログイン時にこれらのグループを自動的に作成することができます。

デフォルトでは、グループはユーザの既存のグループに追加されます。Proxmox VE でユーザが既にメンバーであるグループを OpenID プロバイダからのグループで上書きすることが望ましい場合があります。`groups-overwrite` 設定を有効にすると、OpenID プロバイダから報告されたグループを追加する前に、Proxmox VE でユーザーからすべてのグループが削除されます。

場合によっては、OpenIDサーバーがProxmox VEのグループIDに無効な文字を含むグループクレームを送信することができます。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含まれず、警告がログに送信されます。

## 詳細設定

- `userinfo` エンドポイントに問い合わせる(`query-userinfo`): このオプションを有効にすると、OpenID Connect 認証機能が "userinfo" エンドポイントにクレーム値を問い合わせるようになります。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ていない ID プロバイダ(例えば ADFS)がある場合に便利です。

## 例

Google を使って OpenID レルムを作成する例です。この例では、`--client-id` と`--client-key` に Google OpenID の設定値を指定します。

```
pveum realm add myrealm1 --type openid --issuer-url https://accounts.google.com --client-id XXXX --client-key YYYY --username-claim email
```

上記のコマンドでは `--username-claim email` を使用しているため、Proxmox VE 側のユーザー名は `example.user@google.com@myrealm1` のようになります。

Keycloak (<https://www.keycloak.org/>) は、OpenID ConnectをサポートするオープンソースのIDおよびアクセス管理ツールです。以下の例では、`--issuer-url`と`--client-id`をあなたの情報で置き換える必要があります：

```
pveum realm add myrealm2 --type openid --issuer-url https://your.server:8080/realms/your-realm --client-id XXX --username-claim ユーザー名
```

`--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使用できます。



#### 警告

ユーザーが（Keycloakサーバー上で）ユーザー名の設定自分で編集できないようにする必要があります。

## 14.6 二要素認証

二要素認証を使用するには2つの方法があります：

これは、TOTP (Time-based One-Time Password) または YubiKey OTP のいずれかの認証レルムによって要求されます。この場合、新しく作成されたユーザは、すぐに鍵を追加する必要があります。TOTPの場合、最初にログインできれば、後からTOTPを変更することも可能です。

あるいは、たとえレルムが2ファクタ認証を強制していなくても、ユーザは後から2ファクタ認証を選択することもできます。

### 14.6.1 利用可能なセカンドファクター

スマートフォンやセキュリティキーを紛失してアカウントから永久にロックされる事態を避けるため、複数のセカンドファクターを設定することができます。

レルムで強制されるTOTPとYubiKey OTPに加えて、以下の二要素認証方法が利用できます：

- ユーザーが設定するTOTP（時間ベースのワンタイムパスワード）。共有シークレットと現在時刻から導き出される短いコードで、30秒ごとに変更されます。
- WebAuthn (Web Authentication)。一般的な認証規格。コンピュータやスマートハードウェアキーやTPM (Trusted Platform Module) など、さまざまなセキュリティデバイスによって実装されます。
- シングルユースのリカバリーキー。プリントアウトして安全な施錠するか、電子保管庫にデジタル保存する必要があるキーのリスト。各キーは1回のみ使用できます。他のすべてのセカンドファクターが紛失または破損した場合でも、ロックアウトされないようにするのに最適なキーです。

WebAuthn がサポートされる前は、U2F は設定できました。既存のU2Fファクターは引き続き使用できますが、サーバー上で設定された後は、WebAuthnに切り替えることをお勧めします。

## 14.6.2 レルム強制二要素認証

これは、認証レルムを追加または編集する際に、TFAドロップダウン・ボックスから利用可能な方法の1つを選択することで実行できます。レルムがTFAを有効にすると、それが必須条件となり、TFAを設定したユーザのみがログインできるようになります。

現在、2つの方法が：

### タイムベースOATH (TOTP)

これは標準的な HMAC-SHA1 アルゴリズムを使用し、現在時刻はユーザーが設定したキーでハッシュ化されます。時間ステップとパスワードの長さのパラメータは設定可能です。

ユーザーは複数のキーを（スペースで区切って）設定することができ、キーはBase32（RFC3548）または16進数表記で指定できます。

Proxmox VEは、oathtoolコマンドラインツールやAndroid Google Authenticator、FreeOTP、andOTPなどの様々なOTPツールで直接使用できるBase32記法のランダムキーを出力するキー生成ツール（oathkeygen）を提供しています。

### YubiKey OTP

YubiKeyを使った認証には、Yubico API ID、API KEY、認証サーバのURLが必要です。YubiKeyからキーIDを取得するには、YubiKeyをUSBで接続した後、一度YubiKeyを起動し、入力されたパスワードの最初の12文字をユーザーのキーIDsフィールドにコピーします。

[YubiCloud](#)を使用する方法、または[独自の認証ホストサーバ](#)をする方法については、[YubiKey OTPのドキュメント](#)を参照してください。

## 14.6.3 二要素認証の制限とロックアウト

第二の要因は、パスワードが何らかの形で漏れたり推測されたりした場合に、ユーザーを保護するためのものです。しかし、いくつかの要素は総当たりで破られる可能性があります。このため、第2要素によるログインに何度も失敗すると、ユーザーはロックアウトされます。

TOTPの場合、8回失敗すると、ユーザーのTOTPファクターは無効になります。回復キーでログインすると解除されます。TOTPが唯一の利用可能な要素であった場合、管理者の介入が必要であり、ユーザに直ちにパスワードの変更を要求することが強く推奨されます。

FIDO2/Webauthnとリカバリキーはブルートフォースアタックの影響を受けにくいため、リミットは高くなります（100回）、それを超えるとすべてのセカンドファクターが1時間ロックされます。

管理者は、UIのユーザリストまたはコマンドラインから、いつでもユーザーの二要素認証を解除することができます：

```
pveum ユーザー tfa ロック解除 joe@pve
```

## 14.6.4 ユーザが設定したTOTP認証

ユーザは、ログイン時にTOTPまたはWebAuthnを第二要素として有効にするかどうかを、ユーザ一覧のTFAボタンから選択できます（レルムがYubiKey OTPを強制している場合を除く）。

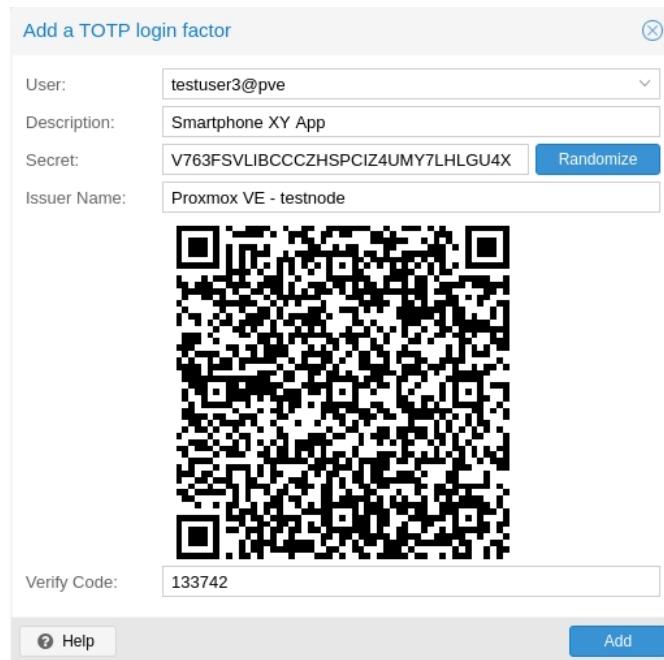
ユーザーはいつでも1回限りのリカバリーキーを追加して使用できます。

The screenshot shows the Proxmox VE management interface. On the left is a sidebar with various management sections: Datacenter, Search, Summary, Notes, Cluster, Ceph, Options, Storage, Backup, Replication, Permissions (with sub-options like Users, API Tokens, Groups, Pools, Roles, and Realms), HA, ACME, Firewall, Metric Server, and Support. The 'Two Factor' section is currently selected and highlighted with a blue background. At the top right of the main content area are buttons for 'Add', 'Edit', and 'Remove'. Below these buttons is a table with columns: User, Enabled, TFA Type, Created, and Description. A single row is visible in the table, showing 'testuser3@pve' as the User, 'Yes' as Enabled, 'recovery' as TFA Type, '2021-11-15 12:20:38' as Created, and an empty Description field.

TFA ウィンドウを開くと、TOTP 認証を設定するためのダイアログが表示されます。Secret (秘密) フィールドには鍵を入力します。この鍵は「Randomize (ランダム化)」ボタンでランダムに生成することができます。オプションで「Issuer Name (発行者名)」を追加すると、TOTPアプリに鍵の所有者情報を提供することができます。ほとんどの TOTP アプリは、対応する OTP 値とともに発行者名を表示します。ユーザ名は TOTP アプリの QR コードにも含まれます。

キーを生成すると、FreeOTP などのほとんどの OTP アプリで使用できる QR コードが表示されます。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除く) と、TOTP キーを正しく使用できることを、現在の OTP 値を検証コードフィールドに入力して適用ボタンを押すことで確認する必要があります。

## 14.6.5 トップキー



サーバーの設定は不要です。スマートフォンにTOTPアプリ（例えばFreeOTP）をインストールし、Proxmox VEのウェブインターフェースを使用してTOTPファクターを追加するだけです。

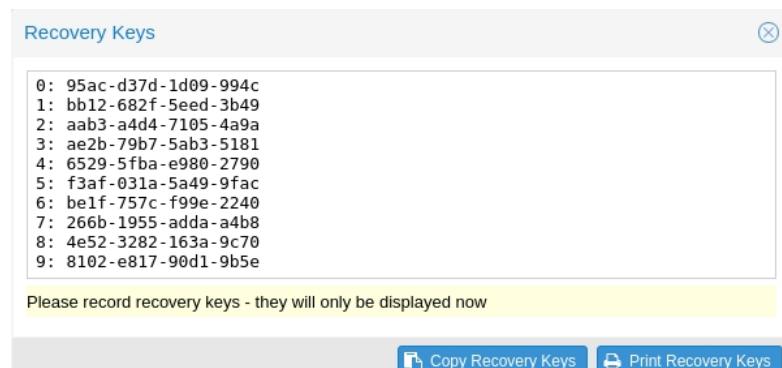
## 14.6.6 ウェブオート

WebAuthnを動作させるには、2つのものが必要です：

- 信頼できる HTTPS 証明書 ([Let's Encrypt を使用するなど](#))。信頼されていない証明書でもおそらく動作しますが、信頼されていない場合、ブラウザによっては WebAuthn の操作を警告したり拒否したりすることがあります。
- WebAuthn 設定をセットアップします (Proxmox VE ウェブインターフェースの **Datacenter→ Options→ WebAuthn Settings** を参照)。これはほとんどのセットアップで自動入力できます。

これらの両方の要件を満たすと、WebAuthn 設定を **Two Factor** パネルで **「Datacenter→ Permissions→ Two Factor」** を選択します。

## 14.6.7 リカバリーキー

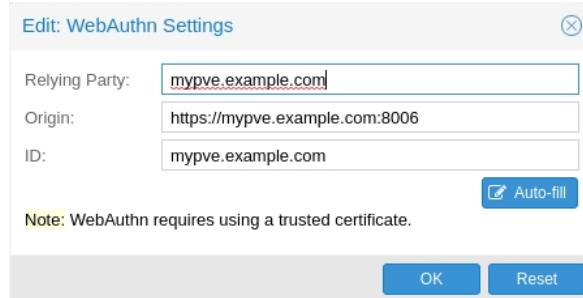


**Datacenter→ Permissions→ Two Factor** の **Two Factor** パネルで回復キーのセットを作成するだけです。

**備考**

回復用鍵は、11セットのみ使用できます。

## 14.6.8 サーバー側のWebauthn設定



ユーザーが *WebAuthn* 認証を使用できるようにするには、有効な SSL 証明書を持つ有効なドメインを使用する必要があります。

**備考**

*WebAuthn* の設定を変更すると、既存のすべての *WebAuthn* 登録が使用できなくなる可能性があります！

/etc/pve/datacenter.cfgで行います。例えば

```
webauthn: rp=mypve.example.com,origin=https://mypve.example.com:8006,id=<rp>'  
        mypve.example.com
```

## 14.6.9 サーバー側のU2F設定

**備考**

代わりにWebAuthnを使用することをお勧めします。

ユーザーがU2F認証を使用できるようにするには、有効なSSL証明書と有効なドメインを使用する必要がある場合があります。そうでない場合、ブラウザによっては警告が表示されたり、U2Fの使用を完全に拒否したりする場合があります。最初に、*AppId* <sup>(1)</sup> を設定する必要があります。

**備考**

*AppId*を変更すると、既存のすべてのU2F登録が使用できなくなります！

/etc/pve/datacenter.cfgで行います。例えば

```
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)

1つのノードの場合`AppId`は単純にウェブインターフェースのアドレスで、ブラウザで使用されるものと全く同じで、上記のように`https://`、ポートも含みます。ブラウザによっては、`AppId`のマッピングが他のブラウザよりも厳しい場合があることに注意してください。

複数のノードを使用する場合、`https` の別サーバで `appid.json`<sup>2</sup> 提供するのが、ほとんどのブラウザと互換性があるようです。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`AppId`としてTLDを使用するだけで十分かもしれません。ただし、ブラウザによってはこれを受け入れない場合があることに注意が必要です。

#### 備考

`AppId`が通常はエラーが発生しますが、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`を使用している場合、エラーが発生しないことがあります。このため、後で`AppId`を変更すると既存のU2F登録が使用できなくなるため、複数のブラウザで設定をテストすることをお勧めします。

### 14.6.10 ユーザーとしてのU2Fのアクティベーション

U2F認証を有効にするには、TFAウィンドウのU2Fタブを開き、現在のパスワードを入力し（rootでログインしていない場合）、Registerボタンを押します。サーバーが正しくセットアップされ、ブラウザがサーバーから提供された`AppId`を受け入れると、U2Fボタンを押すよう促すメッセージが表示されます（YubiKeyの場合、ボタンのランプが1秒間に2回程度、点灯と消灯を繰り返すはずです）。

FirefoxユーザーはU2Fトークンを使う前に`about:config`で`security.webauth.u2f`を有効にする必要があるかもしれません。

## 14.7 許可管理

ユーザがアクション（VMのコンフィギュレーションの一部を一覧表示、変更、削除など）を実行するには、そのユーザに適切なパーミッションが必要です。

Proxmox VEは、ロールおよびスペースの権限管理システムを使用しています。パーミッションテーブルのエントリは、オブジェクトやノードにアクセスする際に、ユーザー、グループ、トークンが特定の役割を担うことを許可します。つまり、アクセスルールは（パス、ユーザー、ロール）、（パス、グループ、ロール）、（パス、トークン、ロール）のトリプルとして表すことができます。

### 14.7.1 役割

ロールは単に権限のリストです。Proxmox VEには定義済みのロールが多数用意されており、ほとんどの要件を満たすことができます。

- **管理者:** すべての権限があります。
- **NoAccess:** 権限を持たない（アクセスを禁止するために使用）
- **PVEAdmin:** ほとんどのタスクを実行できますが、システム設定（`Sys.PowerMgmt`、`Sys.Modify Realm.Allocate`）やパーミッション（`Permissions.Modify`）を変更する権限はありません。
- **PVEAuditor:** 読み取り専用

<sup>2</sup> マルチファセットアプリ：[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と割り当て
- PVEDatastoreUser: バックアップ領域の割り当てとストレージの表示
- PVEMappingAdmin: リソースマッピングの管理
- PVEMappingUser: リソースマッピングの表示と使用
- PVEPoolAdmin: プールの割り当て
- PVEPoolUser: プールを見る
- PVESDNAdmin: SDN コンフィギュレーションの管理
- PVESDNUser: ブリッジ/ネットへのアクセス
- PVESysAdmin: 監査、システムコンソール、システムログ
- PVETemplateUser: テンプレートの表示とクローン
- PVEUserAdmin: ユーザーの管理
- PVEVMAdmin: VM を完全に管理します。
- 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で定義済みのロール一式を確認できます。GUIまたはコマンドラインから新しいロ

ールを追加できます。



GUIからDatacenterのPermissions→ Rolesタブに移動し、 Createボタンをクリックします。そこでロール名を設定し、 [Privileges] ドロップダウンメニューから必要な権限を選択できます。

コマンドラインからロールを追加するには、 pveum CLIツールを使用します：

```
pveum role add VM_Power-only --privs "VM.PowerMgmt VM.Console" pveum role add Sys_Power-only --privs "Sys.PowerMgmt Sys.Console"
```

#### 備考

PVEで始まるロールは常に組み込みのものであり、カスタムロールはこの予約接頭辞を使用することはできません。

### 14.7.2 特典

権限とは、特定のアクションを実行する権利のことです。管理を簡単にするために、権限のリストはロールにグループ化されます。権限はロールの一部でなければユーザやバスに直接割り当てることができないことに注意してください。

現在、以下の特権をサポートしています：

## ノード / システム関連権限

- Group.Allocate: グループの作成/変更/削除
- Mapping.Audit: リソースマッピングの表示
- Mapping.Modify: リソースマッピングの管理
- Mapping.Use: リソースマッピングの使用
- Permissions.Modify: アクセス許可の変更
- Pool.Allocate: プールの作成/変更/削除
- Pool.Audit: プールの表示
- Realm.AllocateUser: ユーザーをレルムに割り当てる
- Realm.Allocate: 認証レルムの作成/変更/削除
- SDN.Allocate: SDNコンフィギュレーションの管理
- SDN.Audit: SDNコンフィギュレーションの表示
- Sys.Audit: ノードステータス/設定、Corosyncクラスタ設定、HA設定の表示
- Sys.Console: ノードへのコンソールアクセス
- Sys.Incoming: 他のクラスタからの受信データストリームを許可（実験的）
- Sys.Modify: ノード・ネットワーク・パラメータの作成/変更/削除
- Sys.PowerMgmt: ノードの電源管理(スタート、ストップ、リセット、シャットダウン、...)
- Sys.Syslog: syslogの表示
- User.Modify: ユーザーアクセスと詳細の作成/変更/削除。

## 仮想マシン関連権限

- SDN.Use: SDN ネットワークとローカルネットワークのブリッジにアクセスします。
- VM.Allocate: サーバー上のVMの作成/削除
- VM.Audit: VMコンフィグを表示
- VM.Backup: VMのバックアップ/リストア
- VM.Clone: VMのクローン/コピー
- VM.Config.CDROM: CD-ROM の取り出し/交換
- VM.Config.CPU: CPU設定の変更
- VM.Config.Cloudinit: クラウドイットパラメータの変更
- VM.Config.Disk: ディスクの追加/変更/削除
- VM.Config.HWType: エミュレートされたハードウェアタイプを変更します。
- VM.Config.Memory: メモリ設定の変更
- VM.Config.Network: ネットワークデバイスの追加/変更/削除
- VM.Config.Options: その他のVM設定を変更します。
- VM.Console: VMへのコンソールアクセス
- VM.Migrate: クラスタ上の代替サーバにVMを移行
- VM.Monitor: VMモニター（kvm）へのアクセス

- VM.PowerMgmt: 電源管理(スタート、ストップ、リセット、シャットダウン、...)
- VM.Snapshot.Rollback: VMをスナップショットの1つにロールバックします。
- VM.Snapshot: VMスナップショットの作成/削除

#### ストレージ関連の権限

- Datastore.Allocate: データストアの作成/変更/削除とボリュームの削除
- Datastore.AllocateSpace: データストアに領域を割り当てる
- Datastore.AllocateTemplate: テンプレートと ISO イメージの割り当て/アップロード
- Datastore.Audit: データストアの表示/閲覧



#### 警告

Permissions.ModifyとSys.Modifyはどちらも、危険であったり機密であったりするシステムやその設定の変更を許可するものであるため、取り扱いには注意が必要です。



#### 警告

以下の継承に関するセクションを注意深く読んで、割り当てられたロール（およびその権限）がACLツリーに沿ってどのように伝搬するかを理解してください。

### 14.7.3 オブジェクトとパス

アクセス許可は、仮想マシン、ストレージ、リソースプールなどのオブジェクトに割り当てられます。これらのオブジェクトのアドレスには、ファイルシステムのようなパスを使用します。これらのパスは自然なツリーを形成し、より高いレベル（より短いパス）のパーミッションは、オプションでこの階層内に伝搬することができます。

パスはテンプレート化できます。API呼び出しがテンプレート化されたパスのパーミッションを必要とする場合、パスにはAPI呼び出しのパラメータへの参照が含まれることがあります。これら の参照は中かっこで指定します。一部のパラメータは、APIコールのURIから暗黙的に取得されます。例えば、/nodes/mynode/status を呼び出すときのパーミッションパス /nodes/{node} は、/nodes/mynode のパーミッションを必要とし、/access/acl へのPUT要求のパス {path} は、メソッドのパスパラメータを参照します。

いくつか例を挙げましょう：

- /ノード/{ノード}: Proxmox VE サーバマシンへのアクセス
- /vms: すべてのVMをカバー
- /vms/{vmid}です: 特定のVMへのアクセス
- /storage/{storeid}: 特定のストレージへのアクセス
- /pool/{プール名}: 特定のプールに含まれるリソースへのアクセス
- /アクセス/グループグループ管理
- /access/realms/{realmid}: レルムへの管理者アクセス

## 相続

前述したように、オブジェクトのパスはファイルシステムのようなツリーを形成し、パーミッションはそのツリーの下のオブジェクトに継承されます（デフォルトではpropagateフラグが設定されています）。以下の継承ルールを使用します：

- 個人ユーザーに対するパーミッションは、常にグループパーミッションに取って代わります。
- グループに対するパーミッションは、ユーザーがそのグループのメンバーである場合に適用されます。
- より深いレベルのパーミッションは、レベルから継承されたものを置き換えます。
- NoAccessは、指定されたパス上の他のすべてのロールをキャンセルします。

さらに、権限分離されたトークンは、関連するユーザーが持っていないパーミッションを、任意のパス上で持つことはできません。

## 14.7.4 プール

プールを使用すると、仮想マシンとデータストアのセットをグループ化できます。そして、プール (/pool/{poolid}) にパーミッションを設定するだけで、そのパーミッションはすべてのプールメンバーに継承されます。これは、アクセス制御を簡素化する素晴らしい方法です。

## 14.7.5 どの権限が必要ですか？

必要なAPIパーミッションは個々のメソッドごとに文書化されており、<https://pve.proxmox.com/pve-docs/api-viewer/>にあります。

パーミッションはリストとして指定され、ロジックとアクセスチェック関数のツリーとして解釈できます：

`["および", <subtests>...] および ["or", <subtests>...] 。`

現在のリストの各要素(and)またはそれ以上の要素(any(or))が真でなければなりません。

`["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](#)」参照）。列挙されている権限のすべて（または any オプションが用いられている場合はいずれか）が、指定されたパス上で許されている必要があります。require-param オプションが指定されているときは、その API 呼び出しのスキーマがそれ以外にはそのパラメタをオプショナルとしてリストしているときでも、その指定されたパラメタは必須です。

`[userid-group", [ <privileges>... ], <options>... ]。`

呼び出し元は、/access/groupsにリストされた特権のいずれかを持っていなければなりません。さらに、groups\_paramオプションが設定されているかどうかによって、2つのチェックが可能です：

- groups\_param が設定されています：APIコールに非オプションのgroupsパラメータがあり、呼び出し元がリストされたすべてのグループに対してリストされた特権のいずれかを持っている必要があります。
- groups\_param が設定されていません：useridパラメータで渡されたユーザは、呼び出し元が(/access/groups/<group>パス経由で)リストされた権限のいずれかを持つグループに存在し、そのグループの一部でなければなりません。

[**userid-param**, "self"]。

APIコールのuseridパラメータに提供される値は、アクションを実行するユーザーを参照する必要があります（通常、昇格権限を持っていなくても、ユーザーが自分自身に対してアクションを実行できるようにするために、orと組み合わせて使用されます）。

[ "userid-param", "Realm.AllocateUser"]。

ユーザは、/access/realm/<realm>へのRealm.AllocateUserアクセスが必要です。<realm>はuseridパラメータで渡されたユーザの指します。 なお、ユーザIDは<ユーザー名>@<領域>。

[ "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](#)参照)。ユーザはPermissions.Modify特権、またはパスによっては以下の特権で代用できます：

- /storage/...: 'Datastore.Allocate' が必要です。
- /vms/...: 'VM.Allocate' が必要です。
- /pool/...: 'Pool.Allocate' が必要です。

パスが空の場合は、/accessのPermissions.Modifyが必要です。

ユーザーがPermissions.Modify権限を持っていない場合、指定されたパス上の自分の権限のサブ委譲することができます（例えば、PVEVMAdminを持つユーザーはPVEVMUserを委譲することはできますが、PVEAdminを委譲することはできません）。

## 14.8 コマンドラインツール

ほとんどのユーザーは GUI を使ってユーザーを管理します。しかし、pveum ("Proxmox VE User Manager "の略) と呼ばれるフル機能のコマンドラインツールもあります。Proxmox VEのコマンドラインツールはすべてAPIのラッパーなので、REST APIからこれらの機能にアクセスすることもできます。

以下は簡単な使用例です。ヘルプを表示するには

はげ

または(特定のコマンドに関する詳細なヘルプを表示するには)

pveum ヘルプ ユーザー追加

新しいユーザーを作成します：

pveum user add testuser@pve -comment "ただのテスト"

パスワードを設定または変更します（すべてのレルムでサポートされているではありません）：

pveum passwd testuser@pve

ユーザーを無効にします：

pveum user modify testuser@pve -enable 0

新しいグループを作成します：

```
pveumグループ追加テストグループ
```

新しいロールを作成します:

```
pveum role add PVE_Power-only -privs "VM.PowerMgmt VM.Console"
```

## 14.9 実例

### 14.9.1 管理者グループ

管理者が、(rootアカウントを使わずに) 完全な管理者権限を持つユーザーグループを作りたいと思うことはあります。

そのためには、まずグループを定義します:

```
pveum group add admin -comment "システム管理者"
```

次に役割を割り当てます:

```
pveum acl modify / -group admin -role 管理者
```

最後に、新しい管理者ユーザーを追加します:

```
pveum user modify testuser@pve -group admin
```

### 14.9.2 監査役

PVEAuditorロールをユーザーまたはグループに割り当てることで、ユーザーに読み取り専用アクセス権を与えることができます。例1: ユーザー

joe@pveにすべての閲覧を許可する場合

```
pveum acl modify / -user joe@pve -role PVEAuditor
```

例2: ユーザ joe@pve にすべての仮想マシンを表示できるようにします。

```
pveum acl modify /vms -user joe@pve -role PVEAuditor
```

### 14.9.3 ユーザー管理の委任

ユーザ管理をユーザjoe@pveに委譲したい場合は、次のようにします:

```
pveum acl modify /access -user joe@pve -role PVEUserAdmin
```

ユーザjoe@pveはユーザの追加と削除、パスワードなどのユーザ属性の変更ができるようになりました。これは非常に強力なロールであり、選択されたrealmとグループに制限したい場合がほとんどでしょう。以下の例では、joe@pveはrealm pve内のユーザを変更することができます:

```
pveum acl modify /access/realm/pve -user joe@pve -role PVEUserAdmin  
pveum acl modify /access/groups/customers -user joe@pve -role PVEUserAdmin
```

**備考**

そのユーザーは他のユーザーを追加することができますが、そのユーザーがcustomersグループのメンバーであり、pveルーム内にいる場合に限られます。

## 14.9.4 モニタリング用限定APIトークン

APIトークンの権限は、常に対応するユーザの権限のサブセットです。つまり、APIトークンを使用して、バックユーザが権限を持たないタスクを実行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、別の権限を持つAPIトークンを使用する方法を示します。

すべてのVMで、ユーザーjoe@pveにロールPVEVMAdminを与えます：

```
pveum acl modify /vms -user joe@pve -role PVEVMAdmin
```

別の権限を持つ新しいAPIトークンを追加します。このAPIトークンは、VM情報の表示のみ許可されます（監視目的など）：

```
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ユーザー権限 joe@pve  
pveumユーザートークンパーミッション joe@pve モニタリング
```

## 14.9.5 リソースプール

通常、企業はいくつかの小さな部門に分かれて、それぞれの部門にリソースを割り当てたり、管理タスクを委任したりするのが一般的です。ここでは、ソフトウェア開発部門のためのプールを設定したいとします。まず、グループを作成します：

```
pveum group add developers -コメント "Our software developers"
```

次に、そのグループのメンバーである新しいユーザーを作成します：

```
pveum user add developer1@pve -group developers -password
```

**備考**

password "パラメーターはパスワードの入力を要求します。

そして、開発部門が使用するリソース・プールを作成します：

```
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ることができます：

```
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

私たちの現代社会は、ネットワークを通じてコンピューターから提供される情報に大きく依存しています。モバイル機器は、人々がいつでもどこからでもネットワークにアクセスできるため、その依存度を高めています。このようなサービスを提供する場合、常に利用可能であることが非常に重要です。

可用性を数学的に定義すると、(A)、ある間隔の間にサービスが使用可能な合計時間と、(B)、間隔の長さの比率となります。これは通常、ある年の稼働時間のパーセンテージで表されます。

表15.1：可用性 - 年間ダウンタイム

空室率	年間ダウンタイム
99	3.65日
99.9	8.76時間
99.99	52分56秒
99.999	5.26分
99.9999	31.5秒
99.99999	3.15秒

可用性を高める方法はいくつかあります。最もエレガントな解決策は、ソフトウェアを書き換えて、複数のホストで同時に実行できるようにすることです。ソフトウェア自体がエラーを検出し、フェイルオーバーを行う方法が必要です。読み取り専用のウェブページを提供するだけなら、これは比較的簡単です。しかし、これは一般的に複雑で、ソフトウェアを自分で修正することができないため、不可能な場合もあります。以下の解決策は、ソフトウェアを修正することなく動作します：

- 信頼性の高い「サーバー」コンポーネントの使用

#### 備考

同じ機能を持つコンピュータ・コンポーネントでも、コンポーネントの品質によって信頼性の数値は異なります。ほとんどのベンダーは、信頼性の高いコンポーネントを「サーバー」コンポーネントとして販売しています。

- 単一障害点の排除（冗長コンポーネント）

- 無停電電源装置（UPS）の使用
  - サーバーに冗長電源を使用
  - ECC-RAMを使用
  - 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データに分散冗長ストレージを使用
- ダウンタイムの削減
    - 迅速にアクセス可能な管理者（24時間365日）
    - スペアパーツの入手可能性（Proxmox VEクラスタの他のノード）
    - 自動エラー検出（ha-managerによって提供されます）
    - 自動フェイルオーバー（ha-managerにより提供）

Proxmox VEのような仮想化環境では、「ハードウェア」への依存がなくなるため、高可用性の実現が非常に容易になります。また、冗長ストレージやネットワークデバイスのセットアップと使用もサポートされているため、1台のホストに障害が発生した場合でも、クラスタ内の別のホストでそれらのサービスを開始するだけで済みます。

さらに良いことに、Proxmox VEはha-managerと呼ばれるソフトウェアスタックを提供しています。エラーを自動的に検出し、自動フェイルオーバーを行うことができます。

Proxmox VE ha-managerは、"自動化された"管理者のように動作します。まず、管理すべきリソース（VM、コンテナなど）を設定します。次に、ha-managerは正しい機能を監視し、エラーが発生した場合に別のノードへのサービスフェイルオーバーを処理します。

しかし、高い可用性には代償が伴います。高品質の部品は高価であり、冗長化することでコストは少なくとも2倍になります。スペアパーツを追加すれば、コストはさらに増加します。そのため、利点を慎重に計算し、追加コストと比較する必要があります。

#### チップ

可用性を99%から99.9%に高めるのは比較的簡単です。しかし、99.9999%から99.99999%に可用性を高めるのは非常に難しく、コストがかかります。ha-managerの典型的なエラー検出とフェイルオーバーの時間は約2分なので、99.999%以上の可用性を得ることはできません。

## 15.1 必要条件

HAを始める前に、以下の条件を満たしている必要があります：

- 少なくとも3つのクラスタノード（信頼できるクオーラムを取得するため）
- VMとコンテナ用の共有ストレージ
- ハードウェアの性
- 信頼性の高い「サーバー」コンポーネントを使用
- ハードウェア・ウォッチドッグ - 利用できない場合は、Linuxカーネル・ソフトウェア・ウォッチドッグ（ソフトドッグ）にフォールバックします。
- オプションのハードウェア・フェンス装置

## 15.2 リソース

ha-managerが扱う主要な管理単位をリソースと呼びます。リソース ("service"とも呼ばれます) は、リソースタイプとタイプ固有のID、例えばvm:100からなるサービスID (SID) によって一意に識別されます。例えば、vm:100のようなものです。この例では、vm (仮想マシン) タイプでIDが100のリソースとなります。

仮想マシンとコンテナです。ここでの基本的な考え方の1つは、関連するソフトウェアをこのような仮想マシンやコンテナにバンドルすることができるということです、rgmanagerの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はありません。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 管理業務

このセクションでは、一般的な管理タスクの簡単な概要を説明します。最初のステップは、リソースのHAを有効にすることです。これは、リソースをHAリソース設定に追加することで行います。これはGUIを使用して行うこともできますし、単にコマンドラインツールなどを使用することもできます：

```
# ha-manager add vm:100
```

HAスタックはリソースを起動し、実行し続けようとします。要求された「リソースの状態を設定できることに注意してください。例えば、HAスタックにリソースを停止させたい場合があります：

```
# ha-manager set vm:100 --state stopped
```

またやり直します：

```
# ha-manager set vm:100 --state started
```

通常のVMやコンテナ管理コマンドも使用できます。コマンドは自動的にHAスタックに転送されるので

```
# qm start 100
```

は単に要求された状態を `started` に設定します。`qm stop` も同様で、要求された状態を `stopped` に設定します。

### 備考

HAスタックは完全に非同期で動作し、他のクラスタメンバーと通信する必要があります。そのため、このようなアクションの結果が表示されるまで数秒かかります。

現在のHAリソース設定を表示するには

```
# ha-manager config vm:100  
    状態停止
```

そして、実際のHAマネージャーとリソースの状態を見ることができます：

```
# ha-manager status quorum OK  
master node1 (active, Wed Nov 23 11:07:23 2016)  
lrm elsa (active, Wed Nov 23 11:07:19 2016) service vm:100 (node1,  
started)
```

他のノードへのリソース移行を開始することもできます:

```
# ha-manager migrate vm:100 node2
```

これはオンラインマイグレーションを使用し、VMを実行し続けようとします。オンライン・マイグレーションでは、使用済み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した方が速い場合があります。これはrelocateコマンドで実行できます:

```
# ha-manager relocate vm:100 node2
```

最後に、以下のコマンドを使用して、HA構成からリソースを削除できます:

```
# ha-manager remove vm:100
```

---

#### 備考

これはリソースを開始したり停止したりするものではありません。

---

しかし、HA関連のタスクはすべてGUIで行えるので、コマンドラインを使う必要はありません。

## 15.4 仕組み

このセクションでは、Proxmox VE HAマネージャの内部について詳しく説明します。関係するすべてのデーモンとその連携方法について説明します。HAを提供するために、各ノードで2つのデーモンが実行されます:

### ペハールム

ローカル・リソース・マネージャ（LRM）は、ローカル・ノード上で実行されているサービスを制御します。現在のマネージャステータスファイルからサービスの要求状態を読み取り、それぞれのコマンドを実行します。

### PVE-HA-CRM

クラスタ全体の意思決定を行うクラスタ・リソース・マネージャ（CRM）。LRMにコマンドを送信し、その結果を処理し、何か障害が発生した場合はリソースを他のノードに移動します。CRMはノードのフェンシングも行います。

---

#### 備考

ロックは分散設定ファイルシステム(pmxefs)によって提供されます。ロックは、各LRMが一度だけアクティブになって動作することを保証するために使用されます。LRMはロックを保持しているときのみアクションを実行するため、ロックを取得することができれば、障害が発生したノードをフェンスされたノードとしてマークすることができます。これにより、障害が発生したHAサービスを、未知の障害ノードからの干渉を受けずに安全に復旧させることができます。これは現在マネージャ・マスター・ロックを保持しているCRMによって監視されます。

---

## 15.4.1 サービス提供国

CRMはサービス状態の列挙を使用して現在のサービス記録します。この状態はGUIに表示され、ha-managerコマンドラインツールを使用して照会できます：

```
# ha-manager status quorum OK
エルザ様 (active, Mon Nov 21 07:23:29 2016)
lrm elsa (active, Mon Nov 21 07:23:22 2016) service ct:100 (elsa,
stopped)
service ct:102 (elsa, started) service
vm:501 (elsa, started)
```

以下は可能な状態のリストです：

### 停止

サービスが停止しました（LRMで確認）。LRMは停止したサービスがまだ実行中であることを検出すると再度停止します。

### リクエストストップ

サービスを停止してください。CRMはLRMからの確認を待ちます。

### へいせん

停止要求の保留。しかし、CRMは今のところリクエストを受け取っていません。

### 開始

サービスがアクティブな場合、LRMは、まだ実行中でなければ、それを早急に開始する必要があります。サービスが失敗し、実行されていないことが検出された場合、LRMはサービスを再起動します（「[開始失敗ポリシー](#)」を参照）。

### スタート

開始要求を保留中です。しかし、CRMはLRMからサービスが実行されていることを確認していません。

### フェンス

サービスノードがクオーレートクラスタパートナーション内にないため、ノードのフェンシングを待ちます（[フェンシングを参照](#)）。ノードのフェンシングが成功すると、サービスはリカバリ状態になります。

### 回復

サービスの復旧を待ちます。HAマネージャは、サービスが実行できる新しいノードを見つけようとします。この探索はオンラインノードと準正規ノードのリストに依存するだけでなく、サービスがグループメンバーであるかどうか、またそのようなグループがどのように制限されているかにも依存します。新しい利用可能なノードが見つかり次第、サービスはそこに移動され、最初は停止状態になります。実行するように設定されている場合は、新しいノードが実行します。

### 凍る

サービス状態には触れないでください。この状態はノードのリブート時やLRMデーモンの再起動時に使用します（[パッケージの更新を参照](#)）。

## 無視

あたかもサービスがHAによって管理されていないかのように振る舞います。一時的にサービスを完全に管理場合に便利です。

## マイグレート

他のノードにサービス（ライブ）を移行します。

## エラー

LRM エラーが原因でサービスが無効になっています。手動操作が必要です（[エラー回復](#)を参照）。

## キューに入れられた

サービスは新しく追加されたもので、CRMは今のところ見ていません。

## 使用禁止

サービスが停止し、無効とマークされました。

## 15.4.2 ローカル・リソース・マネージャー

ローカルリソースマネージャ (`pve-ha-lrm`) は起動時にデーモンとして起動され、HAクォーレートされ、クラスタ全体のロックが機能するまで待機します。

それは3つの状態です：

### エージェントロック待ち

LRMは排他的ロックを待ちます。これは、サービスが設定されていない場合、アイドル状態としても使用されます。

### アクティブ

LRMは排他的ロックを保持し、サービスが設定されています。

### エージェントロックの紛失

LRM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。

LRM がアクティブな状態になると、`/etc/pve/ha/manager_status` にあるマネージャステータスファイルを読み、所有するサービスに対して実行するコマンドを決定します。各コマンドに対してワーカーが開始され、これらのワーカーは並列に実行され、デフォルトでは最大4つまでに制限されています。このデフォルト設定は、データセンター設定キー `max_worker` で変更できます。終了するとワーカープロセスは収集され、その結果は CRM のために保存されます。

---

### 備考

デフォルト値の最大同時ワーカー数は、特定のセットアップには適さないかもしれません。例えば、4つのライブマイグレーションが同時に発生する可能性があり、低速なネットワークや（メモリ的に）大きなネットワークが混雑する可能性があります。また、最悪の場合、`max_worker` の、輻輳が最小になるようしてください。逆に、特にパワフルでハイエンドなセットアップを使用している場合は、`max_worker` の値を大きくすることもできます。

---

CRMが要求する各コマンドは UIDで一意に識別できます。ワーカーが終了すると、その結果は処理され、LRMステータスファイル /etc/pve/nodes/<nodename>/lrm\_status に書き込まれます。そこでCRMはそれを収集し、コマンドの出力に応じたステートマシンを動作させることができます。

通常、CRMとLRM間の各サービスのアクションは常に同期されます。これは、CRMがUIDで一意にマークされた状態を要求し、LRMがそのアクションを1回実行し、同じUIDで識別可能な結果を書き戻すことを意味します。これは、LRMが古いコマンドを実行しないために必要です。この動作の唯一の例外は、停止コマンドとエラーコマンドです。この2つは生成される結果に依存せず、停止状態の場合は常に実行され、エラー状態の場合は1回実行されます。

#### 備考

HA Stackはすべてのアクションをログに記録します。これはクラスタ内で何が、そしてなぜ何かが起こるのかを理解するのに役立ちます。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 journalctl -u pve-ha-lrm を、現在のマスターであるノードで pve-ha-crm に対して同じコマンドを使用することができます。

### 15.4.3 クラスタリソースマネージャ

クラスタリソースマネージャ(pve-ha-crm)は各ノード上で起動し、一度に1つのノードのみが保持できるマネージャロックを待ちます。マネージャロックの取得に成功したノードはCRMマスターに昇格します。

それは3つの状態です：

#### エージェントロック待ち

CRMは排他的ロックを待ちます。サービスが設定されていない場合、これはアイドル状態としても使用されます。

#### アクティブ

CRMは排他的なロックを保持し、サービスが設定されています。

#### エージェントロックの紛失

CRMのロックが失われました。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。

その主なタスクは、高可用性に設定されたサービスを管理し、常に要求された状態を強制しようとすることです。たとえば、要求された状態が開始されたサービスは、まだ実行されていなければ開始されます。クラッシュした場合は自動的に再スタートします。このように、CRMはLRMが実行する必要のあるアクションを指示します。

ノードがクラスタクオーラムから離脱すると、そのノードの状態はunknownに変わります。その後、現在のCRMが故障したノードのロックを確保できれば、サービスは盗まれて別のノードで再起動されます。

クラスタ・メンバがクラスタ・クオーラムから外れたと判断すると、LRMは新しいクオーラムが形成されるのを待ちます。クラスタ・クオーラムが形成されるまで、ノードはウォッチドッグをリセットできません。ノード上にアクティブなサービスがある場合、またはLRMまたはCRMプロセスがスケジュールされていない場合、または強制終了された場合、ウォッチドッグがタイムアウトした後に再起動がトリガされます（これは60秒後に発生します）。

ノードにアクティブなCRMがあり、LRMがアイドル状態である場合、クオーラムが失われてもセルフ・フェンスのリセットはトリガされないことに注意してください。この理由は、CRMがアクセスするすべての状態ファイルとコンフィギュレーションが

クラスタ構成ファイルシステムは、クオーラムが失われると読み取り専用になります。つまり、CRMはプロセスが長時間スケジューリングされないように自身を保護するだけでよく、その場合、別のCRMが状況を知らずに引き継ぎ、HA状態の破損を可能性があります。オープン・ウォッチドッグは、このようなことが起こらないようにします。

サービスが 15 分以上設定されないと、CRM は自動的にアイドル状態に戻り、ウォッチドッグは完全に終了します。

## 15.5 HAシミュレータ

The screenshot shows the Proxmox VE web interface. On the left, there is a sidebar with various management options like 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 sections: 'Status' and 'Resources'. The 'Status' section contains a table with columns 'Type' and 'Status'. It shows a 'quorum' entry with 'OK' status. The 'Resources' section contains a table with columns 'ID', 'State', 'Node', 'Max. Restart', 'Max. Relo...', 'Group', and 'Description'. It lists two entries: 'vm:501' with state 'stopped' on 'demohost1', and 'ct:510' with state 'queued' on 'demohost1'. Both entries have a 'prefer\_node1' group and no description.

HAシミュレータを使用すると、Proxmox VE HAソリューションのすべての機能をテストし、学習することができます。

デフォルトでは、シミュレータは6つのVMを持つ実際の3ノードクラスタの動作を監視し、テストすることができます。また、VMやコンテナを追加したり削除したりすることもできます。

実際のクラスタをセットアップしたり設定したりする必要はありません。aptでインストールしてください:

```
apt install pve-ha-simulator
```

他のProxmox VEパッケージがないDebianベースのシステムにもインストールできます。その場合、パッケージをダウンロードし、インストールしたいシステムにコピーする必要があります。ローカルファイルシステムから apt を使ってパッケージをインストールすると、必要な依存関係も解決されます。

リモートマシンでシミュレータを起動するには、現在のシステムに X11 リダイレクトを設定する必要があります。Linux マシンの場合は、次のようにします:

```
ssh root@<IPofPVE>-Y.
```

Windowsではmobaxtermで動作します。

シミュレータがインストールされた既存の Proxmox VE に接続するか、ローカルの Debian ベースのシステムに手動でインストールした後、以下のようにして試すことができます。

まず、シミュレータが現在の状態を保存し、デフォルトの設定を書き込む作業ディレクトリを作成する必要があります：

```
mkdir 作業中
```

作成したディレクトリを `pve-ha-simulator` にパラメータとして渡します：

```
pve-ha-シミュレータ作業/
```

その後、シミュレーションしたHAサービスを開始、停止、移行したり、ノード障害時に何が起こるかをチェックすることもできます。

## 15.6 構成

HAスタックはProxmox VE APIにうまく統合されています。例えば、`ha-manager` コマンドラインインターフェースや Proxmox VE ウェブインターフェースで HA を設定することができます。自動化ツールはAPIを直接使用できます。

すべてのHA設定ファイルは/`etc/pve/ha`/内にあるため、クラスタノードに自動的に配布され、すべてのノードが同じHA設定を共有します。

### 15.6.1 リソース

The screenshot shows the Proxmox VE web interface under the 'Datacenter' tab. On the left, a sidebar menu is open, showing various management sections like Summary, Options, Storage, Backup, Replication, Permissions, Groups, Pools, Roles, Authentication, and HA. The 'HA' section is currently selected and highlighted in blue. The main content area has two tabs: 'Status' and 'Resources'. The 'Status' tab displays a table with columns 'Type' and 'Status'. It shows one 'quorum' entry labeled 'OK' and two 'master' entries, both 'demohost2' (active, Fri Jun 30 09:41:54 2017). The 'Resources' tab displays a table with columns 'ID', 'State', 'Node', 'Max. Restart', 'Max. Reio...', 'Group', and 'Description'. It lists two resources: 'vm:501' (stopped, demohost1, 1, 1, prefer\_node1) and 'ct:510' (queued, demohost1, 1, 1, mygroup1).

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保存します。そのリスト内のリソース設定は以下のようになります：

```
<タイプ>: <名前  
      <プロパティ> <値  
      ...
```

リソースIDは、リソースの種類から始まり、コロンで区切られたリソース固有の名前が続きます。これはすべてのha-managerコマンドでリソースを一意に識別するために使用されます(例: vm:100またはct:101)。次の行は追加のプロパティです:

### コメント:<文字列

説明

### group: <文字列

HAグループの識別子。

### max\_relocate: <整数> (0 - N) (デフォルト= 1)

サービスの起動に失敗した場合のサービス再配置の最大試行回数。

### max\_restart: <整数> (0 - N) (デフォルト= 1)

起動に失敗したノードでサービスを再起動する最大回数。

### state: <無効 | 有効 | 無視 | 開始 | 停止> (=

#### 開始

要求されたリソースの状態。CRMはこの状態を読み取り、それに従って動作します。有効な  
は単にstartedの別名です。

#### 開始

CRMがリソースの起動を試みます。起動に成功すると、サービス状態はstartedに設定されます。オン  
ノードに障害が発生した場合、または起動に失敗した場合、リソースの回復を試みます。すべてが失敗した場合、サービスの状態はエラーに設  
定されます。

#### 停止

CRMはリソースを停止維持しようとしていますが、ノード障害時にはリソースの再配置を試みます。

#### 使用禁止

CRMはリソースを停止状態にしようとしますが、リソースの再配置は行いません。

ノードの障害時にこのステートの主な目的はエラーリカバリーで、リソースをエラー状態から解放する唯一の方法だからです。

#### 無視

リソースはマネージャーステータスから削除されるため、CRMとLRMはリソースに触れなくなります。このリソースに影響する全ての{pve}API  
コールが実行されます。このリソースに影響を与えるAPIコールは、HAスタックを直接バイパスして実行されます。ソースがこの状態にある間、  
CRMコマンドは破棄されます。リソースはノード障害時に再配置されません。

以下は、1つのVMと1つのコンテナを使った実際の例です。ご覧の通り、これらのファイルの構文は実にシンプルなので、お気に入りのエディタを使ってファイル  
を読んだり編集したりすることも可能です:

**設定例 (/etc/pve/ha/resources.cfg)**

```
vm: 501
    状態開始 最大再配置数 2

CT: 102
    # Note: すべてデフォルト設定を使用
```

上記のコンフィグはha-managerコマンドラインツールを使って生成されました:

Add: Resource: Container/Virtual Machine

CT/VM ID:	510	Group:	prefer_node1
Max. Restart:	1	Request State:	started
Max. Relocate:	1	Comment:	
		<a href="#">Help</a>	<a href="#">Add</a>

```
# ha-manager add vm:501 --state started --max_relocate 2 # ha-manager add ct:102
```

## 15.6.2 グループ

Datacenter

Groups				
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

[Create](#) [Edit](#) [Remove](#)

- Q Search**
- S Summary**
- O Options**
- S Storage**
- B Backup**
- R Replication**
- P Permissions**
  - U Users**
  - G Groups**
  - P Pools**
  - R Roles**
  - A Authentication**
- HA**
  - G Groups**
  - F Fencing**
  - F Firewall**
  - S Support**

クラスタノードのグループを定義するには、HAグループ設定ファイル/etc/pve/ha/groups.cfgを使用します。リソースの実行を制限することができます。グループ設定は以下のようになります:

```
group: <グループ>
    ノード <ノードリスト>
    <プロパティ> <値>
    ...
```

## コメント:<文字列>

説明

### **nodes: <node>[:<pri>] { ,<node>[:<pri>] }\*.**

クラスタ・ノード・メンバーのリストで、各ノードに優先順位を与えることができます。リソースは

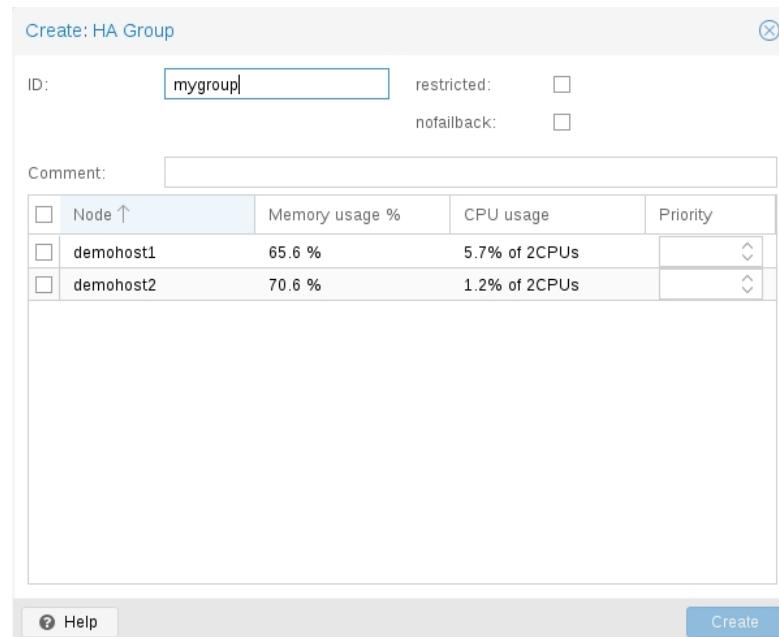
グループは、最も優先度の高い利用可能なノードで実行されます。最も優先度の高いクラスのノードが多ければ、それらのノードにサービスが分散されます。優先順位は相対的な意味しか持ちません。数字が大きいほど優先度が高くなります。

### **nofailback: <boolean> (デフォルト= 0)**

CRM は最も優先度の高いノードでサービスを実行しようとします。より優先度の高いノードがオンラインになると、CRM はサービスをそのノードに移行します。nofailback を有効にすると、この動作が防止されます。

### **restricted: <boolean> (デフォルト= 0)**

制限付きグループにバインドされたリソースは、グループによって定義されたノード上でのみ実行できます。グループ・ノード・メンバがオンラインでない場合、リソースは停止状態になります。非制限グループのリソースは、グループ・メンバがすべてオフラインの場合、どのクラスタ・ノードでも実行できますが、グループ・メンバがオンラインになるとすぐに元に戻ります。メンバが1人だけの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。



一般的な要件として、リソースは特定のノードで実行する必要があります。通常、リソースは他のノードでも実行できるので、単一のメンバーで制限のないグループを定義できます：

```
# ha-manager groupadd prefer_node1 --nodes node1
```

大規模なクラスタでは、より詳細なフェイルオーバー動作を定義することが理にかなっています。例えば、可能であればnode1で一連のサービスを実行したいとします。node利用できない場合はnode2とnode3で均等にサービスを実行します。これらのノードにも障害が発生した場合は、node4でサービスを実行する必要があります。これを実現するには、ノードリストを次のように設定します：

```
# ha-manager groupadd mygroup1 -nodes "node1:2,node2:1,node3:1,node4"
```

もう1つのユースケースは、リソースが特定のノード（例えばnode1とnode2）でのみ利用可能な他のリソースを使用する場合です。HAマネージャが他のノードを使用しないようにする必要があるため、当該ノードで制限グループを作成する必要があります：

```
# 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作成されました：

#### 設定例 (/etc/pve/ha/groups.cfg)

```
グループ: prefer_node1
ノード node1
```

```
グループ: mygroup1
ノード2:1, ノード4, ノード1:2, ノード3:1
```

```
グループ: mygroup2
ノード node2,node1
restricted 1
```

nofailbackオプションは、管理タスク中に不要なリソースの移動を回避するのに便利です。例えば、あるサービスをグループ内で最も高い優先度を持たないノードに移行する必要がある場合、nofailbackオプションを設定することで、このサービスを即座に戻さないようにHAマネージャに指示する必要があります。

もう一つのシナリオは、あるサービスがフェンスにかけられ、別のノードにリカバリされた場合です。管理者はフェンスで保護されたノードを修復して再びオンラインにし、障害の原因を調査して再び安定して動作するかどうかを確認しようとします。nofailbackフラグを設定することで、復旧したサービスがそのままフェンスで保護されたノードに戻ることを防ぎます。

## 15.7 フェンシング

ノードの障害時には、フェンシングによって、誤ったノードがオフラインであることが保証されます。これは、リソースが別のノードで復旧したときに2回実行されないようにするために必要です。これがないと、別のノードでリソースを回復できないため、これは本当に重要なタスクです。

もしノードがフェンスされなければ、共有リソースにアクセスできる可能性がある未知の状態になります。これは本当に危険です！ストレージ以外の全てのネットワークが壊れたとします。パブリック・ネットワークからはアクセスできませんが、VMはまだ稼働しており、共有ストレージに書き込んでいます。

このVMを別のノードで起動すると、両方のノードから書き込みを行うため、危険な競合状態が発生します。このような状態になると、VMのデータがすべて破壊され、VM全体が使用不能になる可能性があります。また、ストレージが複数マウントから保護されている場合、リカバリに失敗する可能性もあります。

### 15.7.1 ProxmoxのVEフェンス

例えば、ノードからの電力を遮断したり、通信を完全に無効にするフェンス・デバイスなどです。これらは非常に高価なことが多く、システムにさらに重要なコンポーネントを追加することになります。

そのため、外部ハードウェアを追加する必要のない、よりシンプルなフェンシング方法を統合したいと考えました。これにはウォッチドッグ・タイマーを使用します。

### 可能なフェンスの方法

- 外部電源スイッチ
- スイッチ上のネットワーク・トラフィックを完全に無効にすることで、ノードを隔離します。
- ウォッチドッグタイマーを使用したセルフフェンシング

ウォッチドッグ・タイマーは、マイクロ・コントローラーが登場した当初から、重要で信頼性の高いシステムで広く使用されてきました。ウォッチドッグ・タイマーは、多くの場合、コンピュータの誤動作を検出して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す。

通常動作中、`ha-manager`は定期的にウォッチドッグ・タイマーをリセットし、タイマーが経過しないようにします。ハードウェア障害またはプログラムエラーにより、コンピューターがウォッチドッグのリセットに失敗した場合、タイマーは経過し、サーバー全体のリセット（再起動）がトリガーされます。

最近のサーバー・マザーボードには、このようなハードウェア・ウォッチドッグが搭載されていることがよくありますが、これらを設定する必要があります。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します。それでも信頼性はありますが、サーバーのハードウェアから独立していないため、ハードウェア・ウォッチドッグよりも信頼性は低くなります。

## 15.7.2 ハードウェア・ウォッチドッグの設定

デフォルトでは、全てのハードウェア・ウォッチドッグ・モジュールはセキュリティ上のブロックされています。正しく初期化されないと、装填された銃のようになります。ハードウェア・ウォッチドッグを有効にするには、ロードするモジュールを

`/etc/default/pve-ha-manager` のします：

```
# ウォッチドッグモジュールの選択 (デフォルトはソフトドッグ) WATCHDOG_MODULE=itco_wdt
```

このコンフィギュレーションは `watchdog-mux` サービスによって読み取られ、起動時に指定されたモジュールがロードされます。

## 15.7.3 リカバー・フェンス・サービス

ノードに障害が発生し、そのフェンシングが成功した後、CRMは障害が発生したノードからオンライン状態のノードにサービスを移動しようとします。

これらのサービスが回復するノードの選択は、リソースグループによって影響されます。

設定、現在アクティブなノードのリスト、およびそれぞれのアクティブなサービス数。

CRMはまず、（グループ設定から）ユーザーが選択したノードと利用可能なノードの交点からセットを構築します。次に、最も優先順位の高いノードのサブセットを選択し、最後に最もアクティブなサービス数が少ないノードを選択します。これにより、ノードが過負荷になる可能性を最小限に抑えます。

### 注意

ノード障害が発生すると、CRMは残りのノードにサービスを分散します。これにより、これらのノードのサービス数が増加し、特に小規模なクラスタでは高負荷になる可能性があります。このような最悪のケースに対応できるようにクラスタを設計してください。

## 15.8 スタート失敗ポリシー

開始失敗ポリシーは、あるノードでサービスの開始が1回失敗した場合に有効になります。このポリシーを使用して、同じノードで再起動をトリガする頻度や、サービスを別のノードで開始できるように再配置する頻度を設定できます。このポリシーの目的は、特定のノードで共有リソースが一時的に利用できなくなるのを回避することです。例えば、共有ストレージがネットワークの問題などで利用できないが、他のノードではまだ利用できる場合、リロケート・ポリシーによってサービスを開始することができます。

各リソースに固有の設定を行うことができる2つのサービス開始回復ポリシー設定があります。

### 最大再起動

実際のノードで障害が発生したサービスの再起動を試行する最大回数。デフォルトは1回です。

### max\_relocate

サービスを別のノードに再配置する最大試行回数。実際のノードでmax\_restart値を超えた場合にのみ再配置が行われます。デフォルトは1です。

### 備考

再配置カウントの状態は、サービスの起動が少なくとも1回成功した場合にのみゼロにリセットされます。つまり、エラーが修正されずにサービスが再スタートした場合、再スタートポリシーだけが繰り返されます。

## 15.9 エラーリカバリー

すべての試行の後、サービスの状態を回復できなかった場合、エラーなります。この状態では、サービスはもうHAスタックに接続されません。唯一の方法は、サ

```
# ha-manager set vm:100 --state disabled
```

ービスを無効にすることです:

これはウェブ・インターフェイスでも可能です。

エラー状態から回復するには、次のようにしてください:

- リソースを安全で一貫性のある状態に戻す（例：サービスを停止できなかった場合、そのプロセスを終了させる）
- リソースを無効にしてエラーフラグを外します。
- 不具合の原因となったエラーを修正
- すべてのエラーを修正した後、サービスの再開をリクエストすることができます。

## 15.10 パッケージの更新

ha-managerをアップデートする際は、様々な、一度に全部をアップデートするのではなく、1つのノードを次々にアップデートしてください。第一に、私たちはソフトウェアを徹底的にテストしていますが、あなたの特定のセットアップに影響するバグを完全に排除することはできません。ノードを1つずつアップデートし、アップデート終了後に各ノードの機能をチェックしてください。

一方、一度にすべてのアップデートを行うとクラスタが壊れる可能性があり、一般的に良い方法ではありません。

また、Proxmox VE HAスタックは、クラスタとローカルリソースマネージャの間でアクションを実行するためにリクエストアクノリッジプロトコルを使用します。再起動のために、LRMはCRMにすべてのサービスをフリーズする要求を行います。これにより、LRMが再起動している短時間の間、クラスタによってこれらのサービスが触れられるのを防ぎます。、LRMは再起動中にウォッチドッグを安全に閉じることができます。このような再起動は通常パッケージの更新中に起こり、すでに述べたように、LRMからの要求を承認するためにアクティブなマスター CRMが必要です。そうでない場合、更新処理に時間がかかりすぎ、最悪の場合ウォッチドッグがリセットされる可能性があります。

## 15.11 ノードのメンテナンス

ハードウェアの交換や新しいカーネルイメージのインストールなど、ノードのメンテナンスが必要になることがあります。これはHAスタックが使用されている間にも当てはまります。

HAスタックは主に2種類のサポートします：

- 一般的なシャットダウンまたはリブートについては、[シャットダウンポリシー](#)を参照してください。
- シャットダウンや再起動を必要としないメンテナンス、または1回の再起動だけで自動的にオフにならないメンテナンスの場合、手動メンテナンスマードを有効にすることができます。

### 15.11.1 メンテナンスマード

手動メンテナンスマードを使用すると、ノードをHA動作不可としてマークし、HAによって管理されるすべてのサービスが他のノードに移行するように促すことができます。

これらの移行のターゲットとなるノードは、現在利用可能な他のノードから選択され、HAグループの構成と設定されているクラスタリソーススケジューラ (CRS) モードによって決定されます。各移行の間、元のノードはHAマネージャの状態に記録されるため、メンテナンスマードが無効になってノードがオンラインに戻ると、サービスは再び自動的に移行れます。

現在、ha-manager CLIを使用してメンテナンスマードを有効または無効にすることができます。

#### ノードのメンテナンスマードの有効化

```
# ha-manager crm-command node-maintenance enable NODENAME
```

これはCRMコマンドをキューに入れ、マネージャがこのコマンドを処理すると、マネージャのステータスにメンテナンスマードのリクエストが記録されます。これにより、メンテナンスマードにしたいノードだけでなく、どのノードでもコマンドを送信することができます。

各ノードの LRM がコマンドを拾うと、それ自身を利用不可としてマークしますが、すべてのマイグレーションコマンドを処理します。つまり、LRM のセルフフェンシング・ウォッチドッグは、すべてのアクティブなサービスが移動し、すべての実行中のワーカーが終了するまでアクティブなままでです。

LRMのステータスがメンテナンスマードと表示されるのは、LRMが要求されたステータスをした時点であり、すべてのサービスがした時点ではないことに注意してください。今のところ、ノード上にアクティブな HA サービスが残っていないか確認したり、次のようなログ行に注意したりすることで、ノードがいつメンテナンスマードへの移行を完了したかを知ることができます: pve-ha-lrm[PID]: watchdog closed (disabled).

**備考**

手動メンテナンスマードはノードのリブート時には自動的に削除されませんが、`ha-manager` CLIを使用して手動で解除するか、`manager-status`を手動でクリアした場合にのみ削除されます。

**ノードのメンテナンスマードの無効化**

```
# ha-manager crm-command node-maintenance disable NODENAME
```

手動メンテナンスマードを無効にするプロセスは、有効にするプロセスと似ています。上記の`ha-manager` CLIコマンドを使用すると、CRMコマンドがキューに入れられ、それが処理されると、それぞれのLRMノードが再び使用可能にマークされます。

メンテナンスマードを解除すると、メンテナンスマードを有効にしたときにノード上にあったサービスはすべて元に戻ります。

## 15.11.2 シャットダウン・ポリシー

以下に、ノードのシャットダウンに関するさまざまなHAポリシーの説明を示します。現在のところ、後方互換性のため条件付きがデフォルトです。ユーザによっては、*Migrate*の方がより期待通りの動作をすると感じるかもしれません。

シャットダウン・ポリシーは、Web UI (Datacenter→ Options→ HA Settings) で設定するか、`datacenter.cfg`で直接設定することができます：

```
ha: shutdown_policy=<値>です。
```

**移住**

ローカルリソースマネージャ(LRM)がシャットダウンリクエストを受け取り、このポリシーが有効になると、現在のHAマネージャに対して利用不可のマークを付けます。これにより、このノードに現在配置されているすべてのHAサービスの移行が開始されます。LRMは、実行中のサービスがすべて移動されるまで、シャットダウン処理を遅らせようとします。しかし、これは実行中のサービスが別のノードに移行できることを想定しています。つまり、ハードウェア・パススルーを使用するなどして、サービスがローカルにバインドされていない必要があります。グループメンバーでないノードは、グループメンバーが利用できない場合、実行可能なターゲットとみなされるため、一部のノードだけを選択してHAグループを利用する場合でも、このポリシーを使用することができます。しかし、グループを制限ノードとしてマークすることで、HAマネージャは選択されたノードセット以外ではサービスを実行できないことを伝えます。これらのノードがすべて利用できない場合、手動で介入するまでシャットダウンは停止します。シャットダウンされたノードが再びオンラインに戻ると、その間にまだ手動で移行されていない場合、以前に置き換えたサービスは元に戻ります。

**備考**

ウォッチャドッグはシャットダウンの移行プロセス中もアクティブです。ノードがクオーラムを失った場合、それはフェンスされ、サービスは回復されます。

現在メンテナンス中のノードで（以前に停止した）サービスを開始する場合、サービスを移動して別の利用可能なノードで開始できるように、そのノードをフェンスで囲む必要があります。

## フェイルオーバー

このモードでは、現在のノードがすぐにオンラインにならない場合、すべてのサービスが停止されますが、リカバリも行われます。一度に多くのノードがパワーオフされるとVMのライブマイグレーションが不可能になる可能性がありますが、それでもHAサービスを確実にリカバリしてできるだけ早く再スタートさせたい場合など、クラスタ規模でメンテナンスを行う場合に便利です。

## フリーズ

このモードでは、すべてのサービスが停止してフリーズし、現在のノードが再びオンラインになるまで復旧しないようにします。

### 条件付き

条件付きシャットダウンポリシーは、シャットダウンまたは再起動が要求されたかどうかを自動的に検出し、それに応じて動作を変更します。

### シャットダウン

シャットダウン（電源オフ）は通常、ノードがしばらくの間停止する予定がある場合に実行されます。この場合、LRMはすべての管理サービスを停止します。これは、その後、他のノードがこれらのサービスを引き継ぐことを意味します。

#### 備考

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てから再起動し、すべてのRAMのオンラインマイグレーションを回避します。オンライン・マイグレーションを使用したい場合は、ノードをシャットダウンする前に手動でそれを起動する必要があります。

### 再起動

ノードの再起動は`reboot`コマンドで行います。これは通常、新しいカーネルをインストールした後に行われます。これは「シャットダウン」とは異なることに注意してください。

LRMはCRMに再起動を指示し、CRMがすべてのリソースをフリーズ状態にするまで待ちます（[Package Updates](#)と同じメカニズム）。これにより、これらのリソースが他のノードに移動することを防ぎます。代わりに、CRMは同じノード上で再起動後にリソースを起動します。

### 手動リソース移動

最後になりますが、ノードをシャットダウンまたは再起動する前に、手動でリソースを他のノードに移動することができます。オンラインマイグレーションを使うかどうかは自分で決めることができます。

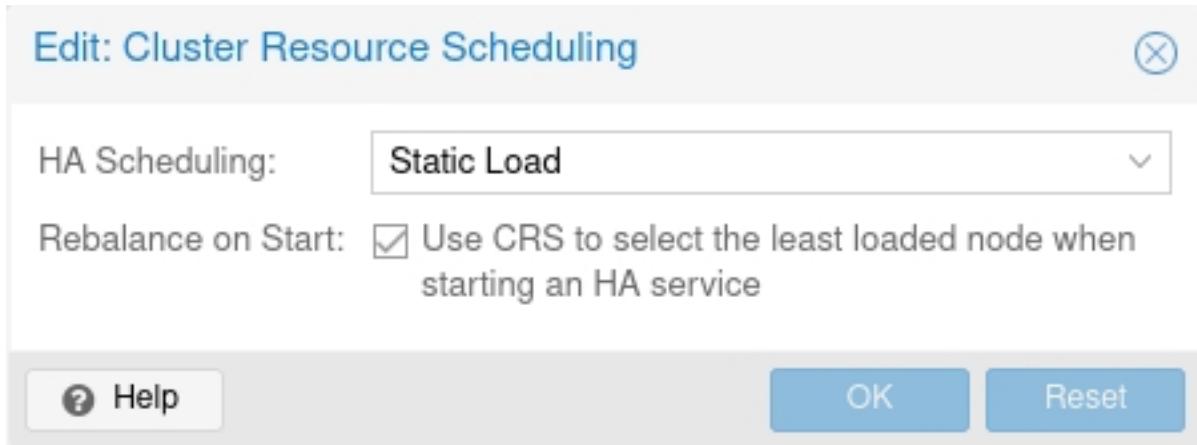
#### 備考

`pve-ha-crm`、`pve-ha-lrm`、`watchdog-mux`のようなサービスを停止しないでください。これらはウォッチドッグを管理・使用しているため、ノードのリブートやリセットを引き起こす可能性があります。

## 15.12 クラスタ・リソース・スケジューリング

クラスタリソーススケジューラ (CRS) モードは、シャットダウンポリシーによってトリガされる移行だけでなく、サービスの回復のためにHAがノードを選択する方法を制御します。デフォルトのモードはベーシックで、Web UI (Datacenter → Options) で変更するか、datacenter.cfgで直接変更できます:

```
crs: ha=静的
```



変更は次の監督ラウンドから有効になります（数後）。

復旧や移行が必要な各サービスに対して、スケジューラはそのサービスのグループ内で最も優先順位の高いノードの中から最適なノードを繰り返し選択します。

### 備考

将来的には、（静的および動的な）負荷分散のモードを追加する予定です。

### 15.12.1 基本スケジューラー

各ノードのアクティブなHAサービスの数は、復旧ノードを選択するために使用されます。HA管理されていない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

静的モードはまだ技術プレビューです。

各ノードのHAサービスからの静的な利用情報は、回復ノードを選択するために使用されます。HAで管理されていないサービスの現在のところ考慮されていません。

この選択のために、各ノードは、関連するゲスト構成からのCPUとメモリ使用量を使用して、そのノード上でサービスがすでに実行されているかのように順番に検討されます。次に、このような各選択肢について、すべてのノードのCPUとメモリの使用量が考慮されます。CPUとメモリの両方について、ノードの中で最も高い使用量（理想的にはどのノードもオーバーコミットされるべきではないため、より重み付けされます）と、すべてのノードの平均使用量（よりコミット率の高いノードがある場合に区別できるようにするため）が考慮されます。

**重要**

サービスの数が多ければ多いほど、可能な組み合わせも増えるため、数千ものHAマネージドサービスをお持ちの場合は、現在のところ使用することはお勧めできません。

### 15.12.3 CRSのスケジューリングポイント

CRS アルゴリズムはラウンドごとにすべてのサービスに適用されるわけではありません。ワークロードによっては、常時バランスするよりもクラスタに負担がかかる可能性があります。そのため、Proxmox VE HAマネージャはサービスを現在のノードに維持することを推奨しています。

CRSは現在、次のようなスケジューリング使用されています：

- サービス復旧（常時アクティブ）。アクティブなHAサービスを持つノードが故障した場合、そのノードの全てのサービスを他の復旧させる必要があります。CRSアルゴリズムは、残りのノードにバランスよく復旧させるために使用されます。
- HAグループ設定の変更（常にアクティブ）あるノードがグループから削除されたり、そのノードの優先度が下がったりすると、HAスタックはCRSアルゴリズムを使って、適応された優先度制約に合う、そのグループ内のHAサービスの新しいターゲットノードを見つけます。
- HAサービス停止→開始移行（オプトイン）。停止しているサービスを開始するように要求することは、CRSアルゴリズムに従って最適なノードをチェックする良い機会です。これは、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にできます。このオプションは Web UI の Datacenter → Options → Cluster Resource Scheduling でも変更できます。

## 第16章

# バックアップと復元

Proxmox VEは、各ストレージおよび各ゲストシステムタイプの機能を使用して、完全に統合されたソリューションを提供します。Proxmox VEは、各ストレージおよび各ゲストシステムのタイプの機能を使用して、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ダウンタイムをモードオプションで微調整することができます。

Proxmox VEのバックアップは、常にVM/CT設定とすべてのデータを含むフルバックアップです。バックアップはGUIまたはvzdumpコマンドラインツールから開始できます。

### バックアップ・ストレージ

バックアップを実行する前に、バックアップストレージを定義する必要があります。ストレージを追加する方法については、[ストレージのドキュメント](#)を参照してください。Proxmox Backup Serverストレージの場合、バックアップは複製解除されたチャンクとメタデータとして保存され、ファイルレベルのストレージの場合、バックアップは通常のファイルとして保存されます。Proxmox Backup Serverは高度な機能を備えているため、専用ホストで使用することをお勧めします。NFSサーバを使用するのも良い方法です。どちらの場合も、バックアップをテープドライブに保存し、オフサイトアーカイブにすることができます。

### 定期バックアップ

バックアップジョブは、選択可能なノードとゲストシステムに対して、特定の日時に自動的に実行されるようにスケジュールできます。詳細はバックアップジョブセクションを参照してください。

## 16.1 バックアップモード

一貫性（オプションモード）を提供するには、ゲストのタイプによっていくつかの方法があります。

VMのバックアップモード：

### 停止モード

このモードでは、VMの動作が短時間停止する代わりに、バックアップの一貫性が最も高くなります。このモードは、VMの整然としたシャットダウンを実行し、バックグラウンドでQEMUプロセスを実行してVMデータをバックアップします。バックアップの開始後、VMはフル稼働モードに移行。ライブバックアップ機能を使用することで、一貫性が保証されます。

## サスPENDモード

このモードは互換性のために提供されており、スナップショット・モードを呼び出す前にVMを一時停止します。VMをサスPENDするとダウンタイムが長くなり、データの一貫性が向上するとは限らないため、スナップショット・モードの使用をお勧めします。

## スナップショットモード

このモードは、わずかな不整合リスクを伴いますが、操作のダウンタイムを最小限に抑えます。このモードはProxmox VEのライブバックアップを実行することで機能し、VMの実行中にデータブロックがコピーされます。ゲストエージェントが有効 (エージェント: 1) で実行中の場合、`guest-fsfreeze-freeze` と `guest-fsfreeze-thaw` を呼び出して整合性を向上させます。

### 備考

Windows ゲストで、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は、ゲストエージェントを設定する必要があります。詳細については、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用Proxmox VEライブバックアップの技術概要は、[こちら](#)をご覧ください。

### 備考

Proxmox VEライブバックアップは、あらゆるストレージタイプでスナップショットのようなセマンティクスを提供します。基礎となるストレージがスナップショットをサポートしている必要はありません。また、バックアップはバックグラウンドのQEMUプロセスで実行されるため、VMディスクがQEMUによって読み込まれている間、停止したVMは短時間実行されているように見えます。ただし、VM自体は起動せず、ディスクだけが読み込まれます。

コンテナのバックアップモード：

## 停止モード

バックアップの間、コンテナを停止します。これにより、ダウンタイムが非常に長くな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のデータを一時的な場所にコピーします（オプション `--tmpdir` を参照）。その後、コンテナが一時停止され、2回目の `rsync` で変更されたコピーされます。コンテナが再び起動（再開）されます。この結果、ダウンタイムは最小限になりますが、コンテナのコピーを保持するための追加の領域が必要になります。

コンテナがローカル・ファイル・システム上にあり、バックアップのターゲット・ストレージが NFS/CIFS サーバである場合、`-tmpdir` もローカル・ファイル・システム上に存在するように設定する必要があります。バックアップ・ストレージがNFSサーバの場合、サスPEND・モードでACLを使用してローカル・コンテナをバックアップする場合にも、ローカルの`tmpdir`を使用する必要があります。

## スナップショットモード

このモードでは、基礎となるストレージのスナップショット機能を使用します。まず、データの一貫性を確保するためにコンテナを一時停止します。コンテナのボリュームの一時スナップショットが作成され、スナップショットの内容がtarアーカイブされます。最後に、一時スナップショットは再び削除されます。

**備考**

スナップショット・モードでは、すべてのバックアップ・ボリュームがスナップショットをサポートするストレージ上にある必要があります。`backup=no`マウントポイントオプションを使用すると、個々のボリュームをバックアップ（およびこの要件）から除外できます。

**備考**

デフォルトでは、ルート ディスク マウント ポイント以外の追加のマウント ポイントはバックアップに含まれません。ボリューム マウント ポイントについては、バックアップ オプションを設定して、マウント ポイントをバックアップに含めることができます。デバイスマウントとバインドマウントは、コンテンツがProxmox VEストレージ ライブラリの外部で管理されるため、バックアップされることはありません。

### 16.1.1 VM/バックアップフリッキング

VMのバックアップが開始されると、QEMUIはブロック層に「copy-before-write」フィルタをインストールします。このフィルタは、新しいゲストの書き込み時に、バックアップにまだ必要な古いデータが最初にバックアップターゲットに送信されることを保証します。ゲストの書き込みはこの操作が終了するまでブロックされるため、まだバックアップされていないセクタへのゲストの IO はバックアップターゲットの速度によって制限されます。

バックアップフリークでは、そのような古いデータはバックアップターゲットに直接送信されるのではなく、フリークイメージにキャッシュされます。これはゲストのIOパフォーマンスを助け、特定のシナリオではハングを防ぐこともできますが、その代償としてより多くのストレージ容量が必要になります。

ストレージlocal-lvm上に作成されたフリーリング・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.
```

いつものように、[設定オプション](#)を使用して、特定のバックアップジョブ、またはノード全体のフォールバックとしてオプションを設定できます。UIでは、`fleecing`はバックアップジョブの編集時に[Advanced](#)タブで設定できます。

フリークするストレージは、シンプロビジョンングとディスクカードサポートを備えた高速なローカルストレージであるべきです。例えば、LVM-thin、RBD、ストレージ構成にスパース1を持つZFS、多くのファイルベースのストレージです。理想的には、fleecingストレージは専用ストレージで、フルに動作しても他のゲストには影響せず、バックアップに失敗するだけです。バックアップされたfleecingイメージの一部は、スペース使用量を低く抑えるために破棄されます。

廃棄をサポートしないファイルベースのストレージ（例えば、バージョン4.2以前のNFS）の場合、ストレージの設定で`preallocation`を`off`に設定する必要があります。`qcow2` (fleecing イメージのフォーマットとして、ストレージがサポートしている場合、自動的に使用されます) と組み合わせることで、すでに割り当てられているイメージの一部を後で再利用できるという利点があります。

**警告**

シンプロビジョンングされていないストレージ、例えばスパースオプションのないLVMやZFSでは、オリジナルディスクのフルサイズを前もってフリーシングイメージ用に予約しておく必要があります。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ビギーになっている間にゲストがディスク全体を再書き込みした場合にのみ、フリーリングイメージはオリジナルイメージと同じサイズまで成長できます。

### 16.1.2 CT変化検出モード

変更検出モードを設定すると、pxar アーカイブのエンコード形式が定義され、Proxmox Backup Server をターゲットとするコンテナバックアップで変更されたファイルや変更されていないファイルがどのように処理されるかが定義されます。

変更検出モードオプションは、ジョブの編集中にAdvancedタブで個々のバックアップジョブに対して設定することができます。さらに、このオプションは[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できます。

3つの変化検出モードがあります：

モード	説明
デフォルト	pxarフォーマットバージョンを使用して、すべてのファイルを読み込み、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んでエンコードしますが、データとメタデータは別々のストリームに分割します、pxarフォーマットバージョンを使用しています。
メタデータ	ストリームを分割し、Dataのようなアーカイブ・フォーマット・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、以前のスナップショットのメタデータ・アーカイブ（存在する場合）を使用し、ファイルの内容をディスクから読み込むことなく、そのデータチャンクを再利用します。 可能です。

変更検出モードのメタデータを使用してバックアップを実行するには、以下を実行します。

```
vzdump 123 --storage pbs-storage --pbs-change-detection-mode='  
メタデータ'
```

#### 備考

VMのバックアップやProxmox Backup Server以外のストレージバックエンドへのバックアップは、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

vzdump の新しいバージョンでは、ゲストの種類とバックアップ時間がファイル名にエンコードされます。

```
vzdump-lxc-105-2009_10_09-11_04_43.tar
```

これにより、同じ複数のバックアップを保存することができます。さまざまな[保持](#)オプションを使用して、保持するバックアップの数を制限できます。

## 16.3 バックアップファイルの圧縮

バックアップファイルは次のいずれかのアルゴリズムで圧縮できます： lzo <sup>1</sup>、gzip <sup>2</sup>、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。マルチスレッドは lzo と gzip に対する zstd のもう一つの利点です。Lzo と gzip はより広く使われており、しばしばデフォルトでインストールされます。

pigz <sup>(4)</sup> を gzip のドロップイン代替としてインストールすることで、マルチスレッドによるより良いパフォーマンスが得られます。pigz と zstd では、スレッド数/コア数を調整できます。以下の[設定オプション](#)を参照してください。

バックアップファイル名の拡張子から、どの圧縮アルゴリズムでバックアップが作成されたかを判断できます。

<sup>1</sup> Lempel-Ziv-Oberhumer a lossless data compression algorithm <https://en.wikipedia.org/wiki/Lempel-Ziv-Oberhumer>

<sup>2</sup> gzip - DEFLATE アルゴリズムに基づく <https://en.wikipedia.org/wiki/Gzip>

<sup>3</sup> Zstandard 可逆データ圧縮アルゴリズム <https://en.wikipedia.org/wiki/Zstandard>

<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	エルゾ 圧縮

バックアップファイル名が上記のファイル拡張子のいずれかで終わっていない場合、vzdumpによって圧縮されません。

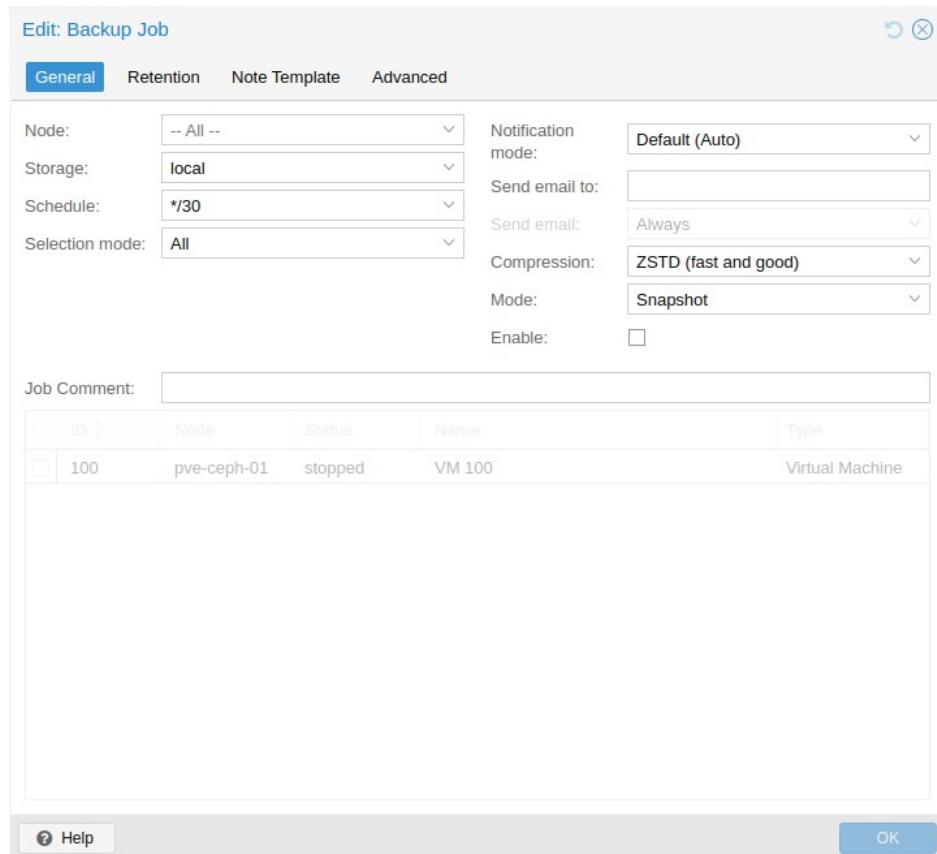
## 16.4 バックアップの暗号化

Proxmox Backup Serverストレージでは、オプションでバックアップのクライアント側暗号化を設定できます。

## 16.5 バックアップ

Enabled	Node	Schedule	Next Run	Storage	Comm...	Retention	Selection
-- All --	*30	2024-04-23 11:00:00	local	Fallback from storag...	-- All --		

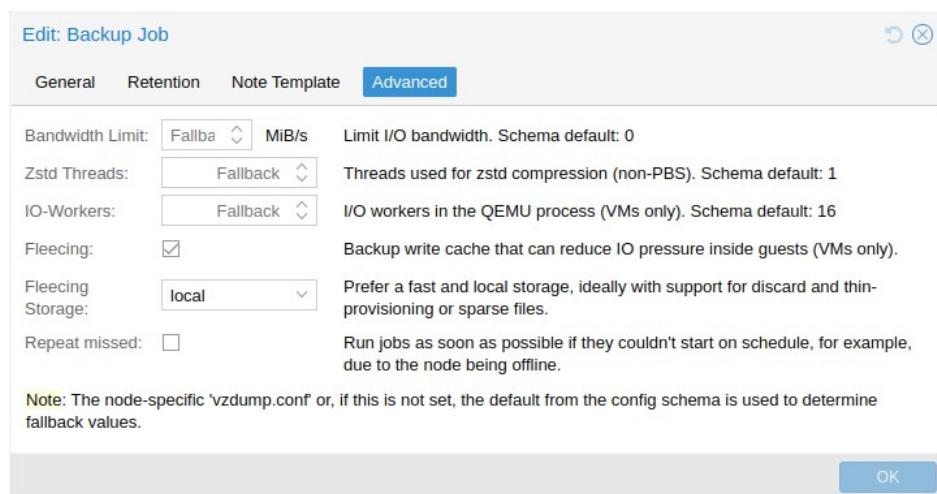
手動でバックアップをトリガーする以外に、すべての仮想ゲストまたは選択した仮想ゲストをストレージにバックアップする定期的なジョブを設定することもできます。 ジョブは UI の Datacenter→Backup または /cluster/backup API エンドポイントを使用します。どちらも /etc/pve/jobs.cfg にジョブエントリを生成し、pvescheduler デーモンによって解析、実行されます。



ジョブは全てのクラスタノードまたは特定のノードに対して設定され、指定されたスケジュールに従って実行されます。スケジュールのフォーマットはsystemd のカレンダーアイベントによく似ています。UI の **Schedule** フィールドは自由に編集することができ、ドロップダウンリストには開始点として使える例がいくつかあります。

ストレージやノードの設定より優先されるジョブ固有の保持オプションや、バックアップと一緒に保存される追加情報のメモの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であったり、スケジュールされた時間中にpveschedulerが無効になっていたりすると、その実行を逃してしまったため、それを取り戻すための動作を設定することができます。Repeat missedオプション (UIのAdvancedタブ、configのrepeat-missed) を有効にすると、見逃したジョブをできるだけ早く実行するようスケジューラに指示できます。

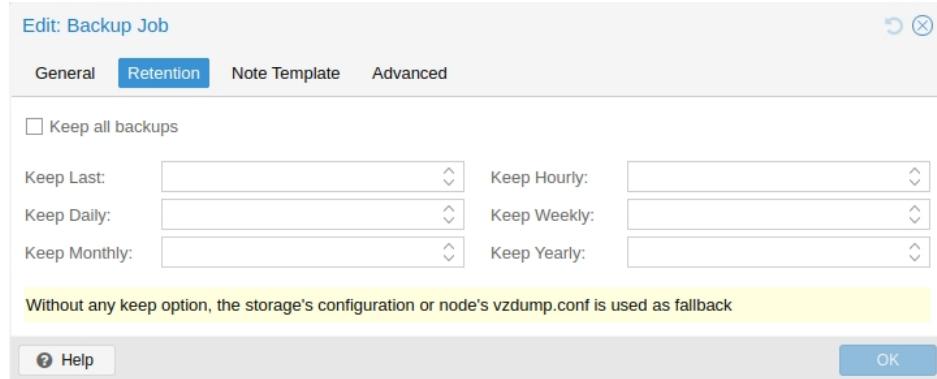


バックアップのパフォーマンスを調整するための設定がいくつかあります（そのうちのいくつかはUIのAdvancedタブで公開されています）。最も注目すべきは、IO帯域幅を制限するためのbwlimitです。compressorに使用されるスレッドの量は、それぞれpigz (gzipの代わり) 、zstd制御できます。さらに

はionice (BFQスケジューラ時)、パフォーマンス設定の一部としてmax-workers (VMバックアップにのみ影響)、pbs-entries-max (コンテナバックアップにのみ影響) です。詳細については、[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

prune-backupsオプションを使用すると、柔軟な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:

### keep-all <ブール値>

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。

### キープラスト <N>

最後の<N>個のバックアップを保持します。

### 時間毎 <N>

過去<N>バックアップを保持します。1時間に複数のバックアップがある場合は、最新のものだけが保持されます。

### キープデイリー <N>

過去<N>週間分のバックアップを保持します。1日に複数のバックアップがある場合は、最新のものだけが保持されます。

### キープウィークリー <N>

過去<N>週間分のバックアップを保持します。1週間に複数のバックアップがある場合は、最新のものだけが保持されます。

### 備考

週は月曜日に始まり、日曜日に終わります。このソフトウェアはISOの週日付システムを使用し、年末の週を正しく処理します。

### キープマンスリー<N>

過去<N>ヶ月分のバックアップを保持します。1つの月に複数のバックアップがある場合は、最新のものだけが保持されます。

## 年間 <N>

過去<N>バックアップを保持します。1年分のバックアップが複数ある場合は、最新のものだけが保存されます。

保持オプションは上記の順序で処理されます。各オプションは、その内のバックアップのみを対象とします。次のオプションでは、すでにカバーされているバックアップは処理されません。古いバックアップのみを考慮します。

使用したい保持オプションを、カンマ区切りのリストで指定:

```
# vzdump 777 --prune-バックアップ keep-last=3,keep-daily=13,keep-yearly=9.
```

prune-backupsを直接vzdumpに渡すこともできますが、ストレージレベルで設定した方が賢明な場合が多く、ウェブインターフェイスから行うことができます。

---

### 備考

古い maxfiles オプションは非推奨です。

maxfilesは、keep-allによって無制限に保持するために0でした。

---

## 16.6.1 プルーンシミュレーター

Proxmox Backup Serverドキュメントの剪定シミュレータを使用して、さまざまなバックアップスケジュールでさまざまな保持オプションの効果を調べることができます。

## 16.6.2 保持設定例

バックアップの頻度や古いバックアップの保持は、データの変更頻度や、特定の作業負荷における古い状態の重要性によって異なります。バックアップが企業の文書アーカイブとして機能する場合、バックアップの保存期間に関する法的要件がある場合もあります。

この例では、バックアップを毎日行い、保存期間を10年とし、保存されるバックアップの間隔は徐々に長くなっていくと仮定します。

keep-last=3 - たとえ毎日バックアップを取るとしても、管理者は大きなアップグレードの直前や直後に追加のバックアップを取りたいと思うかもしれません。keep-lastを設定することで、これが確実になります。

keep-hourly が設定されていません - 毎日のバックアップには関係ありません。keep-lastを使用することで、余分な手動バックアップをカバーすることができます。

keep-daily=13 - 少なくとも1日をカバーするkeep-lastと合わせて、少なくとも2週間分のバックアップを確保します。

keep-weekly=8 - 少なくとも2ヶ月分の週次バックアップを確保します。

keep-monthly=11 - 前回のkeep設定と合わせて、少なくとも1年間の月次バックアップを確保します。

keep-yearly=9 - これは長期アーカイブ用です。前のオプションで現在の年をカバーしたので、残りの年についてはこれを9に設定し、合計で少なくとも10年間をカバーするようにします。

不必要に高いことが判明した場合は、いつでも期間を短縮できますが、一度削除され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して削除できないようにすることができます。Proxmox VE の UI、CLI、または API を使用して保護されたバックアップを削除しようとする、失敗します。ただし、これはProxmox VEによって強制されるものであり、ファイルシステムによって強制されるものではないため、バックアップファイル自身を手動で削除することは、基礎となるバックアップストレージへの書き込みアクセス権を持っている人であれば誰でも可能です。

### 備考

保護されたバックアップはブルーニングによって無視され、保持設定にカウントされません。

ファイルシステムベースのストレージの場合、保護はセンチネルファイル<backup-name>.protectedを介して実装されます。Proxmox Backup Serverの場合は、サーバ側で処理されます（Proxmox Backup Serverバージョン2.1）。

ストレージオプションのmax-protected-backupsを使用して、ゲストごとにストレージで許可される保護バックアップの数を制御します。無制限には -1 を使用します。デフォルトは、Datastore.Allocate権限を持つユーザは無制限、その他のユーザは5です。

## 16.8 バックアップノート

UIのEdit NotesボタンまたはストレージコンテンツAPIを使用して、バックアップにノートを追加できます。



バックアップジョブや手動バックアップのノートを動的に生成するためのテンプレートを指定することもできます。テンプレート文字列には、2つの中括弧で囲まれた変数を含めることができます、バックアップの実行時に対応する値に置き換えられます。

現在サポートされているのは

- {{cluster}} もしあれば、クラスタ名。
- {{ゲスト名}} 仮想ゲストの割り当て名
- {{バックアップが作成されるノードのホスト名。}}
- {{vmid}} ゲストの数値 VMID です。

APIやCLIで指定する場合は、1行で指定し、改行とバックスラッシュはそれぞれリテラル //としてエスケープする必要があります。

## 16.9 復元

バックアップアーカイブは、Proxmox VEのWeb GUIまたは以下のCLIツールからリストアできます：

### pct リストア

コンテナ復元ユーティリティ

### qmrestore

仮想マシン復元ユーティリティ

詳細は各マニュアルページをご覧ください。

### 16.9.1 帯域幅の制限

1つまたは複数の大きなバックアップをリストアする場合、多くのリソース、特にバックアップストレージからの読み取りとターゲットストレージへの書き込みの両方にストレージ帯域幅が必要になることがあります。ストレージへのアクセスが混雑するため、他の仮想ゲストに悪影響を及ぼす可能性があります。

これを回避するには、バックアップジョブに帯域幅の制限を設定します。Proxmox VEはリストアとアーカイブの2種類の制限を実装しています：

- リストアごとの制限：バックアップアーカイブからの読み取りの最大帯域幅を示します。
- ストレージごとの書き込み制限：特定のストレージへの書き込みに使用される最大帯域幅を示します。

読み込み制限は書き込み制限に間接的に影響します。ジョブごとの上限が小さいと、ストレージごとの上限が大きくても上書きされます。より大きなジョブごとの制限は、影響を受けるストレージに 'Data.Allocate' パーミッションがある場合にのみ、ストレージごとの制限を上書きします。

リストアCLIコマンドで '--bwlimit <integer>' オプションを使用すると、リストアジョブ固有の帯域幅制限を設定できます。KiB/sが制限の単位として使用されます。つまり、'10240'を渡すと、バックアップの読み取り速度が10MiB/sに制限され、実行中の仮想ゲストが残りのストレージ帯域幅を使用できるようになります。

#### 備考

`bwlimit` パラメータに '0' を使用すると、特定のリストアジョブのすべての制限を無効にできます。これは非常に重要な仮想ゲストを可能な限り速くリストアする必要がある場合に役立ちます。(ストレージの 'Data.Allocate' パーミッションが必要です)

ほとんどの場合、ストレージの一般的に利用可能な帯域幅は時間の経過とともに変化しません。そのため、設定済みのストレージごとにデフォルトの帯域幅制限を設定で

```
# pvesm set STORAGEID --bwlimit restore=KIBs  
きるようになりました:
```

## 16.9.2 ライブ復元

大きなバックアップのリストアには長い時間がかかり、その間にゲストが利用できなくなることがあります。Proxmox Backup Server上に保存されたVMバックアップの場合、ライブリストアオプションを使用することで、この待ち時間を短縮できます。

GUIのチェックボックスまたはqmrestoreの-live-restore引数のいずれかを使用してライブ・リストアを有効にすると、リストアが開始されると同時に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が優先されます。

なお、これには2つの注意点があります：

- ライブ・リストア中は、データをバックアップ・サーバーからロードする必要があるため、VMは制限されたディスク・リード速度で動作します（ただし、一度ロードされると、宛先ストレージすぐに利用できるため、データに2回アクセスしてもペナルティが発生するのは最初の1回だけです）。書き込み速度はほとんど影響を受けません。
- つまり、すべてのデータがバックアップからコピーされていない可能性があり、失敗したリストア操作中に書き込まれたデータを保持できない可能性が高いのです。

この動作モードは、ウェブ・サーバーなど、初期動作に必要なデータ量が少ない大規模なVMに特に有効です。OSと必要なサービスが開始されると、VMは稼働し、バックグラウンド・タスクはあまり使用されないデータのコピーを続けます。

## 16.9.3 単一ファイルの復元

ストレージGUIの【バックアップ】タブにある【ファイル復元】ボタンを使用すると、バックアップに含まれるデータ上でファイルブラウザを直接開くことができます。この機能はProxmox Backup Server上のバックアップでのみ使用できます。

コンテナの場合、ファイルツリーの最初のレイヤには、含まれるすべてのpxarアーカイブが表示され、自由に開いたり参照したりできます。VMの場合、最初のレイヤには含まれるドライブイメージが表示され、これを開くと、ドライブでサポートされているストレージ技術のリストが表示されます。最も基本的なケースでは、これはパーティションテーブルを表すpartと呼ばれるエントリで、ドライブにある各パーティションのエントリが含まれています。VMの場合、すべてのデータにアクセスできるわけではないことに注意してください（サポートされていないゲストファイルシステム、ストレージテクノロジーなど）。

ファイルやディレクトリはダウンロードボタンを使ってダウンロードすることができ、後者はその場で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に、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、ハイパーテイラー・システムが危険にさらされるのを避けることができます。VMはタイムアウト後に停止します。このプロセス全体はユーザーから見て透過的に行われます。

---

### 備考

についてトラブルシューティング を目的としています、各 一時的な VM インスタンス は a ログ ファイルを生成します。

/var/log/proxmox-backup/file-restore/。ログファイルには、バックアップアーカイブに含まれる個々のファイルのリストアやファイルシステムへのアクセスが失敗した場合の追加情報が含まれている可能性があります。

---

## 16.10 構成

グローバル設定は/etc/vzdump.confに保存されます。このファイルでは、コロンで区切られた単純なキー/値形式を使用します。各行の書式は次のとおりです：

### オプション： 値

ファイル内の空白行は無視され、#文字で始まる行はコメントとして扱われ、これも無視されます。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することができます。

現在、以下のサポートしています：

**bwlimit: <整数> (0 - N) (デフォルト = 0)**

I/O/バンド幅の制限（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: <文字列>**

結果のファイルを指定したディレクトリに保存します。

**除外パス: <配列>**

特定のファイル/ディレクトリ（シェル・グロブ）を除外します。で始まるパスはコンテナのルートに固定され、他のパスは各サブディレクトリに相対的に一致します。

**fleecing: [[enabled=<1|0>] [[enabled=<1|0>]]** です。 [,ストレージ=<ストレージID>]]。

バックアップフリーのオプション（VMのみ）。

**enabled= <boolean> (デフォルト= 0)**

バックアップフリーを有効にします。新しいゲストの書き込みがバックアップターゲットに直接コピーする代わりに、指定されたストレージで発生するブロックからバックアップデータをキャッシュします。これにより、ゲストの IO パフォーマンスを向上させ、ハングを防止することができます。

**ストレージ= <ストレージID>**

このストレージは、ストレージフリーキングイメージに使用します。スペースを効率的に使用するには、廃棄とシンプロビジョニングまたはスペースファイルをサポートするローカルストレージを使用するのが最善です。

**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とサスPENDモードのバックアップでは、これはコンプレッサにのみ影響します。値が8の場合はアイドル優先度が使用され、それ以外の場合は指定した値でベストエフォート優先度が使用されます。

**lockwait: <整数> (0 - N) (デフォルト = 180)**

グローバルロックの最大待機時間（分）。

**mailnotification: <常に | 失敗> (デフォルト= 常に)**

非推奨: 代わりに通知ターゲット/マッチャーを使用してください。通知メールを送るタイミングを指定

**mailto: <文字列>**

非推奨: 代わりに通知先/通知先を使用してください。メール通知を受け取るメールアドレスまたはユーザのカンマ区切りリスト。

**maxfiles: <整数> (1 - N)**

非推奨: 代わりに *prune-backups* を使用してください。ゲストシステムごとのバックアップファイルの最大数。

**mode: <snapshot| stop| suspend> (default= snapshot)**

バックアップモード。

**notes-template: <文字列>**

バックアップのメモを生成するためのテンプレート文字列。値で置き換える変数を含むことができます。現在サポートされているのは{{cluster}}, {{guestname}}, {{node}}, {{vmid}}ですが、今後追加されるかもしれません。改行とバックスラッシュは、それぞれ「\n」と「\\n」としてエスケープする必要があります。

---

**備考**

必要なオプション: ストレージ

---

**notification-mode: <auto| legacy-sendmail| notification-system> です。****(デフォルト= 自動)**

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とmailnotificationは無視されます。*auto* (デフォルト設定) に設定すると、mailto が設定されていればメールが送信され、設定されていなければ通知システムが使用されます。

**notification-policy: <always| failure| never> (デフォルト= always)**

非推奨: を使用しないでください。

**通知先: <文字列>**

非推奨: を使用しないでください。

**pbs-change-detection-mode: <data| legacy| metadata> です。**

ファイルの変更を検出し、コンテナバックアップのエンコード形式を切り替えるために使用されるPBSモード。

**パフォーマンスを向上させます: [max-workers=<integer>] [,pbs-entries-max=<integer>]。**

他のパフォーマンス関連の設定。

**max-workers= <integer> (1 - 256) (デフォルト= 16)**

VMに適用されます。最大この数のIOワーカーを許可します。

**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 状況を回避するために、指定された時間にメモリ内で許可されるエントリ数を制限します。これを増やすと大量のファイルを含むコンテナのバックアップが可能になります。

**pigz: <整数> (デフォルト= 0)**

N>0の場合はgzipの代わりにpigzを使用。N=1はコアの半分を使用し、N>1はNをスレッド数として使用します。

**プール: <文字列>**

指定されたプールに含まれるすべての既知のゲストシステムをバックアップします。

**protected: <ブール値>**

trueの場合、バックアップに保護マークを付けます。

---

**備考**

必要なオプション: ストレージ

---

を削除します: [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)

ストレージ構成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。

**keep-all= <boolean>**

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= <N>**

過去<N>日分のバックアップを保持します。1日に複数のバックアップがある場合は、最新のものだけが保持されます。

**keep-hourly= <N>**

最後の<N>異なる時間のバックアップを保持します。1時間に複数のバックアップがある場合、最新のものだけが保持されます。

**keep-last= <N>**

最後の<N>個のバックアップを保持します。

**keep-monthly= <N>**

過去<N>の異なる月のバックアップを保持します。1つの月に複数のバックアップがある場合、最新のものだけが保持されます。

**keep-weekly= <N>**

過去<N>週間分のバックアップを保持します。1週間に複数のバックアップがある場合、最新のものだけが保持されます。

**keep-yearly= <N>**

過去<N>年分のバックアップを保持します。一つの年に複数のバックアップがある場合、最新のものだけが保持されます。

**remove: <論理値> (デフォルト = 1)**

prune-backupsに従って古いバックアップを削除します。

**スクリプト: <文字列>**

指定されたフックスクリプトを使用します。

**stdexcludes: <boolean> (デフォルト= 1)**

一時ファイルとログを除外します。

**stopwait: <整数> (0 - N) (デフォルト= 10)**

ゲストシステムが停止するまでの最大待機時間 (分)。

**ストレージ: <ストレージID**

結果のファイルをこのストレージに保存します。

**tmpdir: <文字列**

指定したディレクトリに一時ファイルを保存します。

**zstd: <整数> (デフォルト= 1)**

Zstdスレッド。N=0は利用可能なコアの半分を使用し、Nが0より大きな値に設定された場合、Nはスレット数として使用されます。

**vzdump.confの設定例**

```
tmpdir/mnt/fast_local_disk storage:  
my_backup_storage mode: snapshot  
ビューリミット: 10000
```

## 16.11 フックスクリプト

オプション --script でフックスクリプトを指定できます。このスクリプトはバックアップ処理の様々な段階で呼び出され、パラメータが適宜設定されます。ドキュメンテーションディレクトリに例があります (vzdump-hook-script.pl)。

## 16.12 ファイルの除外

---

**備考**

このオプションはコンテナ・バックアップでのみ使用できます。

---

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オプションで無効にできます)。

```
/tmp/*  
/var/tmp/*  
/var/run/*pid
```

手動で（追加の）除外指定することもできます:

```
# vzdump 777 --exclude-path /tmp/ --exclude-path '/var/foo*' 
```

は、/tmp/ディレクトリと、/var/foo、/var/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)へのバックアップとサスPENDモードバックアップでは、末尾のスラッシュが付いたパターンはディレクトリにマッチしますが、ファイルにはマッチしません。一方、PBS以外のスナップショット・モードとサスPEND・モードのバックアップでは、tarコマンドでサポートされていないため、末尾のスラッシュを含むパターンは現在まったく一致しません。

で始まらないパスは、コンテナのルートにアンカーされません。たとえば

```
# vzdum 777 --exclude-path bar
```

は、/bar、/var/bar、/var/foo/bar ファイルやディレクトリを除外しますが、/bar2 は除外しません。

設定ファイルもバックアップアーカイブ内（./etc/vzdum/内）に保存され、正しくリストアされます。

## 16.13 例

単にゲスト777をダンプします - スナップショットではなく、デフォルトのダンプディレクトリ(通常は/var/lib/vz/dump/)にゲストのプライベート領域と設定ファイルをアーカイブするだけです。

```
# vzdum 777
```

rsyncとサスPEND/レジュームを使ってスナップショットを作成します（ダウンタイムは最小限）。

```
# 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します。設定され、notification-modeがデフォルトでautoに設定されているため、通知メールは通知システムの代わりにシステムのsendmailコマンドで送信されます。

```
# 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します。

```
# vzdum 777 --dumpdir /mnt/backup --mode snapshot
```

複数のゲストのバックアップ（選択的）

```
# vzdum 101 102 103 --mailto root
```

101と102を除く全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にリストア

```
# pct restore 600 /mnt/backup/vzdum-lxc-777.tar
```

VM 601へのQemuServer VMのリストア

```
# qmrestore /mnt/backup/vzdum-qemu-888.vma 601
```

パイプを使用して、既存のコンテナ101を、4GBのルートファイルシステムを持つ新しいコンテナ300にクローンします。

```
# vzdum 101 --stdout| pct restore --rootfs 4 300 -.
```

## 第17章 通知

### 17.1 概要

- Proxmox VEは、ストレージのレプリケーション障害、ノードのフェンシング、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通知システムによって処理されます。通知イベントには、タイムスタンプ、重大度レベル、タイプ、その他のオプションのメタデータフィールドなどのメタデータがあります。
- [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](#)、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。ターゲットには複数のタイプがあり、メールベース（SendmailとSMTP）とGotifyがあります。

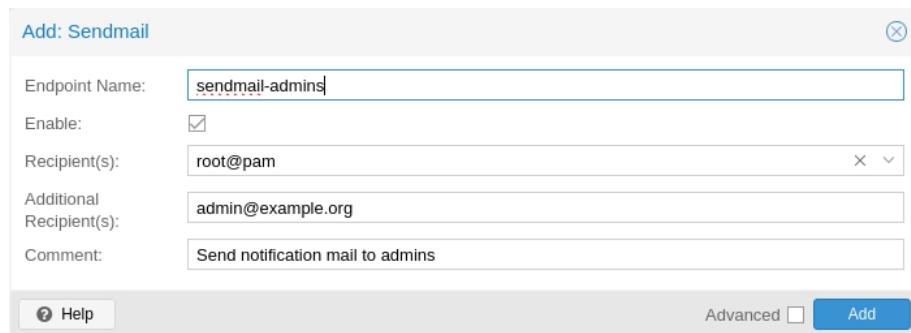
バックアップジョブには、設定可能な[通知モード](#)があります。このモードでは、通知システムと、通知メールを送信するレガシーモードのどちらかを選択できます。レガシーモードはProxmox VE 8.1以前の通知方法と同じです。

通知システムは、GUI の [Datacenter→ Notifications] で構成できます。この構成は、/etc/pve/notifications.cfg および /etc/pve/priv/notifications.cfg に保存されます。/etc/pve/priv/notifications.cfg には、通知ターゲットのパスワードや認証トークンなどの機密構成オプションが含まれており、root によってのみ読み取ることができます。

### 17.2 通知対象

Proxmox VEは複数のタイプの通知提供します。

## 17.2.1 センドメール



sendmail バイナリは、電子メールメッセージの送信を処理する Unix 系オペレーティングシステムでよく見られるプログラムです。コマンドラインユーティリティであり、ユーザやアプリケーションがコマンドラインやスクリプト内から直接電子メールを送信することができます。

sendmail 通知ターゲットは、sendmail バイナリを使用して、設定されたユーザーまたはメールアドレスのリストに電子メールを送信します。ユーザが受信者として選択された場合、ユーザの設定で構成された電子メールアドレスが使用されます。root@pamユーザの場合、これはインストール時に入力された電子メールアドレスです。ユーザの電子メール アドレスは、Datacenter→ Permissions→ Users で設定できます。ユーザーに関連する電子メール・アドレスがない場合、電子メールは送信されません。

### 備考

Proxmox VEの標準インストールでは、sendmailバイナリはPostfixによって提供されます。外部メールリレー(スマートホスト)を設定するなどして、Postfixがメールを正しく配達できるように設定する必要があるかもしれません。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`: 通知を送信する電子メールアドレス。複数の受信者に対応するために複数回設定することができます。
- `mailto-user`: メールを送信ユーザー。ユーザーのメールアドレスは `users.cfg`。複数の受信者に対応するために複数回設定することができます。
- `author`: メールの作成者を設定します。デフォルトはProxmox VEです。
- `from-address`: メールの差出人アドレスを設定します。このパラメータが設定されていない場合、プラグインは`datacenter.cfg`の`email_from`設定にフォールバックします。このパラメータも設定されていない場合、プラグインのデフォルトは`root@$hostname` (`$hostname`はノードのホスト名) です。
- コメントこのターゲットのコメント メールのFromヘッダは`$author <$from-address`に設定されます。

設定例 (/etc/pve/notifications.cfg):

```
sendmail: 例
  mailto-user root@pam mailto-user
  admin@pve max@example.com
  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電子メールを送信できます。このターゲットは、電子メールの配信にシステムのMTAを使用しません。sendmail ターゲットと同様に、ユーザが受信者として選択されている場合、ユーザの構成済み電子メールアドレスが使用されます。

### 備考

sendmail ターゲットとは異なり、SMTP ターゲットにはメール配送に失敗した場合のキューイング/リトライ機構はありません。

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:** 通知を送信する電子メールアドレス。複数の受信者に対応するために複数回設定することができます。
- **mailto-user:** メールを送信ユーザー。ユーザーのメールアドレスは `users.cfg`。複数の受信者に対応するために複数回設定することができます。
- **author:** メールの作成者を設定します。デフォルトはProxmox VEです。
- **From-address:** メールのFromアドレスを設定します。SMTPリレーでは、なりすましを避けるためにこのアドレスがユーザのものであることを要求することがあります。メールのFromヘッダは`$author`に設定されます。`<$from-address>`。
- **username:** 認証時に使用するユーザー名。ユーザ名が設定されていない場合は、認証は行われません。PLAIN および LOGIN 認証方式をサポートしています。
- **password:** 認証時に使用するパスワード。
- **モード:** 暗号化モード (`insecure`、`starttls`、`tls`) を設定します。デフォルトは`tls`です。
- **サーバーを指定します:** SMTPリレーのアドレス/IP
- **ポート:** 接続ポート。設定されていない場合、使用されるポートのデフォルトは、mode の値に応じて 25 (安全でない)、465 (`tls`)、または 587 (`starttls`) になります。
- **コメント:** このターゲットに対するコメント

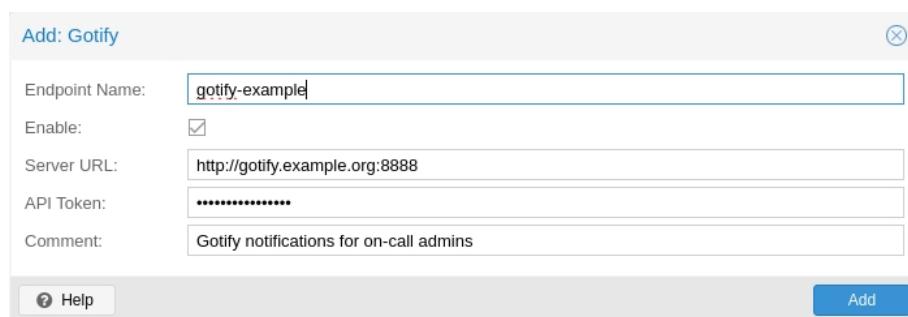
設定例 (/etc/pve/notifications.cfg):

```
smtp: 例
      mailto-user root@pam mailto-user
      admin@pve max@example.com
      from-addresspvel@example.com username pvel
      サーバー mail.example.com モード
      starttls
```

シークレットトークンを含む/etc/pve/priv/notifications.cfgの一致するエントリ:

```
smtp: 例
      パスワード
```

### 17.2.3 ゴティファイ



Gotifyは、様々なデバイスやアプリケーションにプッシュ通知を送受信できるオープンソースのセルフホスト型通知サーバーです。シンプルなAPIとウェブインターフェースを提供し、様々なプラットフォームやサービスと簡単に統合することができます。

Gotifyターゲットプラグインの設定には以下のオプションがあります:

- サーバーを指定します: GotifyサーバーのベースURL、例: `http://<ip>:8888`
- トークン: 認証トークン。トークンはGotifyのWebインターフェイス内で生成することができます。
- コメントこのターゲットに対するコメント

#### 備考

Gotify ターゲットプラグインは、[データセンター設定からの HTTP プロキシ設定](#)を尊重します。

設定例 (/etc/pve/notifications.cfg):

```
例
      サーバー http://gotify.example.com:8888 コメント 複数のユーザー/
      アドレスに送信
```

シークレットトークンを含む/etc/pve/priv/notifications.cfgの一致するエントリ:

```
例
      トークン
```

## 17.2.4 ウェブフック

Webhook 通知ターゲットは、構成可能な URL への HTTP リクエストを実行します。以下の設定オプションがあります：

- `url`: HTTP リクエストを実行する URL。メッセージの内容、メタデータ、秘密を注入するためのテンプレート化をサポートします。
- メソッドを使用します： 使用する HTTP メソッド (POST/PUT/GET)
- ヘッダの配列： リクエストに設定されるべき HTTP ヘッダの配列。メッセージの内容、メタデータ、秘密を注入するためのテンプレート化をサポートします。
- ボディ： 送信されるべきHTTPボディ。メッセージの内容、メタデータ、秘密を注入するためのテンプレート化をサポートします。
- `secret`: 秘密のキーと値のペアの配列。これらはrootだけが読める保護された設定ファイルに保存されます。秘密は`secrets`名前空間を通して body/header/URLテンプレートでアクセスできます。
- コメント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構成オプションでは、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：

- `{タイトル}`：レンダリングされた通知のタイトル
- `{メッセージ}`：レンダリングされた通知本文
- `{{severity}}`：通知の重大度 (情報、通知、警告、エラー、不明)
- `{{timestamp}}`：通知のタイムスタンプをUNIXエポック（秒単位）で指定します。
- `{{fields.<name>}}`：メタデータフィールドのサブネームスペース。例えば `fields.type` は通知タイプを含んでいます - すべての利用可能な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>}}`：`secrets`のサブ名前空間。例えば、`token` という名前の`secrets.token` からアクセスできます。

便宜上、以下のヘルパーをいます：

- `{{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送信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、JSONペイロードの一部としてサブ名前空間全体(`fields`など)を渡すのに便利です(例 `{{json fields}}`)。

## 例

### ntfy.sh

- メソッドPOST
- URL: `https://ntfy.sh/{{ secrets.channel }}`
- ヘッダー
  - マークダウンはい
- ボディ

```
...  
{メッセージ }  
...
```

- 秘密:

- チャネル:<あなたの ntfy.sh チャネル>。

## 不和

- メソッドPOST
- URL: `https://discord.com/api/webhooks/{{ secrets.token }}`
- ヘッダー
  - コンテンツタイプ: `application/json`
- ボディ

```
{  
  "content": " {{ エスケープメッセージ }} "  
}
```

- 秘密:

- トークン:<トークン

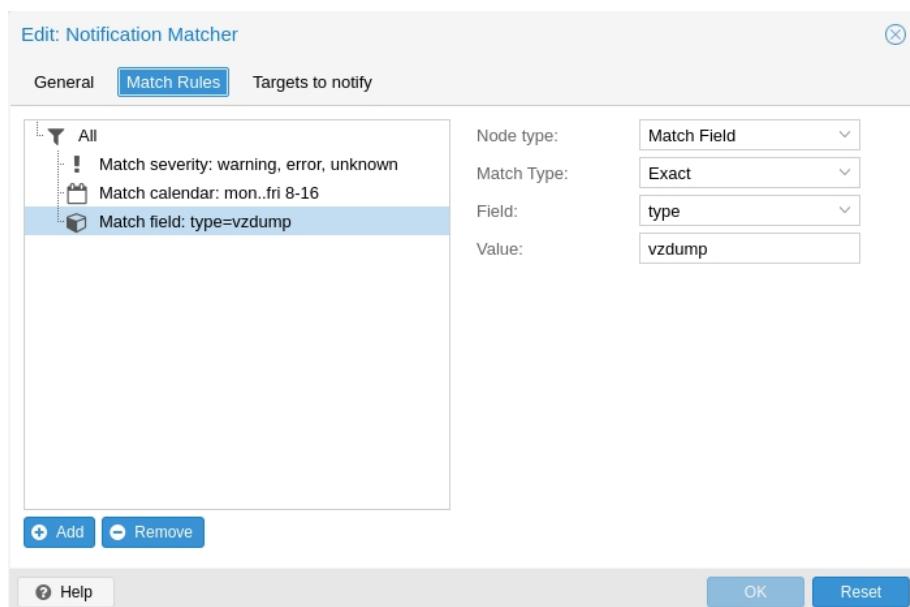
## スラック

- メソッドPOST
- URL: `https://hooks.slack.com/services/{{ secrets.token }}`
- ヘッダー
  - コンテンツタイプ: `application/json`
- ボディ

```
{
  "text": "{{エスケープメッセージ}}",
  "type": "mrkdwn"
}
```

- 秘密:
  - トークン: <トークン>

## 17.3 通知マッチャー



通知マッチャーは、マッチングルールに基づいて通知を通知ターゲットにルーティングします。これらのルールは、タイムスタンプ (match-calendar)、通知の重大度 (match-severity)、メタデータ・フィールド (match-field) など、通知の特定のプロパティに一致させることができます。通知がマッチャーによってマッチされると、マッチャー用に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することができ、それぞれが独自のマッチングルールと通知するターゲットを持ちます。ターゲットが複数のマッチャーで使用されている場合でも、各ターゲットは通知の度に最大一度だけ通知されます。

マッチング・ルールのないマッチャーは常にtrueになり、設定されたターゲットは常に通知されます。

```
matcher: 常にマッチ
        ターゲット管理者
        コメント このマッチャーは常に
```

### 17.3.1 マッチャーのオプション

- ターゲット: マッチャーがマッチした場合、どのターゲットに通知するかを決定します。複数のターゲットに通知するためには複数使用することができます。
- invert-match: マッチャ全体の結果を反転します。
- モード: マッチャー全体の結果を計算するために、個々のマッチ・ルールがどのように評価されるかを決定します。`all` に設定すると、すべての一一致ルールが一致しなければなりません。`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マッチさせます。
- match-field: 通知のメタデータフィールドにマッチします。
- match-severity: 通知の重大度にマッチ
- コメントこのマッチャーのコメント

### 17.3.2 カレンダーのマッチングルール

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。

- マッチカレンダー 8-12
- 試合カレンダー 8:00-15:30
- 試合カレンダー 月～金 9:00～17:00
- 試合カレンダー 日・火・水・金 9-17

### 17.3.3 フィールド・マッチング・ルール

通知には、マッチング可能なメタデータ・フィールドがあります。マッチング・モードとして`exact`を使用する場合、`,` を区切り文字として使用できます。マッチング・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持つ場合にマッチングします。

- `match-field exact:type=vzdump` バックアップに関する通知のみにマッチします。
- `match-field exact:type=replication,fencing` レプリケーションとフェンシングの無検証を一致させます。
- ノードのホスト名にマッチします。

マッチしたメタデータフィールドが存在しない場合、その通知はマッチしません。例えば、`match-field regex:hostname=.*` ディレクティブは任意のホスト名のメタデータフィールドを持つ通知だけにマッチしますが、フィールドが存在しない場合はマッチしません。

### 17.3.4 深刻度マッチングルール

通知には、一致させることができる関連する重大度があります。

- match-severity エラー：マッチエラーのみ
- match-severity 警告, エラー：マッチの警告とエラー

次の深刻度が使用されています: info、notice、warning、error、unknown。

### 17.3.5 例

マッチャー: 平日

マッチカレンダー 月～金 9-17 目標 管理者  
コメント 就業時間内に管理者に通知

マッチャー: 夜と週末

マッチカレンダー 月-金 9-17 反転マッチ true  
オンコール管理者対象  
コメント 非労働時間については別途目標

```
matcher: バックアップの失敗
    match-field exact:type=vzdump match-
    severity エラー
    ターゲットバックアップアドミン
    コメント バックアップ失敗の通知を← の1つのグループに送信 '
        管理者
```

matcher: クラスタ失敗

match-field exact:type=replication, フェンシング対象クラスタ管理者  
コメント クラスタ関連の通知を他の管理者グループに送信します。

## 17.4 通知イベント

イベント	タイプ	重大性	メタデータフィールド (タイプの追加)
システムアップデート 利用可能	パッケージ更新	インフォメーション	ホスト名
クラスターノードのフェンス	フェンシング	エラー	ホスト名
ストレージ・レプリケーション・ジョブ 失敗	レプリケーション	エラー	ホスト名, job-id
バックアップ成功	ブイエスタンプ	インフォメーション	ホスト名, job-id (バックアップジョブのみ)

イベント	タイプ	重大性	メタデータフィールド (タイプの追加)
バックアップ失敗	ブイエスダンプ	エラー	ホスト名, job-id (バックアップジョブのみ)
ルートへのメール	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	ドメイン名を除いたホスト名 (例: pve1)
ジョブID	ジョブID

#### 備考

バックアップジョブの通知には、バックアップジョブがスケジュールに基づいて自動的に実行された場合のみjob-idが設定されます。

## 17.5 システムメール転送

smartdなどの特定のローカル・システム・デーモンは、最初にローカル・ルート・向けられた通知メールを生成します。Proxmox VEはこれらのメールを通じシステムに、タイプsystem-mail、深刻度unknownの通知として送ります。

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。それ以外のターゲットに対しては、システムはメールのコンテンツから件名と本文の両方を抽出しようとします。メールがHTMLコンテンツのみで構成されている場合、このプロセスでプレーンテキスト形式に変換されます。

## 17.6 アクセス許可

通知ターゲットの構成を変更/表示するには、/mapping/notifications ACL ノードの Mapping.Modify/Mapping.Audit パーミッションが必要です。

ターゲットのテストには、/mapping の 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の設定には、notification-modeオプションがあり、3つの値のいずれかを指定できます。

- auto: mailto/Send email to 電子メール・アドレスが入力されていない場合は、legacy-sendmail モードを使用します。メールアドレスが入力されていない場合、notification-system モードが使用されます。
- legacy-sendmail: システムの sendmail コマンドで通知メールを送信します。通知システムはバイパスされ、設定されたターゲット/マッチャーは無視されます。このモードはProxmox VE 8.1より前のバージョンの通知動作と同じです。
- 通知システム: 新しいフレキシブルな通知システムをご利用ください。

notification-modeオプションが設定されていない場合、Proxmox VEのデフォルトはautoになります。この

legacy-sendmailモードは、Proxmox VEの後のリリースで削除される可能性があります。

## 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、Proxmox VE API全体を127.0.0.1:85に公開します。このデーモンはrootとして実行され、すべての特権操作を実行する権限を持っています。

### 備考

デーモンはローカルアドレスのみをリッスンしますので、アクセスすることはできません。 pveproxy  
デーモンはAPIを公開します。

## 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でProxmox VE API全体を公開します。ユーザーwww-dataとして実行されます。  
で、非常に限定されたパーミッションを持ちます。より多くのパーミッションを必要とする操作は、ローカルのpvedaemonに転送されます。  
他のノードをターゲットにしたリクエストは、自動的にそれらのノードに転送されます。つまり、単一のProxmox VEノードに接続してクラスタ全体を管理できます。

### 18.2.1 ホストベースのアクセス制御

apache2" ライクなアクセス制御リストを設定することができます。値は /etc/default/pveproxy ファイルから読み込まれます。

例えば

```
ALLOW_FROM="10.0.0.1-10.0.0.5,192.168.0.0/22" DENY_FROM="all"  
POLICY="allow"
```

IPアドレスは、Net::IPが理解できる構文を使用して指定できます。all という名前は  
0/0および::/0（すべてのIPv4およびIPv6アドレスを意味します）。デフォルトのポ  
リシーは許可です。

試合	POLICY=拒否	POLICY=許可
マッチ許可のみ	認める	認める
マッチ拒否のみ	争う	争う
該当なし	争う	認める
許可と拒否の両方に対応	争う	認める

## 18.2.2 リスニングIPアドレス

デフォルトでは、pveproxyとspiceproxyデーモンはワイルドカード・アドレスをリッスンし、IPv4とIPv6の両方のクライアントからの接続を受け入れます。

etc/default/pveproxy で LISTEN\_IP を設定することで、pveproxy がどの IP アドレスに接続されるかを制御できます。

と spiceproxy デーモンがバインドします。IPアドレスはシステム上で設定する必要があります。

sysctl net.ipv6.bindv6onlyをデフォルトでない1に設定すると、デーモンが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が、通常は他の多くの問題も発生します。この設定を行った場合、sysctl設定を削除、LISTEN\_IPを0.0.0.0 (IPv4クライアントのみを許可する) に設定するお勧めします。

LISTEN\_IPは、ソケットを内部インターフェイスに制限するためだけに使うことができます：

```
LISTEN_IP="192.0.2.1"
```

同様に、IPv6アドレスを設定することもできます：

```
LISTEN_IP="2001:db8:85a3::1"
```

リンクローカルIPv6アドレスを指定したい場合はインターフェース名そのものを指定する必要があることに注意してください。例えば

```
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 にアクセスする必要があります。クラスタ化されたシステムで LISTEN\_IP を設定することはお勧めしません。

変更を適用するには、ノードを再起動するか、pveproxyとspiceproxyを完全に再起動する必要があります。

サービスです：

```
systemctl restart pveproxy.service spiceproxy.service
```

---

### 備考

reload とは異なり、pveproxy サービスの、仮想ゲストからの実行中のコンソールやシェルなど、いくつかの長時間実行中のワーカープロセスを中断する可能性があります。そのため、この変更を有効にするにはメンテナンスウィンドウを使用してください。

## 18.2.3 SSL暗号スイート

/etc/default/pveproxy で CIPHERS (TLS≥ 1.2) と CIPHERSUITE を使って暗号化リストを定義できます。

(TLS≥ 1.3) キー。例えば

```
CIPHERS="ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:← , ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-← , ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-← , AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES128-SHA256:← , ECDHE-RSA-AES128-SHA256"
```

```
CIPHERSUITES="TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:← , TLS_AES_128_GCM_SHA256"
```

上記はデフォルトです。利用可能なオプションの一覧は、openssl パッケージの ciphers(1) man ページを参照してください。

さらに /etc/default/pveproxy で使用される暗号をクライアントが選択するように設定できます (デフォルトは、クライアントと pveproxy の両方で使用可能なリストの最初の暗号です):

```
honor_cipher_order=0
```

## 18.2.4 対応TLSバージョン

安全でない SSL バージョン 2 と 3 は pveproxy では無条件に無効化されます。最近の OpenSSL バージョンでは、1.1 未満の TLS バージョンはデフォルトで無効化され、pveproxy はこれに従います (/etc/ssl/openssl.cnf を参照)。

TLS バージョン 1.2 または 1.3 を無効にするには、/etc/default/pveproxy で以下を設定します:

```
disable_tls_1_2=1
```

または、それぞれ

```
disable_tls_1_3=1
```

---

### 備考

特別な理由がない限り、サポートされる TLS バージョンを手動で調整することは推奨されません。

---

## 18.2.5 ディフィー・ヘルマン・パラメーター

使用する Diffie-Hellman パラメータは、/etc/default/pveproxy で DHPARAMS を設定することで定義できます。  
を PEMDH パラメータを含むファイルのパスに変換します。

```
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。

---

### 備考

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

使用する証明書は、外部証明書または ACME 経由で取得した証明書に変更できます。

pveproxy は /etc/pve/local/pveproxy-ssl.pem と /etc/pve/local/pveproxy-ssl.key を使用します。

が存在する場合、/etc/pve/local/pve-ssl.pem および /etc/pve/local/pve-ssl.key にフォールバックします。秘密鍵はパスフレーズを使用することはできません。

証明書の秘密鍵/etc/pve/local/pveproxy-ssl.keyの場所を上書きすることができます。

例えば、/etc/default/pveproxy で TLS\_KEY\_FILE を設定します：

```
TLS_KEY_FILE="/secrets/pveproxy.key"
```

---

### 備考

付属のACMEインテグレーションはこの設定に対応していません。

---

詳細については、マニュアルの「ホストシステム管理」の章を参照してください。

## 18.2.7 レスポンス・コンプレッション

デフォルトでは、pveproxy は圧縮可能なコンテンツに対して gzip HTTP レベル圧縮を使用します。これは /etc/default/pveproxy で無効にできます。

```
COMPRESSION=0
```

## 18.2.8 リアルクライアントIPログ

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。プロキシサーバが pveproxy の前にある場合プロキシ IP ではなく、リクエストを送ったクライアントの IP をログに記録することが望ましいかもしれません。

プロキシが設定したHTTPヘッダーをロギング処理できるようにするには、PROXY\_REAL\_IP\_HEADERを設定します。

をクライアントIPを取得ヘッダの名前に置き換えます。例えば

```
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。

デフォルトの動作は、すべての受信リクエストでこのヘッダーの値をログに記録します。上記の HTTP ヘッダを設定するために信頼されるべきプロキシサーバのリストを定義するには、例えば PROXY\_REAL\_IP\_ALLOW\_FROM を設定します：

```
PROXY_REAL_IP_ALLOW_FROM="192.168.0.2"
```

PROXY\_REAL\_IP\_ALLOW\_FROM設定も、ALLOW\_FROM設定と同様の値をサポートしています。

DENY\_FROM の設定。

IPアドレスは、Net::IPが理解できる構文を使用して指定できます。all という名前は 0/0と::/0すべてのIPv4とIPv6アドレスを意味します)。

---

## 18.3 pvestatd - Proxmox VE ステータスデーモン

このデーモンは、VM、ストレージ、コンテナのステータスを定期的にクエリします。結果はクラスタ内のすべての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、リモートディスプレイやデバイス（キーボード、マウス、オーディオなど）へのクライアントアクセスを提供します。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 をリッスンし、SPICE クライアントからの CONNECT 要求を正しい Proxmox VE VM に転送する HTTP プロキシを実装しています。ユーザ www-data として実行され、非常に限定された権限を持っています。

### 18.4.1 ホストベースのアクセス制御

アクセス制御リストのように "apache2" を設定することができます。値は /etc/default/pveproxy ファイルから読み込まれます。  
詳細は pveproxy のドキュメントを参照してください。

## 18.5 pvescheduler - Proxmox VE スケジューラデーモン

このデーモンは、レプリケーション ジョブや vzdump ジョブなど、スケジュールに従ってジョブを開始します。vzdumpジョブについては、  
/etc/pve/jobs.cfg ファイルから設定を取得します。

## 第19章

# 便利なコマンドラインツール

## 19.1 pvesubscription - サブスクリプション管理

このツールはProxmox VEのサブスクリプションを処理するために使用されます。

## 19.2 pveperf - Proxmox VE ベンチマークスクリプト

PATHにマウントされたハードディスク（デフォルトは/）のCPU//ハードディスクのパフォーマンスデータの収集を試みます：

### CPUボゴミップス

全CPUのボゴミップス合計

### REGEX/SECOND

1秒あたりの正規表現数（perl/パフォーマンステスト）は300000以上である必要があります。

### HDサイズ

ハードディスクサイズ

### パファードリード

簡単なHD読み取りテスト。最近のHDは少なくとも40MB/秒に達するはずです。

### 平均シーク時間

は平均シーク時間をテストします。高速SCSI HDは8ミリ秒未満の値に達します。一般的なIDE/SATAディスクの値は15~20ミリ秒です。

### FSyncs/SECOND

値を200より大きくする必要があります（RAIDコントローラーのライトバックキャッシュモードを有効にする必要があります - バッテリーバックアップキャッシュ（BBWC）が必要です）。

### DNS EXT

外部DNS名の平均解決時間

### DNS INT

ローカルDNS名の平均解決時間

## 19.3 Proxmox VE API 用シェルインターフェイス

Proxmox VE管理ツール(`pvsh`)を使用すると、REST/HTTPSサーバを使用せずにAPI関数を直接呼び出すことができます。

---

### 備考

それができるのは`root`だけです。

---

### 19.3.1 使用例

クラスタ内のノードのリストを取得

```
# pvsh get /nodes
```

データセンターで利用可能なオプションのリストを取得します。

```
# pvsh usage cluster/options -v
```

HTML5 NoVNCコンソールをデータセンターのデフォルトコンソールに設定

```
# pvsh set cluster/options -console html5
```

## 第20章

### よくある質問

---

#### 備考

新しいFAQは、このセクションの一番下に追加されています。

---

#### 1. Proxmox VEはどのようにいますか?

Proxmox VEはDebian GNU/Linuxをベースにしています。

#### 2. Proxmox VEプロジェクトはどのライセンスを使用していますか?

Proxmox VEコードのライセンスはGNU Affero General Public License, version 3です。

#### 3. Proxmox VEは32ビットプロセッサで動作しますか?

Proxmox VEは64ビットCPU(AMDまたはIntel)でのみ動作します。32ビットのプラットフォームは予定されていません。

---

#### 備考

VMとコンテナは、32ビットと64ビットの両方に対応しています。

---

#### 4. 私のCPUは仮想化をサポートしていますか?

CPUが仮想化に対応しているかどうかを確認するには、このコマンドの出力にvmxまたはsvmタグがあるかどうかを確認します:

```
egrep '(vmx|svm)' /proc/cpuinfo
```

#### 5. 対応インテルCPU

インテル® バーチャライゼーション・テクノロジー（インテル® VT-x）をサポートする64ビット・プロセッサー。（インテル対応プロセッサー一覧VTおよび64ビット）

#### 6. 対応AMD CPU

AMD仮想化テクノロジー（AMD-V）をサポートする64ビットプロセッサー。

#### 7. コンテナ／仮想環境(VE)／仮想専用サーバー(VPS)とは何ですか?

コンテナの文脈では、これらの用語はすべてオペレーティング・システム・レベルの仮想化という概念を指しています。オペレーティング・システム・レベルの仮想化とは、オペレーティング・システムのカーネルが、カーネルを共有する複数の分離されたインスタンスを可能にする仮想化の手法です。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くて済みますが、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）は、QEMU と Linux KVM カーネルモジュールを使用して Proxmox VE で仮想化されたゲストシステムです。

## 9. QEMUとは何ですか？

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ツールです。QEMU は Linux KVM カーネルモジュールを使用し、ホスト CPU 上でゲストコードを直接実行することで、ネイティブに近いパフォーマンスを実現します。Linuxゲストに限らず、任意のオペレーティングシステムを実行できます。

## 10. Proxmox VEのサポート期間はいつまでですか？

Proxmox VEのバージョンは、少なくとも対応するDebianバージョンが`oldstable`である限りサポートされます。Proxmox VE はローリングリリースモデルを採用しており、常に最新の安定版を使用することを推奨します。

プロックスモックスVEバージョン	Debian バージョン	ファースト・リリース	Debian EOL	プロックスモックスEOL
プロックスモックスVE 8	Debian 12 (本の虫)	2023-06	tba	tba
プロックスモックスVE 7	Debian 11 (ブルズアイ)	2021-07	2024-07	2024-07
プロックスモックスVE 6	Debian 10 (バスター)	2019-07	2022-09	2022-09
プロックスモックスVE 5	Debian 9 (ストレッチ)	2017-07	2020-07	2020-07
プロックスモックスVE 4	Debian 8 (ジェシー)	2015-10	2018-06	2018-06
プロックスモックスVE 3	Debian 7 (ウィージー)	2013-05	2016-04	2017-02
プロックスモックスVE 2	Debian 6 (スクイーズ)	2012-04	2014-05	2014-05
プロックスモックスVE 1	Debian 5 (レニー)	2008-10	2012-03	2013-01

## 11. Proxmox VEを次のポイントリリースにアップグレードする方法を教えてください。

マイナーバージョンアップ、例えばバージョン7.1のProxmox VEから7.2や7.3へのアップグレードは、通常のアップデートと同様に行うことができます。

ただし、[リリースノート](#)で関連する重要な変更、または破壊的な変更を確認してください。

アップデート自体には、Web UI の `Node→Updates` パネルを使用するか、CLI を使用します：

```
アプトアップデート
apt full-upgrade
```

### 備考

[パッケージリポジトリ](#)を正しくセットアップし、`apt update` でエラーが出なかった場合のみ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方法を教えてください。

Proxmox VE 4.4から5.0へのメジャーバージョンアップもサポートされています。アップグレードは慎重に計画し、テストする必要があります。

アップグレードの具体的な手順は、それぞれのセットアップによって異なりますが、アップグレードの実行方法に関する一般的な手順とアドバイスを提供します：

- [Proxmox VE 7から8へのアップグレード](#)
- [Proxmox VE 6から7へのアップグレード](#)
- [Proxmox VE 5から6へのアップグレード](#)
- [Proxmox VE 4から5へのアップグレード](#)
- [Proxmox VE 3から4へのアップグレード](#)

### 13. LXC vs LXD vs Proxmox Containers vs Docker

LXCは、Linuxカーネルのコンテナ機能のユーザー空間インターフェイスです。強力なAPIとシンプルなツールにより、Linuxユーザーはシステムコンテナを簡単に作成・管理できます。LXCは、以前のOpenVZと同様に、**システムの仮想化**を目的としています。そのため、コンテナ内で完全なOSを実行し、sshを使ってログインしたり、ユーザーを追加したり、apacheを実行したりすることができます。

LXDはLXCの上に構築され、より優れた新しいユーザー体験を提供します。LXDは、liblxcとそのGoバインディングを通じてLXCを使用し、コンテナの作成と管理を行います。基本的には、LXCのツールやディストリビューションテンプレートシステムに代わるもので、ネットワーク経由で制御可能な機能が追加されています。

Proxmoxコンテナは、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。また、**システム仮想化をターゲット**とし、コンテナ提供の基盤としてLXCを使用します。Proxmox Container Toolkit (pct)はProxmox VEと緊密に連携しています。つまり、クラスタ・セットアップを認識し、QEMU仮想マシン(VM)と同じネットワークおよびストレージ・リソースを使用できます。Proxmox VEのファイアウォールの使用、バックアップの作成と復元、HAフレームワークを使用したコンテナの管理も可能です。Proxmox VE APIを使用して、すべてをネットワーク経由で制御できます。

Dockerは、**単一のアプリケーション**を分離された自己完結型の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」と呼ばれています。Dockerインスタンスは、Docker Engineコマンドラインインタ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。

---

#### 備考

アプリケーションコンテナ、例えばDocker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 書誌

### 21.1 プロックスモックス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] Rik Goldman.Learning Proxmox VE.Packt Publishing, 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 管理者ハンドブック：Debian Bullseye の発見から習得まで](#), Freexian, 2021.ISBN 979-10-91414- 20-3
- [7] [Bir96] Kenneth P. Birman.Building Secure and Reliable Network Applications.Manning Publications Co, 1996.ISBN 978-1884777295
- [8] [Walsh10] Norman Walsh.DocBook 5: The Definitive Guide.O'Reilly & Associates, 2010.  
ISBN 978-0596805029
- [9] [Richardson07] Leonard Richardson & Sam Ruby.RESTful Web Services.O'Reilly Media, 2007.ISBN 978-0596529260
- [10] [Singh15] Karan Singh. Learning Ceph.Packt Publishing, 2015.ISBN 978-1783985623
- [11] [Singh16] Karan Singh.Ceph Cookbook Packt Publishing, 2016.ISBN 978-1784393502

- [12] [Mauerer08] Wolfgang Mauerer. Professional Linux Kernel Architecture. John Wiley & Sons, 2008. ISBN 978-0470343432
- [13] [Loshin03] Pete Loshin, IPv6: Theory, Protocol, and Practice, 2nd Edition. Morgan Kaufmann, 2003. ISBN 978-1558608108
- [14] [Loeliger12] Jon Loeliger & Matthew McCullough. Gitによるバージョン管理: 共同ソフトウェア開発のための強力なツールとテクニック. O'Reilly and Associates, 2012. ISBN 978-1449316389
- [15] [Kreibich10] Jay A. Kreibich. Using SQLite, O'Reilly and Associates, 2010. ISBN 978-0596521189

## 21.3 関連書籍

- [16] [Bessen09] James Bessen & Michael J. Meurer, Patent Failure: How Judges, Bureaucrats, and Lawyers Put Innovators at Risk. Princeton Univ Press, 2009. ISBN 978-0691143217

## 付録A

# コマンドラインインターフェース

## A.1 一般

歴史的に統一されていないオプションのケーシングスタイルについては、[設定ファイルの関連セクション](#)を参照してください。

## A.2 出力フォーマットのオプション [FORMAT\_OPTIONS]

出力形式は--output-formatパラメータで指定できます。デフォルトのフォーマット・テキストは、ASCII-artを使用して表の周りにきれいな枠線を描きます。さらに、いくつかの値を人間が読めるテキストに変換します：

- Unix epoch は ISO 8601 日付文字列として表示されます。
- 期間は週/日/時/分/秒のカウントで表示されます。
- バイトサイズの値には単位 (B、KiB、MiB、GiB、TiB、PiB) が含まれます。
- 分数はパーセントで表示されます。つまり、1.0は100%として表示されます。

オプション--quietを使えば、出力を完全に抑制することもできます。

--human-readable <boolean> (デフォルト= 1)  
出力レンダリング関数を呼び出して、人間が読めるテキストを生成します。

--noborder <boolean> (デフォルト= 0)  
ボーダーを描画しません（テキスト場合）。

--noheader <boolean> (デフォルト= 0)  
カラムヘッダを表示しない（テキスト場合）。

--output-format <json| json-pretty| text| yaml> (デフォルト= text)  
出力フォーマット

--quiet <ブール値  
印刷結果を抑制します。

## A.3 pvesm - Proxmox VE Storage Manager

**pvesm** <COMMAND> [ARGS] [OPTIONS] です。

**pvesm add** <タイプ> <ストレージ> [オプション] .

新しいストレージを作成します。

<type>: <btrfs| cephfs| cifs| dir| esxi| glusterfs| iscsi | iscsidirect | lvm | lvmthin | nfs | pbs | rbd | zfs | zfspool> です。

収納タイプ。

<ストレージ>: <ストレージID

ストレージ識別子。

--authsupported <文字列

認証済み。

--ベース <文字列

基本音量。このボリュームは自動的にアクティブになります。

--ブロックサイズ <文字列

ブロックサイズ

--bwlimit [<clone>=<LIMIT>] [<default>=<LIMIT>] [<migration>=<LIMIT>] [<move>=<LIMIT>] [<restore>=<LIMIT>] となります。

各種操作のI/O帯域幅制限を設定（単位：KiB/s）。

--コムスター\_hg <文字列

コムスタービューのホストグループ

--comstar\_tg <文字列

コムスター・ビューの対象者

--内容 <文字列

許可されるコンテンツタイプ

---

### 備考

コンテナには *rootdir* が、VM には *images* が使用されます。

---

--コンテンツディレクトリ <文字列

デフォルトのコンテンツタイプのディレクトリを上書きします。

-create-base-path<ブール値> (デフォルト= yes)

---

ベース・ディレクトリが存在しない場合は作成します。

--create-subdirs <boolean> (デフォルト= yes)

ディレクトリにデフォルトの入力します。

--データプール <文字列>

データプール（消去符号化専用）

--データストア <文字列>

Proxmox Backup Server データストア名。

--無効 <ブール値>

ストレージを無効にするフラグ。

--ドメイン <文字列>

CIFS ドメイン。

--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。パスフレーズなしで自動生成するには`autogen`を使用します。

--エクスポート <文字列>

NFSエクスポートパス。

--fingerprint ([A-Fa-f0-9]{2}:){31}[A-Fa-f0-9]{2}

証明書の SHA 256 フィンガープリント。

--format <qcow2| raw| subvol| vmdk>

デフォルトの画像フォーマット。

-fs-name<文字列>

Cephファイルシステム名。

--ヒューズ <ブール値>

FUSEを使用してCephFSをマウントします。

--is\_mountpoint <string> (デフォルト= no)

指定されたパスが外部で管理されているマウントポイントであると仮定し、マウントされていない場合はストレージをオフラインと見なします。ブーリアン (yes/no) 値を使用すると、このフィールドでターゲットパスを使用するショートカットとして機能します。

--iscsiprovider <文字列>

iscsi プロバイダ

--Cephクラスタで認証するためのキーリングを含むkeyringファイル

クライアントのキーリングの内容（外部クラスタ用）。

--krbd <boolean> (デフォルト= 0)

常にkrbdカーネルモジュールを通してrbdにアクセスしてください。

**--lio\_tpg <文字列>**

Linux LIOターゲット用ターゲットポータルグループ

**-master-pubkeyPEM形式のマスター公開鍵を含むファイル** Base64エンコードされたPEM形式のRSA公開鍵。暗号化された各バックアップに追加される暗号化キーのコピーを暗号化するために使用します。

**--max-protected-backups <integer> (-1 - N) (デフォルト= Datastore.Allocate権限を持つユーザは無制限、他のユーザは5)**

ゲストごとの保護バックアップの最大数。無制限には-1を使用します。

**-マックスファイル <整数> (0 - N)**

非推奨: 代わりに *prune-backups* を使用してください。VMごとのバックアップファイルの最大数。無制限の場合は0を使用します。

**-mkdir <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を設定します。注意: 非推奨。

*create-base-path* と *create-subdirs* オプションを使用。

**--monhost <文字列>**

モニターのIPアドレス（外部クラスタの場合）。

**--マウントポイント <文字列>**

マウントポイント

**--名前空間 <文字列>**

名前空間。

**--nocow <boolean> (デフォルト= 0)**

ファイルにNOCOWフラグを設定します。データのチェックサムを無効にし、直接I/Oを許可しながら、データエラーを回復できなくします。このフラグを使用するのは、基礎となるレイドシステムがない単一の ext4 フォーマットディスク上よりもデータを安全にする必要がない場合だけです。

**--ノード <文字列>**

ストレージ構成が適用されるノードのリスト。

**--nowritecache <ブール値>**

ターゲットの書き込みキャッシングを無効にします。

**--オプション <文字列>**

NFS/CIFS マウントオプション (*man nfs* または *man mount.cifs* を参照)

**--パスワード <password>**

共有/データストアにアクセスするためのパスワード。

**--パス <文字列>**

ファイルシステムのパス。

**--プール <文字列>**

プール。

**--ポート <整数> (1 - 65535)**

デフォルトのポートの代わりにこのポートを使用してストレージに接続します (PBS や ESXi など)。NFS および CIFS の場合は、*options* オプションを使用して、マウントオプションでポートを設定します。

**--portal <文字列>**

iSCSIポータル (IPまたはDNS名とオプションのポート)。

**--preallocation <falloc | full | metadata | off>** (デフォルト = metadata) raw および qcow2 画像に対するプリアロケーションモード。raw 画像でメタデータを使うと preallocation=off になります。

**--prune-backups [keep-all=<1|0>] [,keep-daily=<N>] [,keep-hourly=<N>] [,keep-last=<N>] [,keep-monthly=<N>] [,keep-weekly=<N>] [,keep-yearly=<N>].**

間隔が短い保持オプションが最初に処理され、--keep-lastが一番最初に処理れます。

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このオプションの対象となります。次のオプションでは、すでにカバーされているバックアップは考慮されず、古いバックアップのみが考慮されます。

**--セーフリムーブ <ブール値>**

LVを削除するとデータがゼロになります。

**-saferemove\_throughput<文字列>**

ワイスループット (cstream -t パラメータ値)。

**--サーバ <文字列>**

サーバーIPまたはDNS名。

**--サーバー2 <文字列>**

バックアップファイルサーバーのIPまたはDNS名。

---

**備考**

必要なオプション: server

---

**--共有 <文字列>**

CIFS 共有。

**--共有 <ブール値>**

すべてのノード (またはnodesオプションにリストされているすべて) で同じ内容を持つ単一のストレージであることを示します。ローカルストレージの内容が自動的に他のノードからアクセスできるようになるわけではなく、すでに共有されているストレージをそのようにマークするだけです!

--skip-cert-verification <boolean> (デフォルト= false)

TLS証明書の検証を無効にし、完全に信頼できるネットワークでのみ有効にします！

--smbversion <2.0| 2.1| 3| 3.0| 3.11| default> (default= default)

SMBプロトコルのバージョン。デフォルトでは、設定されていない場合、クライアントとサーバーの両方でサポートされている最高のSMB2+バージョンをネゴシエートします。

#### -スパース <ブール値>

スパースボリュームを使用

#### --サブディレクトリ <文字列>

マウントするサブディレクトリ。

#### --タグ付きのみ <ブール値>

pve-vm-ID でタグ付けされた論理ボリュームのみを使用します。

#### --ターゲット <文字列>

iSCSI ターゲット。

#### --シンプルル <文字列>

LVMシンプルLV名。

#### --トランスポート <rdma| tcp| unix>

クラスタ・トランスポート: tcpまたはrdma

#### --ユーザー名 <文字列>

RBD Id.

#### --vgname <文字列>

ボリュームグループ名。

#### --ボリューム <文字列>

Glusterfsボリューム。

**pvesm alloc <storage> <vmid> <filename> <size> [OPTIONS].**

ディスクイメージを割り当てます。

#### <ストレージ>: <ストレージID>

ストレージ識別子。

#### <vmid>: <整数> (100 - 999999999)

オーナーVMの指定

#### <ファイル名>: <文字列>

作成するファイル名。

<サイズ>です: \d+[MG]?

キロバイト (1024バイト) 単位のサイズ。オプションの接尾辞M (メガバイト、1024K) およびG (ギガバイト、1024M)。

--format <qcow2| raw| subvol| vmdk>

画像のフォーマット。

---

#### 備考

必要なオプション: サイズ

---

**pvesm apiinfo**

APIVERとAPIAGEを返します。

**pvesm cifsscan**

*pvesm scan cifs* のエイリアス。

**pvesm export** <ボリューム> <フォーマット> <ファイル名> [OPTIONS].

ボリュームのエクスポートに内部的に使用されます。

<ボリューム>: <文字列

ボリューム識別子

<format>: <btrfs| qcow2+size| raw+size| tar+size| vmdk+size zfs>. |

ストリーム形式のエクスポート

<ファイル名>: <文字列

保存先ファイル名

--ベース (?^i:[a-zA-Z0-9\_-]{1,40})

から増分ストリームを開始するスナップショット。

--snapshot (?^i:[a-zA-Z0-9\_-]{1,40})

エクスポートするスナップショット

--スナップショットリスト <文字列

転送するスナップショットの順序付きリスト

--with-snapshots <boolean> (デフォルト= 0)

中間スナップショットをストリームに含めるかどうか

**pvesm extractconfig** <ボリューム>。

vzdumpバックアップアーカイブから設定を抽出します。

<ボリューム>: <文字列

ボリューム識別子

---

**pvesm free <ボリューム> [オプション] .**

ボリュームの削除

#### <ボリューム>: <文字列>

ボリューム識別子

#### --ディレイ <整数> (1 - 30)

タスクが終了するまでの待ち時間。その時間内にタスクが終了した場合は *null* を返します。

#### --ストレージID

ストレージ識別子。

**pvesm glusterfsscan**

*pvesm scan glusterfs* のエイリアス。

**pvesm help [OPTIONS]**

指定したコマンドに関するヘルプを表示します。

#### --extra-args <array> です。

特定のコマンドのヘルプを表示します。

#### --verbose <ブール値>

冗長出力フォーマット。

**pvesm import <volume> <format> <filename> [OPTIONS] .**

ボリュームのインポートに内部的に使用されます。

#### <ボリューム>: <文字列>

ボリューム識別子

<format>: <btrfs| qcow2+size| raw+size| tar+size| vmdk+size zfs>. |  
インポートストリーム形式

#### <ファイル名>: <文字列>

ソースファイル名。stdinが使用される場合、tcp://<IPまたはCIDR>フォーマットはTCP接続を、unix://PATH-TO-SOCKETフォーマットはUNIXソケットを入力として使用できます。

#### --allow-rename <boolean> (デフォルト= 0)

要求されたボリュームIDがすでに存在する場合は、エラーをスローする代わりに新しいボリュームIDを選択します。

#### --ベース (?^i:[a-zA-Z\_-]{1,40})

増分ストリームの基本スナップショット

--delete-snapshot (?^i:[a-zA-Z0-9\_\\-]{1,80})  
成功時に削除するスナップショット

--snapshot (?^i:[a-zA-Z0-9\_\\-]{1,40})  
ストリームにスナップショットが含まれている場合は、現在の状態のスナップショット

--with-snapshots <boolean> (デフォルト= 0)  
ストリームが中間スナップショットを含むかどうか

#### pvesm iscsiscan

*pvesm scan iscsi* のエイリアス。

**pvesm list** <storage> [OPTIONS].

ストレージの内容を一覧表示します。

#### <ストレージ>: <ストレージID

ストレージ識別子。

#### --内容 <文字列

このタイプのコンテンツのみをリストアップします。

--vmid <整数> (100 - 999999999)  
このVMのイメージのみを表示

#### pvesm lvmscan

*pvesm scan lvm* のエイリアス。

#### pvesm lvmthinscan

*pvesm scan lvmthin* のエイリアス。

#### pvesm nfsscan

*pvesm scan nfs* のエイリアス。

#### pvesm パス <ボリューム>

指定されたボリュームのファイルシステムのパスを取得します。

#### <ボリューム>: <文字列

ボリューム識別子

**pvesm prune-backups** <storage> [OPTIONS].

バックアップを刈り込みます。標準の命名スキームを使用しているものだけが考慮されます。keepオプションが指定されていない場合は、ストレージ構成のものが使用されます。

#### <ストレージ>: <ストレージID

ストレージ識別子。

**- ドライラン <ブール値>**

刈り込まれるものだけを表示し、何も削除しないでください。

**--keep-all <ブール値>**

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。

**--キープ・デイリー <N>**

過去<N>日分のバックアップを保持します。1日に複数のバックアップがある場合は、最新のものだけが保持されます。

**--キープ・アワー<N>**

最後の<N>異なる時間のバックアップを保持します。1時間に複数のバックアップがある場合、最新のものだけが保持されます。

**--keep-last <N>**

最後の<N>個のバックアップを保持します。

**--キープマンスリー<N>**

過去<N>の異なる月のバックアップを保持します。1つの複数のバックアップがある場合、最新のものだけが保持されます。

**--キープウィークリー<N>**

過去<N>週間分のバックアップを保持します。1週間に複数のバックアップがある場合、最新のものだけが保持されます。

**--キープ・イヤー・リー<N>**

過去<N>年分のバックアップを保持します。一つの年に複数のバックアップがある場合、最新のものだけが保持されます。

**--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てください。

**--vmid <整数> (100 - 999999999)**

このバックアップのみを考慮してください。

**pvesm remove <ストレージ>**

ストレージ構成を削除します。

**<ストレージ>: <ストレージID>**

ストレージ識別子。

**pvesm scan cifs <server> [OPTIONS].**

リモート CIFS サーバーをスキャンします。

**<サーバー>: <文字列>**

サーバーアドレス（名前またはIP）。

**--ドメイン <文字列>**

SMB ドメイン (ワークグループ)。

**--パスワード <password>**

ユーザー パスワード

**--ユーザー名 <文字列>**

ユーザー名

**pvesm scan glusterfs <server>**

リモートのGlusterFSサーバをスキャンします。

**<サーバー>: <文字列>**

サーバーアドレス (名前またはIP)。

**pvesm scan iscsi <portal>**

リモートiSCSIサーバーをスキャンします。

**<ポータル>: <文字列>**

iSCSIポータル (IPまたはDNS名とオプションのポート)。

**pvesm scan lvm**

ローカル LVM ボリュームグループを一覧表示

します。 **pvesm scan lvmthin <vg>** ローカ

ル LVM シンプルを一覧表示します。

**<vg>: [a-zA-Z0-9\.\+\\_\-] [a-zA-Z0-9\.\+\\_\-]+\+**

説明なし

**pvesm scan nfs <server>**

リモート NFS サーバーをスキャンします。

**<サーバー>: <文字列>**

サーバーアドレス (名前またはIP)。

**pvesm scan pbs <server> <username> --password <string> [OPTIONS] [FORMAT\_OPTIONS]**

リモートのProxmox/バックアップサーバをスキャンします。

**<サーバー>: <文字列>**

サーバーアドレス (名前またはIP)。

**<ユーザー名>: <文字列**

ユーザー名またはAPIトークンID。

**--fingerprint ([A-Fa-f0-9]{2}:){31}[A-Fa-f0-9]{2}**

証明書の SHA 256 フィンガープリント。

**--パスワード <文字列**

ユーザーパスワードまたはAPIトークンの秘密。

**--port <integer> (1 - 65535) (デフォルト= 8007)**

オプションのポート。

**pvesm scan zfs**

ローカルノードのzfsプールリストをスキャンします。

**pvesm set <storage> [OPTIONS] .**

ストレージ構成を更新します。

**<ストレージ>: <ストレージID**

ストレージ識別子。

**--ブロックサイズ <文字列**

ブロックサイズ

**--bwlimit [clone=<LIMIT>] [,default=<LIMIT>] [,migration=<LIMIT>] [,move=<LIMIT>] [,restore=<LIMIT>]** となります。

各種操作のI/O帯域幅制限を設定（単位: KiB/s）。

**--コムスター\_hg <文字列**

コムスター・ビューのホストグループ

**--comstar\_tg <文字列**

コムスター・ビューの対象者

**--内容 <文字列**

許可されるコンテンツタイプ

---

**備考**

コンテナには *rootdir* が、VM には *images* が使用されます。

---

**--コンテンツディレクトリ <文字列**

デフォルトのコンテンツタイプのディレクトリを上書きします。

**-create-base-path<ブール値> (デフォルト= yes)**

---

ベース・ディレクトリが存在しない場合は作成します。

--create-subdirs <boolean> (デフォルト= yes)

ディレクトリにデフォルトの入力します。

--データプール <文字列>

データプール（消去符号化専用）

--削除 <文字列>

削除したい設定のリスト。

-ダイジェスト <文字列>

現在のコンフィギュレーション・ファイルのダイジェストが異なる場合、変更を防止します。これは、現在のコンフィギュレーション・ファイルが変更されないようにするために使用できます。

--無効 <ブール値>

ストレージを無効にするフラグ。

--ドメイン <文字列>

CIFS ドメイン。

--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。パスフレーズなしで自動生成するには`autogen`を使用します。

--fingerprint ([A-Fa-f0-9]{2}::){31}[A-Fa-f0-9]{2}

証明書の SHA 256 フィンガープリント。

--format <qcow2| raw| subvol| vmdk>

デフォルトの画像フォーマット。

-fs-name<文字列>

Cephファイルシステム名。

--ヒューズ <ブール値>

FUSEを使用してCephFSをマウントします。

--is\_mountpoint <string> (デフォルト= no)

指定されたパスが外部で管理されているマウントポイントであると仮定し、マウントされていない場合はストレージをオフラインと見なします。ブーリアン (yes/no) 値を使用すると、このフィールドでターゲットパスを使用するショートカットとして機能します。

--Cephクラスタで認証するためのキーリングを含むkeyringファイル

クライアントのキーリングの内容（外部クラスタ用）。

--krbd <boolean> (デフォルト= 0)

常にkrbdカーネルモジュールを通してrbdにアクセスしてください。

**--lio\_tpg <文字列>**

Linux LIOターゲット用ターゲットポータルグループ

**-master-pubkeyPEM形式のマスター公開鍵を含むファイル** Base64エンコードされたPEM形式のRSA公開鍵。暗号化された各バックアップに追加される暗号化キーのコピーを暗号化するために使用します。

**--max-protected-backups <integer> (-1 - N) (デフォルト= Datastore.Allocate権限を持つユーザは無制限、他のユーザは5)**

ゲストごとの保護バックアップの最大数。無制限には-1を使用します。

**-マックスファイル <整数> (0 - N)**

非推奨: 代わりに *prune-backups* を使用してください。VMごとのバックアップファイルの最大数。無制限の場合は0を使用します。

**-mkdir <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を設定します。注意: 非推奨。

*create-base-path* と *create-subdirs* オプションを使用。

**--monhost <文字列>**

モニターのIPアドレス（外部クラスタの場合）。

**--マウントポイント <文字列>**

マウントポイント

**--名前空間 <文字列>**

名前空間。

**--nocow <boolean> (デフォルト= 0)**

ファイルにNOCOWフラグを設定します。データのチェックサムを無効にし、直接I/Oを許可しながら、データエラーを回復できなくします。このフラグを使用するのは、基礎となるレイドシステムがない単一の ext4 フォーマットディスク上よりもデータを安全にする必要がない場合だけです。

**--ノード <文字列>**

ストレージ構成が適用されるノードのリスト。

**--nowritecache <ブール値>**

ターゲットの書き込みキャッシングを無効にします。

**--オプション <文字列>**

NFS/CIFS マウントオプション (*man nfs* または *man mount.cifs* を参照)

**--パスワード <password>**

共有/データストアにアクセスするためのパスワード。

**--プール <文字列>**

プール。

**--ポート <整数> (1 - 65535)**

デフォルトのポートの代わりにこのポートを使用してストレージに接続します (PBS や ESXi など)。NFS および CIFS の場合は、*options* オプションを使用して、マウントオプションでポートを設定します。

**--preallocation <fallow | full | metadata | off> (デフォルト=metadata)** raw および qcow2 画像に対するプリアロケーションモード。raw 画像でメタデータを使うと preallocati-on=off になります。**--prune-backups [keep-all=<1|0>] [,keep-daily=<N>] [,keep-hourly=<N>] [,keep-last=<N>] [,keep-monthly=<N>] [,keep-weekly=<N>] [,keep-yearly=<N>].**

間隔が短い保持オプションが最初に処理され、--keep-lastが一番最初に処理れます。

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このオプションの対象となります。次のオプションでは、すでにカバーされているバックアップは考慮されず、古いバックアップのみが考慮されます。

**--セーフリムーブ <ブール値>**

LVを削除するとデータがゼロになります。

**-saferemove \_throughput<文字列>**

ワイスループット (cstream -tパラメータ値)。

**--サーバ <文字列>**

サーバーIPまたはDNS名。

**--サーバー2 <文字列>**

バックアップファイルサーバーのIPまたはDNS名。

---

**備考**

必要なオプション: server

---

**--共有 <ブール値>**

すべてのノード (またはnodesオプションにリストされているすべて) で同じ内容を持つ単一のストレージであることを示します。ローカルストレージの内容が自動的に他のノードからアクセスできるようになるわけではなく、すでに共有されているストレージをそのようにマークするだけです!

**--skip-cert-verification <boolean> (デフォルト= false)**

TLS証明書の検証を無効にし、完全に信頼できるネットワークでのみ有効にします!

**--smbversion <2.0| 2.1| 3| 3.0| 3.11| default> (default= default)**

SMBプロトコルのバージョン。デフォルトでは、設定されていない場合、クライアントとサーバーの両方でサポートされている最高のSMB2+バージョンをネゴシエートします。

**-スパース <ブール値>**

スパースボリュームを使用

---

**--サブディレクトリ <文字列>**

マウントするサブディレクトリ。

**--タグ付きのみ <ブール値>**

*pve-vm-ID* でタグ付けされた論理ボリュームのみを使用します。

**--トランスポート <rdma | tcp | unix>**

クラスタ・トランスポート: tcpまたはrdma

**--ユーザー名 <文字列>**

RBD Id.

**pvesm status [OPTIONS]**

すべてのデータストアのステータスを取得します。

**--内容 <文字列>**

このコンテンツタイプをサポートするストアのみをリストします。

**--enabled <boolean> (デフォルト= 0)**

有効になっている(無効になっていない)ストアのみをリストします。

**--format <boolean> (デフォルト= 0)**

フォーマットに関する情報

**--ストレージID**

指定したストレージのステータスのみを一覧表示

**--ターゲット <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でアクセス可能な共有ストレージのみがリストされます。

**pvesm zfs**

*pvesm scan zfs* のエイリアス。

## A.4 pvesubscription - Proxmox VE サブスクリプションマネージャ

**pvesubscription** <COMMAND> [ARGS] [OPTIONS] です。

**pvesubscriptionの削除**

このノードの購読キーを削除します。

**サブスクリプション取得**

**pvesubscription help** [OPTIONS] 指定したコマンド

に関するヘルプを表示します。

**--extra-args <array>** です。

特定のコマンドのヘルプを表示します。

**--verbose <ブール値**

冗長出力フォーマット。

**pvesubscription** セット<キー

購読キーを設定します。

<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>** を設定します。

内部使用のみです！オフラインキーを設定するには、代わりに proxmox-offline-mirror-helper パッケージを使ってください。

<データ>: <文字列

署名された購読情報の塊

**pvesubscription update** [OPTIONS] を実行します。

購読情報を更新しました。

**--force <boolean>** (デフォルト= 0)

ローカルキャッシュがまだ有効であっても、常にサーバーに接続します。

## 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 サービスを管理します。

**pveceph** <COMMAND> [ARGS] [OPTIONS] です。

**pveceph createmgr**

*pveceph mgr create* のエイリアス。

ヴェーセフ・クリエイトモン

*pveceph mon create* のエイリアス。

ブベケフ・クリエイト・オスド

*pveceph osd create* のエイリアス。

**pveceph**はプールを作成します。

*pveceph pool create* のエイリアス。

## ヴェーセフ・デストロイムグ

*pveceph mgr destroy* のエイリアス。

## ブヴェセフ・デストロイモン

*pveceph mon destroy* のエイリアス。

## ヴェセフ・デストロエスティー

*pveceph osd destroy* のエイリアス。

## ブベセフ・デストロイプール

*pveceph pool destroy* のエイリアス。 **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)

作成されたCephFSをこのクラスタのストレージとして構成します。

**--name** (?^: ^[ ^:/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)

パッキング・データ・プールの配置グループ数。メタデータ・プールはこの4分の1を使用します。

**pveceph fs destroy** <name> [OPTIONS].

Cephファイルシステムの破棄

<名前>: <文字列

cephファイルシステム名。

**--remove-pools <boolean>** (デフォルト= 0)

この fs に設定されているデータとメタデータのプールを削除します。

**--remove-storages <boolean>** (デフォルト= 0)

この fs 用に構成された pveceph 管理ストレージをすべて削除します。

**pveceph help** [OPTIONS] (オプション)

指定したコマンドに関するヘルプを表示します。

**--extra-args <array>** です。

特定のコマンドのヘルプを表示します。

**--verbose <ブール値>**

冗長出力フォーマット。

**pveceph init [OPTIONS]** です。

cephの初期デフォルト構成を作成し、シンボリックリンクを設定します。

#### --cluster-network <文字列>

独立したクラスタネットワークを宣言し、OSDはハートビート、オブジェクトレプリケーション、リカバリのトラフィックをルーティングします。

##### 備考

必要なオプション: ネットワーク

#### --disable\_cephx <boolean> (デフォルト= 0)

cephx認証を無効にします。



##### 警告

cephxは中間者攻撃から保護するセキュリティ機能です。ネットワークがプライベートな場合にのみ、cephxの無効検討してください!

#### --min\_size <integer> (1 - 7) (デフォルト= 2)

I/Oを許可するために、オブジェクトごとに利用可能なレプリカの最小数

#### --ネットワーク <文字列>

すべてのceph関連トラフィックに特定のネットワークを使用します。

#### --pg\_bits <整数> (6 - 14) (デフォルト= 6)

配置グループのデフォルト数を指定するために使用される配置グループビット。非推奨。この設定は、最近のCephバージョンで非推奨になりました。

#### --サイズ <整数> (1 - 7) (デフォルト= 3)

オブジェクトあたりの目標レプリカ数

**pveceph install [OPTIONS]** (オプション)

ceph関連パッケージをインストールします。

#### --allow-experimental <boolean> (デフォルト= 0)

実験的な許可します。使用してください!

#### --repository <enterprise| no-subscription| test> (=

企業)

使用するCephリポジトリ。

#### --version <quincy| reef| squid> (default=quincy)

インストールするCephのバージョン。

鞭打ち鞭打ち鞭打ち鞭打ち鞭打ち鞭打ち鞭打ち鞭打ち鞭打ち鞭打ち鞭打ち

**pveceph pool ls**のエイリアス。 **pveceph mds create**

[OPTIONS] Cephメタデータサーバ(MDS)を作成します。

**--hotstandby <boolean>** (デフォルト= 0)

ceph-mdsデーモンがアクティブなデータシートのログをポーリングして再生するかどうかを決定します。データシートの障害時の切り替えが速くなりますが、アイドルリソースがより多く必要になります。

**--name [a-zA-Z0-9]([a-zA-Z0-9-]\*[a-zA-Z0-9])?** (default= nodename)

省略時はノード名と同じ。

**pveceph mds destroy <名前>**。

Cephメタデータサーバの破棄

**<name>: [a-zA-Z0-9]([a-zA-Z0-9\-\-]\*[a-zA-Z0-9])?**  
mdsの名前 (ID)

**pveceph mgr create [OPTIONS] [オプション]**。

Ceph Managerの作成

**--id [a-zA-Z0-9]([a-zA-Z0-9\-\-]\*[a-zA-Z0-9])?**  
省略時はノード名と同じ。

**pveceph mgr 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 [OPTIONS]**を作成します。

Cephモニタとマネージャの作成

**--モン・アドレス <文字列**

自動検出されたモニタIP上書きします。Cephのパブリックネットワーク内にある必要があります。

**--monid [a-zA-Z0-9]([a-zA-Z0-9\-\-]\*[a-zA-Z0-9])?**  
省略時はノード名と同じ。

**pveceph mon**を破壊する <monid

Ceph MonitorとManagerを破棄します。

<monid>: [a-zA-Z0-9] ([a-zA-Z0-9\-\-]\*[a-zA-Z0-9])?  
モニターID

**pveceph osd create <dev> [OPTIONS]**.

OSDの作成

<dev>: <文字列>  
ブロックデバイス名。

**-crush-device-class <文字列>**

OSDのデバイスクラスをクラッシュで設定します。

**--db\_dev <文字列>**

block.db のブロックデバイス名。

**--db\_dev\_size <number> (1 - N) (デフォルト= bluestore\_block\_db\_size または OSD サイズの 10%)**  
block.dbのGiB単位のサイズ。

---

#### 備考

必要なオプション: db\_dev

---

**--encrypted <boolean> (デフォルト= 0)**

OSDの暗号化を有効にします。

**--osds-per-device <整数> (1 - N)**

物理デバイスごとのOSDサービス。高速な NVMe デバイスにのみ有効です。

**--wal\_dev <文字列>**

block.walのブロックデバイス名。

**--wal\_dev\_size <number> (0.5 - N) (デフォルト= bluestore\_block\_wal\_size または OSD サイズの 1%)**  
block.walのGiBサイズ。

---

#### 備考

必要なオプション: wal\_dev

---

**pveceph osd destroy <osdid> [OPTIONS]**.

OSDの破棄

<osdid>: <整数  
OSD ID

--クリーンアップ <ブール値> (デフォルト= 0)  
もしセットされていれば、パーティションテーブルのエントリーを削除します。

**pveceph osd details** <osdid> [OPTIONS] [FORMAT\_OPTIONS].

OSDの詳細を取得します。

<osdid>: <文字列  
OSDのID

--verbose <boolean> (デフォルト= 0)  
json-pretty 出力形式と同じです。

**pveceph pool create** <name> [OPTIONS] [オプション]。

Cephプールの作成

<name>: (?^:^[^:/s]+\$)  
プールの名前。一意でなければなりません。

--add\_storages <boolean> (デフォルト= 0; データ消去プールの場合: 1)  
新しいプールを使用してVMとCTストレージを構成します。

--application <cephfs| rbd| rgw> (デフォルト= rbd)  
プールの応用。

-crush\_rule (クラッシュルール) <文字列  
クラスタ内のオブジェクト配置のマッピングに使用するルール。

--erasure-coding k=<integer>, m=<integer> [,device-class=<class>] [,failure-domain=<domain>] [,profile=<profile>].  
RBD用に消去コード化プールを作成し、メタデータ・ストレージ用にレプリケート・プールを付随させます。  
ECでは、共通のcephオプションのsize、min\_size、crush\_ruleパラメータがメタデータプールに適用されます。

--min\_size <integer> (1 - 7) (デフォルト= 2)  
オブジェクトあたりの最小レプリカ数

--pg\_autoscale\_mode <off| on| warn> (デフォルト= warn)  
プールの自動PGスケーリングモード。

--pg\_num <整数> (1 - 32768) (デフォルト= 128)  
配置グループの数。

--pg\_num\_min <整数> (-N - 32768)

配置グループの最小数。

--サイズ <整数> (1 - 7) (デフォルト=3)

オブジェクトごとのレプリカ数

--target\_size(~--~) ([KMGТ])?

PGオートスケーラのプールの推定目標サイズ。

--target\_size\_ratio <数値> を指定します。

PGオートスケーラのプールの推定目標比率。

**pveceph pool destroy** <name> [OPTIONS] [オプション].

プールを破壊

<名前>: <文字列

プールの名前。一意でなければなりません。

--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棄します。

--remove\_ecprofile <boolean> (デフォルト= 1)

消去コード・プロファイルを削除します。該当する場合、デフォルトは true です。

--remove\_storages <boolean> (デフォルト= 0)

このプールに設定されているすべての pveceph 管理ストレージを削除します。

**pveceph pool get** <name> [OPTIONS] [FORMAT\_OPTIONS] です。

現在のプールの状態を表示します。

<名前>: <文字列

プールの名前。一意でなければなりません。

--verbose <boolean> (デフォルト= 0)

有効にすると、追加データ（統計など）が表示されます。

**pveceph pool ls** [FORMAT\_OPTIONS] です。

すべてのプールとその設定 (POST/PUT エンドポイントで設定可能) を一覧表示します。

**pveceph pool set** <name> [OPTIONS].

プール設定の変更

<name>: (?^:^[^:/s]+\$)

プールの名前。一意でなければなりません。

--アプリケーション <cephfs| rbd| rgw>

プールの応用。

-crush\_rule (クラッシュルール) <文字列>

クラスタ内のオブジェクト配置のマッピングに使用するルール。

--最小サイズ <整数> (1 - 7)

オブジェクトあたりの最小レプリカ数

--pg\_autoscale\_mode <off| on| warn> です。

プールの自動PGスケーリングモード。

--pg\_num <整数> (1 - 32768)

配置グループの数。

--pg\_num\_min <整数> (-N - 32768)

配置グループの最小数。

--サイズ <整数> (1 - 7)

オブジェクトごとのレプリカ数

--target\_size([KMGT])?

PGオートスケーラのプールの推定目標サイズ。

--target\_size\_ratio <数値> を指定します。

PGオートスケーラのプールの推定目標比率。

**pveceph purge [OPTIONS] (ページ・オプション)**

ceph関連のデータと構成破棄します。

--クラッシュ <ブール値>

さらに、/var/lib/ceph/crashのCephクラッシュログをページします。

--ログ <ブール値>

さらに、/var/log/cephのCephログをページします。

**pveceph start [OPTIONS]** です。

cephサービスを開始します。

--サービス

(ceph|mon|mds|osd|mgr) (.|[a-zA-Z0-9]([a-zA-Z0-9\-\-]\*[a-zA-Z0-9]))? (デフォルト=ceph.target)。

Cephサービス名。

**pvecephステータス**

Cephのステータスを取得します。

**pveceph stop** [OPTIONS] [オプション] .

cephサービスを停止します。

**--サービス**

(ceph|mon|mds|osd|mgr) (\. [a-zA-Z0-9] ([a-zA-Z0-9\-\-]\* [a-zA-Z0-9])?) ?

(デフォルト=ceph.target).

Cephサービス名。

## A.7 pvenode - Proxmox VE ノード管理

**pvenode** <COMMAND> [ARGS] [OPTIONS] です。

**pvenode acme account deactivate** [<name>] (アクメアカウントの無効化)

CAで既存のACMEアカウントを停止します。

<name>: <名前> (デフォルト= デフォルト)

ACMEアカウント設定ファイル名。

**pvenode acme account info** [<name>] [FORMAT\_OPTIONS] .

既存のACMEアカウント情報を返します。

<name>: <名前> (デフォルト= デフォルト)

ACMEアカウント設定ファイル名。

**pvenode acme**アカウントリスト

ACMEEAccount インデックス。

**pvenode acme account register** [<name>] {<contact>} [OPTIONS] .

互換性のあるCAに新しいACMEアカウントを登録します。

<name>: <名前> (デフォルト= デフォルト)

ACMEアカウント設定ファイル名。

<連絡先>: <文字列>

連絡先メールアドレス

-ディレクトリ?://.\*

ACME CA ディレクトリエンドポイントの URL。

**pvenode acme account update** [<name>] [OPTIONS].

既存の ACME アカウント情報を CA で更新します。注意：新しいアカウント指定しないと、更新がトリガされます。

<name>: <名前> (デフォルト= デフォルト)

ACMEアカウント設定ファイル名。

--連絡先 <文字列>

連絡先メールアドレス

**pvenode acme cert order** [OPTIONS] (アクメ証明書注文)

ACME 互換の CA に新しい証明書を注文します。

--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** [OPTIONS] (アクメ証明書の更新)

CA から既存の証明書を更新します。

--force <boolean> (デフォルト= 0)

有効期限が30日以上場合でも、強制的に更新されます。

**pvenode acme 認証取り消し**

CA から既存の証明書を失効させます。

**pvenode acme plugin add** <type> <id> [OPTIONS].

ACMEプラグインの設定を追加します。

<type>: <dns | standalone>

ACMEのチャレンジタイプ。

<id>: <文字列>

ACMEプラグインID名

```
--api <1984hosting | acmedns | acmeproxy | active24 | ad | ali | alviy | anx | artfiles | arvan
| aurora | autodns | aws | azion | azure | bookmyname| bunny| cf| clouddns| cloudns| cn| conoha
| constellix| cpanel| curanet| cyon| dal| ddnss| desec| df | dgon | dnsexit | dnshome | dnsimple
| dnservices | doapi | domeneshop | dp | dpi | dreamhost | duckdns | durabledns | dyn | dynu |
dynv6 | easydns | edgedns | euserv | exoscale | fornex | freedns | gandi_livedns | gcloud |
gcore | gd | geoscaling | googledomains| he| hetzner| hexonet| hostingde| huaweicloud |
infoblox | infomaniak | internetbs | inwx | ionos | ionos_cloud | ipv64| ispconfig| jdl| joker|
kappernet| kas| kinghost| knot
| la| leaseweb| lexicon| limacity| linode| linode_v4| loopia
| lua | maradns | me | miab | misaka | myapi | mydevil | mydnsjp | mythic_beasts| namecheap| namecom|
namesilo| nanelo| nederhost
| neodigit | netcup | netlify | nic | njalla | nm | nsd | nsone | nsupdate | nw | oci | omglol
| one | online | openprovider | openstack | opnsense | ovh | pdns | pleskxml | pointhq | porkbun
| rackcorp | rackspace | rage4 | rcode0 | regru | scaleway | schlundtech| selectel| selfhost|
servercow| simply| technitium
| テレ3 | テンセント | タイムウェブ | トランシップ | UDR | ウルトラ | ウノユーロ | バリオメディア | ヴィースプ | ヴェルセル |
| ブイスケール | ヴルトル | ウェブサポート | ウエスト_CN
| world4you| yandex360| yc| zilore| zone| zoneedit| zonomi>
```

APIプラグイン名

--data 1行に1つのキーと値のペアを持つファイルで、プラグイン設定に保存するためにbase64urlエンコードされます。

DNSプラグインのデータ。(base64 エンコード)

--無効 <布尔値>

設定を無効にするフラグ。

--ノード <文字列>

クラスタ・ノード名のリスト。

--validation-delay <integer> (0 - 172800) (デフォルト= 30)

検証を要求する前に待つ、秒単位の追加遅延。DNS レコードの TTL が長い場合に対応できるようにします。

pvenode acme plugin config <id> [FORMAT\_OPTIONS] です。

ACMEプラグインの設定を取得します。

<id>:<文字列>

ACMEプラグインインスタンスの一意な識別子。

pvenode acmeプラグインリスト [OPTIONS] [FORMAT\_OPTIONS]

ACMEプラグインインデックス

```
--type <dns| standalone>
```

特定のタイプのACMEプラグインのみをリストアップします。

```
pvenode acme プラグイン remove <id>
```

ACMEプラグインの設定を削除します。

<id>:<文字列>

ACMEプラグインインスタンスの一意な識別子。

```
pvenode acme plugin set <id> [OPTIONS] [オプション]。
```

ACMEプラグインの設定を更新します。

<id>:<文字列>

ACMEプラグインID名

```
--api <1984hosting | acmedns | acmeproxy | active24 | ad | ali | alviy | anx | artfiles | arvan | aurora | autodns | aws | azion | azure | bookmyname | bunny | cf | clouddns | cloudns | cn | conoha | constellix | cpanel | curanet | cyon | da | ddns | desec | df | dgon | dnsexit | dnshome | dnsimple | dnservices | doapi | domeneshop | dp | dpi | dreamhost | duckdns | durabledns | dyn | dynu | dynv6 | easydns | edgedns | euserv | exoscale | fornex | freedns | gandi_livedns | gcloud | gcore | gd | geoscaling | googledomains | he | hetzner | hexonet | hostingde | huaweicloud | infoblox | infomaniak | internetbs | inwx | ionos | ionos_cloud | ipv64 | ispconfig | jd | joker | kappernet | kas | kinghost | knot | la | leasesweb | lexicon | limacity | linode | linode_v4 | loopia | lua | maradns | me | miab | misaka | myapi | mydevil | mydnsjp | mythic_beasts | namecheap | namecom | namesilo | nanelo | nederhost | neodigit | netcup | netlify | nic | njalla | nm | nsd | nsone | nsupdate | nw | oci | omglol | one | online | openprovider | openstack | opnsense | ovh | pdns | pleskxml | pointhq | porkbun | rackcorp | rackspace | rage4 | rcode0 | regru | scaleway | schlundtech | selectel | selfhost | servercow | simply | technitium | テレ3 | テンセント | タイムウェブ | トランシップ | UDR | ウルトラ | ウノユーロ | バリオメディア | ヴィースプ | ベルセル | ブイスケール | ヴルトル | ウェブサポート | ウエスト_CN | world4you | yandex360 | yc | zilore | zone | zoneedit | zonomi>
```

APIプラグイン名

```
--data 1行に1つのキーと値のペアを持つファイルで、プラグイン設定に保存するためにbase64urlエンコードされます。
```

DNSプラグインのデータ。(base64 エンコード)

--削除 <文字列>

削除したい設定のリスト。

**-ダイジェスト <文字列>**

現在のコンフィギュレーション・ファイルのダイジェストが異なる場合、変更を防止します。これは、現在のコンフィギュレーション・ファイルが変更されないようにするために使用できます。

**--無効 <ブール値>**

設定を無効にするフラグ。

**--ノード <文字列>**

クラスタ・ノード名のリスト。

**--validation-delay <integer> (0 - 172800) (デフォルト= 30)**

検証を要求する前に待つ、秒単位の追加遅延。DNS レコードの TTL が長い場合に対応できるようにします。

**pvenode cert delete [<restart>]。**

カスタム証明書チェーンとキーを削除します。

**<restart>: <boolean> (デフォルト= 0)**

pveproxyを再起動します。

**pvenode証明書情報 [FORMAT\_OPTIONS]。**

ノードの証明書に関する情報を取得します。

**pvenode cert set <certificates> [<key>] [OPTIONS] [FORMAT\_OPTIONS].**

カスタム証明書チェーンとキーをアップロードまたは更新します。

**<証明書>: <文字列>**

PEM エンコードされた証明書（チェーン）。

**<キー>: <文字列>**

PEM エンコードされた秘密鍵。

**--force <boolean> (デフォルト= 0)**

既存のカスタムまたは ACME 証明書ファイルを上書きします。

**--restart <boolean> (デフォルト= 0)**

pveproxyを再起動します。

**pvenode config get [OPTIONS] [OPTIONS]** を取得します。

ノード設定オプションを取得します。

**-プロパティ <acme | acmedomain0 | acmedomain1 | acmedomain2 | acmedomain3 | acmedomain4 | acmedomain5 | ballooning-target | description | startall-onboot-delay | wakeonlan> (デフォルト= all)**

ノード構成から特定のプロパティのみを返します。

**pvenode config set [OPTIONS]** を設定します。

ノード設定オプションを設定します。

**--acme [アカウント=<名前> [, ドメイン=<ドメイン[; ドメイン; ...]>]]。**

ノード固有のACME設定。

**--acmedomain[n] [domain=<domain> [,alias=<ドメイン> [,plugin=<プラグイン設定名>]]。**

ACMEドメインとバリデーションプラグイン

**--ballooning-target <integer> (0 - 100) (デフォルト= 80)**

バルーニングの対象となるRAM使用量(総メモリ量に対する割合)

**--削除 <文字列**

削除したい設定のリスト。

**--説明 <文字列**

ノードの説明。ウェブインターフェースのノードノートパネルに表示されます。これは設定ファイル内にコメントとして保存されます。

**-ダイジェスト <文字列**

現在の設定ファイルの SHA1 ダイジェストが異なる場合に変更を防止します。これは同時変更を防ぐために使用できます。

**--startall-onboot-delay <integer> (0 - 300) (デフォルト= 0)**

オンブートが有効なすべての仮想ゲストを開始するまでの初期遅延時間(秒)。

**--wakeonlan [mac=<MACアドレス> [,bind-interface=<バインドインターフェース>] [,broadcast-address=<IPv4プロードキャストアドレス>]]。**

ノード固有のウェイクオンLAN設定。

**pvenodeヘルプ [オプション]**

指定したコマンドに関するヘルプを表示します。

**--extra-args <array>** です。

特定のコマンドのヘルプを表示します。

**--verbose <ブール値**

冗長出力フォーマット。

**pvenode migrateall <ターゲット> [オプション] .**

すべてのVMとコンテナを移行します。

**<ターゲット>: <文字列**

対象ノード

**-最大労働者数 <整数> (1 - N)**

並列マイグレーションジョブの最大数。設定されていない場合は、datacenter.cfgの'max\_workers'を使用します。両方を設定する必要があります！

**--vms <文字列>**

これらのIDをお持ちのお客様のみご検討ください。

**--ローカルディスクあり <ブール値>**

ローカルディスクのライブストレージ移行を有効にします。

**pvenode startall** [OPTIONS] [OPTIONS]。

このノードにあるすべてのVMとコンテナ（デフォルトではonboot=1のもののみ）を起動します。

**--force <boolean> (デフォルト= off)**

仮想ゲストのonbootが設定されていない、またはoffに設定されてもstartコマンドを発行します。

**--vms <文字列>**

コマセで区切られたVMIDのリストからのゲストのみを考慮します。

**pvenode stopall** [OPTIONS] (ブペノード・ストップオール)

すべてのVMとコンテナを停止します。

**--force-stop <boolean> (デフォルト= 1)**

タイムアウト後の強引なストップ。

**--タイムアウト <整数> (0 - 7200) (デフォルト = 180)**

各ゲストシャットダウンタスクのタイムアウト。強制停止に応じて、シャットダウンは単に中止されるか、ハード停止が強制されます。

**--vms <文字列>**

これらのIDをお持ちのお客様のみご検討ください。

**pvenodeタスクリスト** [OPTIONS] [FORMAT\_OPTIONS]

1つのノードのタスクリスト（終了したタスク）を読み取ります。

**--エラー <ブール値> (デフォルト= 0)**

ステータスが ERROR のタスクのみをリストします。

**--リミット <整数> (0 - N) (デフォルト = 50)**

この量のタスクだけをリストアップしてください。

**--以降 <整数>**

この UNIX エポック以降のタスクのみをリストします。

**--source <active| all| archive> (default = archive)**  
アーカイブされたタスク、アクティブなタスク、すべてのタスクを一覧表示します。

**--start <整数> (0 - N) (デフォルト = 0)**  
このオフセットから始まるタスクをリストします。

**--statusfilter <文字列**  
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（例: vzstart、vzdump）のみをリストアップします。

**<integer> まで**  
このUNIXエポックまでのタスクだけをリストアップします。

**--userfilter <文字列**  
このユーザーのタスクのみをリストアップします。

**--vmid <整数> (100 - 999999999)**  
このVMのタスクのみをリストします。

**pvenode タスクログ <upid> [OPTIONS].**

タスクログを読む

**<upid>: <文字列**  
タスク固有のID。

**--ダウンロード <ブール値**  
タスクログファイルをダウンロードするかどうか。このパラメータは他のパラメータと併用することはできません。

**--start <整数> (0 - N) (デフォルト = 0)**  
タスクログを読むときは、この行から始めます。

**pvenode タスクステータス <upid> [FORMAT\_OPTIONS].**

タスクのステータスを読み取ります。

**<upid>: <文字列**  
タスク固有のID。

**pvenode wakeonlan <ノード>.**  
*wake on LAN* ネットワークパケットでノードをウェイクアップします。

**<ノード>: <文字列**

ウェイクオンLAN/パケットのターゲットノード

## A.8 pvesh - Proxmox VE API 用シェルインターフェイス

**pvesh** <COMMAND> [ARGS] [OPTIONS].

**pvesh create** <api\_path> [OPTIONS] [FORMAT\_OPTIONS] です。

<api\_path>でAPI POSTを呼び出します。

<api\_path>: <文字列

APIパス。

--noproxy <ブール値

自動プロキシを無効にします。

**pvesh delete** <api\_path> [OPTIONS] [FORMAT\_OPTIONS].

<api\_path>でAPI DELETEを呼び出します。

<api\_path>: <文字列

APIパス。

--noproxy <ブール値

自動プロキシを無効にします。

**pvesh get** <api\_path> [OPTIONS] [FORMAT\_OPTIONS].

<api\_path>でAPI GETを呼び出します。

<api\_path>: <文字列

APIパス。

--noproxy <ブール値

自動プロキシを無効にします。

**pvesh help** [OPTIONS]

指定したコマンドに関するヘルプを表示します。

--extra-args <array> です。

特定のコマンドのヘルプを表示します。

--verbose <ブール値

冗長出力フォーマット。

**pvesh ls** <api\_path> [OPTIONS] [FORMAT\_OPTIONS].

<api\_path>上の子オブジェクトを一覧表示します。

<api\_path>: <文字列>

APIパス。

--noproxy <ブール値>

自動プロキシを無効にします。

**pvesh set** <api\_path> [OPTIONS] [FORMAT\_OPTIONS].

<api\_path>でAPI PUTを呼び出します。

<api\_path>: <文字列>

APIパス。

--noproxy <ブール値>

自動プロキシを無効にします。

**pvesh usage** <api\_path> [OPTIONS] を使用します。

<api\_path> の API 使用情報を表示します。

<api\_path>: <文字列>

APIパス。

-コマンド <create| delete| get| set>

APIコマンド。

<boolean>を返します。

返されたスキーマを含みます。

--verbose <ブール値>

冗長出力フォーマット。

## A.9 qm - QEMU/KVM 仮想マシンマネージャー

**qm** <COMMAND> [ARGS] [OPTIONS] です。

qmエージェント

qm guest cmd のエイリアス。

**qm cleanup** <vmid> <clean-shutdown> <guest-requested>.

タップデバイスやvgpusなどのリソースをクリーンアップします。vm のシャットダウンやクラッシュなどの後に呼び出されます。

<vmid>: <整数> (100 - 999999999)

VMの（一意の）ID。

**<クリーンシャットダウン>: <ブール値**

qemuがクリーンにシャットダウンしたかどうかを示します。

**<ゲストリクエスト>: <ブール値**

シャットダウンがゲストによって要求されたか、qmp を介して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>: <整数> (100 - 999999999)**

VMの（一意の）ID。

**<newid>: <整数> (100 - 999999999)**

クローンの VMID。

**--bwlimit <integer> (0 - N) (デフォルト= データセンターまたはストレージ設定からのクローン制限)**

I/O帯域幅制限のオーバーライド（単位: KiB/s）。

**--説明 <文字列>**

新しいVMの説明。

**--format <qcow2 | raw | vmdk>**

ファイルストレージのターゲットフォーマット。フルクローンの場合のみ有効です。

**--full <ブール値>**

すべてのディスクの完全コピーを作成します。これは通常のVMをクローンするときに必ず行われます。VMテンプレートでは、デフォルトでリンクされたクローンを作成しようとします。

**--name <文字列>**

新しいVMの名前を設定します。

**--プール <文字列>**

新しいVMを指定したプールに追加します。

**--スナップネーム <文字列>**

スナップショットの名前。

**--ストレージID**

対象ストレージ。

**--ターゲット <文字列>**

ターゲット・ノード。元のVMが共有ストレージ上にある場合にのみ許可されます。

**qm cloudinit dump <vmid> <type> です。**

自動生成された cloudinit 設定を取得します。

**<vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

**<type>: <meta| network| user>**

コンフィグタイプ。

**qm cloudinit pending <vmid>** です。

cloudinitコンフィギュレーションを現在の値と保留中の値の両方で取得します。

**<vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

**qm cloudinit update <vmid>** です。

cloudinit設定ドライブを再生成して変更します。

**<vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

**qm config <vmid> [OPTIONS].**

保留中の構成変更を適用した仮想マシン構成を取得します。現在のコンフィギュレーションを取得するには、*current* パラメータを設定します。

**<vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

**--current <boolean> (デフォルト= 0)**

保留値ではなく）現在の値を取得します。

**--スナップショット <文字列>**

指定されたスナップショットから設定値を取得します。

**qm create <vmid> [OPTIONS].**

仮想マシンを作成またはリストアします。

**<vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

**--acpi <boolean> (デフォルト= 1)**

ACPI を有効/無効にします。

**--アフィニティ <文字列>**

ゲストプロセスの実行に使用されるホストコアのリスト（例：0,5,8-11

```
--agent [enabled=<1|0> [,freeze-fs-on-backup=<1|0>]  
[,fstrim_cloned_disks=<1|0>] [,type=<virtio|isa>].
```

QEMU ゲストエージェントとそのプロパティとの通信を有効/無効にします。

```
--amd-sev [type=<sev-type>[,allow-smt=<1|0>].  
[,kernel-hashes=<1|0>] [,no-debug=<1|0>] [,no-key-sharing=<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・アーキテクチャ。デフォルトはホスト。

#### --アーカイブ <文字列>

バックアップアーカイブ。.tarまたは.vmaファイルへのファイルシステムのパス（標準入力からデータをパイプする場合は-を使用）、またはproxmoxストレージのバックアップボリューム識別子。

```
--args <文字列>
```

kvm に渡される任意の引数。

```
--audio0 device=<ich9-intel-hda|intel-hda|AC97> [,driver=<spice|none>].
```

QXL/Spiceと組み合わせると便利です。

```
--autostart <boolean> (デフォルト= 0)
```

クラッシュ後の自動再起動（現在は無視）。

```
--バルーン <整数> (0 - N)
```

VM のターゲット RAM の量 (MiB 単位)。ゼロを使用すると、バロンドライバが無効になります。

```
--bios <ovmf| seabios> (デフォルト= seabios)
```

BIOSの実装を選択します。

```
--boot [[legacy=]<[acdn]{1,4}>] [,order=<デバイス[,デバイス...]>]].
```

ゲストの起動順序を指定します。order= サブプロパティを使用してください。

```
--ブートディスク (ide|sata|scsi|virtio)◆d.
```

指定したディスクからの起動を有効にします。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リストア制限)
```

I/O帯域幅制限のオーバーライド（単位: KiB/s）。

```
--CDROM <ボリューム>
```

これは -ide2 オプションのエイリアスです。

```
--cicustom [meta=<volume>] [,network=<volume>] [,user=<volume>] [,vendor=<volume>].
```

cloud-init: 開始時に自動生成されるファイルを置き換えるカスタムファイルを指定します。

## --パスワード

cloud-init: ユーザーに割り当てるパスワード。一般的に、これを使うことは推奨されません。代わりに ssh 鍵を使います。また、古いバージョンの cloud-init はハッシュ化されたパスワードをサポートしていないことに注意しましょう。

## --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、設定されているオペレーティングシステムの種類 (ostype.Linuxではnocloud形式、Windowsではconfigdrive2を使用します。

## --ciupgrade <boolean> (デフォルト= 1)

cloud-init: 最初の起動後にパッケージの自動アップグレードを行います。

## --ciuser <文字列>

cloud-init: イメージに設定されているデフォルトユーザーの代わりに、sshキーとパスワードを変更するユーザー名。

## --コア数 <整数> (1 - N) (デフォルト=1)

ソケットあたりのコア数。

## --cpu [[cputype=<string>] [,flags=<+FLAG[-FLAG...]>] [,hidden=<1|0>]

[,hv-vendor-id=<vendor-id>] となります。 [,hidden=<1|0>] [,hv-vendor-id=<vendor-id>] [,hv-vendor-id=<vendor-id>]

[,phys-bits=<8-64|host>] [,reported-model=<enum> ]。

エミュレートされたCPUタイプ。

## --cpulimit <number> (0 - 128) (デフォルト= 0)

CPU使用量の上限。

## --cpuunits <integer> (1 - 262144) (デフォルト= cgroup v1: 1024, cgroup v2: 100)

VM の CPU ウェイトは、cgroup v2 では [1, 10000] にクランプされます。

## --説明 <文字列>

VM の説明。WebインターフェイスのVMのサマリーに表示されます。コンフィギュレーション・ファイルのコメントとして保存されます。

## --efidisk0 [file=<volume> [,efitype=<2m|4m>] [,format=<enum>] [,import-from=<ソースボリューム>] [,pre-enrolled-keys=<1|0>] [,size=<DiskSize>].

EFIバーを格納するディスクを設定します。STORAGE\_ID:SIZE\_IN\_GiBという特殊な構文を使用して、EFIバーを格納するディスクを割り当てます。

を新しいボリュームにコピーします。SIZE\_IN\_GiBはここでは無視され、代わりにデフォルトのEFIバー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 <ブール値>

既存のVMの上書きを許可します。

---

### 備考

必要なオプション: archive

---

**--freeze <ブール値>**

起動時にCPUをフリーズ（c monitorコマンドで実行開始）。

**--hookscript <文字列>**

vmsのライフタイムの様々なステップで実行されるスクリプト。

```
--hostpci[n] [[host=<HOSTPCIID[,HOSTPCIID2...]>][,device-id=<hex id>][,legacy-igd=<1|0>][,mapping=<mapping-id>][,mdev=<string>][,pcie=<1|0>][,rombar=<1|0>][,romfile=<string>][[host=<HOSTPIID[,HOSTPCIID2...]>]] --hostpci[n[,sub-device-id=<hex id>][,sub-vendor-id=<hex id>][,vendor-id=<hex id>][,x-vga=<1|0>]]
```

ホストのPCIデバイスをゲストにマッピングします。

**--hotplug <string> (デフォルト= ネットワーク,ディスク,USB)**

ホットプラグ機能を選択的に有効にします。これはカンマ区切りのホットプラグ機能のリストです： ネットワーク、ディスク、CPU、メモリ、USB、*cloudinit*。ホットプラグを完全に無効にするには0を使用します。値として1を使用すると、デフォルトの network、disk、usb のエイリアスになります。USB ホットプラグは、マシン>=7.1 および ostype l26 または windows>7.

**--hugepages <1024| 2| any>**

hugepagesメモリの有効/無効。

```
--ide[n] [file=<volume>[,aio=<native|threads|io_uring>][,backup=<1|0>][,bps=<bps>][,bps_max_length=<seconds>][,bps_rd=<bps>][,bps_rd_max_length=<seconds>][,bps_wr=<bps>][,bps_wr_max_length=<seconds>][,cache=<enum>][,cyls=<integer>][,detect_zeroes=<1|0>][,discard=<ignore|on>][,format=<enum>][,heads=<integer>][,import-from=<ソース・ボリューム>][,iops=<iops>][,iops_max=<iops>][,iops_max_length=<seconds>][,iops_rd=<iops>][,iops_rd_max=<iops>][,iops_rd_max_length=<seconds>][,iops_wr=<iops>][,iops_wr_max=<iops>][,iops_wr_max_length=<seconds>][,mbps=<mbps>][,mbps_max=<mbps>][,mbps_rd=<mbps>][,mbps_rd_max=<mbps>][,mbps_wr=<mbps>][,mbps_wr_max=<mbps>][,media=<CDROM|DISK>][,model=<model>][,replicate=<1|0>][,rerror=<ignore|report|stop>][,secs=<integer>][,serial=<serial>][,shared=<1|0>][,size=<DiskSize>][,snapshot=<1|0>][,ssd=<1|0>][,trans=<none|lba|auto>][,werror=<enum>][,wwn=<wwn>]となります。
```

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0～3）。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を使用します。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。

**--import-working-storage <ストレージID>**

インポート時に中間抽出ストレージとして使用されます。デフォルトはソースストレージです。

```
--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。

特別な文字列`dhcp`は、DHCPを使用するIPアドレスに使用でき、その場合、明示的な ゲートウェイは提供されるべきではありません。IPv6では、ステートレス自動設定を使うために特別な文字列`auto`を使うことができます。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=<整数> [,name=<文字列>]** です。

VM間共有メモリ。VM間やホストとの直接通信に便利。

**--keephugepages <boolean> (デフォルト= 0)**

`hugepages`と一緒に使用します。有効にすると、`hugepages`はVMシャットダウン後も削除されず、その後の起動に使用できます。

**--キーボード <da| de| de-ch| en-gb| en-us| es| fi| fr| fr-be  
|fr-ca| fr-ch| hu| is| it| ja| lt| mk| nl| no| pl| pt| pt-br| sl| sv| tr>**

VNC サーバーのキーボードレイアウト。このオプションは一般的には必要なく、ゲストOS内で処理した方が良い場合が多いです。

**--kvm <boolean> (デフォルト= 1)**

KVMハードウェア仮想化の有効/無効を設定します。

**--ライブリストア <ブール値>**

バックグラウンドでインポートまたはリストアしている間、VMを直ちに起動します。

**--ローカルタイム <ブール値>**

リアルタイムクロック (RTC) をローカルタイムに設定します。 これは、`ostype`が以下を示す場合、デフォルトで有効になります。

Microsoft Windows OS。

**--lock <backup| clone| create| migrate| rollback| snapshot | snapshot-delete | suspended | suspending>.**

VMをロック/アンロックします。

**-マシン [[タイプ=<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。

**--メモリ [current=<整数>]**

メモリ特性。

**--migrate\_downtime <number> (0 - N) (デフォルト= 0.1)**

移行の最大許容ダウンタイム (秒) を設定します。マイグレーションが

新しく汚れたRAMを転送する必要がありすぎるため、移行が収束するまで、自動的に段階的に制限を。

**--migrate\_speed <integer> (0 ~ N) (デフォルト= 0)**

マイグレーションの最大速度 (MB/s) を設定します。値 0 は制限なしです。

**--name <文字列>**

VM の名前を設定します。コンフィギュレーションWebインターフェイスでのみ使用します。

**--ネームサーバー <文字列>**

cloud-init: コンテナの DNS サーバー IP アドレスを設定します。searchdomain も nameserver も設定されていない場合、Create は自動的にホストからの設定を使用します。

**--net[n] [model=<enum> [,bridge=<bridge>] [,firewall=<1|0>] [,link\_down=<1|0>] [,macaddr=<XX:XX:XX:XX:XX:>] [,mtu=<integer>] [,queues=<integer>] [,rate=<number>] [,tag=<integer>] [,trunks=<vlanid[;vlanid...]>] [<model>=<macaddr>] [<model>=<macaddr>] [<model>=<macaddr>]**  
ネットワーク機器を指定します。

**--numa <boolean> (デフォルト= 0)**

NUMAの有効/無効。

**--numa[n] cpus=<id[-id];...> [,hostnodes=<id[-id];...>] を指定します。**

**[,memory=<number>] [,policy=<preferred|bind|interleave>] となります。**

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 <l24| l26| other| solaris| w2k| w2k3| w2k8| win10 | win11 | win7 | win8 | wvista | 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/parportd+|/dev/usb/lpd+**

ホスト・パラレル・デバイスをマップします (nは0~2)。

**--プール <文字列>**

VMを指定したプールに追加します。

**--プロテクション <ブール値> (デフォルト= 0)**

VM の保護フラグを設定します。これにより、VM の削除とディスクの削除操作が無効になります。

**--reboot <boolean> (デフォルト= 1)**

再起動を許可。0 に設定すると、VM は時に終了します。

```
--rng0 [source=</dev/urandom|/dev/random|/dev/hwrng> [,max_bytes=<integer>] [,period=<integer>].  
VirtIO ベースの乱数ジェネレータを設定します。
```

```
--sata[n] [[file=<volume> [,aio=<native|threads|io_uring> [,backup=<1|0>] [,bps=<bps>]  
[,bps_max_length=<seconds> [,bps_rd=<bps>] [,bps_rd_max_length=<seconds>] [,bps_wr=<bps>]  
[,bps_wr_max_length=<seconds>] [,cache=<enum>] [,cyls=<integer>] [,detect_zeroes=<1|0>]  
[,discard=<ignore|on>] [,format=<enum>] [,heads=<integer>] [,import-from=<ソースボリューム>]  
[,iops=<iops>] [,iops_max=<iops>] [,iops_max_length=<seconds>] [,iops_rd=<iops>]  
[,iops_rd_max=<iops>] [,iops_rd_max_length=<seconds>] [,iops_wr=<iops>] [,iops_wr_max=<iops>]  
[,iops_wr_max_length=<seconds>] [,mbps=<mbps>] [,mbps_max=<mbps>] [,mbps_rd=<mbps>]  
[,mbps_rd_max=<mbps>] [,mbps_wr=<mbps>] [,mbps_wr_max=<mbps>] [,media=<cdrom|disk>]  
[,replicate=<1|0>] [,rerror=<ignore|report|stop>] [,secs=<integer>] [,serial=<serial>]  
[,shared=<1|0> ]。] [,size=<DiskSize>] [,snapshot=<1|0>] [,ssd=<1|0>] [,trans=<none|lba|auto>]  
[,werror=<enum>] [,wwn=<wwn> ]。
```

ボリュームをSATA/ハードディスクまたはCD-ROMとして使用します (nは0~5)。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。既存のボリュームからインポートするには、`STORAGE_ID:0`と`import-from`パラメータを使用します。

```
--scsi[n] [[file=<volume> [,aio=<native|threads|io_uring> [,backup=<1|0>] [,bps=<bps>]  
[,bps_max_length=<seconds>] [,bps_rd=<bps>] [,bps_rd_max_length=<seconds>] [,bps_wr=<bps>]  
[,bps_wr_max_length=<seconds>] [,cache=<enum>] [,cyls=<integer>] [,detect_zeroes=<1|0>]  
[,discard=<ignore|on>] [,format=<enum>] [,heads=<integer>] [,import-from=<ソースボリューム>]  
[,iops=<iops>] [,iops_max=<iops>] [,iops_max_length=<seconds>] [,iops_rd=<iops>]  
[,iops_rd_max=<iops>] [,iops_rd_max_length=<seconds>] [,iops_wr=<iops>] [,iops_wr_max=<iops>]  
[,iops_wr_max_length=<seconds>] [,iothread=<1|0>] [,mbps=<mbps>] [,mbps_max=<mbps>]  
[,mbps_rd=<mbps>] [,mbps_rd_max=<mbps>] [,mbps_wr=<mbps>] [,mbps_wr_max=<mbps>]  
[,media=<cdrom|disk>] [,product=<product>] [,queues=<integer>] [,replicate=<1|0>]  
[,rerror=<ignore|report|stop>] [,ro=<1|0>] [,scsiblock=<1|0>] [,secs=<integer>]  
[,serial=<serial>] [,shared=<1|0> ]。] [,size=<DiskSize>] [,snapshot=<1|0>] [,ssd=<1|0>]  
[,trans=<none|lba|auto>] [,vendor=<vendor>] [,werror=<enum>] [,wwn=<wwn>] となります。
```

ボリュームをSCSI/ハードディスクまたはCD-ROMとして使用します (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` を使用します。新しいボリュームを割り当てるには、`STORAGE_ID:0`と`import-from`パラメータを使用します。  
設定します。

```
--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
```

**--検索ドメイン <文字列>**

cloud-init: コンテナのDNS検索ドメインを設定します。searchdomainもnameserverも設定されていない場合、Createはホストからの設定を自動的に使用します。

**-シリアル[n] (/dev/.+|socket)**

VM内にシリアル・デバイスを作成 (nは0~3)

**--shares <integer> (0 - 50000) (デフォルト=1000)**

オート・バルーニングのメモリ・シェア量。この数値が大きいほど、このVMはより多くのメモリを取得します。数値は、他のすべての実行中のVMの重みに対する相対値です。ゼロを使用すると、オートバルーニングは無効になります。オートバルーニングはpvestatdによって実行されます。

**--smbios1 [base64=<1|0>] [,family=<Base64エンコードされた文字列>] [,manufacturer=<Base64エンコードされた文字列>] [,product=<Base64エンコードされた文字列>] [,serial=<Base64エンコードされた文字列>] [,sku=<Base64エンコードされた文字列>] [,uuid=<UUID>] [,version=<Base64エンコードされた文字列>] となります。**

SMBIOSタイプ1のフィールドを指定します。

**--smp <整数> (1 - N) (デフォルト=1)**

CPU数。代わりに -sockets オプションを使用してください。

**--ソケット <整数> (1 - N) (デフォルト=1)**

CPUソケットの数。

**--spice\_enhancements [folderssharing=<1|0>] [,videostreaming=<off|all|filter>]。**

SPICE の追加機能拡張を設定します。

**--sshkeys <ファイルパス>**

cloud-init: 公開 SSH 鍵を設定します (1 行に 1 つの鍵、OpenSSH 形式)。

**--start <boolean> (デフォルト= 0)**

VMの作成に成功したら、VMを起動します。

**--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。有効な日付の書式は、'now' または 2006-06-17T16:01:21 または 2006-06-17.

**--startup `[[order]=\d+] [,up=d+] [,down=d+`**

スタートアップとシャットダウンの動作。Orderは一般的な起動順序を定義する非負の数値です。シャットダウンは逆の順序で行われます。さらに、次のVMが起動または停止するまでの待ち時間を秒単位で設定できます。

**--ストレージID**

デフォルトのストレージ。

--tablet <boolean> (デフォルト= 1)  
USBタブレットデバイスを有効/無効にします。

--タグ <文字列>  
VM のタグ。これは単なるメタ情報です。

--tdf <ブール値> (デフォルト= 0)  
時間ドリフト有効/無効。

--テンプレート <ブール値> (デフォルト= 0)  
テンプレートの有効/無効。

--tpmstate0 [<ファイル=><ボリューム> [, インポート元=<ソースボリューム>] [, サイズ=<ディスクサイズ>] [, バージョン=<v1.2|v2.0>]。

TPM の状態を保存するDiskを設定します。フォーマットはrawに固定されています。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。SIZE\_IN\_GiBはここでは無視され、代わりに4MiBが使用されることに注意してください。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 <ブール値>  
一意のランダムなイーサネットアドレスを割り当てます。

---

#### 備考

必要なオプション: archive

---

--unused[n] [<file=><volume>]  
未使用ボリュームへの参照。これは内部的に使用されるので、手動で変更しないでください。

--usb[n] [[host=]<HOSTUSBDEVICE|spice>]の場合 [, マッピング=<マッピングID>] [, usb3=<1|0>]  
USBデバイスを設定します (nは0~4、マシンのバージョンが7.1以上、かつostype l26またはwindows > 7の場合、nは最大14まで)。

--vcpus <整数> (1 ~ N) (デフォルト= 0)  
ホットプラグされたvcpusの数。

--vga [[タイプ=<enum>]] [[クリップボード=<vnc[クリップボード=<vnc>] [, メモリ=<整数>]

VGAハードウェアを設定します。

---

```
--virtio[n] [file=<volume> [,aio=<native|threads|io_uring>] [,backup=<1|0>] [,bps=<bps>]
[,bps_max_length=<seconds> [,bps_rd=<bps>] [,bps_rd_max_length=<seconds>] [,bps_wr=<bps>]
[,bps_wr_max_length=<seconds>] [,cache=<enum>] [,cyls=<integer>] [,detect_zeroes=<1|0>]
[,discard=<ignore|on>] [,format=<enum>] [,heads=<integer>] [,import-from=<ソース・ボリューム>]
[,iops=<iops>] [,iops_max=<iops>] [,iops_max_length=<seconds>] [,iops_rd=<iops>]
[,iops_rd_max=<iops>] [,iops_rd_max_length=<seconds>] [,iops_wr=<iops>] [,iops_wr_max=<iops>]
[,iops_wr_max_length=<seconds>] [,iothread=<1|0>] [,mbps=<mbps>] [,mbps_max=<mbps>]
[,mbps_rd=<mbps>] [,mbps_rd_max=<mbps>] [,mbps_wr=<mbps>] [,mbps_wr_max=<mbps>]
[,media=<cdrom|disk>] [,replicate=<1|0>] [,rerror=<ignore|report|stop>] [,ro=<1|0>]
[,secs=<integer>] [,serial=<serial>] [,serial=<serial>] [,shared=<1|0>] [,size=<DiskSize>>]
[,snapshot=<1|0>] [,trans=<none|lba|auto>] [,werror=<enum>] [,secs=<integer>>]のように指定します。
```

ボリュームをVIRTIOハードディスクとして使用します (nは0~15)。新しいボリュームを割り当てるには、特別な構文 STORAGE\_ID:SIZE\_IN\_GiB を使用します。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。

```
--virtiofs[n] [dirid=<mapping-id> [,cache=<enum>].
[,direct-io=<1|0>] [,expose-acl=<1|0>] [,expose-xattr=<1|0>] 。
```

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生成IDを設定します。作成または更新時に自動生成する場合は 1 を使用し、明示的に無効にする場合は 0 を渡します。

**--vmstatestorage <ストレージID>**

VM 状態ボリューム/ファイルのデフォルトストレージ。

**--watchdog [[model=<i6300esb|ib700>][アクション]**

仮想ハードウェア・ウォッチドッグ・デバイスを作成します。

**qm delsnapshot <vmid> <snapname> [OPTIONS]**

VMスナップショットを削除します。

**<vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

**<スナップ名>: <文字列>**

スナップショットの名前。

**--force <ブール値>**

ディスクスナップショットの削除に失敗した場合でも、設定ファイルから削除できます。

**qm destroy <vmid> [OPTIONS]**

VM とすべての使用済み/所有ボリュームを破棄します。VM固有の権限とファイアウォールルールを削除します。

**<vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

**--destroy-unreferenced-disks <boolean> (デフォルト= 0)**

設定されている場合、有効になっているすべてのストレージから、VMIDが一致し、設定内で参照されていないすべてのディスクを追加で破棄します。

**--ページ <ブール値**

バックアップ＆レプリケーションジョブやHAなどの構成からVMIDを削除します。

**--スキップロック <ブール値**

Ignore locks - rootのみがこのオプションを使用できます。

**qm disk import <vmid> <source> <storage> [OPTIONS].**

VM の未使用ディスクとして外部ディスクイメージをインポート。イメージフォーマットは qemu-img(1) でサポートされている必要があります。

**<vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

**<ソース>: <文字列**

インポートするディスクイメージへのパス

**<ストレージ>: <ストレージID**

対象ストレージID

**--format <qcow2| raw| vmdk>**

対象フォーマット

```
--target-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| unused0| unused1| unused10| unused100|  
unused101| unused102| unused103  
| 未使用104| 未使用105| 未使用106| 未使用107| 未使用108| 未使用109| 未使用11| 未使用110| 未使用111| 未使用112| 未  
使用113| 未使用114| 未使用115| 未使用116| 未使用117| 未使用118| 未使用119| 未使用12| 未使用120| 未使用121| 未使用  
122| 未使用123| 未使用124| 未使用125| 未使用126| 未使用127| 未使用128| 未使用129| 未使用13| 未使用130| 未  
使用131| 未使用132| 未使用133| 未使用134| 未使用135| 未使用136| 未使用137| 未使用138| 未使用139| 未使用14|  
未使用140| 未使用141| 未使用142| 未使用143| 未使用144| 未使用145| 未使用146| 未使用147| 未使用148| 未使用  
149| 未使用15| 未使用150| 未使用151| unused152| unused153| unused154| unused155| unused156|  
unused157| unused158| unused159| unused16| unused160| unused161| unused162| unused163|  
unused164| unused165| unused166| unused167| unused168| unused169| unused17| unused170|  
unused171| unused172| unused173| unused174| unused175| 未使用176円| 未使用177円| 未使用178円|  
未使用179円| 未使用18円| 未使用180円| 未使用181円| 未使用182円| 未使用183円| 未使用184円| 未使用185円| 未使用  
186円| 未使用187円| 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名| 未使用237名| 未使用238名| 未使用239名| 未使用24名| 未使用240名| 未使用241名| 未使用  
242名| 未使用243名| 未使用244名| 未使用245名| 未使用246名| 未使用247名| 未使用248名| 未使用未使用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5:未使用50:未使用51:未  
使用52:未使用53:未使用54:未使用55:未使用56:未使用57:未使用58:未使用59:未使用6:未使用60:未使用61:未使用62:未  
使用63:未使用64:未使用65:未使用66:未使用67:未使用68:未使用69:未使用7:未使用70:未使用71:未使用72:未使用  
未使用73| 未使用74| 未使用75| 未使用76| 未使用77| 未使用78| 未使用
```

---

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9| 未使用90| 未使用91| 未使用92| 未使用93| 未使用

ボリュームのインポート先のディスク名（例: scsi1）。

**qm disk move** <vmid> <disk> [<storage>] [OPTIONS].

ボリュームを別のストレージまたは別のVMに移動します。

<vmid>: <整数> (100 - 999999999)

VMの（一意の）ID。

<デイスク>: <efidisk0 | ide0 | ide1 | ide2 | ide3 | sata0 | sata1 | sata2 | sata3 | sata4 | sata5 | scsi0 | scsi1 | scsi10 | scsi11 | scsi12 | scsi13 | scsi14 | scsi15 | scsi16 | scsi17 | scsi18 | scsi19 | scsi2 | scsi20 | scsi21 | scsi22 | scsi23 | scsi24 | scsi25 | scsi26 | scsi27 | scsi28 | scsi29 | scsi3 | scsi30 | scsi4 | scsi5 | scsi6 | scsi7 | scsi8 | scsi9 | tpmstate0 | unused0 | unused1 | unused10 | unused100 | unused101 | unused102 | unused103 | 未使用104 | 未使用105 | 未使用106 | 未使用107 | 未使用108 | 未使用109 | 未使用11 | 未使用110 | 未使用111 | 未使用112 | 未使用113 | 未使用114 | 未使用115 | 未使用116 | 未使用117 | 未使用118 | 未使用119 | 未使用12 | 未使用120 | 未使用121 | 未使用122 | 未使用123 | 未使用124 | 未使用125 | 未使用126 | 未使用127 | 未使用128 | 未使用129 | 未使用13 | 未使用130 | 未使用131 | 未使用132 | 未使用133 | 未使用134 | 未使用135 | 未使用136 | 未使用137 | 未使用138 | 未使用139 | 未使用14 | 未使用140 | 未使用141 | 未使用142 | 未使用143 | 未使用144 | 未使用145 | 未使用146 | 未使用147 | 未使用148 | 未使用149 | 未使用15 | 未使用150 | 未使用151 | unused152 | unused153 | unused154 | unused155 | unused156 | unused157 | unused158 | unused159 | unused16 | unused160 | unused161 | unused162 | unused163 | unused164 | unused165 | unused166 | unused167 | unused168 | unused169 | unused17 | unused170 | unused171 | unused172 | unused173 | unused174 | unused175 | 未使用176円 | 未使用177円 | 未使用178円 | 未使用179円 | 未使用18円 | 未使用180円 | 未使用181円 | 未使用182円 | 未使用183円 | 未使用184円 | 未使用185円 | 未使用186円 | 未使用187円 | 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229 | 未使用235 | 未使用231 | 未使用232 | 未使用233 | 未使用234 | 未使用235 | 未使用236名 | 未使用237名 | 未使用238名 | 未使用239名 | 未使用24名 | 未使用240名 | 未使用241名 | 未使用242名 | 未使用243名 | 未使用244名 | 未使用245名 | 未使用246名 | 未使用247名 | 未使用248名 | 未使用未使用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5 | 未使用50 | 未使用51 | 未使用52 | 未使用53 | 未使用54 | 未使用55 | 未使用56 | 未使用57 | 未使用58 | 未使用59 | 未使用6 | 未使用60 | 未使用61 | 未使用62 | 未使用63 | 未使用64 | 未使用65 | 未使用66 | 未使用67 | 未使用68 | 未使用69 | 未使用7 | 未使用70 | 未使用71 | 未使用72 | 未使用  
未使用73 | 未使用74 | 未使用75 | 未使用76 | 未使用77 | 未使用78 | 未使用

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9 | 未使用90 | 未使用91 | 未使用92 | 未使用93 | 未使用 |

移動したいディスク。

**<ストレージ>: <ストレージID**

対象のストレージ

**--bwlimit <integer> (0 - N) (デフォルト= データセンターまたはストレージ設定からの移動制限)**

I/O帯域幅制限のオーバーライド（単位: KiB/s）。

**--delete <boolean> (デフォルト= 0)**

コピー成功後、元のディスクを削除します。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。

**-ダイジェスト <文字列>**

現在の設定ファイルのSHA1" . "ダイジェストが異なる場合、変更を防止します。これは、同時変更を防ぐために使用できます。

**--format <qcow2| raw| vmdk>**

対象フォーマット

**-ターゲット・ダイジェスト <文字列>**

ターゲットVMの現在のコンフィグファイルのSHA1ダイジェストが" . "異なる場合、変更を防止します。これは、同時変更を検出するために使用できます

。

```
--target-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| unused0| unused1| unused10| unused100|  
unused101| unused102| unused103  
| 未使用104| 未使用105| 未使用106| 未使用107| 未使用108| 未使用109| 未使用11| 未使用110| 未使用111| 未使用112| 未  
使用113| 未使用114| 未使用115| 未使用116| 未使用117| 未使用118| 未使用119| 未使用12| 未使用120| 未使用121| 未使用  
122| 未使用123| 未使用124| 未使用125| 未使用126| 未使用127| 未使用128| 未使用129| 未使用13| 未使用130| 未  
使用131| 未使用132| 未使用133| 未使用134| 未使用135| 未使用136| 未使用137| 未使用138| 未使用139| 未使用14|  
未使用140| 未使用141| 未使用142| 未使用143| 未使用144| 未使用145| 未使用146| 未使用147| 未使用148| 未使用  
149| 未使用15| 未使用150| 未使用151| unused152| unused153| unused154| unused155| unused156|  
unused157| unused158| unused159| unused16| unused160| unused161| unused162| unused163|  
unused164| unused165| unused166| unused167| unused168| unused169| unused17| unused170|  
unused171| unused172| unused173| unused174| unused175| 未使用176円| 未使用177円| 未使用178円|  
未使用179円| 未使用18円| 未使用180円| 未使用181円| 未使用182円| 未使用183円| 未使用184円| 未使用185円| 未使用  
186円| 未使用187円| 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名| 未使用237名| 未使用238名| 未使用239名| 未使用24名| 未使用240名| 未使用241名| 未使用  
242名| 未使用243名| 未使用244名| 未使用245名| 未使用246名| 未使用247名| 未使用248名| 未使用未使用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5:未使用50:未使用51:未  
使用52:未使用53:未使用54:未使用55:未使用56:未使用57:未使用58:未使用59:未使用6:未使用60:未使用61:未使用62:未  
使用63:未使用64:未使用65:未使用66:未使用67:未使用68:未使用69:未使用7:未使用70:未使用71:未使用72:未使用  
未使用73| 未使用74| 未使用75| 未使用76| 未使用77| 未使用78| 未使用
```

---

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9| 未使用90| 未使用91| 未使用92| 未使用93| 未使用

ターゲット VM 上でディスクが移動される設定キー (例えば、ide0 または scsi1)。デフォルトはソースディスクキーです。

#### --ターゲット-vmid <整数> (100 - 999999999)

VMの（一意の）ID。

#### qm disk rescan [OPTIONS] (ディスク再スキャン)

すべてのストレージを再スキャンし、ディスクサイズと未使用ディスク更新します。

#### --dryrun <boolean> (デフォルト= 0)

実際にVMコンフィグに変更を書き出さないでください。

#### --vmid <整数> (100 - 999999999)

VMの（一意の）ID。

#### qm disk resize <vmid> <disk> <size> [OPTIONS].

ボリュームサイズを拡張します。

#### <vmid>: <整数> (100 - 999999999)

VMの（一意の）ID。

<ディスク>: <efidisk0 | ide0 | ide1 | ide2 | ide3 | sata0 | sata1 | sata2 | sata3 | sata4 | sata5 | scsi0 | scsi1 | scsi10 | scsi11 | scsi12 | scsi13 | scsi14 | scsi15 | scsi16 | scsi17 | scsi18 | scsi19 | scsi2 | scsi20 | scsi21 | scsi22 | scsi23 | scsi24 | scsi25 | scsi26 | scsi27 | scsi28 | scsi29 | scsi3 | scsi30 | scsi4 | scsi5 | scsi6 | scsi7 | scsi8 | scsi9 | tpmstate0 | virtio0 | virtio1 | virtio10 | virtio11 | virtio12 | virtio13 | virtio14 | virtio15 | virtio2 | virtio3 | virtio4 | virtio5 | virtio6 | virtio7 | virtio8 | virtio9>

サイズを変更したいディスク。

#### <サイズ>です: \+?\d+(\.\d+)?[KMGT]?

新しいサイズ。+記号を付けると、値はボリュームの実際のサイズに加算され、付けないと、値は絶対値とみなされます。ディスク・サイズの縮小はサポートされていません。

#### -ダイジェスト <文字列>

現在の設定ファイルの SHA1 ダイジェストが異なる場合に変更を防止します。これは、同時変更を防ぐために使用できます。

#### --スキップロック <ブール値>

Ignore locks - rootのみがこのオプションを使用できます。

#### qm disk unlink <vmid> --idlist <string> [OPTIONS].

ディスクイメージのリンク解除/削除

**<vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

**--force <ブール値>**

物理的な削除を強制します。これがないと、単純に設定ファイルからディスクを削除し、ボリュームIDを含む`unused[n]`という追加の設定エントリを作成します。`unused[n]`のリンク解除は、常に物理的な削除を引き起こします。

**--idlist <文字列>**

削除したいディスク ID のリスト。

**qm guest cmd <vmid> <command>**

QEMU Guest Agentのコマンドを実行します。

**<vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

**<コマンド>: <fsfreeze-freeze| fsfreeze-status| fsfreeze-thaw | fstrim | get-fsinfo | get-host-name | get-memory-block-info | get-memory-blocks | get-osinfo | get-time | get-timezone | get-users| get-vcpus| info| network-get-interfaces| ping | shutdown | suspend-disk | suspend-hybrid | suspend-ram> です。**

QGAコマンド。

**qm guest exec <vmid> [<extra-args>] [OPTIONS].**

ゲストエージェント経由で指定されたコマンドを実行します。

**<vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

**<extra-args>: <array> です。**

配列としての追加引数

**--pass-stdin <boolean> (デフォルト= 0)**

設定されると、EOFまで STDIN を読み込み、`input-data` 経由でゲストエージェントに転送します（通常、ゲストエージェントによって起動されるプロセスの STDIN として扱われます）。最大 1MiB を許可します。

**--同期 <boolean> (デフォルト= 1)**

`off` に設定すると、コマンドの終了やタイムアウトを待たずに、即座に pid を返します。

**--タイムアウト <整数> (0 - N) (デフォルト= 30)**

コマンドが終了するまで同期的に待つ最大時間。到達した場合、pidが返されます。無効にするには0を設定します。

**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>: <整数> (100 - 999999999)**

VMの（一意の）ID。

**<pid>: <整数>**

問い合わせるPID

**qm guest passwd <vmid> <username> [OPTIONS].**

指定されたユーザのパスワードを指定されたパスワードに設定します。

**<vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

**<ユーザー名>: <文字列>**

パスワードを設定ユーザー。

**--暗号化 <ブール値> (デフォルト= 0)**

パスワードがすでに crypt() を通過している場合は 1 を設定します。

**qm help [OPTIONS]**

指定したコマンドに関するヘルプを表示します。

**--extra-args <array> です。**

特定のコマンドのヘルプを表示します。

**--verbose <ブール値>**

冗長な出力形式。

**qm import <vmid> <source> --storage <string> [OPTIONS].**

ESXi ストレージなど、サポートされているインポートソースから外固の仮想ゲストをインポートします。

**<vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

**<ソース>: <文字列>**

インポート元のボリューム ID。

**--acpi <boolean> (デフォルト= 1)**

ACPI を有効/無効にします。

**--アフィニティ <文字列>**

ゲストプロセスの実行に使用されるホストコアのリスト（例：0,5,8-11

--agent [enabled=<1|0> [,freeze-fs-on-backup=<1|0>]  
[,fstrim\_cloned\_disks=<1|0>] [,type=<virtio|isa>].

QEMU ゲストエージェントとそのプロパティとの通信を有効/無効にします。

--amd-sev [type=<sev-type>[,allow-smt=<1|0>]  
[,kernel-hashes=<1|0>] [,no-debug=<1|0>] [,no-key-sharing=<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・アーキテクチャ。デフォルトはホスト。

--args <文字列>

kvm に渡される任意の引数。

--audio0 device=<ich9-intel-hda|intel-hda|AC97> [,driver=<spice|none>].

QXL/Spiceと組み合わせると便利です。

--autostart <boolean> (デフォルト= 0)

クラッシュ後の自動再起動 (現在は無視されています)。

--バーレーン <整数> (0 - N)

VM のターゲット RAM の量 (MiB 単位)。ゼロを使用すると、バロンドライバが無効になります。

--bios <ovmf| seabios> (デフォルト= seabios)

BIOSの実装を選択します。

--boot [[legacy=]<[acdn]{1,4}>] [,order=<デバイス[:デバイス...]>]].

ゲストの起動順序を指定します。order= サブプロパティを使用してください。

--ブートディスク (ide|sata|scsi|virtio)◆d+.

指定したディスクからの起動を有効にします。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 -ide2 オプションのエイリアスです。

--cicustom [meta=<volume>] [,network=<volume>] [,user=<volume>] [,vendor=<volume>].

cloud-init: 開始時に自動生成されるファイルを置き換えるカスタムファイルを指定します。

-パスワード <文字列>

cloud-init: ユーザーに割り当てるパスワード。一般的に、これを使うことは推奨されません。代わりに ssh 鍵を使います。また、古いバージョンの cloud-init はハッシュ化されたパスワードをサポートしていないことに注意しましょう。

--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、設定されているオペレーティングシステムの種類 (ostype.Linuxではnocloud形式、Windowsではconfigdrive2を使用します)。

**--ciupgrade <boolean> (デフォルト= 1)**

cloud-init: 最初の起動後にパッケージの自動アップグレードを行います。

**--ciuser <文字列>**

cloud-init: イメージに設定されているデフォルトユーザーの代わりに、sshキーとパスワードを変更するユーザー名。

**--コア数 <整数> (1 - N) (デフォルト=1)**

ソケットあたりのコア数。

**--cpu [[cputype=<string>] [,flags=<+FLAG[-FLAG...]>] [,hidden=<1|0>]**

[,hv-vendor-id=<vendor-id>] となります。[,hidden=<1|0>] [,hv-vendor-

id=<vendor-id>] [,hv-vendor-id=<vendor-id

,phys-bits=<8-64|host>] [,reported-model=<enum> ]。

エミュレートされたCPUタイプ。

**--cpulimit <number> (0 - 128) (デフォルト= 0)**

CPU使用量の上限。

**--cpuunits <integer> (1 - 262144) (デフォルト= cgroup v1: 1024, cgroup v2: 100)**

VM の CPU ウェイトは、cgroup v2 では [1, 10000] にクランプされます。

**--削除 <文字列>**

削除したい設定のリスト。

**--説明 <文字列>**

VM の説明。WebインターフェイスのVMのサマリーに表示されます。コンフィギュレーション・ファイルのコメントとして保存されます。

**--dryrun <boolean> (デフォルト= 0)**

createコマンドを表示し、何もせずに終了します。

**--efidisk0 [file=<volume> [,efitype=<2m|4m>] [,format=<enum>] [,pre-enrolled-keys=<1|0>] [,size=<DiskSize>].**

EFIバーを保存するディスクを設定します。

**--format <qcow2| raw| vmdk>**

対象フォーマット

**--フリーズ <ブール値>**

起動時にCPUをフリーズ (c monitorコマンドで実行開始)。

**--hookscript <文字列>**

vmsのライフタイムの様々なステップで実行されるスクリプト。

```
--hostpci[n] [[host=<HOSTPCIID[,HOSTPCIID2...]>][,device-id=<hex id>][,legacy-igd=<1|0>]
[,mapping=<mapping-id>][,mdev=<string>][,pcie=<1|0>][,rombar=<1|0>][,romfile=<string>]
[[host=<HOSTPIID[,HOSTPCIID2...]>]] --hostpci[n
[,sub-device-id=<hex id>][,sub-vendor-id=<hex id>][,vendor-id=<hex
id>][,x-vga=<1|0>]。
```

ホストのPCIデバイスをゲストにマッピングします。

#### --hotplug <string> (デフォルト= ネットワーク,ディスク,USB)

ホットプラグ機能を選択的に有効にします。これはカンマ区切りのホットプラグ機能のリストです: ネットワーク、ディスク、CPU、メモリ、USB、*cloudinit*。ホットプラグを完全に無効にするには0を使用します。値として1を使用すると、デフォルトの network、disk、usb のエイリアスになります。USB ホットプラグは、マシン>=7.1 および ostype l26 または windows>7.

#### --hugepages <1024| 2| any>

hugepagesメモリの有効/無効。

```
--ide[n] [file=<volume>[,aio=<native|threads|io_uring>][,backup=<1|0>][,bps=<bps>]
[,bps_max_length=<seconds>][,bps_rd=<bps>][,bps_rd_max_length=<seconds>][,bps_wr=<bps>]
[,bps_wr_max_length=<seconds>][,cache=<enum>][,cyls=<integer>][,detect_zeroes=<1|0>]
[,discard=<ignore|on>][,format=<enum>][,heads=<integer>][,iops=<iops>][,iops_max=<iops>]
[,iops_max_length=<seconds>][,iops_rd=<iops>][,iops_rd_max=<iops>][,iops_rd_max_length=<秒
>][,iops_wr=<iops>][,iops_wr_max=<iops>][,iops_wr_max_length=<seconds>][,mbps=<mbps>]
[,mbps_max=<mbps>][,mbps_rd=<mbps>][,mbps_rd_max=<mbps>][,mbps_wr=<mbps>]
[,mbps_wr_max=<mbps>][,media=<cdrom|disk>][,model=<model>][,replicate=<1|0>]
[,rerror=<ignore|report|stop>][,secs=<integer>][,serial=<serial>][,shared=<1|0>]
[,size=<Disk>][,size=<DiskSize>][,snapshot=<1|0>][,ssd=<1|0>][,trans=<none|lba|auto>]
[,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0~3)。

#### --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。

特別な文字列*dhcp*は、DHCPを使用するIPアドレスに使用でき、その場合、明示的な ゲートウェイは提供されるべきではありません。IPv6では、ステートレス自動設定を使うために特別な文字列*auto*を使うことができます。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=<整数> [,name=<文字列>] です。

VM間共有メモリ。VM間やホストとの直接通信に便利。

**--keephugepages <boolean> (デフォルト= 0)**

hugepagesと一緒に使用します。有効にすると、hugepagesはVMシャットダウン後も削除されず、その後の起動に使用できます。

**--キーボード <da| de| de-ch| en-gb| en-us| es| fi| fr| fr-be****|fr-ca| fr-ch| hu| is| it| ja| lt| mk| nl| no| pl| pt| pt-br| sl| sv| tr>**

VNC サーバーのキーボードレイアウト。このオプションは一般的には必要なく、ゲストOS内で処理した方が良い場合が多いです。

**--kvm <boolean> (デフォルト= 1)**

KVMハードウェア仮想化の有効/無効を設定します。

**--live-import <boolean> (デフォルト= 0)**

すぐにVMを起動し、バックグラウンドでデータをコピーします。

**--ローカルタイム <ブール値>**

リアルタイムクロック (RTC) をローカルタイムに設定します。これは、ostypeが以下を示す場合、デフォルトで有効になります。

Microsoft Windows OS。

**--lock <backup| clone| create| migrate| rollback| snapshot | snapshot-delete | suspended | suspending>。**

VMをロック/アンロックします。

**-マシン [[タイプ=<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。

**--メモリ [current=<整数>**

メモリ特性。

**--migrate\_downtime <number> (0 - N) (デフォルト= 0.1)**

マイグレーションの最大許容ダウンタイム (秒) を設定します。新しく汚れたRAMを転送する必要があるため、マイグレーションが最後まで収束しない場合、マイグレーションが収束するまで、この上限は自動的に段階的に増加します。

**--migrate\_speed <integer> (0 - N) (デフォルト= 0)**

マイグレーションの最大速度 (MB/s) を設定します。値 0 は制限なしです。

**--name <文字列>**

VM の名前を設定します。コンフィギュレーションWebインターフェイスでのみ使用します。

**--ネームサーバー <文字列>**

cloud-init: コンテナの DNS サーバー IP アドレスを設定します。searchdomain も nameserver も設定されていない場合、Create は自動的にホストからの設定を使用します。

```
--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>] [,<model>=<macaddr>] [,<model>=<macaddr>] ネットワーク機器を指定します。
```

--numa <boolean> (デフォルト= 0)  
NUMAの有効/無効。

```
--numa[n] cpus=<id[-id];...> [,hostnodes=<id[-id];...>] を指定します。  
[,memory=<number>] [,policy=<preferred|bind|interleave>] となります。  
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 <l24| 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/parportd+|/dev/usb/lpd+  
ホスト・パラレル・デバイスをマップします (nは0~2)。

--プロテクション <ブール値> (デフォルト= 0)  
VM の保護フラグを設定します。これにより、VM の削除とディスクの削除操作が無効になります。

--reboot <boolean> (デフォルト= 1)  
再起動を許可。0に設定すると、VM は時に終了します。

```
--rng0 [source=</dev/urandom|/dev/random|/dev/hwrng> [,max_bytes=<integer>] [,period=<integer>]. VirtIO ベースの乱数ジェネレータを設定します。
```

```
--sata[n] [file=<volume> [,aio=<native|threads|io_uring>] [,backup=<1|0>] [,bps=<bps>]
[,bps_max_length=<seconds>] [,bps_rd=<bps>] [,bps_rd_max_length=<seconds>] [,bps_wr=<bps>]
[,bps_wr_max_length=<seconds>] [,cache=<enum>] [,cyls=<integer>] [,detect_zeroes=<1|0>]
[,discard=<ignore|on>] [,format=<enum>] [,heads=<integer>] [,iops=<iops>] [,iops_max=<iops>]
[,iops_max_length=<seconds>] [,iops_rd=<iops>] [,iops_rd_max=<iops>] [,iops_rd_max_length=<秒
>] [,iops_wr=<iops>] [,iops_wr_max=<iops>] [,iops_wr_max_length=<seconds>] [,mbps=<mbps>]
[,mbps_max=<mbps>] [,mbps_rd=<mbps>] [,mbps_rd_max=<mbps>] [,mbps_wr=<mbps>]
[,mbps_wr_max=<mbps>] [,media=<cdrom|disk>] [,replicate=<1|0>] [,rerror=<ignore|report|stop>]
[,secs=<integer>] [,serial=<serial>] [,shared=<1|0>] [,size=<DiskSize>] [,snapshot=<1|0>]
[,ssd=<1|0>] [,trans=<none|lba|auto>] [,werror=<enum>] [,wwn=<wwn> ]。
```

ボリュームをSATA/ハードディスクまたはCD-ROMとして使用します (nは0~5)。

```
--scsi[n] [file=<volume> [,aio=<native|threads|io_uring>] [,backup=<1|0>] [,bps=<bps>]
[,bps_max_length=<seconds>] [,bps_rd=<bps>] [,bps_rd_max_length=<seconds>] [,bps_wr=<bps>]
[,bps_wr_max_length=<seconds>] [,cache=<enum>] [,cyls=<integer>] [,detect_zeroes=<1|0>]
[,discard=<ignore|on>] [,format=<enum>] [,heads=<integer>] [,iops=<iops>] [,iops_max=<iops>]
[,iops_max_length=<seconds>] [,iops_rd=<iops>] [,iops_rd_max=<iops>]
[,iops_rd_max_length=<seconds>] [,iops_wr=<iops>] [,iops_wr_max=<iops>]
[,iops_wr_max_length=<seconds>] [,iothread=<1|0>] [,mbps=<mbps>] [,mbps_max=<mbps>]
[,mbps_rd=<mbps>] [,mbps_rd_max=<mbps> ]。 [,mbps_wr=<mbps>] [,mbps_wr_max=<mbps>]
[,media=<cdrom|disk>] [,product=<product>] [,queues=<integer>] [,replicate=<1|0>]
[,rerror=<ignore|report|stop>] [,ro=<1|0>] [,scsiblock=<1|0>] [,secs=<integer>]
[,serial=<serial>] [,shared=<1|0>] [,size=<DiskSize>] [,snapshot=<1|0>] [,ssd=<1|0>]
[,trans=<none|lba|auto>] [,vendor=<vendor>] [,werror=<enum>] [,wwn=<wwn>] となります。
```

ボリュームをSCSI/ハードディスクまたはCD-ROMとして使用します (nは0~30)。

```
--scsihw <lsi| lsi53c810| megasas| pvscsi| virtio-scsi-pci | virtio-scsi-single> (デフォルト
= lsi)
SCSIコントローラモデル
```

#### --検索ドメイン <文字列>

cloud-init: コンテナのDNS検索ドメインを設定します。searchdomainもnameserverも設定されていない場合、Createはホストからの設定を自動的に使用します。

#### -シリアル[n] (/dev/.+|socket)

VM内にシリアル・デバイスを作成 (nは0~3)

--shares <integer> (0 - 50000) (デフォルト=1000)

オート・バルーニングのメモリ・シェア量。この数値が大きいほど、このVMはより多くのメモリを取得します。数値は、他のすべての実行中のVMの重みに対する相対値です。ゼロを使用すると、オートバルーニングは無効になります。オートバルーニングはpvestatdによって実行されます。

--smbios1 [base64=<1|0>] [,family=<Base64エンコードされた文字列>] [,manufacturer=<Base64エンコードされた文字列>] [,product=<Base64エンコードされた文字列>] [,serial=<Base64エンコードされた文字列>] [,sku=<Base64エンコードされた文字列>] [,uuid=<UUID>] [,version=<Base64エンコードされた文字列>] となります。

SMBIOSタイプ1のフィールドを指定します。

--smp <整数> (1 - N) (デフォルト=1)

CPU数。代わりに -sockets オプションを使用してください。

--ソケット <整数> (1 - N) (デフォルト=1)

CPUソケットの数。

--spice\_enhancements [folderssharing=<1|0>] [,videostreaming=<off|all|filter>]。

SPICE の追加機能拡張を設定します。

--sshkeys <文字列>

cloud-init: 公開 SSH 鍵を設定します (1 行に 1 つの鍵、OpenSSH 形式)。

--startdate (now| YYYY-MM-DD| YYYY-MM-DDTHH:MM:SS) (デフォルト= now)

リアルタイムクロックの初期日付を設定します。有効な日付の書式は、'now' または 2006-06-17T16:01:21 または 2006-06-17.

--startup `[[order=]\d+] [,up=d+] [,down=d+`

スタートアップとシャットダウンの動作。Orderは一般的な起動順序を定義する非負の数値です。シャットダウンは逆の順序で行われます。さらに、次のVMが起動または停止するまでの待ち時間を秒単位で設定できます。

--ストレージID

デフォルトのストレージ。

--tablet <boolean> (デフォルト= 1)

USBタブレットデバイスを有効/無効にします。

--タグ <文字列>

VM のタグ。これは単なるメタ情報です。

--tdf <ブール値> (デフォルト= 0)

時間ドリフト有効/無効。

--テンプレート <ブール値> (デフォルト= 0)

テンプレートの有効/無効。

```
--tpmstate0 [file=<volume> [,size=<DiskSize>] [,version=<v1.2|v2.0>] .
TPMの状態を保存するDiskを設定します。フォーマットはraw固定。
```

```
--unused[n] [file=<volume>
未使用ボリュームへの参照。これは内部的に使用されるので、手動で変更しないでください。
```

```
--usb[n] [[host=<HOSTUSBDEVICE|spice>] の場合[,マッピング=<マッピングID>] [,usb3=<1|0>]
USBデバイスを設定します（nは0～4、マシンのバージョンが7.1以上、かつostype l26またはwindows > 7の場合、nは最大14まで）。
```

```
--vcpus <整数> (1 - N) (デフォルト=0)
ホットプラグされたvcpusの数。
```

```
--vga [[タイプ=<enum>]] [[クリップボード=<vnc[クリップボード=<vnc>] [,メモリ=<整数>
VGAハードウェアを設定します。
```

```
--virtio[n] [file=<volume> [,aio=<native|threads|io_uring>] [,backup=<1|0>] [,bps=<bps>]
[,bps_max_length=<seconds> [,bps_rd=<bps>] [,bps_rd_max_length=<seconds>] [,bps_wr=<bps>]
[,bps_wr_max_length=<seconds>] [,cache=<enum>] [,cyls=<integer>] [,detect_zeroes=<1|0>]
[,discard=<ignore|on>] [,format=<enum>] [,heads=<integer>] [,iops=<iops>] [,iops_max=<iops>]
[,iops_max_length=<seconds>] [,iops_rd=<iops>] [,iops_rd_max=<iops>] [,iops_rd_max_length=<
秒>] [,iops_wr=<iops>] [,iops_wr_max=<iops>] [,iops_wr_max_length=<seconds>]
[,iothread=<1|0>] [,mbps=<mbps>] [,mbps_max=<mbps>] [,mbps_rd=<mbps>] [,mbps_rd_max=<mbps>]
[,mbps_wr=<mbps>] [,mbps_wr_max=<mbps>] [,media=<cdrom|disk>] [,replicate=<1|0>]
[,error=<ignore|report|stop>] [,ro=<1|0>] [,secs=<integer>] [,serial=<serial>]
[,shared=<1|0>] [,size=<DiskSize>] [,snapshot=<1|0>] [,trans=<none|lba|auto>]
[,werror=<enum>] となります。
```

ボリュームを VIRTIO ハード ディスクとして使用します (n は 0 ~ 15)。

```
--virtiofs[n] [dirid=<マッピングID> [,cache=<enum>].
[,direct-io=<1|0>] [,expose-acl=<1|0>] [,expose-xattr=<1|0>] 。
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生成IDを設定します。作成または更新時に自動生成する場合は 1 を使用し、明示的に無効にする場合は 0 を渡します。
```

```
--vmstatestorage <ストレージID>。
VM 状態ボリューム/ファイルのデフォルトストレージ。
```

```
--watchdog [[model=<i6300esb|ib700>][アクション]
仮想ハードウェア・ウォッチドッグ・デバイスを作成します。
```

**qmインポートディスク**

*qm disk import* のエイリアス。

**qm importovf** <vmid> <manifest> <storage> [OPTIONS].

OVFマニフェストから読み込んだパラメータを使用して新しいVMを作成します。

<vmid>: <整数> (100 - 999999999)

VMの（一意の）ID。

<マニフェスト>: <文字列>

ovfファイルへのパス

<ストレージ>: <ストレージID>

対象ストレージID

--ドライラン <ブール値>

抽出されたOVFパラメータの解析済み表現を出力しますが、VMは作成しません。

--format <qcow2| raw| vmdk>

対象フォーマット

**qmリスト [オプション]**

仮想マシンのインデックス（ノードごと）。

--full <ブール値>

アクティブなVMの完全なステータスを決定します。

**qm listsnapshot <vmid> です。**

すべてのスナップショットを一覧表示します。

<vmid>: <整数> (100 - 999999999)

VMの（一意の）ID。

**qm migrate <vmid> <target> [OPTIONS].**

仮想マシンを移行します。新しい移行タスクを作成します。

<vmid>: <整数> (100 - 999999999)

VMの（一意の）ID。

<ターゲット>: <文字列>

対象ノード

--bwlimit <integer> (0 - N) (デフォルト= データセンターまたはストレージ設定からのマイグレーション制限)

I/O帯域幅制限のオーバーライド（単位: KiB/s）。

#### --force <ブール値

ローカルデバイスを使用するVMのマイグレーションを許可します。このオプションはrootのみが使用できます。

#### --マイグレーションネットワーク <文字列

移行に使用する（サブ）ネットワークのCIDR。

#### --マイグレーションタイプ <insecure| secure>

マイグレーショントラフィックはデフォルトでSSHトンネルを使用して暗号化されます。安全で完全にプライベートなネットワークでは、パフォーマンスを上げるためにこれを無効にすることができます。

#### --オンライン <ブール値

VMが実行中の場合、オンライン/ライブマイグレーションを使用。VMが停止している場合は無視されます。

#### -ターゲットストレージ <文字列

ソースストレージからターゲットストレージへのマッピング。単一のストレージIDのみを指定すると、すべてのソース・ストレージがそのストレージにマッピングされます。特別な値1を指定すると、各ソース・ストレージはそれ自体にマッピングされます。

#### --ローカルディスクあり <ブール値

ローカルディスクのライブストレージ移行を有効にします。

### qm モニタ <vmid>

QEMU Monitorインターフェイスに入ります。

#### <vmid>: <整数> (100 - 999999999)

VMの（一意の）ID。

### qm ディスク移動

*qm disk move* のエイリアス。

### qm move\_disk

*qm disk move* のエイリアス。

### qm mtunnel

*qmigrate* によって使用されます - 手動で使用しないでください。

### qm nbdstop <vmid> です。

組み込み nbd サーバーを停止します。

#### <vmid>: <整数> (100 - 999999999)

VMの（一意の）ID。

**qm pending <vmid>**

現在の値と保留中の値の両方を含む仮想マシン設定を取得します。

<vmid>: <整数> (100 - 999999999)

VMの（一意の）ID。

**qm reboot <vmid> [OPTIONS] .**

VMをシャットダウンします。保留中の適用します。

<vmid>: <整数> (100 - 999999999)

VMの（一意の）ID。

**--タイムアウト <整数> (0 - N)**

シャットダウンまで最大タイムアウト秒数待ちます。

**qm remote-migrate <vmid> [<target-vmid>] <target-endpoint> --target-bridge <string> --target-storage <string> [OPTIONS].**

仮想マシンをリモートクラスタに移行します。新しい移行タスクを作成します。EXPERIMENTAL 機能!

<vmid>: <整数> (100 - 999999999)

VMの（一意の）ID。

**<ターゲット-vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

**<ターゲットエンドポイント>: apitoken=<PVEAPIToken=user@realm! token=SECRET>。**

ホスト=<ADDRESS> [, フィンガープリント=<FINGERPRINT>] [, ポート=<PORT>]。

リモートターゲットエンドポイント

**--bwlimit <integer> (0 - N) (デフォルト= データセンターまたはストレージ設定からのマイグレーション制限)**

I/O帯域幅制限のオーバーライド（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は停止した状態で移行元クラスタに保持されます。

**--オンライン <ブール値>**

VMが実行中の場合、オンライン/ライブマイグレーションを使用。VMが停止している場合は無視されます。

**--ターゲットブリッジ <文字列>**

ソース・ブリッジからターゲット・ブリッジへのマッピング。単一のブリッジ ID だけを指定すると、すべてのソース・ブリッジがそのブリッジにマッピングされます。特別な値 1 を指定すると、各ソース・ブリッジはそれ自身にマッピングされます。

### -ターゲットストレージ <文字列>

ソースストレージからターゲットストレージへのマッピング。単一のストレージIDのみを指定すると、すべてのソース・ストレージがそのストレージにマッピングされます。特別な値 1 を指定すると、各ソース・ストレージはそれ自体にマッピングされます。

#### qm再スキャン

*qm disk rescan* のエイリアス。

**qm reset <vmid> [OPTIONS] .**

仮想マシンをリセットします。

**<vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

### --スキップロック <ブール値>

Ignore locks - rootのみがこのオプションを使用できます。

#### qm リサイズ

*qm disk resize* のエイリアス。

**qm resume <vmid> [OPTIONS] [オプション] .**

仮想マシンを再開します。

**<vmid>: <整数> (100 - 999999999)**

VM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>

Ignore locks - rootのみがこのオプションを使用できます。

**qm rollback <vmid> <スナップ名> [オプション] .**

VMの状態を指定したスナップショットにロールバックします。

**<vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

### <スナップ名>: <文字列>

スナップショットの名前。

### --start <boolean> (デフォルト= 0)

ロールバック成功後にVMを起動するかどうか。(注意: スナップショットにRAMが含まれている場合、VMは自動的に起動します)

```
qm sendkey <vmid> <key> [OPTIONS] .
```

仮想マシンにキーイベントを送信します。

<vmid>: <整数> (100 - 999999999)

VMの（一意の）ID。

<キー>: <文字列

キー（qemuモニターエンコーディング）。

--スキップロック <ブール値

Ignore locks - rootのみがこのオプションを使用できます。

```
qm set <vmid> [OPTIONS] .
```

仮想マシン・オプションの設定（同期API） - ホットプラグまたはストレージ割り当てを含むアクションでは、代わりにPOSTメソッドの使用を検討する必要があります。

<vmid>: <整数> (100 - 999999999)

VMの（一意の）ID。

--acpi <boolean> (デフォルト= 1)

ACPI を有効/無効にします。

--アフィニティ <文字列

ゲストプロセスの実行に使用されるホストコアのリスト（例：0,5,8-11

--agent [enabled=<1|0> [,freeze-fs-on-backup=<1|0>]  
[,fstrim\_cloned\_disks=<1|0>] [,type=<virtio|isa>] .

QEMU ゲストエージェントとそのプロパティとの通信を有効/無効にします。

--amd-sev [type=<sev-type> [,allow-smt=<1|0>]。

[,kernel-hashes=<1|0>] [,no-debug=<1|0>] [,no-key-sharing=<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・アーキテクチャ。デフォルトはホスト。

--args <文字列

kvm に渡される任意の引数。

--audio0 device=<ich9-intel-hda|intel-hda|AC97> [,driver=<spice|none>] .

QXL/Spiceと組み合わせると便利です。

--autostart <boolean> (デフォルト= 0)

クラッシュ後の自動再起動（現在は無視されています）。

**--バルーン <整数> (0 - N)**

VM のターゲット RAM の量 (MiB 単位)。ゼロを使用すると、バロンドライバが無効になります。

**--bios <ovmf| seabios> (デフォルト=seabios)**

BIOSの実装を選択します。

**--boot [[legacy]=[<[acdn]{1,4}>] [,order=<デバイス[;デバイス...]>]]。**

ゲストの起動順序を指定します。order= サブプロパティを使用してください。

**--ブートディスク(ide|sata|scsi|virtio)◆d+.**

指定したディスクからの起動を有効にします。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 -ide2 オプションのエイリアスです。

**--cicustom [<volume> [,network=<volume>] [,user=<volume>] [,vendor=<volume>]].**

cloud-init: 開始時に自動生成されるファイルを置き換えるカスタムファイルを指定します。

**--パスワード**

cloud-init: ユーザーに割り当てるパスワード。一般的に、これを使うことは推奨されません。代わりに ssh 鍵を使います。また、古いバージョンの cloud-init はハッシュ化されたパスワードをサポートしていないことに注意しましょう。

**--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、設定されているオペレーティングシステムの種類 (ostype.Linuxではnocloud形式、Windowsではconfigdrive2を使用します。

**--ciupgrade <boolean> (デフォルト= 1)**

cloud-init: 最初の起動後にパッケージの自動アップグレードを行います。

**--ciuser <文字列>**

cloud-init: イメージに設定されているデフォルトユーザーの代わりに、sshキーとパスワードを変更するユーザー名。

**--コア数 <整数> (1 - N) (デフォルト= 1)**

ソケットあたりのコア数。

**--cpu [<cputype=><string>] [,flags=<+FLAG[-FLAG...]>] [,hidden=<1|0>]**

[,hv-vendor-id=<vendor-id>]となります。[,hidden=<1|0>] [,hv-vendor-id=<vendor-id>] [,hv-vendor-id=<vendor-id>]

[,phys-bits=<8-64|host>] [,reported-model=<enum> ]。

エミュレートされたCPUタイプ。

**--cpulimit <number> (0 - 128) (デフォルト= 0)**

CPU使用量の上限。

```
--cpuunits <integer> (1 - 262144) (デフォルト= cgroup v1: 1024, cgroup v2: 100)
VM の CPU ウエイトは、cgroup v2 では [1, 10000] にクランプされます。
```

#### --削除 <文字列>

削除したい設定のリスト。

#### --説明 <文字列>

VM の説明。WebインターフェイスのVMのサマリーに表示されます。コンフィギュレーション・ファイルのコメントとして保存されます。

#### -ダイジェスト <文字列>

現在の設定ファイルの SHA1 ダイジェストが異なる場合に変更を防止します。これは同時変更を防ぐために使用できます。

```
--efidisk0 [file=<volume> [,efitype=<2m|4m>] [,format=<enum>] [,import-from=<ソースボリューム>] [,pre-enrolled-keys=<1|0>] [,size=<DiskSize>].
```

EFIバーを格納するディスクを設定します。STORAGE\_ID:SIZE\_IN\_GiBという特殊な構文を使用して、EFIバーを格納するディスクを割り当てます。を新しいボリュームにコピーします。SIZE\_IN\_GiBはここでは無視され、代わりにデフォルトのEFIバー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 <ブール値>

物理的な削除を強制します。これがないと、単純に設定ファイルからディスクを削除し、ボリュームIDを含むunused[n]という追加の設定エントリを作成します。unused[n]のリンク解除は、常に物理的な削除を引き起こします。

---

#### 備考

必要なオプション: 削除

---

#### --フリーズ <ブール値>

起動時にCPUをフリーズ (c monitorコマンドで実行開始)。

#### --hookscript <文字列>

vmsのライフタイムの様々なステップで実行されるスクリプト。

```
--hostpci[n] [[host=<HOSTPCIID[,HOSTPCIID2...]>][,device-id=<hex id>] [,legacy-igd=<1|0>]
[,mapping=<mapping-id>] [,mdev=<string>] [,pcie=<1|0>] [,rombar=<1|0>] [,romfile=<string>]
[[host=<HOSTPIID[,HOSTPCIID2...]>]] --hostpci[n
[,sub-device-id=<hex id>] [,sub-vendor-id=<hex id>] [,vendor-id=<hex id>]
[,x-vga=<1|0>]。
```

ホストのPCIデバイスをゲストにマッピングします。

#### --hotplug <string> (デフォルト= ネットワーク,ディスク,USB)

ホットプラグ機能を選択的に有効にします。これは、ホットプラグコンマ区切りのリストです: ネットワーク、ディスク

CPU、メモリ、USB、cloudinit。ホットプラグを完全に無効にするには0を使います。値として1を使用すると、デフォルトの network、disk、usb のエイリアスになります。USB ホットプラグは、マシン>=7.1 および ostype l26 または windows>7.

**--hugepages <1024| 2| any>**

hugepagesメモリの有効/無効。

--ide[n] [file=<volume> [,aio=<native|threads|io\_uring>] [,backup=<1|0>] [,bps=<bps>]  
[,bps\_max\_length=<seconds> [,bps\_rd=<bps>] [,bps\_rd\_max\_length=<seconds>] [,bps\_wr=<bps>]  
[,bps\_wr\_max\_length=<seconds>] [,cache=<enum>] [,cyls=<integer>] [,detect\_zeroes=<1|0>]  
[,discard=<ignore|on>] [,format=<enum>] [,heads=<integer>] [,import-from=<ソース・ボリューム>]  
[,iops=<iops>] [,iops\_max=<iops>] [,iops\_max\_length=<seconds>] [,iops\_rd=<iops>]  
[,iops\_rd\_max=<iops>] [,iops\_rd\_max\_length=<seconds>] [,iops\_wr=<iops>] [,iops\_wr\_max=<iops>]  
[,iops\_wr\_max\_length=<seconds>] [,mbps=<mbps>] [,mbps\_max=<mbps>] [,mbps\_rd=<mbps>]  
[,mbps\_rd\_max=<mbps>] [,mbps\_wr=<mbps>] [,mbps\_wr\_max=<mbps>] [,media=<CDROM|DISK>]  
[,model=<model>] [,replicate=<1|0>] [,rerror=<ignore|report|stop>] [,secs=<integer>]  
[,serial=<serial>] [,shared=<1|0>] [,size=<DiskSize>] [,snapshot=<1|0>] [,ssd=<1|0>]  
[,trans=<none|lba|auto>] [,werror=<enum>] [,wwn=<wwn>] となります。

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0~3)。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を使用します。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。

--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。

特別な文字列dhcpは、DHCPを使用するIPアドレスに使用でき、その場合、明示的な ゲートウェイは提供されるべきではありません。IPv6では、ステートレス自動設定を使うために特別な文字列autoを使うことができます。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=<整数> [,name=<文字列>] です。

VM間共有メモリ。VM間やホストとの直接通信に便利。

--keephugepages <boolean> (デフォルト= 0)

hugepagesと一緒に使用します。有効にすると、hugepagesはVMシャットダウン後も削除されず、その後の起動に使用できます。

--キーボード <da| de| de-ch| en-gb| en-us| es| fi| fr| fr-be  
|fr-ca| fr-ch| hu| is| it| ja| lt| mk| nl| no| pl| pt| pt-br| sl| sv| tr>

VNC サーバーのキーボードレイアウト。このオプションは一般的には必要なく、ゲストOS内で処理した方が良い場合が多いです。

--kvm <boolean> (デフォルト= 1)

KVMハードウェア仮想化の有効/無効を設定します。

--ローカルタイム <ブール値>

リアルタイムクロック (RTC) をローカルタイムに設定します。 これは、ostypeが以下を示す場合、デフォルトで有効になります。

Microsoft Windows OS。

--lock <backup| clone| create| migrate| rollback| snapshot | snapshot-delete | suspended | suspending>

VMをロック/アンロックします。

-マシン [[タイプ=]<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。

--メモリ [current=<整数>]

メモリ特性。

--migrate\_downtime <number> (0 - N) (デフォルト= 0.1)

マイグレーションの最大許容ダウンタイム (秒) を設定します。新しく汚れたRAMを転送する必要があるため、マイグレーションが最後まで収束しない場合、マイグレーションが収束するまで、この制限を段階的に自動的に増やします。

--migrate\_speed <integer> (0 - N) (デフォルト= 0)

マイグレーションの最大速度 (MB/s) を設定します。値 0 は制限なしです。

--name <文字列>

VM の名前を設定します。コンフィギュレーションWebインターフェイスでのみ使用します。

--ネームサーバー <文字列>

cloud-init: コンテナの DNS サーバー IP アドレスを設定します。searchdomain も nameserver も設定されていない場合、Create は自動的にホストからの設定を使用します。

--net[n] [model=<enum> [,bridge=<bridge>] [,firewall=<1|0>] [,link\_down=<1|0>] [,macaddr=<XX:XX:XX:XX:XX:>] [,mtu=<integer>] [,queues=<integer>] [,rate=<number>] [,tag=<integer>] [,trunks=<vlanid[,vlanid...]>] [,<model>=<macaddr>] [,<model>=<macaddr>] [,<model>=<macaddr>]

ネットワーク機器を指定します。

--numa <boolean> (デフォルト= 0)

NUMAの有効/無効。

--numa[n] cpus=<id[-id];...> [,hostnodes=<id[-id];...>]を指定します。

[,memory=<number>] [,policy=<preferred|bind|interleave>]となります。

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ます。

--parallel[n] /dev/parportd+|/dev/usb/lpd+

ホスト・パラレル・デバイスをマップします (nは0~2)。

--プロテクション <ブール値> (デフォルト= 0)

VM の保護フラグを設定します。これにより、VM の削除とディスクの削除操作が無効になります。

--reboot <boolean> (デフォルト= 1)

再起動を許可。0に設定すると、VM は時に終了します。

--復帰 <文字列

保留中の元に戻します。

--rng0 [source=</dev/urandom|/dev/random|/dev/hwrng> [,max\_bytes=<integer>] [,period=<integer>].  
VirtIO ベースの乱数ジェネレータを設定します。

--sata[n] [file=<volume> [,aio=<native|threads|io\_uring>] [,backup=<1|0>] [,bps=<bps>]  
[,bps\_max\_length=<seconds> [,bps\_rd=<bps>] [,bps\_rd\_max\_length=<seconds>] [,bps\_wr=<bps>]  
[,bps\_wr\_max\_length=<seconds>] [,cache=<enum>] [,cyls=<integer>] [,detect\_zeroes=<1|0>]  
[,discard=<ignore|on>] [,format=<enum>] [,heads=<integer>] [,import-from=<ソース・ボリューム>]  
[,iops=<iops>] [,iops\_max=<iops>] [,iops\_max\_length=<seconds>] [,iops\_rd=<iops>]  
[,iops\_rd\_max=<iops>] [,iops\_rd\_max\_length=<seconds>] [,iops\_wr=<iops>] [,iops\_wr\_max=<iops>]  
[,iops\_wr\_max\_length=<seconds>] [,mbps=<mbps>] [,mbps\_max=<mbps>] [,mbps\_rd=<mbps>]  
[,mbps\_rd\_max=<mbps>] [,mbps\_wr=<mbps>] [,mbps\_wr\_max=<mbps>] [,media=<cdrom|disk>]  
[,replicate=<1|0>] [,rerror=<ignore|report|stop>] [,secs=<integer>] [,serial=<serial>]  
[,shared=<1|0> ]。] [,size=<DiskSize>] [,snapshot=<1|0>] [,ssd=<1|0>] [,trans=<none|lba|auto>]  
[,werror=<enum>] [,wwn=<wwn> ]。

ボリュームをSATA/ハードディスクまたはCD-ROMとして使用します (nは0~5)。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。既存のボリュームからインポートするには、STORAGE\_ID:0とimport-fromパラメータを使用します。

```
--scsi[n] [[file=<volume> [,aio=<native|threads|io_uring>]] [,backup=<1|0>] [,bps=<bps>]
[,bps_max_length=<seconds>] [,bps_rd=<bps>] [,bps_rd_max_length=<seconds>] [,bps_wr=<bps>]
[,bps_wr_max_length=<seconds>] [,cache=<enum>] [,cyls=<integer>] [,detect_zeroes=<1|0>]
[,discard=<ignore|on>] [,format=<enum>] [,heads=<integer>] [,import-from=<ソースボリューム>]
[,iops=<iops>] [,iops_max=<iops>] [,iops_max_length=<seconds>] [,iops_rd=<iops>]
[,iops_rd_max=<iops>] [,iops_rd_max_length=<seconds>] [,iops_wr=<iops>] [,iops_wr_max=<iops>]
[,iops_wr_max_length=<seconds>] [,iothread=<1|0>] [,mbps=<mbps>] [,mbps_max=<mbps>]
[,mbps_rd=<mbps>] [,mbps_rd_max=<mbps>] [,mbps_wr=<mbps>] [,mbps_wr_max=<mbps>]
[,media=<cdrom|disk>] [,product=<product>] [,queues=<integer>] [,replicate=<1|0>]
[,rerror=<ignore|report|stop>] [,ro=<1|0>] [,scsiblock=<1|0>] [,secs=<integer>]
[,serial=<serial>] [,shared=<1|0>] [,size=<DiskSize>] [,snapshot=<1|0>] [,ssd=<1|0>]
[,trans=<none|lba|auto>] [,vendor=<vendor>] [,werror=<enum>] [,wwn=<wwn>] となります。
```

ボリュームをSCSIハードディスクまたはCD-ROMとして使用します（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

を使用します。新しいボリュームを割り当てるには、STORAGE\_ID:0とimport-fromパラメータを使用します。

設定します。

```
--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
```

#### --検索ドメイン <文字列>

cloud-init: コンテナのDNS検索ドメインを設定します。searchdomainもnameserverも設定されていない場合、Createはホストからの設定を自動的に使用します。

#### -シリアル[n] (/dev/.+|socket)

VM内にシリアル・デバイスを作成（nは0～3）

#### --shares <integer> (0 - 50000) (デフォルト = 1000)

オート・バルーニングのメモリ・シェア量。この数値が大きいほど、このVMはより多くのメモリを取得します。数値は、他のすべての実行中のVMの重みに対する相対値です。ゼロを使用すると、オートバルーニングは無効になります。オートバルーニングはpvestatdによって実行されます。

#### --スキップロック <ブール値>

Ignore locks - rootのみがこのオプションを使用できます。

```
--smbios1 [base64=<1|0>] [,family=<Base64エンコードされた文字列>] [,manufacturer=<Base64エンコードされた文字列>]
[,product=<Base64エンコードされた文字列>] [,serial=<Base64エンコードされた文字列>] [,sku=<Base64エンコードされた文字列>]
[,uuid=<UUID>] [,version=<Base64エンコードされた文字列>] となります。
```

SMBIOSタイプ1のフィールドを指定します。

--smp <整数> (1 - N) (デフォルト=1)

CPU数。代わりに -sockets オプションを使用してください。

--ソケット <整数> (1 - N) (デフォルト=1)

CPUソケットの数。

--spice\_enhancements [foldersharing=<1|0>] [,videostreaming=<off|all|filter>]。

SPICE の追加機能拡張を設定します。

--sshkeys <ファイルパス>

cloud-init: 公開 SSH 鍵を設定します (1 行に 1 つの鍵、OpenSSH 形式)。

--startdate (now | YYYY-MM-DD | YYYY-MM-DDTHH:MM:SS) (デフォルト= now)

リアルタイムクロックの初期日付を設定します。有効な日付の書式は、'now' または 2006-06-17T16:01:21 または 2006-06-17.

--startup `[[order=]\d+] [,up=d+] [,down=d+]

スタートアップとシャットダウンの動作。Orderは一般的な起動順序を定義する非負の数値です。シャットダウンは逆の順序で行われます。さらに、次のVMが起動または停止するまでの待ち時間を秒単位で設定できます。

--tablet <boolean> (デフォルト= 1)

USBタブレットデバイスを有効/無効にします。

--タグ <文字列>

VM のタグ。これは単なるメタ情報です。

--tdf <ブール値> (デフォルト= 0)

時間ドリフト有効/無効。

--テンプレート <ブール値> (デフォルト= 0)

テンプレートの有効/無効。

--tpmstate0 [ファイル=<ボリューム> [,インポート元=<ソースボリューム>] [,サイズ=<ディスクサイズ>] [,バージョン=<v1.2|v2.0>]。

TPM の状態を保存するDiskを設定します。フォーマットはrawに固定されています。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。SIZE\_IN\_GiBはここでは無視され、代わりに4MiBが使用されることに注意してください。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=<volume>

未使用ボリュームへの参照。これは内部的に使用されるので、手動で変更しないでください。

```
--usb[n] [[host=]<HOSTUSBDEVICE|spice>]の場合 [,マッピング=<マッピングID>] [,usb3=<1|0>]
```

USBデバイスを設定します（nは0～4、マシンのバージョンが7.1以上、かつostype l26またはwindows > 7の場合、nは最大14まで）。

```
--vcpus <整数> (1 - N) (デフォルト = 0)
```

ホットプラグされたvcpusの数。

```
--vga [[タイプ=]<enum>]] [[クリップボード=<vnc[クリップボード=<vnc>] [,メモリ=<整数>]
```

VGAハードウェアを設定します。

```
--virtio[n] [file=<volume> [,aio=<native|threads|io_uring>>] [,backup=<1|0>] [,bps=<bps>]
[,bps_max_length=<seconds> [,bps_rd=<bps>] [,bps_rd_max_length=<seconds>] [,bps_wr=<bps>]
[,bps_wr_max_length=<seconds>] [,cache=<enum>] [,cyls=<integer>] [,detect_zeroes=<1|0>]
[,discard=<ignore|on>] [,format=<enum>] [,heads=<integer>] [,import-from=<ソース・ボリューム>]
[,iops=<iops>] [,iops_max=<iops>] [,iops_max_length=<seconds>] [,iops_rd=<iops>]
[,iops_rd_max=<iops>] [,iops_rd_max_length=<seconds>] [,iops_wr=<iops>] [,iops_wr_max=<iops>]
[,iops_wr_max_length=<seconds>] [,iothread=<1|0>] [,mbps=<mbps>] [,mbps_max=<mbps>]
[,mbps_rd=<mbps>] [,mbps_rd_max=<mbps>] [,mbps_wr=<mbps>] [,mbps_wr_max=<mbps>]
[,media=<cdrom|disk>] [,replicate=<1|0>] [,rerror=<ignore|report|stop>] [,ro=<1|0>]
[,secs=<integer>] [,serial=<serial>] [,serial=<serial>] [,shared=<1|0>] [,size=<DiskSize>>]
[,snapshot=<1|0>] [,trans=<none|lba|auto>] [,werror=<enum>] [,secs=<integer>>]のように指定します。
```

ボリュームをVIRTIOハードディスクとして使用します（nは0～15）。新しいボリュームを割り当てるには、特別な構文 STORAGE\_ID:SIZE\_IN\_GiB を使用します。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。

```
--virtiofs[n] [dirid=<mapping-id> [,cache=<enum>].
```

```
,direct-io=<1|0>] [,expose-acl=<1|0>] [,expose-xattr=<1|0>] 。
```

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生成IDを設定します。作成または更新時に自動生成する場合は 1 を使用し、明示的に無効にする場合は 0 を渡します。

```
--vmstatestorage <ストレージID>。
```

VM 状態ボリューム/ファイルのデフォルトストレージ。

```
--watchdog [[model=]<i6300esb|ib700>][アクション]
```

仮想ハードウェア・ウォッチドッグ・デバイスを作成します。

```
qm showcmd <vmid> [OPTIONS].
```

VMの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>: <整数> (100 - 999999999)**

VMの（一意の）ID。

**--pretty <boolean> (デフォルト= 0)**

各オプションを改行し、可読性を高めます。

**--スナップショット <文字列>**

指定されたスナップショットから設定値を取得します。

**qm shutdown <vmid> [OPTIONS] .**

仮想マシンをシャットダウンします。これは、物理マシンの電源ボタンを押すのと似ています。これにより、ゲスト OS に ACPI イベントが送信され、クリーンシャットダウンに進むはずです。

**<vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

**--forceStop <boolean> (デフォルト= 0)**

VMが停止していることを確認してください。

**--keepActive <boolean> (デフォルト= 0)**

ストレージボリュームを停止しないでください。

**--スキップロック <ブール値>**

Ignore locks - rootのみがこのオプションを使用できます。

**--タイムアウト <整数> (0 - N)**

最大タイムアウト秒数まで待ちます。

**qm snapshot <vmid> <snapname> [OPTIONS] .**

VMをスナップショットします。

**<vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

**<スナップ名>: <文字列>**

スナップショットの名前。

**--説明 <文字列>**

テキストによる説明やコメント。

**--vmstate <ブール値>**

vmstateの保存

**qm start <vmid> [OPTIONS] .**

仮想マシンを起動します。

**<vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

**--force-cpu <文字列>**

QEMUの-cpu引数を与えられた文字列でオーバーライドします。

**-マシン [[タイプ=]<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 <文字列>。**

クラスタ・ノード名。

**--マイグレーションネットワーク <文字列>**

移行に使用する（サブ）ネットワークのCIDR。

**--マイグレーションタイプ <insecure| secure>**

マイグレーショントラフィックはデフォルトでSSHトンネルを使用して暗号化されます。安全で完全にプライベートなネットワークでは、パフォーマンスを上げるためにこれを無効にすることができます。

**--スキップロック <ブール値>**

Ignore locks - rootのみがこのオプションを使用できます。

**--stateuri <文字列>**

いくつかのコマンドは、場所から状態を保存/復元します。

**-ターゲットストレージ <文字列>**

ソースストレージからターゲットストレージへのマッピング。単一のストレージIDのみを指定すると、すべてのソース・ストレージがそのストレージにマッピングされます。特別な値 1 を指定すると、各ソース・ストレージはそれ自体にマッピングされます。

**--timeout <integer> (0 - N) (デフォルト= max(30, vm メモリは GiB))**

最大タイムアウト秒数まで待ちます。

**qm status <vmid> [OPTIONS].**

VMのステータスを表示します。

**<vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

**--verbose <ブール値>**

冗長出力フォーマット

**qm stop <vmid> [OPTIONS].**

仮想マシンを停止します。qemuプロセスは直ちに終了します。これは実行中のコンピュータの電源プラグを抜くようなもので、仮想マシンのデータを損傷する可能性があ

ります。

**<vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

**--keepActive <boolean> (デフォルト= 0)**

ストレージボリュームを停止しないでください。

**-migratedfrom <文字列>**

クラスタ・ノード名。

**--overrule-shutdown <boolean> (デフォルト= 0)**

停止する前に、アクティブな *qmshutdown* タスクを中止してください。

**--スキップロック <ブール値>**

Ignore locks - rootのみがこのオプションを使用できます。

**--タイムアウト <整数> (0 - N)**

最大タイムアウト秒数まで待ちます。

**qm suspend <vmid> [OPTIONS].**

仮想マシンを一時停止します。

**<vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

**--スキップロック <ブール値>**

Ignore locks - rootのみがこのオプションを使用できます。

**--ストレージID**

VMの状態を保存するストレージ

---

#### 備考

必要なオプション: *todisk*

---

**--todisk <boolean> (デフォルト= 0)**

設定されている場合、VMをディスクにサスPENDします。次回のVM起動時に再開されます。

**qm テンプレート <vmid> [OPTIONS].**

テンプレートを作成します。

**<vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

---

```
--ディスク <efidisk0 | ide0 | ide1 | ide2 | ide3 | sata0 | sata1 | sata2 | sata3 | sata4 |  
sata5 | scsi0 | scsi1 | scsi10 | scsi11 | scsi12 | scsi13 | scsi14 | scsi15| scsi16 | scsi17 |  
scsi18 | scsi19 | scsi2 | scsi20 | scsi21 | scsi22 | scsi23 | scsi24 | scsi25| scsi26| scsi27|  
scsi28| scsi29| scsi3| scsi30| scsi4  
| scsi5 | scsi6 | scsi7 | scsi8 | scsi9 | tpmstate0 | virtio0 | virtio1| virtio10| virtio11|  
virtio12| virtio13| virtio14 | virtio15 | virtio2 | virtio3 | virtio4 | virtio5 | virtio6 |  
virtio7 | virtio8 | virtio9>
```

1つのディスクだけをベースイメージに変換したい場合。

**qm terminal** <vmid> [OPTIONS] .

シリアル・デバイスを使用してターミナルを開きます（VMにはシリアル・デバイスが設定されている必要があります。）

<vmid>: <整数> (100 - 999999999)

VMの（一意の）ID。

--escape <string> (デフォルト= ^O)

エスケープ文字。

--iface <シリアル0| シリアル1| シリアル2| シリアル3>

シリアル・デバイスを選択します。デフォルトでは、単に最初の適切なデバイスを使用します。

**qmアンリンク**

**qm unlock** <vmid> VM のロックを解

除します。

<vmid>: <整数> (100 - 999999999)

VMの（一意の）ID。

**qm vncproxy** <vmid> です。

VMのVNCトラフィックを標準入力/標準出力にプロキシ

<vmid>: <整数> (100 - 999999999)

VMの（一意の）ID。

**qm wait** <vmid> [OPTIONS] .

VMが停止するまで待ちます。

<vmid>: <整数> (100 - 999999999)

VMの（一意の）ID。

--タイムアウト <整数> (1 - N)

秒単位のタイムアウト。デフォルトは永久に待ちます。

## A.10 qmrestore - QemuServer vzdump バックアップのリストア

**qmrestore** ヘルプ

**qmrestore** <archive> <vmid> [OPTIONS].

QemuServer vzdump/バックアップのリストア。

**<アーカイブ>: <文字列**

バックアップ・ファイル。標準入力から読み込む場合は - を渡します。

**<vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

**-bwlimit<数値> (0 - N)**

I/O帯域幅制限のオーバーライド（単位：KiB/s）。

**--force <ブール値**

既存のVMの上書きを許可します。

**--ライブリストア <ブール値**

バックアップから直ちにVMを起動し、バックグラウンドでリストアします。PBSのみ。

**--プール <文字列**

VMを指定したプールに追加します。

**--ストレージID**

デフォルトのストレージ。

**--unique <ブール値**

一意のランダムなイーサネットアドレスを割り当てます。

## A.11 pct - Proxmox コンテナツールキット

**pct**<COMMAND> [ARGS] [OPTIONS] です。

**pct clone** <vmid> <newid> [OPTIONS].

コンテナのクローン/コピーの作成

**<vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

**<newid>: <整数> (100 - 999999999)**

クローンの VMID。

--bwlimit <number> (0 - N) (デフォルト= データセンターまたはストレージ設定からのクローン制限)

I/O帯域幅制限のオーバーライド（単位: KiB/s）。

#### --説明 <文字列>

新しいCTの説明。

#### --full <ブール値>

すべてのディスクの完全コピーを作成します。これは通常のCTをクローンするときに必ず行われます。CTテンプレートでは、デフォルトでリンクされたクローンを作成しようとなります。

#### --ホスト名 <文字列>

新しいCTのホスト名を設定します。

#### --プール <文字列>

新しいCTを指定されたプールに追加します。

#### --スナップネーム <文字列>

スナップショットの名前。

#### --ストレージID

対象ストレージ。

#### --ターゲット <文字列>

ターゲット・ノード。元のVMが共有ストレージ上にある場合にのみ許可されます。

pct config <vmid> [OPTIONS].

コンテナ構成を取得します。

<vmid>: <整数> (100 - 999999999)

VMの（一意の）ID。

--current <boolean> (デフォルト= 0)

保留値ではなく）現在の値を取得します。

#### --スナップショット <文字列>

指定されたスナップショットから設定値を取得します。

pct コンソール <vmid> [OPTIONS].

指定したコンテナのコンソールを起動します。

<vmid>: <整数> (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接頭辞。例えば、<Ctrl+b q> をエスケープシーケンスとして使うには ^b。

#### pct cpusets

割り当てられたCPUセットのリストを印刷します。

**pct create <vmid> <ostemplate> [OPTIONS]**を作成します。

コンテナを作成または復元します。

<vmid>: <整数> (100 - 999999999)

VMの（一意の）ID。

<ostemplate>: <文字列>

OSテンプレートまたはバックアップファイル。

--arch <amd64| arm64| armhf| i386| riscv32| riscv64> (= amd64)

OSアーキテクチャの種類。

--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リストア制限)

I/O帯域幅制限のオーバーライド（単位: KiB/s）。

--cmode <コンソール| シェル| tty> (デフォルト= tty)

コンソールモード。デフォルトでは、consoleコマンドは利用可能なttyデバイスの1つに接続を開こうとします。cmodeを console に設定すると、代わりに /dev/console にアタッチしようとします。cmodeを shell に設定すると、コンテナ内で単にシェルを起動します（ログインはしません）。

--コンソール <ブール値> (デフォルト= 1)

コンテナにコンソールデバイス (/dev/console) をアタッチします。

--コア数 <整数> (1 - 8192)

コンテナに割り当てられたコアの数。コンテナは、デフォルトで使用可能なすべてのコアを使用できます。

--cpulimit <number> (0 - 8192) (デフォルト= 0)

CPU使用量の上限。

---

#### 備考

コンピュータに2つのCPUがある場合、合計2つのCPU時間があります。値0はCPU制限なしを示します。

---

--cpuunits <integer> (0 - 500000) (デフォルト= cgroup v1: 1024, cgroup v2: 100)

コンテナのCPU重量は、cgroup v2 では [1, 10000] にクランプされます。

---

--debug <boolean> (デフォルト= 0)

もっと冗長にしてみてください。今のところ、これは起動時にデバッグルoglevelを有効にするだけです。

#### --説明 <文字列>

コンテナの説明。WebインターフェイスCTのサマリーに表示されます。これは設定ファイルのコメントとして保存されます。

--dev[n] [[path=</パス> [,deny-write=<1|0>] [,gid=<integer>] [,mode=<オクタルアクセスモード>] [,uid=<integer>] [,n

コンテナへの通過装置

--features [force\_rw\_sys=<1|0>] [,fuse=<1|0>] [,keyctl=<1|0>] [,mknod=<1|0>] [,mount=<fstype;fstype;...>] [,nesting=<1|0>] となります。

コンテナが高度なアクセスできるようにします。

#### --force <ブール値>

既存のコンテナの上書きを許可します。

#### --hookscript <文字列>

コンテナのライフタイムのさまざまなステップで実行されるスクリプト。

#### --ホスト名 <文字列>

コンテナのホスト名を設定します。

#### --ignore-unpack-errors <ブール値>

テンプレートの抽出時にエラーを無視します。

--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

#### --メモリ <整数> (16 - N) (デフォルト=512)

コンテナのRAM容量 (MB)。

--mp[n] [volume=<ボリューム>,mp=<パス> [,acl=<1|0>] [,backup=<1|0>] [,mountoptions=<opt[;opt...]>] [,quota=<1|0>] [,replicate=<1|0>] [,ro=<1|0>] [,shared=<1|0>]

[,size=<DiskS[quota=<1|0>][,replicate=<1|0>][,ro=<1|0>][,shared=<1|0>][,size=<DiskSize>]]。

ボリュームをコンテナ・マウント・ポイントとして使用します。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。

#### --ネームサーバー <文字列>

コンテナの DNS サーバー IP アドレスを設定します。searchdomainもnameserverも設定しない場合、Createは自動的にホストの設定を使用します。

```
--net[n] name=<string> [,bridge=<bridge>] [,firewall=<1|0>] [,gw=<GatewayIPv4>]
[,gw6=<GatewayIPv6>] [,hwaddr=<XX:XX:XX:XX:XX: XX>] [,ip=<(IPv4/CIDR|dhcp|manual)>]
[,ip6=<(IPv6/CIDR|auto|dhcp|manual)>] [,link_down=<1|0>] [,mtu=<integer>] [,rate=<mbps>]
[,tag=<integer>] [,trunks=<vlanid[;vlanid...]>] [,type=<veth>]
    コンテナのネットワーク・インターフェースを指定します。

--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。これは、コンテナ内の設定を行うために使用され、/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 は、OS 固有のセットアップをスキップするために使用できます。

--password <password>
    コンテナ内の root パスワードを設定します。

--pool <文字列>
    VMを指定したプールに追加します。

--protection <ブール値> (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/更新操作を防止します。

--restore <ブール値>
    これを復元タスクとしてマークします。

--rootfs [ボリューム=<ボリューム> [,acl=<1|0>] [,mountoptions=<opt[;opt...]>] [,quota=<1|0>]
[,replicate=<1|0>] [,ro=<1|0>] [,shared=<1|0>]
[,size=<DiskS[quota=<1|0>][,replicate=<1|0>][,ro=<1|0>][,shared=<1|0>][,size=<DiskSize>]。
    ボリュームをコンテナのルートとして使用します。
```

ボリュームをコンテナのルートとして使用します。

```
--searchdomain <文字列>
    コンテナのDNS検索ドメインを設定します。searchdomainもnameserverも設定しない場合、Createはホストからの設定を自動的に使用します。
```

```
-ssh-publickey <ファイルパス>
    公開SSH鍵を設定します（1行に1つの鍵、OpenSSH形式）。
```

```
--start <boolean> (デフォルト= 0)
    CTの作成が正常に終了したら、CTを開始します。
```

```
--startup `[[order=] \d+] [,up=d+] [,down=d+`]
    スタートアップとシャットダウンの動作。Orderは一般的な起動順序を定義する非負の数値です。シャットダウンは逆の順序で行われます。さらに、次の
VMが起動または停止するまでの待ち時間を秒単位で設定できます。
```

--storage <ストレージID> (デフォルト= ローカル)

デフォルトのストレージ。

--スワップ <整数> (0 - N) (デフォルト= 512)

MB単位のコンテナのSWAP量。

--タグ <文字列>

コンテナのタグ。これは単なるメタ情報です。

--テンプレート <ブール値> (デフォルト= 0)

テンプレートの有効/無効。

--タイムゾーン <文字列>

コンテナで使用するタイムゾーン。オプションが設定されていない場合は、何もれません。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 から任意のタイムゾーンオプションを設定することができます。

--tty <整数> (0 - 6) (デフォルト= 2)

コンテナで利用可能なttyの数を指定

--unique <ブール値>

一意のランダムなイーサネットアドレスを割り当てます。

---

#### 備考

必要なオプション: リストア

---

--unprivileged <boolean> (デフォルト= 0)

コンテナを非特権ユーザーとして実行します。(手動で変更してはいけません)。

--未使用 [n] [ボリューム=]<ボリューム>

未使用ボリュームへの参照。これは内部的に使用されるので、手動で変更しないでください。

pct delsnapshot <vmid> <snapname> [OPTIONS].

LXCスナップショットを削除します。

<vmid>: <整数> (100 - 999999999)

VMの（一意の）ID。

<スナップ名>: <文字列>

スナップショットの名前。

--force <ブール値>

ディスクスナップショットの削除に失敗した場合でも、設定ファイルから削除できます。

---

**pct destroy <vmid> [OPTIONS] .**

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>: <整数> (100 - 999999999)

VMの（一意の）ID。

**-destroy-unreferenced-disks<boolean>.**

設定されている場合、設定内で参照されていないすべての有効なストレージから、VMIDを持つすべてのディスクを追加で破棄します。

**--force <boolean> (デフォルト= 0)**

走っていても強制破壊。

**--ページ <ブール値> (デフォルト= 0)**

関連するすべての構成からコンテナを削除します。たとえば、バックアップジョブ、レプリケーションジョブ、HAなどです。関連するACLとファイアウォールのエントリは常に削除されます。

**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>: <整数> (100 - 999999999)

VMの（一意の）ID。

**pct enter <vmid> [OPTIONS] .**

指定したコンテナのシェルを起動します。

<vmid>: <整数> (100 - 999999999)

VMの（一意の）ID。

**--keep-env <boolean> (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になります。事前に提供された環境に依存している場合は、このオプションを使用して将来に備えてください。

**pct exec <vmid> [<extra-args>] [OPTIONS] .**

指定したコンテナ内でコマンドを起動します。

<vmid>: <整数> (100 - 999999999)

VMの（一意の）ID。

**<extra-args>: <array> です。**

配列としての追加引数

**--keep-env <boolean> (デフォルト= 1)**

現在の環境を維持します。このオプションは PVE 9 ではデフォルトで無効になります。事前に提供された環境に依存している場合は、このオプションを使用して将来に備えてください。

**pct fsck <vmid> [OPTIONS] .**

コンテナ・ボリュームでファイルシステム・チェック (fsck) を実行します。

**<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>**

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クリーンであるように見えても強制チェック

**pct fstrim <vmid> [OPTIONS] .**

選択したCTとそのマウントポイント（バインドまたは読み取り専用マウントポイントを除く）に対してfstrimを実行します。

**<vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

**--ignore-mountpoints <boolean>**

すべてのマウントポイントをスキップし、コンテナルート上のfstrimのみを実行します。

**pct help [OPTIONS]**

指定したコマンドに関するヘルプを表示します。

**--extra-args <array>** です。

特定のコマンドのヘルプを表示します。

**--verbose <ブール値**

冗長出力フォーマット。

#### パーセントリスト

**pct listsnapshot <vmid>** すべてのスナップ

ショットを一覧表示します。

**<vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

**pct migrate <vmid> <target> [OPTIONS] (マイグレート <vmid> <ターゲット> [オプション])**

コンテナを別のノードに移行します。新しいマイグレーションタスクを作成します。

**<vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

**<ターゲット>: <文字列**

対象ノード

**--bwlimit <number> (0 - N) (デフォルト= データセンターまたはストレージ設定からのマイグレーション制限)**

I/O帯域幅制限のオーバーライド（単位: KiB/s）。

**--オンライン <ブール値**

オンライン/ライブマイグレーションを使用します。

**--restart <ブール値**

リスタートマイグレーションを使用

**-ターゲットストレージ <文字列>**

ソースストレージからターゲットストレージへのマッピング。単一のストレージIDのみを指定すると、すべてのソース・ストレージがそのストレージにマッピングされます。特別な値 1 を指定すると、各ソース・ストレージはそれ自体にマッピングされます。

**--タイムアウト <整数> (デフォルト= 180)**

再移行のためのシャットダウンのタイムアウト時間（秒）

**pct マウント <vmid>**

コンテナのファイルシステムをホストにマウントします。これはコンテナに対するロックを保持し、コンテナに対する起動と停止以外の操作を防止するため、緊急時の保守のみを目的としています。

**<vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

**pct move-volume <vmid> <volume> [<storage>] [<target-vmid>] [<target-volume>] [OPTIONS].**

rootfs/mp-ボリュームを別のストレージまたは別のコンテナに移動します。

**<vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

<volume>:<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| unused114| unused115| unused116| unused117| 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134| 未使用135| 未使用136| 未使用137| 未使用138| 未使用139| 未使用未使用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名| 未使用166名| 未使用167名| 未使用168名| 未使用169名| 未使用170名| 未使用171名| 未使用172名| 未使用173名| 未使用174名| 未使用175名| 未使用176名| 未使用177名| 未使用178名| 未使用179名| 未使用18名| 未使用180名| 未使用181名| 未使用182名| 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

移動するボリューム。

<ストレージ>: <ストレージID>

対象ストレージ

<ターゲット-vmid>: <整数> (100 - 999999999)

VMの（一意の）ID。

<ターゲットボリューム>: <mp0| mp1| mp10| mp100| mp101| mp102| mp103 | mp104 | mp105 | mp106 | mp107 | mp108 | mp109 | mp11 | 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 | 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 | unused114 | unused115 | unused116 | unused117 | unused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 | 未使用未使用160 | 未使用161 | 未使用162 | 未使用163 | 未使用164 | 未使用165 | 未使用166 | 未使用167 | 未使用168 | 未使用169 | 未使用17 | 未使用170 | 未使用171 | 未使用172 | 未使用173 | 未使用174 | 未使用175 | 未使用176 | 未使用177 | 未使用178 | 未使用179 | 未使用18 | 未使用180 | 未使用181 | 未使用未使用182 | 未使用183 | 未使用184 | 未使用185 | 未使用186 | 未使用未使用187 | 未使用188 | 未使用189 | 未使用19 | 未使用190 | 未使用未使用191 | 未使用192 | 未使用193 | 未使用194 | 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

ボリュームの移動コンフィグキー。デフォルトは移動元のボリュームキーです。

**--bwlimit <number> (0 - N) (デフォルト= データセンターまたはストレージ設定からのクローン制限)**

I/O帯域幅制限のオーバーライド（単位: KiB/s）。

**--delete <boolean> (デフォルト= 0)**

コピー成功後、元のボリュームを削除します。デフォルトでは、オリジナルは未使用のボリューム・エントリとして保持されます。

#### -ダイジェスト <文字列>

現在の設定ファイルのSHA1"."ダイジェストを使用します。これは同時変更を防ぐために使用できます。

#### -ターゲット・ダイジェスト <文字列>

ターゲット"."コンテナの現在の設定ファイルが異なる SHA1 ダイジェストを持っている場合に変更を防止します。これは、" ."同時変更を防ぐことができます。

#### pct 移動量

*pct move-volume*のエイリアス。

#### pct 保留中 <vmid>

保留中の含むコンテナ構成を取得します。

**<vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

**pct pull <vmid> <path> <destination> [OPTIONS] .**

コンテナからローカルファイルをコピー。

**<vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

#### <path>: <文字列>

プルするコンテナ内のファイルへのパス。

#### <目的地>: <文字列>

目的地

#### --グループ <文字列>

所有者のグループ名または ID。

#### --perms <文字列>

使用するファイルパーミッション（デフォルトは8進数、16進数の場合は先頭に0xを付けます）。

**--ユーザー <文字列>**

所有者のユーザー名またはID。

**pct push <vmid> <file> <destination> [OPTIONS].**

ローカルファイルをコンテナにコピー。

<vmid>: <整数> (100 - 999999999)

VMの（一意の）ID。

**<ファイル>: <文字列>**

ローカルファイルへのパス。

**<目的地>: <文字列>**

コンテナ内の書き込み。

**--グループ <文字列>**

オーナー・グループ名またはID。名前を使用する場合は、コンテナ内に存在する必要があります。

**--perms <文字列>**

使用するファイルパーミッション（デフォルトは8進数、16進数の場合は先頭に0xを付けます）。

**--ユーザー <文字列>**

所有者のユーザー名またはID。名前を使用する場合は、コンテナ内に存在する必要があります。

**pct reboot <vmid> [OPTIONS].**

コンテナをシャットダウンします。保留中の適用します。

<vmid>: <整数> (100 - 999999999)

VMの（一意の）ID。

**--タイムアウト <整数> (0 - N)**

シャットダウンまで最大タイムアウト秒数待ちます。

**pct remote-migrate <vmid> [<target-vmid>] <target-endpoint> --target-bridge <string> --target-storage <string> [OPTIONS].**

コンテナをリモートクラスタに移行します。新しいマイグレーションタスクを作成します。EXPERIMENTAL機能！

<vmid>: <整数> (100 - 999999999)

VMの（一意の）ID。

<ターゲット-vmid>: <整数> (100 - 999999999)

VMの（一意の）ID。

<ターゲットエンドポイント>: apitoken=<PVEAPIToken=user@realm! token=SECRET>。

ホスト=<ADDRESS> [, フィンガープリント=<FINGERPRINT>] [, ポート=<PORT>]。

リモートターゲットエンドポイント

--bwlimit <integer> (0 - N) (デフォルト= データセンターまたはストレージ設定からのマイグレーション制限)

I/O帯域幅制限のオーバーライド (単位: KiB/s)。

--delete <boolean> (デフォルト= 0)

移行成功後、元のCTと関連データを削除します。デフォルトでは、元のCTは停止状態で移行元クラスタに保持されます。

--オンライン <ブール値

オンライン/ライブマイグレーションを使用します。

--restart <ブール値

リスタートマイグレーションを使用

--ターゲットブリッジ <文字列

ソース・ブリッジからターゲット・ブリッジへのマッピング。単一のブリッジ IDだけを指定すると、すべてのソース・ブリッジがそのブリッジにマッピングされます。特別な値 1 を指定すると、各ソース・ブリッジはそれ自身にマッピングされます。

-ターゲットストレージ <文字列

ソースストレージからターゲットストレージへのマッピング。単一のストレージIDのみを指定すると、すべてのソース・ストレージがそのストレージにマッピングされます。特別な値 1 を指定すると、各ソース・ストレージはそれ自体にマッピングされます。

--タイムアウト <整数> (デフォルト= 180)

再移行のためのシャットダウンのタイムアウト時間 (秒)

pct rescan [OPTIONS]

すべてのストレージを再スキャンし、ディスクサイズと未使用ディスク更新します。

--dryrun <boolean> (デフォルト= 0)

実際にコンフィギュレーションに変更を書き込まないでください。

--vmid <整数> (100 - 999999999)

VMの（一意の）ID。

pct resize <vmid> <disk> <size> [OPTIONS].

コンテナマウントポイントのサイズを変更します。

<vmid>: <整数> (100 - 999999999)

VMの（一意の）ID。

<ディスク>: <mp0 | mp1 | mp10 | mp100 | mp101 | mp102 | mp103 | mp104 | mp105 | mp106 | mp107 | mp108 | mp109 | mp11 | mp110 | mp111 | mp112 | mp113 | mp114 | mp115 | mp116 | mp117 | mp118 | mp119 | mp12 | mp120 | mp121 | mp122 | mp123 | mp124 | mp125 | mp126 | mp127 | mp128 | mp129 | mp13 | mp130 | mp131 | mp132 | mp123 | mp124 | mp125 | mp126 | mp127 | mp128 | mp129 | mp123 | mp124 | mp125 | mp126 | mp127 | mp128 | mp129 | mp123 | 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>

リサイズしたいディスク。

<サイズ>です: \+?\d+(\.\d+)? [KMGT] ?

新しいサイズ。+記号を付けると、値はボリュームの実際のサイズに加算され、付けないと、値は絶対値となります。ディスク・サイズの縮小はサポートされていません。

### -ダイジェスト <文字列>

現在の設定ファイルの SHA1 ダイジェストが異なる場合に変更を防止します。これは同時変更を防ぐために使用できます。

**pct restore** <vmid> <ostemplate> [OPTIONS].

コンテナを作成または復元します。

<vmid>: <整数> (100 - 999999999)

VMの（一意の）ID。

**<ostemplate>: <文字列>**

OSテンプレートまたはバックアップファイル。

--arch <amd64| arm64| armhf| i386| riscv32| riscv64> (= amd64)

OSアーキテクチャの種類。

--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リストア制限)

I/O帯域幅制限のオーバーライド（単位: KiB/s）。

--cmode <コンソール| シェル| tty> (デフォルト= tty)

コンソールモード。デフォルトでは、consoleコマンドは利用可能なttyデバイスの1つに接続を開こうとします。cmodeをconsoleに設定すると、代わりに /dev/console にアタッチしようとします。cmodeを shell に設定すると、コンテナ内で単にシェルを起動します（ログインはしません）。

--コンソール <ブール値> (デフォルト= 1)

コンテナにコンソールデバイス (/dev/console) をアタッチします。

--コア数 <整数> (1 - 8192)

コンテナに割り当てられたコアの数。コンテナは、デフォルトで使用可能なすべてのコアを使用できます。

--cpulimit <number> (0 - 8192) (デフォルト= 0)

CPU使用量の上限。

---

**備考**

コンピュータに2つのCPUがある場合、合計2つのCPU時間があります。値0はCPU制限なしを示します。

---

--cpuunits <integer> (0 - 500000) (デフォルト= cgroup v1: 1024, cgroup v2: 100)

コンテナのCPU重量は、cgroup v2 では [1, 10000] にクランプされます。

--debug <boolean> (デフォルト= 0)

もっと冗長にしてみてください。今のところ、これは起動時にデバッグログレベルを有効にするだけです。

--説明 <文字列>

コンテナの説明。WebインターフェイスCTのサマリーに表示されます。これは設定ファイルのコメントとして保存されます。

--dev[n] [[path=</パス> [,deny-write=<1|0>] [,gid=<integer>] [,mode=<オクタルアクセスモード>] [,uid=<integer>] [,n  
コンテナへの通過装置

--features [force\_rw\_sys=<1|0>] [,fuse=<1|0>] [,keyctl=<1|0>] [,mknod=<1|0>]  
[,mount=<fstype;fstype;...>] [,nesting=<1|0>] となります。

コンテナが高度なアクセスできるようにします。

**--force <ブール値>**

既存のコンテナの上書きを許可します。

**--hookscript <文字列>**

コンテナのライフタイムのさまざまなステップで実行されるスクリプト。

**--ホスト名 <文字列>**

コンテナのホスト名を設定します。

**--ignore-unpack-errors <ブール値>**

テンプレートの抽出時にエラーを無視します。

**--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

**--メモリ <整数> (16 - N) (デフォルト=512)**

コンテナのRAM容量 (MB)。

**--mp[n] [volume=<ボリューム>,mp=<パス> [,acl=<1|0>] [,backup=<1|0>]  
[,mountoptions=<opt[;opt...]>] [,quota=<1|0>] [,replicate=<1|0>] [,ro=<1|0>]  
[,shared=<1|0>]  
,size=<DiskSize[quota=<1|0>][,replicate=<1|0>][,ro=<1|0>][,shared=<1|0>][,size=<DiskSize>]**。

ボリュームをコンテナ・マウント・ポイントとして使用します。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。

**--ネームサーバー <文字列>**

コンテナの DNS サーバー IP アドレスを設定します。searchdomainもnameserverも設定しない場合、Createは自動的にホストの設定を使用します。

**--net[n] name=<string> [,bridge=<bridge>] [,firewall=<1|0>] [,gw=<GatewayIPv4>]  
[,gw6=<GatewayIPv6>] [,hwaddr=<XX:XX:XX:XX:XX:XX>] [,ip=<(IPv4/CIDR|dhcp|manual)>]  
[,ip6=<(IPv6/CIDR|auto|dhcp|manual)>] [,link\_down=<1|0>] [,mtu=<integer>] [,rate=<mbps>]  
[,tag=<integer>] [,trunks=<vlanid[;vlanid...]>] [,type=<veth>]**

コンテナのネットワーク・インターフェースを指定します。

**--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。これは、コンテナ内の設定を行うために使用され、/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* /は、OS 固有のセットアップをスキップするために使用できます。

**--パスワード <password>**

コンテナ内の root パスワードを設定します。

**--プール <文字列>**

VMを指定したプールに追加します。

**--プロテクション <ブール値> (デフォルト= 0)**

コンテナの保護フラグを設定します。これにより、CT または CT のディスクの削除/更新操作を防止します。

**--rootfs [ボリューム=>ボリューム] [,acl=<1|0>] [,mountoptions=<opt[:opt...]>] [,quota=<1|0>] [,replicate=<1|0>] [,ro=<1|0>] [,shared=<1|0>]**

**[,size=<DiskSize[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。

**--検索ドメイン <文字列>**

コンテナのDNS検索ドメインを設定します。searchdomainもnameserverも設定しない場合、Createはホストからの設定を自動的に使用します。

**-ssh-公開鍵 <ファイルパス>**

公開SSH鍵を設定します（1行に1つの鍵、OpenSSH形式）。

**--start <boolean> (デフォルト= 0)**

CTの作成が正常に終了したら、CTを開始します。

**--startup `[[order=] \d+] [,up=d+] [,down=d+`**

スタートアップとシャットダウンの動作。Orderは一般的な起動順序を定義する非負の数値です。シャットダウンは逆の順序で行われます。さらに、次のVMが起動または停止するまでの待ち時間を秒単位で設定できます。

**--storage <ストレージID> (デフォルト= ローカル)**

デフォルトのストレージ。

**--スワップ <整数> (0 - N) (デフォルト= 512)**

MB単位のコンテナのSWAP量。

**--タグ <文字列>**

コンテナのタグ。これは単なるメタ情報です。

**--テンプレート <ブール値> (デフォルト= 0)**

テンプレートの有効/無効。

**--タイムゾーン <文字列>**

コンテナで使用するタイムゾーン。オプションが設定されていない場合は、何も行われません。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 から任意のタイムゾーンオプションを設定することができます。

**--tty <整数> (0 - 6) (デフォルト=2)**

コンテナで利用可能なttyの数を指定

**--unique <ブール値>**

一意のランダムなイーサネットアドレスを割り当てます。

**備考**

必要なオプション: リストア

**--unprivileged <boolean> (デフォルト= 0)**

コンテナを非特権ユーザーとして実行します。(手動で変更してはいけません)。

**--未使用[n] [ボリューム=]<ボリューム>**

未使用ボリュームへの参照。これは内部的に使用されるので、手動で変更しないでください。

**pct レジューム <vmid>**

コンテナを再開します。

**<vmid>: <整数> (100 - 999999999)**

VMの(一意の)ID。

**pct rollback <vmid> <スナップ名> [オプション].**

LXCの状態を指定したスナップショットにロールバックします。

**<vmid>: <整数> (100 - 999999999)**

VMの(一意の)ID。

**<スナップ名>: <文字列>**

スナップショットの名前。

**--start <boolean> (デフォルト= 0)**

ロールバック成功後にコンテナを起動するかどうか

**pct set <vmid> [OPTIONS].**

コンテナのオプションを設定します。

**<vmid>: <整数> (100 - 999999999)**

VMの(一意の)ID。

**--arch <amd64| arm64| armhf| i386| riscv32| riscv64> (= amd64)**

OSアーキテクチャの種類。

**--cmode <コンソール| シェル| tty> (デフォルト= tty)**

コンソールモード。デフォルトでは、consoleコマンドは利用可能なttyデバイスの1つに接続を開こうとします。cmodeをconsoleに設定すると、代わりに/dev/consoleにアタッチしようとします。cmodeをshellに設定すると、コンテナ内で単にシェルを起動します(ログインはしません)。

**--コンソール <ブール値> (デフォルト= 1)**

コンテナにコンソールデバイス (/dev/console) をアタッチします。

**--コア数 <整数> (1 - 8192)**

コンテナに割り当てられたコアの数。コンテナは、デフォルトで使用可能なすべてのコアを使用できます。

**--cpulimit <number> (0 - 8192) (デフォルト= 0)**

CPU使用量の上限。

**備考**

コンピュータに2つのCPUがある場合、合計2つのCPU時間があります。値0はCPU制限なしを示します。

**--cpuunits <integer> (0 - 500000) (デフォルト= cgroup v1: 1024, cgroup v2: 100)**

コンテナのCPU重量は、cgroup v2では[1, 10000]にクランプされます。

**--debug <boolean> (デフォルト= 0)**

もっと冗長にしてみてください。今のところ、これは起動時にデバッグログレベルを有効にするだけです。

**--削除 <文字列>**

削除したい設定のリスト。

**--説明 <文字列>**

コンテナの説明。WebインターフェイスCTのサマリーに表示されます。これは設定ファイルのコメントとして保存されます。

**--dev[n] [[path=<パス> [,deny-write=<1|0>] [,gid=<integer>] [,mode=<オクタルアクセスモード>] [,uid=<integer>] [,n**

コンテナへの通過装置

**-ダイジェスト <文字列>**

現在の設定ファイルのSHA1ダイジェストが異なる場合に変更を防止します。これは同時変更を防ぐために使用できます。

**--features [force\_rw\_sys=<1|0>] [,fuse=<1|0>] [,keyctl=<1|0>] [,mknod=<1|0>] [,mount=<fstype;fstype;...>] [,nesting=<1|0>]**

となります。

コンテナが高度なアクセスできるようにします。

**--hookscript <文字列>**

コンテナのライフタイムのさまざまなステップで実行されるスクリプト。

**--ホスト名 <文字列>**

コンテナのホスト名を設定します。

```
--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

```
--メモリ <整数> (16 - N) (デフォルト=512)
```

コンテナのRAM容量 (MB)。

```
--mp[n] [volume=<ボリューム>, mp=<パス> [,acl=<1|0>] [,backup=<1|0>]  
[,mountoptions=<opt[;opt...]>] [,quota=<1|0>] [,replicate=<1|0>] [,ro=<1|0>]  
[,shared=<1|0>]
```

```
[,size=<DiskS[quota=<1|0>][,replicate=<1|0>][,ro=<1|0>][,shared=<1|0>][,size=<DiskSize>]]。
```

ボリュームをコンテナ・マウント・ポイントとして使用します。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。

```
--ネームサーバー <文字列>
```

コンテナの DNS サーバー IP アドレスを設定します。searchdomain も nameserver も設定しない場合、Create は自動的にホストの設定を使用します。

```
--net[n] name=<string> [,bridge=<bridge>] [,firewall=<1|0>] [,gw=<GatewayIPv4>]  
[,gw6=<GatewayIPv6>] [,hwaddr=<XX:XX:XX:XX:XX:XX>] [,ip=<(IPv4/CIDR|dhcp|manual)>]  
[,ip6=<(IPv6/CIDR|auto|dhcp|manual)>] [,link_down=<1|0>] [,mtu=<integer>] [,rate=<mbps>]  
[,tag=<integer>] [,trunks=<vlanid[;vlanid...]>] [,type=<veth>]
```

コンテナのネットワーク・インターフェースを指定します。

```
--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。これは、コンテナ内の設定を行うために使用され、/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 は、OS 固有のセットアップをスキップするために使用できます。

```
--プロテクション <ブール値> (デフォルト= 0)
```

コンテナの保護フラグを設定します。これにより、CT または CT のディスクの削除/更新操作を防止します。

```
--復帰 <文字列>
```

保留中の元に戻します。

```
--rootfs [ボリューム=<ボリューム> [,acl=<1|0>] [,mountoptions=<opt[;opt...]>] [,quota=<1|0>]  
[,replicate=<1|0>] [,ro=<1|0>] [,shared=<1|0>]  
[,size=<DiskS[quota=<1|0>][,replicate=<1|0>][,ro=<1|0>][,shared=<1|0>][,size=<DiskSize>]]。
```

ボリュームをコンテナのルートとして使用します。

**--検索ドメイン <文字列>**

コンテナのDNS検索ドメインを設定します。searchdomainもnameserverも設定しない場合、Createはホストからの設定を自動的に使用します。

**--startup [[order=] \d+] [,up=d+] [,down=d+]**

スタートアップとシャットダウンの動作。Orderは一般的な起動順序を定義する非負の数値です。シャットダウンは逆の順序で行われます。さらに、次のVMが起動または停止するまでの待ち時間を秒単位で設定できます。

**--スワップ <整数> (0 - N) (デフォルト = 512)**

MB単位のコンテナのSWAP量。

**--タグ <文字列>**

コンテナのタグ。これは単なるメタ情報です。

**--テンプレート <ブール値> (デフォルト = 0)**

テンプレートの有効/無効。

**--タイムゾーン <文字列>**

コンテナで使用するタイムゾーン。オプションが設定されていない場合は、何も行われません。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から任意のタイムゾーンオプションを設定することができます。

**--tty <整数> (0 - 6) (デフォルト = 2)**

コンテナで利用可能なttyの数を指定

**--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。(手動で変更してはいけません)。

**--未使用[n] [ボリューム=>ボリューム**

未使用ボリュームへの参照。これは内部的に使用されるので、手動で変更しないでください。

**pct shutdown <vmid> [OPTIONS].**

コンテナをシャットダウンします。詳細はlxc-stop(1)を参照。

**<vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

**--forceStop <boolean> (デフォルト = 0)**

コンテナが停止していることを確認してください。

**--タイムアウト <整数> (0 - N) (デフォルト = 60)**

最大タイムアウト秒数まで待ちます。

**pct snapshot <vmid> <snapname> [OPTIONS].**

コンテナをスナップショットします。

**<vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

**<スナップ名>: <文字列>**

スナップショットの名前。

**--説明 <文字列>**

テキストによる説明やコメント。

**pct start <vmid> [OPTIONS] .**

コンテナを起動します。

**<vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

**--debug <boolean> (デフォルト= 0)**

設定すると、起動時に非常に冗長なデバッグ・ログ・レベルを有効にします。

**--スキップロック <ブール値>**

Ignore locks - rootのみがこのオプションを使用できます。

**pct status <vmid> [OPTIONS] [オプション] .**

CTステータスを表示します。

**<vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

**--verbose <ブール値>**

冗長出力フォーマット

**pct stop <vmid> [OPTIONS] .**

コンテナを停止します。これにより、コンテナ内で実行されているすべてのプロセスが突然停止します。

**<vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

**--overrule-shutdown <boolean> (デフォルト= 0)**

停止する前に、アクティブなvzshutdownタスクを中止してみてください。

**--スキップロック <ブール値>**

Ignore locks - rootのみがこのオプションを使用できます。

**pct サスペンド <vmid>**

コンテナを一時停止します。これは実験的なものです。

**<vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

**pct テンプレート <vmid>**

テンプレートを作成します。

**<vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

**pct アンロック <vmid>**

VMのロックを解除します。

**<vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

**pct アンマウント <vmid>**

コンテナのファイルシステムをアンマウントします。

**<vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

## A.12 pveam - Proxmox VE アプライアンスマネージャ

**pveam <COMMAND> [ARGS] [OPTIONS]** です。

**pveamあり [OPTIONS]**

利用可能な一覧表示します。

**--section <mail| system| turnkeylinux>**

リストを指定したセクションに制限します。

**pveam ダウンロード <ストレージ> <テンプレート>**

アプライアンスのテンプレートをダウンロード

**<ストレージ>: <ストレージID>**

テンプレートが保存されるストレージ

**<テンプレート>: <文字列>**

ダウンロードされるテンプレート

**pveam help [OPTIONS]**

指定したコマンドに関するヘルプを表示します。

**--extra-args <array>** です。

特定のコマンドのヘルプを表示します。

**--verbose <ブール値**

冗長出力フォーマット。

**pveam リスト <ストレージ**

ストレージ上の全テンプレートのリストを取得

**<ストレージ>: <ストレージID**

指定したストレージ上のテンプレートのみをリストアップ

**pveam remove <template\_path>** です。

テンプレートを削除します。

**<template\_path>: <文字列**

するテンプレート。

**pveamアップデート**

コンテナテンプレートデータベースの更新

## A.13 pvecm - Proxmox VE クラスタマネージャ

**pvecm <COMMAND> [ARGS] [OPTIONS]** です。

**pvecm add <ホスト名> [OPTIONS].**

現在のノードを既存のクラスタに追加します。

**<ホスト名>: <文字列**

既存のクラスタ・メンバーのホスト名（またはIP）。

**--fingerprint ([A-Fa-f0-9]{2}:){31}[A-Fa-f0-9]{2}**

証明書の SHA 256 フィンガープリント。

**--force <ブール値**

ノードが既に存在する場合はエラーをスローしません。

-リンク [n] [アドレス=<IP> [, 優先度=<整数>]。

1つのコロシンク・リンクのアドレスとプライオリティ情報。(最大8リンクまでサポート; link0..link7)

--nodeid <整数> (1 - N)

このノード ID。

--use\_ssh <ブール値>

相手がAPI経由で参加する場合でも、常にSSHを使用してください。

--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です。

<ノード>: <文字列>

クラスタ・ノード名。

--apiversion <整数>

新しいノードの JOIN\_API\_VERSION。

--force <ブール値>

ノードが既に存在する場合はエラーをスローしません。

-リンク [n] [アドレス=<IP> [, 優先度=<整数>]。]

1つのコロシング・リンクのアドレスとプライオリティ情報。(最大8リンクまでサポート; link0..link7)

--new\_node\_ip <文字列>

追加するノードのIPアドレス。ない場合のフォールバックとして使用されます。

--nodeid <整数> (1 - N)

このノード ID。

--votes <整数> (0 - N)

このノードの投票数

**pvecm** アビバー

このノードで利用可能なクラスタ結合 API のバージョンを返します。

**pvecm create** <clustername> [OPTIONS] [オプション].

新しいクラスタ構成を生成します。リンクが指定されていない場合、デフォルトでローカルIPアドレスがlink0になります。

<クラスタ名>: <文字列>

クラスタの名前。

-リンク [n] [アドレス=<IP> [, 優先度=<整数>]。

1つのコロシンク・リンクのアドレスとプライオリティ情報。(最大8リンクまでサポート; link0..link7)

--nodeid <整数> (1 - N)

このノード ID。

--投票数 <整数> (1 - N)

このノードの投票数。

**pvecm delnode** <ノード>。

クラスタ構成からノードを削除します。

<ノード>: <文字列>

クラスタ・ノード名。

**pvecm 予想** <予想>

corosyncに期待される投票数の新しい値を伝えます。

<予想>: <整数> (1 - N)

予想得票数

**pvecm help** [OPTIONS]

指定したコマンドに関するヘルプを表示します。

--extra-args <array> です。

特定のコマンドのヘルプを表示します。

--verbose <ブール値>

冗長出力フォーマット。

**pvecm keygen** <ファイル名>.

corosync用の新しい暗号鍵を生成します。

<ファイル名>: <文字列>

出力ファイル名

**pvecm mtunnel** [<extra-args>] [OPTIONS].

VM/CT使用します。

<extra-args>: <array> です。

配列としての追加引数

--get\_migration\_ip <boolean> (デフォルト= 0)

設定されている場合、マイグレーションIPを返します

**--マイグレーションネットワーク <文字列>**

ローカルマイグレーションIPの検出に使用されるマイグレーションネットワーク

**--run-command <ブール値>**

tcpソケットを標準入力としてコマンドを実行します。IPアドレスとポートが次のように表示されます。

コマンドの標準出力を最初に、それぞれ別の行に出力します。

**pvecmノード**

クラスタノードのローカルビューを表示します。

**pvecm qdevice remove**

設定されたQDeviceの削除

**pvecm qdevice setup <アドレス> [オプション] .**

QDeviceの使用設定

**<アドレス>: <文字列>**

外部corosync QDeviceのネットワークアドレスを指定します。

**--force <ブール値>**

危険な可能性のある操作でエラーを投げないでください。

**--ネットワーク <文字列>**

外部qdeviceへの接続に使用するネットワーク。

**pvecmステータス**

クラスタステータスのローカルビューを表示します。

**pvecm updatecerts [OPTIONS] (証明書の更新)**

ノード証明書を更新します（必要なファイル/ディレクトリをすべて生成します）。

**--force <ブール値>**

新しいSSL証明書を強制的に生成します。

**--サイレント <ブール値>**

エラーを無視します（クラスタに定足数がない場合など）。

**--unmerge-known-hosts <boolean> (デフォルト= 0)**

レガシーSSH既知のホストをアンマージします。

## A.14 pvesr - Proxmox VEストレージレプリケーション

**pvesr <COMMAND> [ARGS] [OPTIONS]** です。

**pvesr create-local-job <id> <target> [OPTIONS] .**

新しいレプリケーション・ジョブの作成

```
<id>: [1-9][0-9]{2,8}-\d{1,9}
レプリケーション・ジョブID。IDはゲストIDとジョブ番号で構成され、ハイフンで区切られます。
<ゲスト>-<ジョブ名>.
```

#### <ターゲット>: <文字列>

対象ノード

#### --コメント <文字列>

説明

#### --無効 <ブール値>

エントリーを無効化/非活性化するフラグ。

#### --レート <数値> (1 - N)

Mbps (メガバイト/秒) 単位のレート制限を浮動小数点数で指定します。

#### --remove\_job <full| local>

レプリケーション・ジョブを削除するようにマークします。このジョブはすべてのローカルレプリケーションスナップショットを削除します。フルに設定すると、ターゲット上のレプリケートされたボリュームも削除しようとします。その後、ジョブは構成ファイルから自身を削除します。

#### --スケジュール <文字列> (デフォルト= \*/15)

ストレージのレプリケーションスケジュール。このフォーマットは `systemd` カレンダーサブセットです。

#### --ソース <文字列>

ゲストが盗まれたかどうかを検出するための内部使用。

```
pvesr delete <id> [OPTIONS].
```

削除するレプリケーション・ジョブをマークします。

#### <id>: [1-9][0-9]{2,8}-\d{1,9}

```
レプリケーション・ジョブID。IDはゲストIDとジョブ番号で構成され、ハイフンで区切られます。
<ゲスト>-<ジョブ名>.
```

#### --force <boolean> (デフォルト= 0)

jobconfig エントリを削除しますが、クリーンアップはしません。

#### --keep <boolean> (デフォルト= 0)

レプリケートされたデータをターゲットに保持します（削除しないでください）。

```
pvesr disable <id>
```

レプリケーション・ジョブを無効にします。

#### <id>: [1-9][0-9]{2,8}-\d{1,9}

```
レプリケーション・ジョブID。IDはゲストIDとジョブ番号で構成され、ハイフンで区切られます。
<ゲスト>-<ジョブ名>.
```

**pvesr enable <id>**

レプリケーション・ジョブを有効にします。

**<id>: [1-9] [0-9]{2,8}-\d{1,9}**

レプリケーション・ジョブID。IDはゲストIDとジョブ番号で構成され、ハイフンで区切られます。

<ゲスト>-<ジョブ名>.

**pvesr finalize-local-job <id> [<extra-args>] [OPTIONS].**

レプリケーション・ジョブを確定します。これにより、<last\_sync>と異なるタイムスタンプを持つすべてのレプリケーション・スナップショットが削除されます。

**<id>: [1-9] [0-9]{2,8}-\d{1,9}**

レプリケーション・ジョブID。IDはゲストIDとジョブ番号で構成され、ハイフンで区切られます。

<ゲスト>-<ジョブ名>.

**<extra-args>: <array> です。**

検討するボリュームIDのリスト。

**--last\_sync <整数> (0 - N)**

最後に同期に成功した時刻（UNIXエポック）。指定しない場合は、すべてのレプリケーションスナップショットが削除されます。

**pvesr help [OPTIONS]**

指定したコマンドに関するヘルプを表示します。

**--extra-args <array> です。**

特定のコマンドのヘルプを表示します。

**--verbose <ブール値>**

冗長出力フォーマット。

## pvesrリスト

レプリケーション・ジョブを一覧表示します。

**pvesr prepare-local-job <id> [<extra-args>] [OPTIONS].**

レプリケーション・ジョブの開始準備。これはレプリケーション開始前にターゲットノードで呼び出されます。この呼び出しは内部用で、標準出力にJSONオブジェクトを返します。このメソッドは、まず VM <vmid> ガ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、このメソッドはスナップショットについてすべてのボリュームIDをスキャン、<last\_sync>と異なるタイムスタンプを持つすべてのレプリケーション・スナップショットを削除します。また、未使用のボリュームも削除します。既存のレプリケーション・スナップショットを持つすべてのボリュームのブール値マーカーを持つハッシュを返します。

**<id>: [1-9] [0-9]{2,8}-\d{1,9}**

レプリケーション・ジョブID。IDはゲストIDとジョブ番号で構成され、ハイフンで区切られます。

<ゲスト>-<ジョブ名>.

**<extra-args>: <array> です。**

検討するボリューム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できます（空のボリュームリスト）。

**--last\_sync <整数> (0 - N)**

最後に同期に成功した時刻（UNIXエポック）。指定しない場合は、すべてのレプリケーション・スナップショットが削除されます。

**-親スナップ名 <文字列>**

スナップショットの名前。

**--スキャン <文字列>**

古くなったボリュームをスキャンするストレージIDのリスト。

**pvesr read <id>**

レプリケーション・ジョブの設定を読み取ります。

**<id>: [1-9][0-9]{2,8}-\d{1,9}**

レプリケーション・ジョブID。IDはゲストIDとジョブ番号で構成され、ハイフンで区切られます。

<ゲスト>-<ジョブ名>.

**pvesr run [OPTIONS]**

このメソッドは systemd-timer によって呼び出され、すべての（または特定の）同期ジョブを実行します。

**--id [1-9][0-9]{2,8}-\d{1,9}**

レプリケーション・ジョブID。IDはゲストIDとジョブ番号で構成され、ハイフンで区切られます。

<ゲスト>-<ジョブ名>.

**--メール <ブール値> (デフォルト= 0)**

障害発生時にメール通知を送信します。

**--verbose <boolean> (デフォルト= 0)**

標準出力に詳細なログを出力します。

**pvesr schedule-now <id>**

レプリケーション・ジョブをできるだけ早く開始するようにスケジュールします。

**<id>: [1-9][0-9]{2,8}-\d{1,9}**

レプリケーション・ジョブID。IDはゲストIDとジョブ番号で構成され、ハイフンで区切られます。

<ゲスト>-<ジョブ名>.

**pvesr set-state <vmid> <state> です。**

マイグレーション時にジョブのレプリケーション状態を設定します。この呼び出しは内部用です。ジョブの状態をja JSON objとして受け取ります。

**<vmid>: <整数> (100 - 999999999)**

VMの（一意の）ID。

**<状態>: <文字列**

JSON デコードされた文字列としてのジョブ状態。

**pvesr status [OPTIONS]**

このノードのすべてのレプリケーション・ジョブのステータスを一覧表示します。

**--ゲスト <整数> (100 - 999999999)**

このゲストのレプリケーションジョブのみをリストします。

**pvesr update <id> [OPTIONS] .**

レプリケーション・ジョブ構成を更新します。

**<id>: [1-9][0-9]{2,8}-\d{1,9}**

レプリケーション・ジョブID。IDはゲストIDとジョブ番号で構成され、ハイフンで区切られます。

<ゲスト>-<ジョブ名>.

**--コメント <文字列**

説明

**--削除 <文字列**

削除したい設定のリスト。

**-ダイジェスト <文字列**

現在のコンフィギュレーション・ファイルのダイジェストが異なる場合、変更を防止します。これは、現在のコンフィギュレーション・ファイルが変更されないようにするために使用できます。

**--無効 <布尔值**

エントリーを無効化/非活性化するフラグ。

**--レート <数値> (1 - N)**

mbps (メガバイト/秒) 単位のレート制限を浮動小数点数で指定します。

**--remove\_job <full| local>**

レプリケーション・ジョブを削除するようにマークします。このジョブはすべてのローカルレプリケーションスナップショットを削除します。フルに設定すると、ターゲット上のレプリケートされたボリュームも削除しようとしています。その後、ジョブは構成ファイルから自身を削除します。

**--スケジュール <文字列> (デフォルト= \*/15)**

ストレージのレプリケーションスケジュール。このフォーマットは `systemd` カレンダーサブセットです。

**--ソース <文字列**

ゲストが盗まれたかどうかを検出するための内部使用。

## A.15 pveum - Proxmox VE ユーザーマネージャ

**pveum** <COMMAND> [ARGS] [OPTIONS] です。

**pveum acl delete** <path> --roles <string> [OPTIONS].

アクセスコントロールリストの更新（パーミッションの追加または削除）。

<path>: <文字列

アクセス制御パス

--グループ <文字列

グループ一覧。

--プロパゲート <ブール値> (デフォルト= 1)

パーミッションの伝搬（継承）を許可します。

--roles <文字列

役割の一覧。

--トークン <文字列

APIトークンのリスト。

--users <文字列

ユーザー一覧。

**pveum acl list** [FORMAT\_OPTIONS]。

アクセス制御リスト（ACL）を取得します。

**pveum acl modify** <path> --roles <string> [OPTIONS].

アクセスコントロールリストの更新（権限の追加または削除）。

<path>: <文字列

アクセス制御パス

--グループ <文字列

グループ一覧。

--プロパゲート <ブール値> (デフォルト= 1)

パーミッションの伝搬（継承）を許可します。

--roles <文字列

役割の一覧。

**-- トークン <文字列**

APIトークンのリスト。

**--users <文字列>**

ユーザー一覧。

**ペベウム・アクデル**

*pveum acl delete* のエイリアス。

**pveum aclmod**

*pveum acl modify* のエイリアス。

**pveum group add <groupid> [OPTIONS] (グループ追加 <グループID>[オプション])**

新しいグループを作成します。

**<groupid>: <文字列>**

説明なし

**--コメント <文字列>**

説明なし

**pveum グループ削除 <グループID>**

グループを削除します。

**<groupid>: <文字列>**

説明なし

**pveum グループ・リスト [FORMAT\_OPTIONS].**

グループインデックス

**pveum group modify <groupid> [OPTIONS] [オプション].**

グループデータを更新します。

**<groupid>: <文字列>**

説明なし

**--コメント <文字列>**

説明なし

**グループ追加**

*pveum group add* のエイリアス。

**ペベウム・グループデル**

*pveum group delete* のエイリアス。

**ペベウム・グループモッド**

*pveum group modify* のエイリアス。

**pveum help [OPTIONS]**

指定したコマンドに関するヘルプを表示します。

--extra-args <array> です。

特定のコマンドのヘルプを表示します。

--verbose <ブール値

冗長出力フォーマット。

**pveum passwd <ユーザーID> [OPTIONS]**.

ユーザーのパスワードを変更します。

<ユーザーID>: <文字列

name@realm完全なユーザーID。

--確認パスワード <文字列

変更を実行するユーザーの現在のパスワード。

**pveum pool add <poolid> [OPTIONS] (プール追加 <プール> [オプション])**

新しいプールを作成します。

<プール>: <文字列

説明なし

--コメント <文字列

説明なし

プール削除 <プール>

プールを削除します。

<プール>: <文字列

説明なし

**pveum プールリスト [OPTIONS] [FORMAT\_OPTIONS]**.

プールの一覧表示またはプール設定の取得

--プール <文字列

説明なし

--type <lxc| qemu| storage>

説明なし

---

#### 備考

必要なオプション: poolid

---

**pveum pool modify <poolid> [OPTIONS] [オプション].**

プールを更新。

#### <プール>: <文字列>

説明なし

#### --allow-move <boolean> (デフォルト= 0)

すでに別のプールにいる場合でも、ゲストを追加できるようにします。ゲストは現在のプールから削除され、このプールに追加されます。

#### --コメント <文字列>

説明なし

#### --delete <boolean> (デフォルト= 0)

渡されたVMIDやストレージIDを追加代わりに削除します。

#### --ストレージ <文字列>

このプールから追加または削除するストレージ ID のリスト。

#### --vms <文字列>

このプールから追加または削除するゲスト VMID のリスト。

**pveum realm add <realm> --type <string> [OPTIONS]** です。

認証サーバーを追加します。

#### <realm>: <文字列>

認証ドメインID

#### --acr-values[^x00-xxx1F <>#]\*,\$

認証サーバーが使用する認証コンテキスト・クラス参照値を指定します。

認証リクエストに使用するようリクエストされました。

#### --autocreate <boolean> (デフォルト= 0)

ユーザーが存在しない場合、自動的にユーザーを作成します。

#### --base\_dn <文字列>

LDAPベースドメイン名

#### --bind\_dn <文字列>

LDAPバインドドメイン名

#### --capath <string> (デフォルト= /etc/ssl/certs)

CA 証明書ストアへのパス

**--大文字小文字を区別する <論理値> (デフォルト= 1)**

ユーザー名は大文字と小文字を区別します。

**--証明書 <文字列>**

クライアント証明書へのパス

**--認証キー <文字列>**

クライアント証明書キーへのパス

**--check-connection <boolean> (デフォルト= 0)**

サーバーへのバインド接続を確認します。

**--クライアントID <文字列>**

OpenIDクライアントID

**--client-key <文字列>**

OpenIDクライアント・キー

**--コメント <文字列>**

説明

**--default <ブール値>**

デフォルトのレルムとして使用します。

**--ドメイン**

AD ドメイン名

**--フィルタ <文字列>**

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)**

グループのオブジェクトクラス。

**--group\_dn <文字列>**

グループ同期用のLDAPベース・ドメイン名。設定されていない場合は、base\_dnが使用されます。

**--グループフィルター <文字列>**

グループ同期用のLDAPフィルタ。

**--group\_name\_attr <文字列>**

グループ名を表すLDAP属性。設定されていない場合、または見つかった場合は、DN の最初の値が name として使用されます。

**--groups-autocreate <boolean> (デフォルト= 0)**

グループが存在しない場合は、自動的にグループを作成します。

--groups-claim (?^:[A-Za-z0-9\.\-\_]+)

グループを取得ために使用される OpenID クレーム。

--groups-overwrite <boolean> (デフォルト= 0)

すべてのグループはログイン時に上書きされます。

--issuer-url <文字列>

OpenID発行者URL

--mode <ldap| ldap+starttls| ldaps> (デフォルト= ldap)

LDAPプロトコルモード。

--パスワード <文字列>

LDAP バインドパスワード。*etc/pve/priv/realm/<REALM>.pw* に格納されます。

--ポート <整数> (1 - 65535)

サーバーポート

--プロンプト (?::none|login|consent|select\_account|S+)

認証サーバがエンドユーザに再認証と同意を求めるかどうかを指定します。

--query-userinfo <boolean> (デフォルト= 1)

クレーム値の userinfo エンドポイントへのクエリを有効にします。

--scopes <string> (デフォルト= 電子メールプロファイル)

認可され返されるべきスコープ(ユーザの詳細)を指定します。

--secure <ブール値>

安全な LDAPS プロトコルを使用します。廃止: 代わりに mode を使用してください。

--サーバ1 <文字列>

サーバーIPアドレス (またはDNS名)

--サーバー2 <文字列>

フォールバックサーバーのIPアドレス (またはDNS名)

--sslversion <tlsv1| tlsv1\_1| tlsv1\_2| tlsv1\_3>

LDAPS TLS/SSLのバージョン。1.2より古いバージョンを使用することはお勧めしません!

-sync-defaults-options [enable-new=<1|0>] [,full=<1|0>] [,purge=<1|0>] [,remove-vanished=([acl];[properties];[entry])|none] [,scope=<users|groups|both>] となります。

同期の動作に関するデフォルトのオプション。

```
--sync_attributes \w+=[^,]+(,\s*\w+=[^,]+)*
  どの LDAP 属性がどの PVE ユーザーに対応するかを指定するための key=value ペアのカンマ区切りリスト。
  フィールドを指定します。例えば、LDAP 属性 mail PVE の email にマッピングするには email=mail と記述します。デフォルトでは、各 PVE ユーザーフィールドは同じ名前の LDAP 属性で表されます。
```

```
--tfa type=<TFATYPE> [,digits=<COUNT>] [,id=<ID>] [,key=<KEY>] [,step=<SECONDS>] [,url=<URL>].
  二要素認証を使用してください。
```

```
--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)
  ユーザーのオブジェクトクラス。
```

```
--username-claim <string>
  一意のユーザー名を生成するために使用されるOpenIDクレーム。
```

```
--verify <boolean> (デフォルト= 0)
  サーバーのSSL証明書の確認
```

```
pveum realm delete <realm>
  認証サーバーを削除します。
```

```
<realm>: <文字列>
  認証ドメインID
```

```
pveum realm list [FORMAT_OPTIONS].
  認証ドメインインデックス。
```

```
pveum realm modify <realm> [OPTIONS] [オプション].
  認証サーバーの更新します。
```

```
<realm>: <文字列>
  認証ドメインID
```

```
--acr-values[^x00-xxxx1F <>#]*.$
  認証サーバが使用する認証コンテキスト・クラス参照値を指定します。
  認証リクエストに使用するようリクエストされました。
```

```
--autocreate <boolean> (デフォルト= 0)
  ユーザーが存在しない場合、自動的にユーザーを作成します。
```

**--base\_dn <文字列>**

LDAPベースドメイン名

**--bind\_dn <文字列>**

LDAPバインドドメイン名

**--capath <string> (デフォルト= /etc/ssl/certs)**

CA 証明書ストアへのパス

**--大文字小文字を区別する <論理値> (デフォルト= 1)**

ユーザー名は大文字と小文字を区別します。

**--証明書 <文字列>**

クライアント証明書へのパス

**--認証キー <文字列>**

クライアント証明書キーへのパス

**--check-connection <boolean> (デフォルト= 0)**

サーバーへのバインド接続を確認します。

**--クライアントID <文字列>**

OpenIDクライアントID

**--client-key <文字列>**

OpenIDクライアント・キー

**--コメント <文字列>**

説明

**--default <ブール値>**

デフォルトのレルムとして使用します。

**--削除 <文字列>**

削除したい設定のリスト。

**-ダイジェスト <文字列>**

現在のコンフィギュレーション・ファイルのダイジェストが異なる場合、変更を防止します。これは、現在のコンフィギュレーション・ファイルが変更されないようにするために使用できます。

**--ドメイン**

AD ドメイン名

**--フィルタ <文字列>**

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)  
グループのオブジェクトクラス。

--group\_dn <文字列

グループ同期用のLDAPベース・ドメイン名。設定されていない場合は、base\_dnが使用されます。

--グループフィルター <文字列

グループ同期用のLDAPフィルタ。

--group\_name\_attr <文字列

グループ名を表すLDAP属性。設定されていない場合、または見つかった場合は、DN の最初の値が name として使用されます。

--groups-autocreate <boolean> (デフォルト= 0)

グループが存在しない場合は、自動的にグループを作成します。

--groups-claim (?^:[A-Za-z0-9\.\-\_]+)

グループを取得ために使用される OpenID クレーム。

--groups-overwrite <boolean> (デフォルト= 0)

すべてのグループはログイン時に上書きされます。

--issuer-url <文字列

OpenID発行者URL

--mode <ldap| ldap+starttls| ldaps> (デフォルト= ldap)

LDAPプロトコルモード。

--パスワード <文字列

LDAP バインドパスワード。etc/pve/priv/realm/<REALM>.pw に格納されます。

--ポート <整数> (1 - 65535)

サーバーポート

--プロンプト (?:none|login|consent|select\_account|S+)

認証サーバがエンドユーザに再認証と同意を求めるかどうかを指定します。

--query-userinfo <boolean> (デフォルト= 1)

クレーム値の userinfo エンドポイントへのクエリを有効にします。

--scopes <string> (デフォルト= 電子メールプロファイル)

認可され返されるべきスコープ(ユーザの詳細)を指定します。

--secure <ブール値

安全な LDAPS プロトコルを使用します。廃止: 代わりに mode を使用してください。

**--サーバ1 <文字列>**

サーバーIPアドレス（またはDNS名）

**--サーバー2 <文字列>**

フォールバックサーバーのIPアドレス（またはDNS名）

**-sslversion <tlsver1| tlsv1\_1| tlsv1\_2| tlsv1\_3>**

LDAP TLS/SSLのバージョン。1.2より古いバージョンを使用することはお勧めしません！

**-sync-defaults-options [enable-new=<1|0>] [,full=<1|0>] [,purge=<1|0>] [,remove-vanished=([acl];[properties];[entry])|none] [,scope=<users|groups|both>]** となります。

同期の動作に関するデフォルトのオプション。

**--sync\_attributes \w+=[^,]+(,\s\*\w+=[^,]+)\***

どの LDAP 属性がどの PVE ユーザーに対応するかを指定するための key=value ペアのカンマ区切りリスト。

フィールドを指定します。例えば、LDAP 属性 mail PVE の email にマッピングするには email=mail と記述します。デフォルトでは、各 PVE ユーザーフィールドは同じ名前の LDAP 属性で表されます。

**--tfa type=<TFATYPE> [,digits=<COUNT>] [,id=<ID>] [,key=<KEY>] [,step=<SECONDS>] [,url=<URL>].**  
二要素認証を使用してください。**--user\_attr \S{2,}.**

LDAPユーザー属性名

**--user\_classes <string> (デフォルト= inetorgperson, posixaccount, person, user)**  
ユーザーのオブジェクトクラス。**--verify <boolean> (デフォルト= 0)**

サーバーのSSL証明書の確認

**pveum realm sync <realm> [OPTIONS] [オプション].**

設定されたLDAPからuser.cfgにユーザおよび/またはグループを同期します。注意: 同期されたグループは name-\$realm を上書きしないように、これらのグループが存在しないことを確認してください。

**<realm>: <文字列>**

認証ドメインID

**--dry-run <boolean> (デフォルト= 0)**

設定されている場合、何も書きません。

**--enable-new <boolean> (デフォルト= 1)**

新しく同期されたユーザーをすぐに有効にします。

**--full <ブール値>**

廃止: 代わりに *remove-vanished* を使用してください。設定されている場合、真実のソースとしてLDAPディレクトリを使用し、同期から返されなかったユーザまたはグループを削除し、同期されたユーザのローカルで変更されたすべてのプロパティを削除します。設定されていない場合、同期されたデータに存在する情報のみが同期され、それ以外のものは削除または変更されません。

**--ページ <ブール値>**

廃止: 代わりに *remove-vanished* を使用してください。同期中に設定から削除されたユーザまたはグループの ACL を削除します。

**--remove-vanished ([acl]; [properties]; [entry]) | none (default= none)**

*entry*は、同期中にユーザーまたはユーザーが消えたときに削除するものをセミコロンで区切ったリストです。*acl*は、ユーザー/グループが同期から返さないときに*acl*を削除します。リストの代わりに*none*（デフォルト）を指定することもできます。

**--スコープ <両方の| グループ| ユーザー>**

同期するものを選択します。

**pveum role add <roleid> [OPTIONS] [オプション].**

新しいロールを作成します。

**<roleid>: <文字列>**

説明なし

**--privs <文字列>**

説明なし

**pveumロール削除<ロールID>**

役割を削除します。

**<roleid>: <文字列>**

説明なし

**pveumロールリスト [FORMAT\_OPTIONS].**

役割インデックス

**pveum role modify <roleid> [オプション].**

既存のロールを更新します。

**<roleid>: <文字列>**

説明なし

**--append <ブール値>**

説明なし

---

**備考**

必要なオプション: privs

---

**--privs <文字列>**

説明なし

**プベウム・ロールアド**

*pveum role add*のエイリアス。

**プベウム・ローデル**

*pveum role delete*のエイリアスです。

**pveumロールモッド**

*pveum role modify*のエイリアス。

**pveum ticket <ユーザー名> [オプション].**

認証チケットを作成または検証します。

**<ユーザー名>: <文字列>**

ユーザー名

**--new-format <boolean> (デフォルト= 1)**

このパラメータは無視され、みなされます。

**--otp <文字列>**

二要素用のワンタイムパスワード。

**--パス <文字列>**

チケットを確認し、ユーザがパスにアクセス権限を持っているかチェックします。

---

**備考**

必要なオプション: privs

---

**--privs <文字列>**

チケットを確認し、ユーザがパスにアクセス権限を持っているかチェックします。

---

**備考**

必要なオプション: path

---

**--realm <文字列>**

オプションでこのパラメータに realm を渡すことができます。通常、レルムは単にユーザ名 <username>@<realm> 追加されます。

**-tfa-チャレンジ <文字列>**

ユーザーがしたい署名済みTFAチャレンジ文字列。

**pveum user add <userid> [OPTIONS] [オプション].**

新しいユーザーを作成します。

**<ユーザーID>: <文字列>**

name@realm完全なユーザーID。

**--コメント <文字列>**

説明なし

**--email <文字列>**

説明なし

**--有効 <ブール値> (デフォルト= 1)**

アカウントを有効にします (デフォルト)。アカウントを無効にするには、これを0に設定します。

**--expire <整数> (0 - N)**

アカウントの有効期限 (エポックからの秒数)。0は有効ないことを意味します。

**--名 <文字列>**

説明なし

**--グループ <文字列>**

説明なし

**--keys [0-9a-zA-Z!=]{0,4096}。**

二要素認証 (yubico) 用のキー。

**--ラストネーム <文字列>**

説明なし

**--パスワード <文字列>**

初期パスワード

**pveum ユーザー削除 <ユーザーID>**

ユーザーを削除します。

**<ユーザーID>: <文字列>**

name@realm完全なユーザーID。

**pveum ユーザーリスト** [OPTIONS] [FORMAT\_OPTIONS].

ユーザーインデックス

**--有効 <ブール値>**

enableプロパティのオプションフィルタ。

**--full <ブール値> (デフォルト= 0)**

グループおよびトークン情報を含みます。

**pveum user modify** <userid> [OPTIONS] [オプション].

ユーザー設定を更新します。

**<ユーザーID>: <文字列>**

name@realm完全なユーザーID。

**--append <ブール値>**

説明なし

---

**備考**

必要なオプション: グループ

---

**--コメント <文字列>**

説明なし

**--email <文字列>**

説明なし

**--有効 <ブール値> (デフォルト= 1)**

アカウントを有効にします（デフォルト）。アカウントを無効にするには、これを0に設定します。

**--expire <整数> (0 - N)**

アカウントの有効期限（エポックからの秒数）。0は有効ないことを意味します。

**--名 <文字列>**

説明なし

**--グループ <文字列>**

説明なし

**--keys [0-9a-zA-Z!=]{0,4096}。**

二要素認証（yubico）用のキー。

**--ラストネーム <文字列>**

---

説明なし

**pveum user permissions** [<userid>] [OPTIONS] [FORMAT\_OPTIONS].

指定されたユーザ/トークンの有効なパーミッションを取得します。

<userid>:

(?^:^(?^:[^\s:/]+)\@(?^:[A-Za-z][A-Za-z0-9\.\\-\_]+)(?:! (?^:[A-Za-z][A-Za-z0-

ユーザーIDまたは完全なAPIトークンID

--パス <文字列

ツリー全体ではなく、特定のパスだけをダンプします。

**pveum user tfa delete** <userid> [OPTIONS] [オプション].

ユーザーから TFA エントリを削除します。

<ユーザーID>: <文字列

name@realm完全なユーザーID。

--id <文字列

TFA ID。指定がない場合は、すべての TFA エントリが削除されます。

**pveumユーザーtfaリスト** [<ユーザーID>]。

TFAエントリー一覧。

<ユーザーID>: <文字列

name@realm完全なユーザーID。

**pveumユーザーtfaロック解除**<ユーザーID>

ユーザーの TFA 認証を解除します。

<ユーザーID>: <文字列

name@realm完全なユーザーID。

**pveum user token add** <userid> <tokenid> [OPTIONS] [FORMAT\_OPTIONS].

特定のユーザー用に新しいAPIトークンを生成します。注意: APIトークンの値を返します。この値は後から取得できないため、保存しておく必要があります！

<ユーザーID>: <文字列

name@realm完全なユーザーID。

<tokenid>: (?^:[A-Za-z][A-Za-z0-9\.\\-\_]+)

ユーザー固有のトークン識別子。

**--コメント <文字列>**

説明なし

**--expire <integer> (0 - N) (デフォルト= user と同じ)**

APIトークンの有効期限（エポックからの秒数）。0は有効なしを意味します。

**--privsep <boolean> (デフォルト= 1)**

個別のACL（デフォルト）でAPIトークンの権限を制限するか、対応する全権限を与えます。

**pveum user token delete** <userid> <tokenid> [FORMAT\_OPTIONS]。

特定のAPIトークンを削除します。

**<ユーザーID>: <文字列>**

name@realm完全なユーザーID。

**<tokenid>: (?^:[A-Za-z][A-Za-z0-9\.\-\_]+)**

ユーザー固有のトークン識別子。

**pveum ユーザー・トークン・リスト** <ユーザーID> [FORMAT\_OPTIONS].

ユーザー API トークンを取得します。

**<ユーザーID>: <文字列>**

name@realm完全なユーザーID。

**pveum user token modify** <userid> <tokenid> [OPTIONS] [FORMAT\_OPTIONS].

特定のAPIトークンを更新します。

**<ユーザーID>: <文字列>**

name@realm完全なユーザーID。

**<tokenid>: (?^:[A-Za-z][A-Za-z0-9\.\-\_]+)**

ユーザー固有のトークン識別子。

**--コメント <文字列>**

説明なし

**--expire <integer> (0 - N) (デフォルト= user と同じ)**

APIトークンの有効期限（エポックからの秒数）。0は有効なしを意味します。

**--privsep <boolean> (デフォルト= 1)**

個別のACL（デフォルト）でAPIトークンの権限を制限するか、対応する全権限を与えます。

**pveum user token permissions** <userid> <tokenid> [OPTIONS] [FORMAT\_OPTIONS].

与えられたトークンの有効なパーミッションを取得します。

**<ユーザーID>: <文字列>**

*name@realm*完全なユーザーID。

**<tokenid>: (?^:[A-Za-z][A-Za-z0-9\.\-\_]+)**

ユーザー固有のトークン識別子。

**--パス <文字列>**

ツリー全体ではなく、特定のパスだけをダンプします。

**pveum ユーザートークンの削除**

*pveum user token delete* のエイリアス。

**ユーザー追加**

*pveum user add* のエイリアス。

**ユーザー削除**

*pveum user delete* のエイリアス。

**pveum usermod**

*pveum user modify* のエイリアス。

## A.16 vzdump - VMとコンテナのバックアップユーティリティ

**vzdumpヘルプ**

**vzdump {<vmid>} [OPTIONS]**.

バックアップを作成します。

**<vmid>: <文字列>**

バックアップするゲストシステムのID。

**--all <boolean> (デフォルト= 0)**

この上のすべての既知のゲストシステムをバックアップします。

**--bwlimit <整数> (0 - N) (デフォルト= 0)**

I/Oバンド幅の制限（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 <文字列>**

結果のファイルを指定したディレクトリに保存します。

**--除外 <文字列>**

指定したゲストシステムを除外 (--all を想定)

**--除外パス <配列>**

特定のファイル/ディレクトリ（シェル・グループ）を除外します。で始まるパスはコンテナのルートに固定され、その他のパスは各サブディレクトリに相対的に一致します。

**--フリーク [[enabled=]<1|0>]。[,ストレージ=<ストレージID>]]。**

バックアップフリートのオプション（VMのみ）。

**--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とサスペンドモードのバックアップでは、これはコンプレッサにのみ影響します。値が8の場合はアイドル優先度が使用され、それ以外の場合は指定した値でベストエフォート優先度が使用されます。

**--ジョブID**

バックアップジョブのID。設定されると、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 <整数> (0 - N) (デフォルト= 180)**

グローバルロックの最大待機時間（分）。

**--メール通知 <常に | 失敗> (デフォルト= 常に)**

非推奨: 代わりに通知ターゲット/マッチャーを使用してください。通知メールを送るタイミングを指定

**--宛先 <文字列>**

非推奨: 代わりに通知先/通知先を使用してください。メール通知を受け取るメールアドレスまたはユーザのカンマ区切りリスト。

**-マックスファイル <整数> (1 - N)**

非推奨: 代わりに *prune-backups* を使用してください。ゲストシステムごとのバックアップファイルの最大数。

**-モード <スナップショット | 停止 | サスPEND> (デフォルト= スナップショット)**

バックアップモード。

**--ノード <文字列>**

このノードで実行された場合のみ実行されます。

**--notes-template <文字列>**

バックアップのメモを生成するためのテンプレート文字列。値で置き換えられる変数を含むことができます。現在サポートされているのは{{cluster}}, {{guestname}}, {{node}}, {{vmid}}ですが、今後追加されるかもしれません。改行とバックスラッシュは、それぞれ「\n」と「\n」としてエスケープする必要があります。

---

**備考**

必要なオプション: ストレージ

---

--notification-mode <auto| legacy-sendmail| notification-system>

(デフォルト= 自動)

使用する通知システムを決定します。*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とmailnotificationは無視されます。*auto* (デフォルト設定) に設定すると、mailto が設定されていればメールが送信され、設定されていなければ通知システムが使用されます。

--notification-policy <always| failure| never> (デフォルト= always)

非推奨：を使用しないでください。

--通知先 <文字列>

非推奨：を使用しないでください。

--pbs-change-detection-mode <data| legacy| metadata>

ファイルの変更を検出し、コンテナバックアップのエンコード形式を切り替えるために使用される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の場合はgzipの代わりにpigzを使用。N=1 はコアの半分を使用し、N>1 は N をスレッド数として使用します。

--プール <文字列>

指定されたプールに含まれるすべての既知のゲストシステムをバックアップします。

--プロジェクト <プール>

trueの場合、バックアップに保護マークを付けます。

---

備考

必要なオプション：ストレージ

---

--prune-backups [keep-all=<1|0>] [,keep-daily=<N>] [,keep-hourly=<N>] [,keep-last=<N>] [,keep-monthly=<N>] [,keep-weekly=<N>] [,keep-yearly=<N>] (デフォルト= keep-all=1)

ストレージ構成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)

静かにしてください。

--remove <boolean> (デフォルト= 1)

*prune-backups*に従って古いバックアップを削除します。

--スクリプト <文字列>

指定されたフックスクリプトを使用します。

---

--stdexcludes <boolean> (デフォルト= 1)

一時ファイルとログを除外します。

--stdout <ブール値

tar をファイルではなく標準出力に書き込みます。

--stop <boolean> (デフォルト= 0)

このバックアップジョブの実行を停止します。

--stopwait <整数> (0 ~ N) (デフォルト= 10)

ゲストシステムが停止するまでの最大待機時間 (分)。

--ストレージID

結果のファイルをこのストレージに保存します。

--tmpdir <文字列

指定したディレクトリに一時ファイルを保存します。

--zstd <整数> (デフォルト= 1)

Zstdスレッド。N=0は利用可能なコアの半分を使用し、Nが0より大きな値に設定された場合、Nはスレッド数として使用されます。

## A.17 ha-manager - Proxmox VE HA マネージャ

ha-manager <COMMAND> [ARGS] [OPTIONS].

ha-manager add <sid> [OPTIONS] を追加します。

新しいHAリソースを作成します。

<sid>: <type>: <name>.

HAリソースID。これは、リソースの種類と、コロンで区切られたリソース固有の名前で構成されます（例：vm:100 / ct:100）。仮想マシンとコンテナの場合は、VMまたはCT IDをショートカットとして使用できます（例：100）。

--コメント <文字列

説明

--グループ <文字列

HAグループの識別子。

--max\_relocate <integer> (0 ~ N) (デフォルト= 1)

サービスの起動に失敗した場合のサービス再配置の最大試行回数。

--max\_restart <integer> (0 ~ N) (デフォルト= 1)

起動に失敗したノードでサービスを再起動する最大回数。

**--state <無効| 有効| 無視| 開始| 停止>** (=

開始)

要求されたリソースの状態。

**--type <ct| vm>**

リソースの種類

**ha-manager config [OPTIONS]**

HAのリソースをリストアップします。

**--type <ct| vm>**

特定のタイプのリソースのみをリストアップ

**ha-manager crm-command migrate <sid> <node>**

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。これは、リソースの種類と、コロンで区切られたリソース固有の名前で構成されます（例：vm:100 / ct:100）。仮想マシンとコンテナの場合は、VMまたはCT IDをショートカットとして使用できます（例：100）。

**<ノード>: <文字列**

対象ノード

**ha-manager crm-command ノードメンテナンス無効化 <ノード>**

ノード保守要求の状態を変更します。

**<ノード>: <文字列**

クラスタ・ノード名。

**ha-manager crm-commandノードメンテナンスイネーブル<ノード>**

ノード保守要求の状態を変更します。

**<ノード>: <文字列**

クラスタ・ノード名。

**ha-manager crm-コマンド relocate <sid> <node>**

別のノードへのリソース再配置を要求します。これにより、古いノードでサービスが停止し、ターゲット・ノードでサービスが再起動します。

**<sid>: <type>: <name>**

HAリソースID。これは、リソースの種類と、コロンで区切られたリソース固有の名前で構成されます（例：vm:100 / ct:100）。仮想マシンとコンテナの場合は、VMまたはCT IDをショートカットとして使用できます（例：100）。

**<ノード>: <文字列**

対象ノード

**ha-manager crm-コマンド停止 <sid> <タイムアウト**

サービスの停止を要求します。

**<sid>: <type>: <name>**

HAリソースID。これは、リソースの種類と、コロンで区切られたリソース固有の名前で構成されます（例：vm:100 / ct:100）。仮想マシンとコンテナの場合は、VMまたはCT IDをショートカットとして使用できます（例：100）。

**<タイムアウト>: <整数> (0 - N)**

秒単位のタイムアウト。0に設定するとハードストップが実行されます。

**ha-manager groupadd <group> --nodes <string> [OPTIONS].**

新しいHAグループを作成します。

**<グループ>: <文字列**

HAグループの識別子。

**--コメント <文字列**

説明

**--nodes <node>[:<pri>],<node>[:<pri>]\*.**

任意の優先度を持つクラスタ・ノード名のリスト。

**--nofailback <boolean> (デフォルト= 0)**

CRMは最も優先順位の高いノードでサービスを実行しようとします。より優先度の高いノードがオンラインになると、CRMはサービスをそのノードに移行します。nofailbackを有効にすると、この動作が防止されます。

**--restricted <boolean> (デフォルト= 0)**

制限付きグループにバインドされたリソースは、グループによって定義されたノード上でのみ実行できます。

**--タイプ <グループ**

グループタイプ。

**ha-manager groupconfig**

HAグループを取得します。

**ha-manager groupremove <グループ削除**

haグループの設定を削除します。

**<グループ>: <文字列>**

HAグループの識別子。

**ha-manager groupset <group> [OPTIONS] .**

ha グループの設定を更新します。

**<グループ>: <文字列>**

HAグループの識別子。

**--コメント <文字列>**

説明

**--削除 <文字列>**

削除したい設定のリスト。

**-ダイジェスト <文字列>**

現在のコンフィギュレーション・ファイルのダイジェストが異なる場合、変更を防止します。これは、現在のコンフィギュレーション・ファイルが変更されないようにするために使用できます。

**--nodes <node>[:<pri>] { ,<node>[:<pri>] }\***

任意の優先度を持つクラスタ・ノード名のリスト。

**--nofailback <boolean> (デフォルト= 0)**

CRM は最も優先度の高いノードでサービスを実行しようとします。より優先度の高いノードがオンラインになると、CRM はサービスをそのノードに移行します。nofailback を有効にすると、この動作が防止されます。

**--restricted <boolean> (デフォルト= 0)**

制限付きグループにバインドされたリソースは、グループによって定義されたノード上でのみ実行できます。

**ha-manager help [OPTIONS]**

指定したコマンドに関するヘルプを表示します。

**--extra-args <array> です。**

特定のコマンドのヘルプを表示します。

**--verbose <ブール値>**

冗長出力フォーマット。

**ha-manager migrate**

*ha-manager crm-command migrate* のエイリアス。

**HAマネージャー移転**

*ha-manager crm-command relocate* のエイリアス。

**ha-manager remove <sid>**

リソース構成を削除します。

**<sid>: <type>: <name>**

HAリソースID。これは、リソースの種類と、コロンで区切られたリソース固有の名前で構成されます（例：vm:100 / ct:100）。仮想マシンとコンテナの場合は、VMまたはCT IDをショートカットとして使用できます（例：100）。

**ha-manager set <sid> [OPTIONS]**

リソース構成を更新します。

**<sid>: <type>: <name>**

HAリソースID。これは、リソースの種類と、コロンで区切られたリソース固有の名前で構成されます（例：vm:100 / ct:100）。仮想マシンとコンテナの場合は、VMまたはCT IDをショートカットとして使用できます（例：100）。

**--コメント <文字列**

説明

**--削除 <文字列**

削除したい設定のリスト。

**-ダイジェスト <文字列**

現在のコンフィギュレーション・ファイルのダイジェストが異なる場合、変更を防止します。これは、現在のコンフィギュレーション・ファイルが変更されないようにするために使用できます。

**--グループ <文字列**

HAグループの識別子。

**--max\_relocate <integer> (0 - N) (デフォルト= 1)**

サービスの起動に失敗した場合のサービス再配置の最大試行回数。

**--max\_restart <integer> (0 - N) (デフォルト= 1)**

起動に失敗したノードでサービスを再起動する最大回数。

**--state <無効| 有効| 無視| 開始| 停止> (=**

**開始**)

要求されたリソースの状態。

**ha-manager status [OPTIONS]**

HA管理者のステータスを表示します。

**--verbose <boolean> (デフォルト= 0)**

詳細出力。完全なCRMとLRMステータス（JSON）を含みます。

## 付録B サービスデーモン

### B.1 pve-firewall - Proxmox VE ファイアウォールデーモン

**pve-firewall <COMMAND> [ARGS] [OPTIONS].**

**pve-firewall** コンパイル

ファイアウォールのコンパイルして印刷します。これはテストに便利です。

**pve-firewallヘルプ [オプション]。**

指定したコマンドに関するヘルプを表示します。

**--extra-args <array>** です。

特定のコマンドのヘルプを表示します。

**--verbose <ブール値**

冗長出力フォーマット。

**pve-ファイアウォール**

ローカルネットワークの情報を印刷します。

**pve-ファイアウォール再起動**

Proxmox VEファイアウォール再起動します。

**pve-firewall simulate [OPTIONS]** をシミュレートします。

ファイアウォールのシミュレートします。これはカーネルのルーティングテーブルをシミュレートするのではなく、単にソースゾーンから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 <文字列**

宛先IPアドレス。

**--ポート <整数**

宛先ポート

--from (host|outside|vm\d+|ct\d+|([a-zA-Z][a-zA-Z0-9]{0,9})/(\s+))

(デフォルト= 外側)

ソースゾーン。

--プロトコル (tcp|udp) (デフォルト= tcp)

プロトコル

--ソース <文字列>

ソースIPアドレス。

--スポーツ <整数>

ソースポート

--to (host|outside|vm\d+|ct\d+|([a-zA-Z][a-zA-Z0-9]{0,9})/(\s+))

(デフォルト= ホスト)

デスティネーションゾーン

--verbose <boolean> (デフォルト= 0)

冗長出力。

**pve-firewall start** [OPTIONS] [オプション]。

Proxmox VEファイアウォールサービスを開始します。

--debug <boolean> (デフォルト= 0)

デバッグモード - フォアグラウンドのまま

**pve-firewall status** ファイア

ウォールの状態を取得します

。

Proxmox VEファイアウォールサービスを停止します。停止すると、Proxmox VE関連のiptablesルールがすべてアクティブに削除され、ホストが保護されなくなる可能性があることに注意してください。

## B.2 pvedaemon - Proxmox VE API デーモン

**pvedaemon** <COMMAND> [ARGS] [OPTIONS].

**pvedaemonヘルプ** [オプション]

指定したコマンドに関するヘルプを表示します。

--extra-args <array> です。

特定のコマンドのヘルプを表示します。

**--verbose <ブール値**

冗長出力フォーマット。

**pvedaemon再起動**

デーモンを再起動します（起動していない場合は起動します）。

**pvedaemonスタート [OPTIONS]**

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 - フォアグラウンドのまま

**pvedaemon status** デーモンの

状態を取得します。 **pvedaemon**

**stop** デーモンを停止します。

## B.3 pveproxy - Proxmox VE API プロキシデーモン

**pveproxy <COMMAND> [ARGS] [OPTIONS]** です。

**pveproxy help [OPTIONS]**.

指定したコマンドに関するヘルプを表示します。

**--extra-args <array>** です。

特定のコマンドのヘルプを表示します。

**--verbose <ブール値**

冗長な出力形式。

**pveproxyの再起動**

デーモンを再起動します（起動していない場合は起動します）。

**pveproxy start [OPTIONS]**.

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 - フォアグラウンドのまま

**pveproxy status** デーモンの

状態を取得します。

**pveproxy stop** デーモンを停

止します。

## B.4 pvestatd - Proxmox VE ステータスデーモン

**pvestatd** <COMMAND> [ARGS] [OPTIONS].

**pvestatdヘルプ** [オプション]。

指定したコマンドに関するヘルプを表示します。

**--extra-args <array>** です。

特定のコマンドのヘルプを表示します。

**--verbose <ブール値**

冗長出力フォーマット。

**pvestatd再起動**

デーモンを再起動します（起動していない場合は起動します）。

**pvestatd start** [OPTIONS].

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 - フォアグラウンドのまま

**pvestatd stop** デーモンを停

止します。

## B.5 spiceproxy - SPICE プロキシサービス

**spiceproxy** <COMMAND> [ARGS] [OPTIONS].

**spiceproxy help** [OPTIONS].

指定したコマンドに関するヘルプを表示します。

**--extra-args <array>** です。

特定のコマンドのヘルプを表示します。

**--verbose <ブール値**

冗長出力フォーマット。

**spiceproxy の再起動**

デーモンを再起動します（起動していない場合は起動します）。

**spiceproxy start** [OPTIONS].

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 - フォアグラウンドのまま
```

**spiceproxy status** デーモン

の状態を取得します。

**spiceproxy stop** デーモンを

停止します。

## B.6 pmxcfs - Proxmox クラスタファイルシステム

**pmxcfs** [OPTIONS]

ヘルプオプション:

```
-h, --help
ヘルプオプションの表示 アプリケーションオプション:
```

ションオプション:

```
-d, --debug
デバッグメッセージをオンにします。
```

```
-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 (corosync.conf を無視し、クオーラムを強制)
```

このサービスは通常 systemd ツールセットを使って起動・管理されます。このサービスは *pve-cluster* と呼ばれます。

```
systemctl start pve-cluster systemctl stop
```

```
pve-cluster systemctl status pve-cluster
```

## B.7 pve-ha-crm - クラスタリソースマネージャーデーモン

**pve-ha-crm** <COMMAND> [ARGS] [OPTIONS] です。

**pve-ha-crmヘルプ** [オプション]。

指定したコマンドに関するヘルプを表示します。

**--extra-args <array>** です。

特定のコマンドのヘルプを表示します。

**--verbose <ブール値**

冗長出力フォーマット。

**pve-ha-crm start** [OPTIONS] [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 - フォアグラウンドのまま

**pve-ha-crm status** デーモン

の状態を取得します。 **pve-**

**ha-crm stop** デーモンを停止

します。

## B.8 pve-ha-lrm - ローカルリソースマネージャーデーモン

**pve-ha-lrm <COMMAND> [ARGS] [OPTIONS]** です。

**pve-ha-lrm help** [OPTIONS] (日本語)

指定したコマンドに関するヘルプを表示します。

**--extra-args <array>** です。

特定のコマンドのヘルプを表示します。

**--verbose <ブール値**

冗長出力フォーマット。

**pve-ha-lrm start** [OPTIONS] [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 - フォアグラウンドのまま

**pve-ha-lrm status** デーモン

の状態を取得します。 **pve-**

**ha-lrm stop** デーモンを停止

します。

## B.9 pvescheduler - Proxmox VE スケジューラーデーモン

**pvescheduler** <COMMAND> [ARGS] [OPTIONS].

**pveschedulerヘルプ** [オプション]。

指定したコマンドに関するヘルプを表示します。

--extra-args <array> です。

特定のコマンドのヘルプを表示します。

--verbose <ブール値

冗長な出力形式。

**pveschedulerの再起動**

デーモンを再起動します（起動していない場合は起動します）。

**pvescheduler start** [OPTIONS] (プベスケジューラ・スタート)。

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 - フォアグラウンドのまま

**pvescheduler status** デーモン

の状態を取得します。

**pvescheduler stop** デーモンを

停止します。

## 付録C 設定ファイル

### C.1 一般

Proxmox VEのほとんどの設定ファイルは、/etc/pveにマウントされた[共有クラスタファイルシステム](#)に存在します。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のような例外もあります。

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ます。

#### C.1.1 物件名のケーシング

歴史的に、長いプロパティ（およびサブプロパティ）はしばしばsnake\_caseを使用するか、1つの単語として書かれていきました。これは、Proxmox VEスタッフのほとんどがプログラミング言語Perlで開発されておりkebab-caseを使用したプロパティへのアクセスには追加の引用符が必要であったこと、および開発初期にスタイルがあまり強制されていなかったため、開発者によって異なる規約が使用されていたことが原因であると考えられます。

新しいプロパティについては、kebab-case が望ましい方法で、既存の snake\_case プロパティのエイリアスを導入し、長期的には API、CLI、および使用中の設定ファイルを kebab-case に切り替える予定です。

### C.2 データセンターの構成

etc/pve/datacenter.cfgファイルはProxmox VEの設定ファイルです。このファイルには、すべてのノードで使用されるクラスタ全体のデフォルト値が含まれています。

#### C.2.1 ファイル形式

このファイルでは、コロンで区切られた単純なキー/値形式を使用します。各行の書式は以下の通りです：

オプション： 値

ファイル内の空白行は無視され、#文字で始まる行はコメントとして扱われ、これも無視されます。

## C.2.2 オプション

**bwlimit: [clone=<LIMIT>] [,default=<LIMIT>] [,migration=<LIMIT>] [,move=<LIMIT>] [,restore=<LIMIT>]** です。  
各種操作のI/O帯域幅制限を設定（単位: KiB/s）。

**クローン= <LIMIT>**

クローン作成ディスクの帯域制限（KiB/s単位）

**デフォルト= <LIMIT>**

デフォルトの帯域幅制限（単位: KiB/s）

**マイグレーション= <LIMIT>**

ゲストの移行（ローカルディスクの移動を含む）の帯域幅制限（KiB/s単位）

**move= <LIMIT>**

移動ディスクの帯域幅制限（KiB/s単位）

**リストア= <LIMIT>**

バックアップからゲストをリストアする際の帯域幅の制限（KiB/s単位）

**consent-text: <文字列>**

ログイン前に表示される同意テキスト。

**コンソール: <applet| html5| vnc| xtermjs>**

デフォルトのConsoleビューアを選択します。ビルトインのJavaアプレット(VNC。非推奨でhtml5にマップされます)、外部のvirt-viewer互換アプリケーション(SPICE)、HTML5ベースのvncビューア(noVNC)、またはHTML5ベースのコンソールクライアント(xtermjs)のいずれかを使用できます。選択したビューアが利用できない場合(例えば、SPICEがVMに対して有効化されていない場合)、フルバックはnoVNCになります。

**crs: [ha=<基本|静的>] [,ha-rebalance-on-start=<1|0>]。**

クラスタリソースのスケジューリング設定。

**ha= <basic| static> (デフォルト= basic)**

HAマネージャがサービスを開始または回復するノードを選択する方法を設定します。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とメモリ構成が考慮されます。

**ha-rebalance-on-start= <boolean> (デフォルト= 0)**

HAサービスの要求状態が停止から開始へ変更された場合に、CRSを使用して対象ノードを選択するように設定します。

**説明: <文字列>**

データセンターの説明。Webインターフェイスのデータセンター・ノート・パネルに表示されます。これは設定ファイル内のコメントとして保存されます。

**email\_from: <文字列>**

通知を送信するメールアドレスを指定（デフォルトはroot@\$hostname）

**fencing: <both| hardware| watchdog> (default= 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。両方では、2つのモードがすべて使用されます。

**警告**

ハードウェアは、どちらも実験的でWIPです。

**ha: shutdown\_policy=<enum> です。**

クラスタ全体のHA設定。

**shutdown\_policy= <conditional| failover| freeze| migrate>****(デフォルト= 条件付き)**

ノードの電源オフまたは再起動時のHAサービスの処理ポリシーを記述します。Freezeは、シャットダウン時にノードに残っているサービスを常にフリーズします。フェイルオーバーは、シャットダウンしたノードがすぐに（1分未満で）復帰しない場合、サービスを凍結としてマークしないため、サービスは他のノードに回復されます。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されます。リブートまたはシャットダウンがトリガーされると、Migrateは実行中のサービスをすべて別のノードに移動しようとします。ノード上に実行中のなくなると、パワーオフプロセスが続行されます。ノードが再び起動した場合、少なくとも他のマイグレーション、リローアクション、リカバリがいなければ、サービスはパワーオフされたノードに戻されます。

**http\_proxy: http://.\***

ダウンロードに使用する外部 http プロキシを指定します (例: <http://username:password@host:port/>)

**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デフォルトキーボードレイアウト。

**language: <ar| ca| da| de| en| es| eu| fa| fr| he| hr| it| ja| ka| kr| nb| nl| nn| pl| pt\_BR| ru| sl| sv| tr| ukr| zh\_CN| zh\_TW>。**

デフォルトのGUI言語。

**mac\_prefix: <string> (デフォルト= BC:24:11)**

仮想ゲストの自動生成 MAC アドレスのプレフィックス。デフォルトのBC:24:11は、MAC Address Block Large (MA-L)のためにIEEEからProxmox Server Solutions GmbHに割り当てられたOrganizationally Unique Identifier (OUI)です。これはローカルネットワーク、つまり一般ユーザーから直接アクセスできないネットワーク (LANやNAT/マスカレードなど) で使用できます。

仮想ゲストのネットワークを（部分的に）共有する複数のクラスタを実行する場合は、デフォルトのMACプレフィックスを拡張するか、カスタムの（有効な）プレフィックスを生成することを強くお勧めします。

MAC衝突のたとえば、各クラスタのProxmox OUIに、最初のクラスタはBC:24:11:0、2番目のクラスタはBC:24:11:1というように、個別の16進数を追加します。また、VLANを使用するなどして、ゲストのネットワークを論理的に分離することもできます。

+ 一般にアクセス可能なゲストの場合は、[IEEEにOUIを登録されている取得するか](#)、あなたまたはあなたのホスティング・プロバイダーのネットワーク管理者と調整することをお勧めします。

#### max\_workers: <整数> (1 - N)

ha-managerからのstopall VMやタスクのようなアクションで、(ノードあたり) 最大何人のワーカーが起動するかを定義します。

マイグレーションを使用します: [type=]<secure|insecure>[,network=<CIDR>] です。

クラスタ全体のマイグレーション設定

#### network= <CIDR>

移行に使用する(サブ)ネットワークのCIDR。

#### タイプ= <インセキュア|セキュア> (デフォルト=セキュア)

マイグレーショントラフィックはデフォルトでSSHトンネルを使用して暗号化されます。安全で完全にプライベートなネットワークでは、パフォーマンスを上げるためにこれを無効にすることができます。

#### migration\_unsecure: <論理値>

移行はデフォルトでSSHトンネルを使用して安全に行われます。安全なプライベートネットワークでは、マイグレーションを高速化するためにこれを無効にすることができます。非推奨。代わりにマイグレーションプロパティを使用してください!

#### next-id: [lower=<整数>] [,upper=<整数>]。

フリーVMID自動選択プールの範囲を制御します。

#### = <整数> (デフォルト=100)

自由な next-id API 範囲の下限、包含境界。

#### upper= <整数> (デフォルト=1000000)

自由な next-id API 範囲の上限、排他的境界。

を通知します: [fencing=<always|never>> [,package-updates=<auto|always|never>> ] [,replication=<always|never>> [,target-fencing=<TARGET>] [,target-package-updates=<TARGET>] [,target-replication=<TARGET>] となります。

クラスタ全体の通知設定。

#### フェンシング= <いつも|決して>

UNUSED - 代わりにデータセンター通知設定を使用します。

#### package-updates= <always| auto| never> (デフォルト= auto)

廃止されました: 代わりにデータセンター通知設定を使用してください。日次更新ジョブが通知を送信する頻度を制御します:

- 有効なサブスクリプションを持つシステムについては、毎日自動更新されます。
- 新しい保留中の更新がある場合は、更新のたびに常に更新されます。
- 新しい保留中の更新の通知を送信することはありません。

#### レプリケーション= <常に|決して>

UNUSED - 代わりにデータセンター通知設定を使用します。

#### target-fencing= <TARGET>

UNUSED - 代わりにデータセンター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通知設定を使用します。

#### target-replication= <TARGET>

UNUSED - 代わりにデータセンター通知設定を使用します。

#### 登録タグ: <tag>[;<tag>...].

タグの設定と削除には、`Sys.Modify on`/が必要です。ここで設定されたタグは `user-tag-access` も `Sys.Modify.User` タグアクセスが必要です。

#### タグのスタイルを指定します: [case-sensitive=<1|0>] です。

[,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= <boolean> (デフォルト= 0)

更新時の一意なタグのフィルタリングで、大文字小文字を区別してチェックするかどうかを制御します。

#### color-map= <tag>:<hex-color>[:<hex-color-for-text>] [;<tag>=...]. . .

タグの手動カラーマッピング（セミコロン区切り）。

#### ordering= <alphabetical| config> (default= alphabetical)

ウェブインターフェースおよびAPIアップデートにおけるタグのソートを制御します。

#### shape= <circle| dense| full| none> (デフォルト= circle)

fullは完全なタグを描画します。circleは背景色のある円だけを描画します。denseは小さな長方形だけを描画します(多くのタグがそれぞれのゲストに割り当てられている場合に便利です).noneはタグの表示を無効にします。

#### u2fです: [appid=<APPID>] [,origin=<URL>] です。

u2f

#### appid= <APPID>

U2F AppId URLのオーバーライド。デフォルトはオリジン。

#### オリジン= <URL>

U2F Originのオーバーライド。主に1つの持つ1つのノードに便利です。

```
user-tag-access: [user-allow=<enum>] [,user-allow-list=<tag>[;<tag>...]]。
```

ユーザー設定可能タグの特権オプション

```
user-allow= <既存の| free| list| none> (デフォルト= free)
```

ユーザが制御するリソース (ゲストなど) で設定または削除できるタグを制御します。Sys.Modify 権限を持つユーザは、常に制限されません。

- タグは使用できません。
- user-allow-list* のリストタグが使用可能です。
- のような既存のリストも使えますが、すでにあるリソースのタグも使えます。
- タグの制限はありません。

```
user-allow-list= <tag>[;<tag>...].
```

ユーザが設定および削除を許可されたタグのリスト (セミコロンで区切られています) *user-allow values list*用  
そして既存。

```
webauthn: [allow-subdomains=<1|0>] [,id=<DOMAINNAME>] [,origin=<URL>]  
[,rp=<RELYING_PARTY>] です。
```

webauthnの設定

```
allow-subdomains= <boolean> (デフォルト= 1)
```

正確な URL ではなく、サブドメインのオリジンを許可するかどうか。

```
id= <DOMAINNAME>
```

依拠当事者ID。プロトコル、ポート、場所を含まないドメイン名でなければなりません。これを変更すると  
既存のクレデンシャルを破棄します。

```
オリジン= <URL>
```

サイトのオリジン。<https:// URL> (または <http://localhost>)。でなければなりませんウェブ・インターフェイスにアクセスする  
ためにユーザーがブラウザに入力するアドレスが含まれていなければなりません。これを変更すると、既存の認証情報が壊れる可能性があります  
。

```
rp= <RELYING_PARTY>
```

依拠当事者名。任意のテキスト識別子。これを変更すると、既存のクレデンシャルが壊れ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リング設定を持っています。これは systemd time カレンダーイベントフォーマットに基づいています<sup>1</sup> カレンダーイベントは1つの式で1つ以上の時点を参照するために使用できます。

このようなカレンダーイベントは、次のような書式を使用します：

平日] [[年-]月-日] [時:分[:秒]] [年-]月-日

このフォーマットでは、ジョブを実行する日を設定できます。また、1つ以上の開始時間を設定することができます。これはレプリケーション・スケジューラに、ジョブを開始する時刻を指定します。この情報を使って、毎日午後10時に実行するジョブを作成することができます：'月,火,水,木,金,22'と省略できます：ほとんどの合理的なスケジュールはこのように直感的に書くことができます。

---

#### 備考

時間は24時間です。

---

便利で短い設定を可能にするために、ゲストごとに1つ以上の繰り返し時間を設定できます。これらは、開始時刻そのものと開始時刻に繰り返し値の倍数を加えた時刻にレプリケーションが行われることを示します。午前8時にレプリケーションを開始し、午前9時まで15分ごとにレプリケーションを繰り返す場合、次のようにになります：'8:00/15'

ここでは、時区切り (:) が使用されていない場合、値は分として解釈されることがわかります。このような区切りが使用された場合、左側の値は「時」を表し、右側の値は「分」を表します。さらに、(\*) を使用、すべての可能な値にマッチさせることができます。

さらにアイデアを得るには、[以下の例をご覧ください](#)。

### D.2 詳細仕様

---

<sup>1</sup> 詳しくは `man 7 systemd.time` を参照してください。

## 平日

曜日は、sun、mon、tue、wed、thu、fri、sat のように省略された指定します。複数の曜日をカンマ指定することもできます。開始日と終了日を「...」で区切って指定することで、日数の範囲を設定することもできます。これらの書式は混在可能です。省略した場合は '\*' とみなされます。

## タイムフォーマット

時刻の書式は、時間と分の間隔のリストで構成されます。時間と分は' ':'で区切られます。時」と「分」は、「日」と同じ書式で、リストや値の範囲を指定することができます。最初に時間です、

次に分。必要なければ、時間は省略できます。この場合、時間の値は'\*'とみなされます。有効な値の範囲は、時：0～23、分：0～59です。

## D.2.1 例

特定の意味を持つ特別な値がいくつかあります：

表 D.1: 特別な値

値	構文
ここまかに	*-*-* *:*:00
毎時	*-*-* *:00:00
デイリー	*-*-* 00:00:00
Wiークリー	月 *-*-* 00:00:00
毎月	*-*01 00:00:00
毎年または毎年	*-01-01 00:00:00
クオータリー	*-01,04,07,10-01 00:00:00
半年ごとまたは半期ごと	*-01,07-01 00:00:00

表D.2: スケジュールの例

スケジュール文字列	オルタナティブ	意味
月、火、水、木、金	月・金	毎営業日0:00
土・日	日	週末のみ0:00
月、水、金	-	月曜日、水曜日のみ と金曜日の0:00
12:05	12:05	毎日午後12時5分
*5	0/5	5分ごと
10月30日（月	月、火、水 30/10	月曜日、火曜日、水曜日 30分後、40分後、50分 後 毎時
月・金 8・17・22:0/15	-	毎営業日、午前8時から午後6時までと午後10時 から11時までの15分ごと PM

表D.2: (続き)

スケジュール文字列	オルタナティブ	意味
12日（金）13:5/20	12,13日(金):5/20	金曜日の12:05、12:25、12:45、 13:05、13:25、13:45
12,14,16,18,20,22:5	12/2:5	毎日12:05から 22:05、2時間ごと
*	*/1	毎分（最小間隔）
*-05	-	毎月5日
土*-1.7 15:00	-	毎月第一土曜日 15:00
2015-10-21	-	2015年10月21日 00:00

## 付録E QEMU vCPUリスト

### E.1 はじめに

これは、QEMUで定義されているAMDおよびIntel x86-64/amd64のCPUタイプのリストで、2007年までさかのぼります。

### E.2 インテルCPUの種類

#### インテルプロセッサー

- *ナヘム*: 第1世代インテル・コア・プロセッサー
- *Nahalem-IBRS (v2)*: Spectre v1 保護の追加 (+spec-ctrl)
- *Westmere*: 第1世代のインテル・コア・プロセッサー (Xeon E7-)。
- *Westmere-IBRS (v2)*: Spectre v1 プロテクションの追加 (+spec-ctrl)
- *SandyBridge*: 第2世代インテル・コア・プロセッサー
- *SandyBridge-IBRS (v2)*: Spectre v1 プロテクションの追加 (+spec-ctrl)
- *IvyBridge*: 第3世代インテル・コア・プロセッサー
- *IvyBridge-IBRS (v2)*: Spectre v1 保護の追加 (+spec-ctrl)
- *Haswell*: 第4世代インテル・コア・プロセッサー
- *Haswell-noTSX (v2)*: TSX (-hle, -rtm) を無効にします。
- *Haswell-IBRS (v3)*: TSX の再追加、Spectre v1 保護の追加 (+hle, +rtm, +spec-ctrl)
- *Haswell-noTSX-IBRS (v4)*: TSX (-hle, -rtm) を無効にします。
- *Broadwell*: 第5世代インテル・コア・プロセッサー
- *Skylake*: 第1世代Xeonスケーラブル・サーバー・プロセッサー
- *Skylake-IBRS (v2)*: Spectre v1プロテクションの追加、CLFLUSHOPTの無効化 (+spec-ctrl, -clflushopt)

- *Skylake-noTSX-IBRS (v3)* : TSX (-hle, -rtm) を無効にします。
- *Skylake-v4* : EPTスイッチングを追加 (+vmx-eptp-switching)
- *Cascadelake*: 第2世代Xeonスケーラブル・プロセッサ
- *Cascadelake-v2* : add arch\_capabilities msr (+arch-capabilities, +rdctl-no, +ibrs-all, +skip-l1dfl-vmentry, +mds-no)。
- *Cascadelake-v3* : TSX (-hle, -rtm) を無効にします。
- *Cascadelake-v4* : EPTスイッチングを追加 (+vmx-eptp-switching)
- *Cascadelake-v5* : XSAVESの追加 (+xsaves, +vmx-xsaves)
- *Cooperlake*: 4ソケットおよび8ソケットサーバー用第3世代Xeonスケーラブル・プロセッサ
- *Cooperlake-v2* : XSAVESの追加 (+xsaves, +vmx-xsaves)
- アイスレイク第3世代Xeonスケーラブル・サーバ・プロセッサ
- *Icelake-v2* : TSX (-hle, -rtm) を無効にします。
- *Icelake-v3* : arch\_capabilities msr (+arch-capabilities, +rdctl-no, +ibrs-all, +skip-l1dfl-vmentry, +mds-no, +pschange-mc-no, +taa-no) を追加。
- *Icelake-v4* : 不足していたフラグを追加 (+sha-ni, +avx512ifma, +rdpid, +fsrm, +vmx-rdseed-exit, +vmx-pml, +vmx-eptp-switching)
- *Icelake-v5* : XSAVESの追加 (+xsaves, +vmx-xsaves)
- *Icelake-v6* : 「5レベルEPT」を追加 (+vmx-page-walk-5)
- *SapphireRapids* : 第4世代Xeonスケーラブル・サーバ・プロセッサ

## E.3 AMD CPUの種類

### AMDプロセッサー

- *Opteron\_G3* : K10
- *Opteron\_G4* : ブルドーザー
- *Opteron\_G5* : パイルドライバー
- *EPYC* : Zenプロセッサの第1世代
- *EPYC-IBPB (v2)* : Spectre v1プロテクションを追加 (+ibpb)
- *EPYC-v3* : 不足していたフラグを追加 (+perfctr-core, +clzero, +xsaveerptr, +xsaves)
- *EPYC-Rome*: 第2世代Zenプロセッサ
- *EPYC-Rome-v2* : Spectre v2, v4プロテクションの追加 (+ibrs, +amd-ssbd )
- *EPYC-Milan*: 第3世代のZenプロセッサ
- *EPYC-Milan-v2* : 不足していたフラグを追加 (+vaes, +vpclmulqdq, +stibp-always-on, +amd-psfd, +no-nested-data-bp, +fence-always-serializing, +null-sei-clr-base)

## 付録F

### ファイアウォールマクロ定義

*アマンダ*

アマンダ・バックアップ

アクション	プロト	ポート	スポーツ
パラム	udp	10080	
パラム	ティーシーピー	10080	

*認証*

認証 (identd) トラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	113	

*BGP*

ポーダーゲートウェイプロトコルのトラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	179	

*BitTorrent*

BitTorrent 3.1 以前の BitTorrent トラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	6881:6889	
パラム	udp	6881	

*BitTorrent32*

BitTorrent 3.2 以降用の BitTorrent トラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	6881:6999	
パラム	udp	6881	

CVS

同時バージョンシステム pserver トラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	2401	

Ceph

Cephストレージクラスタ トラフィック (Cephモニタ、OSD、MDSデーモン)

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	6789	
パラム	ティーシーピー	3300	
パラム	ティーシーピー	6800:7300	

シトリックス

Citrix/ICA トラフィック (ICA、ICAブラウザ、CGP)

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	1494	
パラム	udp	1604	
パラム	ティーシーピー	2598	

DAAP

Digital Audio Access Protocol トラフィック (iTunes、Rhythmboxデーモン)

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	3689	
パラム	udp	3689	

DCC

分散型チェックサム・クリアリングハウス・スパム・フィルタリング機構

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	6277	

DHCPfwd

転送されたDHCP トラフィック

アクション	プロト	ポート	スポーツ
パラム	udp	67:68	67:68

*DHC Pv6*

DHC Pv6 ト ラフィック

アクション	プロト	ポート	スポーツ
パラム	udp	546:547	546:547

*DNS*

ドメイン名システムのト ラフィック (updおよびtcp)

アクション	プロト	ポート	スポーツ
パラム	udp	53	
パラム	ティーシーピー	53	

*ディストック*

分散コンパイラサービス

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	3632	

*FTP*

ファイル転送プロトコル

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	21	

*フィンガー*

フィンガープロトコル (RFC 742)

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	79	

*GNUnet*

GNUnet 安全なピアツーピアネットワーキングト ラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	2086	
パラム	udp	2086	
パラム	ティーシーピー	1080	

アクション	プロト	ポート	スポーツ
パラム	udp	1080	

*GRE* Generic Routing Encapsulation トンネリングプロトコル。

アクション	プロト	ポート	スポーツ
パラム	47		

*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	ポート	スポーツ
パラム	ティーシーピー	9418	

*HKP* OpenPGP HTTP 鍵サーバプロトコルトラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	11371	

*HTTP* ハイパーテキスト転送プロトコル (WWW)

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	80	

*HTTPS* SSL 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	ポート	スポーツ
パラム	ティーシーピー	443	

*ICPV2* インターネット・キャッシュ・プロトコルV2 (Squid) のトラフィック

アクション	プロト	ポート	スポーツ
パラム	udp	3130	

*ICQ*

AOLインスタントメッセンジャーのトラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	5190	

*IMAP*

インターネット・メッセージ・アクセス・プロトコル

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	143	

*IMAPS*

インターネット・メッセージ・アクセス・プロトコル・オーバーSSL

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	993	

*IP/IP*

IP/IPカプセル化トラフィック

アクション	プロト	ポート	スポーツ
パラム	94		

*IPsec*

IPsecトラフィック

アクション	プロト	ポート	スポーツ
パラム	udp	500	500
パラム	50		

*IPsecah*

IPsec 認証 (AH) トラフィック

アクション	プロト	ポート	スポーツ
パラム	udp	500	500
パラム	51		

*IPsecnat*

IPsecトラフィックとNat-Traversal

アクション	プロト	ポート	スポーツ
パラム	udp	500	
パラム	udp	4500	
パラム	50		

*IRC*

インターネット・リレー・チャットのトラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	6667	

*ジェットダイレクト*

HP Jetdirectプリント

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	9100	

*L2TP*

レイヤー2トンネリングプロトコルのトラフィック

アクション	プロト	ポート	スポーツ
パラム	udp	1701	

*LDAP*

ライトウェイト・ディレクトリ・アクセス・プロトコル・トラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	389	

*LDAPS*

セキュア・ライトウェイト・ディレクトリ・アクセス・プロトコル・トラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	636	

*MDNS*

マルチキャストDNS

アクション	プロト	ポート	スポーツ
パラム	udp	5353	

*MSNP*

マイクロソフト通知プロトコル

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	1863	

*MSSQL*

マイクロソフトSQLサーバー

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	1433	

*メール*

メールトラフィック (SMTP、SMTPE、送信)

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	25	
パラム	ティーシーピー	465	
パラム	ティーシーピー	587	

*ムニン*

ムニン・ネットワーク・リソース・モニタリング・トラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	4949	

*MySQL*

MySQLサーバー

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	3306	

**エヌエヌティーピー** NNTPトラフィック（ユースネット）。

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	119	

**NNTPS** 暗号化されたNNTPトラフィック（ユースネット）

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	563	

**NTP** ネットワークタイムプロトコル (ntpd)

アクション	プロト	ポート	スポーツ
パラム	udp	123	

**NeighborDiscoveryPv6** ネイバー勧誘、ネイバー広告、ルーター広告

アクション	プロト	ポート	スポーツ
パラム	アイシーエムピーブイシックス	ルーター募集	
パラム	アイシーエムピーブイシックス	ルーター広告	
パラム	アイシーエムピーブイシックス	隣人募集	
パラム	アイシーエムピーブイシックス	隣人 広告	

**OSPF** OSPFマルチキャストトラフィック

アクション	プロト	ポート	スポーツ
パラム	89		

**OpenVPN** OpenVPNトラフィック

アクション	プロト	ポート	スポーツ
パラム	udp	1194	

*PCA*

シマンテックPCAnywhere (tm)

アクション	プロト	ポート	スポーツ
パラム	udp	5632	
パラム	ティーシーピー	5631	

*PMG*

Proxmox Mail Gatewayウェブインターフェース

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	8006	

*POP3*

POP3トラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	110	

*POP3S*

暗号化されたPOP3トラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	995	

*PPtP*

ポイントツーポイントトンネリングプロトコル

アクション	プロト	ポート	スポーツ
パラム	47		
パラム	ティーシーピー	1723	

*ピング*

ICMPエコー要求

アクション	プロト	ポート	スポーツ
パラム	アイシーエムピー	エコー要求	

*PostgreSQL*

PostgreSQLサーバー

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	5432	

プリンタ

ラインプリンタープロトコル印刷

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	515	

*RDP*

マイクロソフト・リモート・デスクトップ・プロトコル・トラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	3389	

*RIP*

ルーティング情報プロトコル（双方向）

アクション	プロト	ポート	スポーツ
パラム	udp	520	

*RNDC*

BINDリモート管理プロトコル

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	953	

レイザー

アンチスパムシステム

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	2703	

*Rdate*

リモート時刻検索(rdate)

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	37	

*Rsync*

Rsyncサーバー

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	873	

*SANE*

SANEネットワークスキャン

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	6566	

*SMB*

マイクロソフトSMBトラフィック

アクション	プロト	ポート	スポーツ
パラム	udp	135,445	
パラム	udp	137:139	
パラム	udp	1024:65535	137
パラム	ティーシーピー	135,139,445	

*SMBswat*

Sambaウェブ管理ツール

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	901	

*SMTP*

簡易メール転送プロトコル

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	25	

*SMTPS*

暗号化簡易メール転送プロトコル

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	465	

**SNMP** 簡易ネットワーク管理プロトコル

アクション	プロト	ポート	スポーツ
パラム	udp	161:162	
パラム	ティーシーピー	161	

**SPAMD** スパムアサシン SPAMD トラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	783	

**SPICEproxy** Proxmox VE SPICEディスプレイのプロキシトラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	3128	

**SSH** 安全なシェルトラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	22	

**SVN** Subversion サーバー (svnserv)

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	3690	

**SixXS** SixXS IPv6導入とトンネルプローカー

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	3874	
パラム	udp	3740	
パラム	41		

アクション	プロト	ポート	スポーツ
パラム	udp	5072,8374	

イカ SquidのWebプロキシトラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	3128	

投稿 メール送信トラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	587	

システム Syslogプロトコル (RFC 5424) のトラフィック

アクション	プロト	ポート	スポーツ
パラム	udp	514	
パラム	ティーシーピー	514	

TFTP Trivial File Transfer Protocol トラフィック

アクション	プロト	ポート	スポーツ
パラム	udp	69	

テルネット Telnetトラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	23	

テルネット Telnet over SSL

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	992	

時間

RFC 868 時間プロトコル

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	37	

Trort

トレースルート（最大30ホップ）トラフィック

アクション	プロト	ポート	スポーツ
パラム	udp	33434:33524	
パラム	アイシーエムピー	エコー要求	

VNC

VNCディスプレイのVNCトラフィック 0 - 99

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	5900:5999	

VNCL

VncsサーバーからVncビューアーへのVNCトラフィック（リッスンモード）

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	5500	

ウェブ

WWWトラフィック（HTTPおよびHTTPS）

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	80	
パラム	ティーシーピー	443	

ウェブキャッシュ

ウェブキャッシング/プロキシトラフィック（ポート8080）

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	8080	

ウェブミン

Webmin トラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	10000	

Whois

Whois (nickname、RFC 3912) トラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	43	

## 付録G マークダウン入門

MarkdownはウェブライターのためのテキストからHTMLへの変換ツールです。Markdownを使えば、読みやすく、書きやすいプレーンテキスト形式を使って文章を書き、それを構造的に妥当なXHTML（またはHTML）に変換することができます。

- ジョン・グルーバー <https://daringfireball.net/projects/markdown/>

Proxmox VEウェブインターフェースは、ノードおよび仮想ゲストノートのリッチテキストフォーマットをレンダリングするMarkdownの使用をサポートしています。

Proxmox VEは、テーブルやタスクリストのようなGFM（GitHub Flavoured Markdown）のほとんどの拡張機能でCommonMarkをサポートしています。

### G.1 マークダウンの基本

ここでは基本的なことしか説明していませんので、例えば<https://www.markdownguide.org/>。

#### G.1.1 見出し

```
# This is 見出し h1 ## This is 見出  
し h2  
##### これは見出します。
```

#### G.1.2 強調

強調したい場合は、\*text\*または`_text_`を使用してください。

`**text**`または`_____テキスト__`を使います。組み合わせも可能です：

```
"あなたは**それらを組み合わせることができます**"
```

## G.1.3 リンク

リンクの自動検出を使用することができます。例えば、<https://forum.proxmox.com/>、クリック可能なリンクに変換します。

、リンクテキストをコントロールすることもできます：

さて、[カッコ内がリンクテキストになります] (<https://forum.proxmox.com/>)。

## G.1.4 リスト

### 順序なしリスト

例えば、順序なしリストには \* または - を使用します：

- \* 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することができます。

### 注文リスト

1. 項目 1
1. 項目 2
1. 項目 3
  - 1. 項目 3a
  - 1. 項目 3b

---

### 備考

順番に並べられたリストの整数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、未完了のタスクは空白のボックス [ ]、完了したタスクはXのボックスを使用します。例えば

- [X] 最初のタスクはすでに完了しました!
- [X] 2つ目も
- [ ] これはまだto-do
- [これも]

## G.1.5 テーブル

テーブルでは、パイプ記号 | を使って列を区切り、- を使って表のヘッダーと本文を区切れます。この区切りでは、テキストのアラインメントを設定することもでき、1つの列を左揃え、中央揃え、右揃えにすることができます。

左列も	右	Some  More  Cols.  Centering Works←'
	- - - - - ←'	
左 フー	右 フー	最初   列   こちら   >center<'
左 バー	右 バー	セカンド   列   こちら   12345←'
左 バズ	右 バズ	サード   列   こちら   テスト←'
左ザブ	右ザブ	第四   行   こちら   ←'
&#x2601; &#xfe0f; &#x2601; &#xfe0f; &#x2601; &#xfe0f;		
左ラヴ	右ラヴ	そして   最後   ここで   終わり
		←'

列を空白できれいに揃える必要はありませんが、その方が表の編集が簡単になることに注意してください。

## G.1.6 ブロック名言集

ブロック引用符は、プレーンテキストの電子メールと同様に、行の先頭に> を付けることで入力できます。

- > 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、
- > 2004年、ジョン・グルーバーがアーロン・スワーツとともに作成。
- > >>Markdownは、Readmeファイルのフォーマットや、←'のメッセージの記述によく使用されます。  
オンライン・ディスカッション・フォーラム
- >>また、プレーンテキストエディタを使ってリッチテキストを作成することもできます。

## G.1.7 コードとスニペット

バックスティックを使って、いくつかの単語や段落の処理を避けることができます。これはコードや設定の塊が誤ってマークダウンとして解釈されるのを避けるのに便利です。

### インライントコード

行の一部をバックスティックで囲むと、インラインでコードを書くことができます：

```
このホストIPアドレスは です。
```

### コードブロック全体

複数の行にまたがるコード・ブロックでは、トリプル・バックスティックを使ってブロックの開始と終了を行うことができます：

```
...
# ここで覚えておきたいネットワーク設定 auto vmbr2
iface vmbr2 inet static
    アドレス 10.0.0.1/24 bridge-
    ports ens20 bridge-stp off
    bridge-fd 0
    bridge-vlan-aware yes bridge-
    vids 2-4094
...
```

## 付録H

### GNU自由文書ライセンス

バージョン1.3、2008年11月3日

著作権 (C) 2000, 2001, 2002, 2007, 2008 フリーソフトウェアファウンデーション, ←'

株式会社

<<http://fsf.org/>>

誰もがこのライセンス文書の逐語的なコピーし配布することは許可されていますが、変更することは許可されていません。

#### 0. 前文

本許諾書の目的は、マニュアルや教科書、あるいはその他の機能的で有用な文書を、自由という意味で「自由」にすることです。つまり、商業的であれ非商業的であろうと、改変の有無に関わらず、誰もがそれを複製し再配布できる実効的な自由を保証することです。第二に、本許諾書は作者や出版社に、他者による改変に対して責任を負わされることなく、自分たちの作品に対する信用を得る方法を保持します。

このライセンスは一種の「コピーレフト」であり、この文書の派生同じ意味で自由でなければなりません。これはGNU一般公衆ライセンス(GNU General Public License)を補完するもので、自由ソフトウェアのために設計されたコピーレフトのライセンスです。

なぜなら、自由ソフトウェアには自由な文書が必要だからです：自由なプログラムには、ソフトウェアと同じ自由を提供するマニュアルが付属しているべきです。しかし、本許諾書はソフトウェアのマニュアルに限定されるものではありません。題材や印刷された書籍として出版されるかどうかに関わらず、あらゆるテキスト作品に利用することができます。私たちは、この利用許諾書を、主に指導や参照を目的とした作品に推奨します。

#### 1. 適用と定義

本許諾書は、著作権者によって本許諾書の条項の下で頒布可能である旨の告知が含まれた、あらゆる媒体のマニュアルやその他の作品に適用されます。このような告知は、本許諾書に記載された条件の下で、その作品を使用するための、世界的な、期間無制限の、ロイヤリティフリーのライセンスを付与するものです。以下の「文書」とは、そのようなマニュアルや作品を指します。一般公衆はライセンサーであり、「あなた」として扱われます。著作権法上の許可を必要とする方法で作品を複製、変更、頒布する場合、あなたはこのライセンスに同意するものとします。

本件文書の「改変版」とは、本件文書またはその一部をそのまま、あるいは改変を加えて、および/または他の言語に翻訳して複製した著作物を意味します。

「二次的セクション(Secondary Section)」とは、その文書の出版者または著者と、その文書全体の主題との関係(または関連事項)のみを扱い、その文書全体の主題に直接含まれる可能性のあるものを含まない、名称の付録または文書の前文セクションのことです。(したがって、『文書』の一部が数学の教科書である場合、二次セクションでは数学について一切説明しないことができる)。その関係とは、その主題や関連事項との歴史的な関係、あるいはそれらに関する法的、商業的、哲学的、倫理的、政治的な立場の問題である可能性があります。

「変更不可セクション」とは、『文書』が本許諾書の下でリリースされたことを示す告知において、そのタイトルが変更不可セクションであるとして指定されている特定のセカンダリ・セクションのことです。あるセクションが上記の「二次的セクション」の定義に当てはまらない場合、そのセクションを「変更不可セクション」として指定することは認められません。文書には「変更不可セクション」がゼロであってもかまいません。もし『文書』がいかなるInvariant Sectionも特定しない場合、それはInvariant Sectionが存在しないことを意味します。

「表紙テキスト」とは、『文書』が本許諾書の下でリリースされることを示す告知の中で、表紙テキストまたは裏表紙テキストとして列挙される特定の短い文章のことです。表紙テキストは最大5語、裏表紙テキストは最大25語です。

「透明なTransparent)」『文書』のコピーとは、その仕様が一般公衆に利用可能な形式で表現され、一般的なテキストエディタや(ピクセルで構成される画像については)一般的なペイントプログラムや(画面については)広く利用可能な何らかの描画エディタを用いて文書を直接修正するのに適しており、テキストフォーマットへの入力や、テキストフォーマットへの入力に適した様々な形式への自動翻訳に適している、機械可読なコピーを意味します。そうでなければTransparentなファイルフォーマットで作成されたコピーで、マークアップがある、あるいはマークアップがないものが、読者によるその後の変更を妨げたり、思いどまらせたりするように配置されているものは、Transparentではありません。画像フォーマットは、かなりの量のテキストに使われている場合、「透過的」ではありません。「透明」でないコピーは「不透明」と呼ばれます。

透過コピーに適したフォーマットの例としては、マークアップのないプレーンなASCII、Texinfo入力フォーマット、LaTeX入力フォーマット、一般に公開されているDTDを使ったSGMLやXML、人間が修正できるように設計された標準準拠のシンプルなHTML、PostScript、PDFなどがあります。透過画像フォーマットの例としては、PNG、XCF、JPGがあります。不透明なフォーマットには、プロプライエタリなワードプロセッサーによってのみ読み取りや編集が可能なプロプライエタリなフォーマット、DTDや処理ツールが一般に利用可能でないSGMLやXML、出力のみを目的として一部のワードプロセッサーによって生成される機械生成のHTML、PostScript、PDFなどがあります。

「タイトルページ」とは、印刷された書籍の場合、タイトルページそれ自体と、本許諾書がタイトルページに掲載することを要求する資料を読みやすく掲載するために必要な次のページを加えたものを意味します。そのようなタイトルページを持たない形式の著作物については、「タイトルページ」とは、本文の冒頭に先立つ、著作物のタイトルが最も目立つ位置にあるテキストを意味します。

「発行者」とは、『文書』の複製物を公衆に頒布する個人または団体を意味します。

「XYZと題された」セクションとは、そのタイトルが正確にXYZであるか、またはXYZを他の翻訳したテキストの後に括弧で括られたXYZを含む、『文書』の名前付きサブユニットを意味します。(ここでXYZは、「謝辞」、「献辞」、「裏書」、「歴史」など、後述の特定のセクション名を表します)。あなたが文書を変更するときに、そのようなセクションの「タイトルを保持する」ということは、この定義に従って、そのセクションが「XYZというタイトル」のままであることを意味します。

『文書』には、本許諾書が『文書』に適用されることを示す告示の隣に、保証の否認が含まれている場合があります。これらの保証の否認は本許諾書に参考として含まれるものとみなされますが、それは保証の否認に関してのみです。

## 2. まるうつし

あなたは、本許諾書、著作権表示、および本許諾書が『文書』に適用されることを示す使用許諾表示がすべての複製物に複製され、本許諾書の条件にいかなる他の条件も付加されないことを条件として、営利・非営利を問わず、『文書』をいかなる媒体にも複製・頒布することができます。あなたは、あなたが作成または頒布した複製物の閲覧やさらなる複製を妨害または制御する技術的手段を用いてはなりません。ただし、あなたはコピーの対価として報酬を受け取ることができます。あなたが十分な数の複製物を頒布する場合、あなたは第3項の条件にも従わなければなりません。

また、上記と同じ条件でコピーを貸与し、コピーを公に展示することもできます。

## 3. 量り売り

あなたが『文書』の印刷物(または一般的に印刷された表紙を持つ媒体の複製物)を100部以上発行し、『文書』のライセンス告知でカバー・テキストが要求されている場合、あなたはその複製物を、以下のカバー・テキストをすべて明瞭かつ読みやすく記載した表紙に封入しなければなりません：「Front-Cover Texts」裏表紙には「Back-Cover Texts」。また、両表紙とも、あなたがこれらのコピーの者であることを明確かつ判読しやすいように表示してください。表表紙は、タイトルの全単語を等しく目立たせ、見えるようにしなければなりません。また、表紙に他の素材を加えてもかまいません。文書の題名を保持し、これらの条件を満たす限り、表紙に限定して変更を加えた複製は、その他の点では逐語的複製として扱うことができます。

どちらの表紙にも必要な文章が多すぎて読みにくい場合は、最初に挙げたもの（無理のない範囲で）を実際の表紙に載せ、残りは隣のページに続けて載せます。

あなたが『文書』の『不透明な複製物』を100部以上発行または頒布する場合、あなたは各『不透明な複製』に機械可読の『透明な複製物』を同梱するか、あるいは各『不透明な複製物』に、一般的なネットワーク利用者が公衆標準ネットワーク・プロトコルを使用して『文書』の完全な『透明な複製物』をダウンロードできるコンピュータ・ネットワーク上の場所を明記しなければなりません。後者の選択肢を使用する場合、あなたは『不透明コピー』の配布を大量に開始する際に、この『透明コピー』が、あなたがその版の『不透明コピー』を(直接、あるいはあなたの代理人や小売店を通じて)公衆に配布した最後の時点から少なくとも1年経過するまでは、記載されたこのようにアクセス可能であり続けることを保証するために、合理的に慎重な手段を講じなければなりません。

大量のコピーを再配布する前に、文書の作成者によく連絡し、最新版の文書を提供する機会を与えることが要求されますが、必須ではありません。

## 4. 変更点

あなたは、上記第2項および第3項の条件の下で、『文書』の改変された版を複製し頒布することができます。ただし、改変された版をまさに本許諾書の下でリリースし、改変された版が『文書』の役割を果たし、その結果、改変された版の頒布と改変がそのコピーを所持する誰に対しても許諾されることを条件とします。加えて、あなたは『改変されたバージョン』において以下のことを行わなければなりません：

- A. タイトルページ（および表紙がある場合は表紙）には、『文書』のタイトルおよび旧版のタイトル（旧版がある場合は、『文書』の「歴史」の項に記載されているはずです）とは異なるタイトルを使用してください。旧版の発行元が許可している場合は、旧版と同じタイトルを使用してもかまいません。
- B. タイトルページには、改変されたバージョンの改変の著者として、その文書の主要な著者のうち少なくとも5人（主要な著者の数が5人未満の場合はその全員）とともに責任を負う1人または複数の個人または団体を記載してください（ただし、その個人または団体がこの要件からあなたを免除する場合はこの限りではありません）。

- C. タイトルページには、発行者として修正版の発行者名を明記してください。
- D. 文書のすべての著作権表示を保存してください。
- E. 他の著作権隣接して、あなたの改変に対する適切な著作権表示を追加してください。
- F. 著作権表示の直後に、以下の補遺に示す形式で、本許諾書の条項の下で改変されたバージョンの利用を公衆に許可する利用許諾表示を含めてください。
- G. そのライセンス通知には、『文書』のライセンス通知で指定されている、変更不可のセクションと必要なカバーテキストの完全なリストを保存してください。
- H. 本使用許諾の変更されていないコピーを同封してください。
- I. "History"と題されたセクションを保存し、そのタイトルを保存し、少なくともタイトルページに記載されている修正版のタイトル、年、新しい著者、出版社を記載した項目をそのセクションに追加してください。もし『文書』に「歴史」と題されたセクションがない場合、『文書』のタイトルページに記載されているように、『文書』のタイトル、年、著者、出版社を記載したセクションを作成し、前文でようやく、修正版について記述した項目を追加してください。
- J. 文書の透過的なコピーを公開するために、その文書で指定されているネットワークの場所(もしあれば)を保存し、同様に、その文書の基になった旧版の文書で指定されているネットワークの場所も保存してください。これらは「履歴」セクションに置くことができます。あなたは、『文書』自身よりも少なくとも4年前に出版された著作物、あるいはその著作物が参照する版の原版発行者が許可した場合、その著作物のネットワーク上の位置を省略することができます。
- K. 「謝辞」または「献辞」と題されたセクションについては、そのセクションのタイトルを保持し、そこに記載された各寄稿者の謝辞および／または献辞の内容および論調をすべて保持します。
- L. 文書のすべての不变セクションを、そのテキストもタイトルも変更せずに保存します。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見なされません。
- M. 「裏書」と題されたセクションを削除してください。このようなセクションは、修正版には含まれない場合があります。
- N. 既存のセクションのタイトルを「エンドースメント」に変更したり、不变のセクションのタイトルと矛盾させたりしないでください。
- O. 保証の免責事項を守ってください。

変更後のバージョン』に、『二次的著作物』(Secondary Section)として適格であり、『文書』からコピーされた素材を含まない、新しい前文セクションや付録が含まれる場合、あなたは任意でこれらのセクションの一部または全部を変更不可のセクションとして指定することができます。これを行うには、改変された版のライセンス告知にある「変更不可のセクション」のリストに、それらのセクションのタイトルを追加してください。これらのタイトルは、他のいかなるセクションのタイトルとも区別されなければなりません。

「エンドースメント」というセクションを追加してもかまいませんが、そのセクションには、さまざまな当事者によるあなたの修正版に対するエンドースメント-たとえば、査読の記述や、ある標準の権威ある定義としてある組織によって承認されたという記述など-だけを含めるようにしてください。

表紙テキストとして5語までの文章を、裏表紙テキストとして25語までの文章を、修正版の表紙テキストリストの最後に追加することができます。表表紙テキストおよび裏表紙テキストの一節のみを、一団体によって（または一団体による取り決めによって）追加することができます。もし『文書』に既に同じカバー・テキストが含まれており、それが以前にあなたによって追加されたものであるか、またはあなたの取り決めによって追加されたものである場合

しかし、古いものを追加した以前の発行者から明示的な許可を得れば、古いものを置き換えることができます。

本許諾書によって、『文書』の著者および発行者は、その名前を宣伝のために使用すること、あるいは改変された『文書』の支持を表明したり示唆したりすることを許可しないものとします。

## 5. 文書結合

あなたは、本許諾書の下でリリースされた他の文書と、改変されたバージョンについて上記第4項で定義された条件の下で、『文書』を結合することができます。ただし、その結合の中に、改変されていないすべてのオリジナル文書の変更不可部分を含め、そのライセンス告知にあなたの結合著作物の変更不可部分としてそれらすべてを記載し、かつそれらのすべての保証の否認を保持することを条件とします。

結合された著作物には本許諾書が一部含まれていればよく、複数の同一の変更不可部分が一つのコピーで置き換えられてもよい。同じ名前で異なる内容の複数の「変更不可部分」が存在する場合、そのような各「変更不可部分」のタイトルの末尾に、その「変更不可部分」の原著作者または発行者の名前(既知であれば)、あるいは一意の番号を括弧書きで追加することによって、その「変更不可部分」を一意なものとしてください。結合著作物の利用許諾の告知にある、変更されないセクションの一覧のセクションタイトルにも、同じ調整を加えてください。

同様に、「謝辞」と題されたセクションと「献辞」と題されたセクションもすべて組み合わせてください。裏書」と題されたセクションはすべて削除してください。

## 6. 文献集

あなたは、『文書』および本許諾書の下でリリースされたその他の文書から成る文書集を作成し、様々な文書に含まれる本許諾書の個々のコピーを、その文書集中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、その他のすべての点において、各文書の逐語的コピーに関する本許諾書の規則に従う必要があります。

あなたは、そのような文書集から一つの文書を抜き出し、本許諾書の下で個別に頒布することができます。ただし、抜き出した文書に本許諾書のコピーを挿入し、その文書の逐語的な複製に関する他のすべての点において本許諾書に従うことを条件とします。

## 7. 独立作品とのアグリゲーション

文書』またはその派生物と、他の別個独立の文書や著作物とを、記憶媒体や頒布媒体の一巻の中に、あるいは一巻の上に編集したものは、その編集物から生じる著作権が、その編集物の利用者の法的権利を、個々の著作物が許容する範囲を超えて制限するために利用されない場合、「総体」と呼ばれます。文書』が総集編に含まれる場合、本許諾書は総集編に含まれる他の著作物であって、それ自体が『文書』の二次的著作物ではないものには適用されない。

第3項のカバーテキストの要件が当該コピーに適用される場合、当該文書が総体全体の2分の1未満であれば、当該文書のカバーテキストは、総体内で当該文書を囲むカバー、または当該文書が電子形式であればカバーに相当する電子表紙に掲載することができます。それ以外の場合は、総体全体を囲む印刷された表紙に掲載しなければなりません。

## 8. 翻訳

翻訳は一種の改変とみなされるため、あなたは第4項の条件に従って『文書』の翻訳を頒布することができます。変更不可セクションを翻訳に置き換えるには、そのセクションの特別な許可が必要です。

ただし、あなたはこれらの変更不可部分の原版に加えて、一部または全部の変更不可部分の翻訳を含めることができます。あなたは、本許諾書の翻訳、および『文書』中の全てのライセンス表示、そして保証の否認を含めることができますが、その場合、本許諾書の英語原文、およびこれらの表示や否認の原文も含めるものとします。翻訳版と本許諾書の原版、または通知や免責事項の原版との間に不一致がある場合は、原版が優先されます。

文書のセクションのタイトルが「謝辞」、「献辞」、「歴史」である場合、そのタイトル（セクション1）を保持するための要件（セクション4）では、通常、実際のタイトルを変更する必要があります。

## 9. 終了

あなたは、本許諾書の下で明示的に規定されている場合を除き、『文書』を複製、変更、サブライセンス、または頒布することはできません。それ以外の方法で複製、変更、サブライセンス、または頒布しようとする試みは無効であり、本使用許諾に基づくあなたの権利は自動的に消滅します。

ただし、あなたが本許諾書への違反をすべて止めた場合、特定の著作権者からのあなたのライセンスは、(a)著作権者が明示的かつ最終的にあなたのライセンスを終了させない限り、またそれまでは暫定的に、(b)著作権者が違反の停止後60日以前に何らかの合理的なあなたに違反を通知しなかった場合は、永久的に復活します。

さらに、著作権者が何らかの合理的な手段であなたに違反を通知し、あなたがその著作権者から本許諾書に対する違反の通知を(いかなる作品についても)初めて受け取り、あなたがその通知を受け取ってから30日以内に違反を是正した場合、特定の著作権者からのあなたのライセンスは永久に復活します。

本節に基づくあなたの権利の終了は、本許諾書に基づいてあなたから複製物や権利を受領した当事者のライセンスを終了させるものではありません。あなたの権利が終了し、恒久的に復活しない場合、同じ資料の一部または全部の複製物を受け取ったとしても、それを使用する権利はあなたに与えられません。

## 10. 本ライセンスの将来の改訂

フリーソフトウェアファウンデーションは、GNU自由文書利用許諾書の新しい改訂版を隨時発表することができます。そのような新バージョンは、現在のバージョンと精神的には似ていますが、新たな問題や懸念に対処するために細部において異なるかもしれません。<http://www.gnu.org/copyleft/> をご覧ください。

本許諾書の各バージョンには、バージョン番号が付けられています。『文書』において、本許諾書の特定のバージョン番号「またはそれ以降のバージョン」が適用されると指定されている場合、あなたはその指定されたバージョンか、フリーソフトウェア財団によって公表された(草案としてではなく)それ以降のバージョンのいずれかの条項と条件に従うという選択肢を持つことになります。『文書』に本許諾書のバージョン番号が明記されていない場合、あなたはフリーソフトウェア財団によって(草案としてではなく)公表されたどのバージョンでも選ぶことができます。『文書』において、代理人が本許諾書の将来のどの版を使用できるかを決定できると指定されている場合、その代理人がある版を受諾すると公言することで、あなたは『文書』においてその版を選択することが永久に許可されることになります。

## 11. リライセンシング

「大規模多作者コラボレーション・サイト」(あるいは「MMCサイト」)とは、著作権で保護される作品を公開し、また、誰もがそれらの作品を編集できるような著名な設備を提供するワールド・ワイド・ウェブ・サーバを意味します。誰でも編集できる公開ウィキは、そのようなサーバの一例です。MMCサイトに含まれる「大規模複数著者の共同作業」(Massive Multiauthor Collaboration) (あるいは「MMC」)とは、このようにしてMMCサイトで公表される、著作権で保護される著作物のあらゆる集合を意味します。

「CC-BY-SA」とは、カリフォルニア州サンフランシスコに主たる事業所を置く非営利法人であるCreative Commons Corporationが発行するCreative Commons Attribution-Share Alike 3.0ライセンス、および同組織が発行する同ライセンスの将来のコピーレフト版を意味します。

「組込」とは、文書の全部または一部を他の文書の一部として発行または再出版することを意味します。

MMCは、それが本許諾書の下でライセンスされ、かつ、本許諾書の下で本MMC以外のどこかで最初に公表され、その後にそのMMCに全部または一部が組み込まれたすべての作品が、(1)カバーテキストや不变部分がなく、(2)こうして2008年11月1日より前に組み込まれた場合、「再ライセンスの対象」となります。

MMCサイトの運営者は、2009年8月1日以前であればいつでも、CC-BY-SAのもと、同サイトに含まれるMMCを再公開することができます。