MIRANTIS

k0RDENT AI

リファレンスアーキテクチャ

Mirantis k0rdent AI による AI Factory の構築

Translated by: CREATIONLINE

v1.1

目次

はじめに

人工知能（AI）および 機械学習（ML）のワークロードの増加は、既存および新規のインフラストラクチャへの投資に計り知れない影響を与えています。これにより、処理の複雑性は増し、要件は絶えず拡大し多様化し続けます。

さらに、新たな仕組みやアプローチが発見されるに従い、急速に変化し続けるユースケースの広大な範囲をサポートする必要性も生じています。

AI の重要トレンド

トレーニングから推論へ

これまでAIおよびMLのワークロードは、主にAIモデルのトレーニングと大規模機械学習のユースケースに焦点が当てられていました。しかし現在は推論ワークロードへとリソースが大きく移行し始め、推論タスクに最適化されたGPUやその他のアクセラレータを提供するベンダーが増えています。

示唆

- 複雑性：特殊なインフラストラクチャのニーズに対応する、広範囲に分散した**GPU**依存型のワークロードを管理するため、複雑性が増大します。
- コスト効率：組織はワークロードのニーズに基づき、コスト効率を優先させ、リソースの割り当てを最適化する必要があります。

アプリケーションロジックからAIへ

現在アプリケーション内で定まった処理手順として捉えられている業務処理の規則は、将来、さまざまな**LLM**（大規模言語モデル） やMLベースのモデルを活用する**AI**駆動型の推論・意思決定ロジックシステムに置き換えられるでしょう。

これらのシステムは今後、より柔軟な自然言語インターフェースと、独自のロジックを学習し自ら洗練させられる**AI**エージェントを展開するようになります。

示唆

- インフラストラクチャのリソースが増加：新たなモデルを実行し、良好なユーザーエクスペリエンスを提供するため、**GPU**や**CPU**など特殊なリソースの必要性が大幅に増加します。
- ガバナンスとコンプライアンスのリスク：結果が常に一定ではないものへとシステムを移行することでリスクが増大します。そのため、より多くの監視と均衡の仕組み を導入する必要性が高まります。

エージェント型AI

エージェント型AIモデルは、単一の大規模モデルを利用する手法から、複数のモデルを組み合わせて連携させる手法への転換を意味します。これにより外部データソースを利用して外部システムと統合・連携し、問題を解決するために動的に調整できるロジックチェーンを構築し、外部システムを直接制御できるようになります。

示唆

- 永続的でステートフルなワークロード： エージェント型モデルは、永続的でステートフルなワークロードを生成します。これらは時系列に沿ったメモリやコンテキストを維持するためベクターストアなどの多様な外部システムに依存します。基盤となるシステムは、これらのモデルと関連システムの可用性を確保できるようサポートする必要があります。
- 高速性と低遅延性：高パフォーマンスとユーザーエクスペリエンスを担保するため、高速で信頼性の高い通信とリソースへのアクセスが不可欠となります。

課題

効果的なAIおよびMLプラットフォームを導入する際の課題は、大規模な **IaaS**（**Infrastructure as a Service**）ソリューションを導入する場合と類似しています。さらに**HPC**（**High-Performance Computing**） 特有の複雑な問題が加わります。具体的には、以下の課題が含まれます。

- **RDMA**ネットワーキング
- GPUの割り当てと分割
- 高度なスケジューリング（処理の計画・実行） 要件
- パフォーマンスのチューニング、最適化
- **Kubernetes**の拡張

AIおよびMLプラットフォームは、インフラストラクチャのプロビジョニング、メンテナンスに関連する新たな課題に対処し、さらに各ワークロードには次のような固有のニーズがありそれぞれをサポートする必要があります。

- 価値創出までの時間：AIおよびMLのインフラストラクチャは、専門的なファインチューニングやプロビジョニングを必要とします。そのため従来のコンピューティングシステムよりも設定に時間がかかり、学習コストも高くなる可能性があります。
- マルチテナンシー：データのセキュリティ、リソースの割り当て、競合管理のために、厳格かつ柔軟なマルチテナンシーが必要となります。
- データ主権：通常、AIおよびMLのワークロードは、データドリブンです。あるいはモデルや重みに固有の知的財産（**IP**） を含みます。そのためこれらのデータをどこでどのように使用するかを制御することが極めて重要です。
- 規模と拡散：通常、AIおよびMLに使用されるインフラストラクチャは多数のコンピューティン

グシステムで構成されているか、エッジや**IoT**のようなワークロードのために高度に分散されています。
- リソースの希少性：GPUやその他の重要なコンピューティングリソースは希少であるため、適材適所で活用する必要があります。
- スキルの利用可能性：多くのAIおよびMLプロジェクトは、インフラストラクチャの専門家ではない、または専門家になりたいと思っていないデータサイエンティストや開発者によって実行されていることを念頭に置くべきです。

クラウドとGPU as a Serviceプロバイダー

CSPと**GPUaaS**プロバイダーは、先述の一般的なAI/MLプラットフォームの課題に加え、従量課金制のマルチテナントサービスを提供するために乗り越えるべき独自の課題を抱えています。

これらの環境では次のような要件が求められます。

- 地域的および規制上の要件の遵守
- ユースケース固有のモデルとデータ標準へのサポート
- データプライバシー、ビジネス、アプリケーションのセキュリティの保護
- クォータ管理、リソースのコミットメント比率、パフォーマンス管理
- 厳格なテナント分離の常時確保
- 効率性とリソースの最大利用率の確保
    - GPUの選択と分割 の最適化
    - ネットワーク遅延とパフォーマンスの最適化
    - ストレージのパフォーマンスとコストの最適化
- サービス継続性とサービスレベル（**SLA**） を保護するためのモニタリング
    - CPUとGPUの健全性と利用率
    - ネットワーキング、ストレージ、サポートインフラストラクチャ
    - モデルとアプリケーションのパフォーマンスと遅延
- モデルとアプリケーションを迅速に反復し、デプロイする能力
- コストの効果的な追跡と管理
- 上記すべての迅速かつ継続的な管理
-

AIインフラストラクチャのトレンド

クラウドネイティブ**AI**と**Kubernetes**の全面的な採用

AIインフラストラクチャは、クラウドネイティブでコンテナベースのアーキテクチャへと急速な移行を遂げています。KubernetesはAIワークロードのためのデファクトスタンダードとなるオーケストレーションレイヤーとして活用されています。さらに、AIワークロードはオンプレミスとクラウドの両者で実行されています。

OpenAIやGrokなど主要なAI企業は、Kubernetesをシステム内のすべてのソフトウェア基盤となる構成

要素として使用しています。

これらの新しいパターンには、次が含まれます。

- GPU オペレーター、AI オペレーター、K8s ネイティブな ML パイプラインを活用した Kubernetes 上での AI 実行
- Federated K8s と ML Ops を使用したハイブリッド/マルチクラウド AI
- AI デプロイメントのための Infrastructure-as-Code および Policy-as-Code（例：Terraform、Helm、ArgoCD）

AI 向けに最適化された高性能なネットワークとストレージ

GPUや**NPU**（ニューラル プロセッシング ユニット） などの新興技術が急速に進化し、増大するネットワーキングやストレージのパフォーマンス要件をサポートする新たな技術が創出されました。

このようにしてネットワーキングとストレージのオプションにより性能が向上した結果、データセンター内で管理する必要のある新たな技術レイヤーが生まれています。それは次のようなものです。

- 超高速ネットワーキング：AIのデータ移動ニーズに対応するための、超高速なネットワーキング（Ethernet、InfiniBand、NVLink、NVSwitch、CXLなど）。
- ストレージアーキテクチャの進化：AIデータセットをサポートするために進化したストレージアーキテクチャ（オブジェクトストア、GPUDirect Storage、データレイクハウスなど）。
- メモリの革新：AIアクセラレータに効率的にデータを供給するためのメモリの革新（HBM、CXL、共有メモリファブリックなど）。

電力供給とサステナビリティ

電力消費の大きいGPUやデプロイ規模の拡大など、複合的な要因でAIインフラストラクチャは莫大なエネルギーを必要としています。これにより電力供給と消費に関して次のようなトレンドを生んでいます。

- データセンターの立地が今後ますます重要になります。電力の可用性と代替冷却オプションが重要な要素となったことで、一部のデータセンタープロバイダーは、独自の発電設備を建設し始めています。
- 電力と冷却の可用性を考慮し、ワークロードスケジューリングの最適化と連携した、エネルギー効率の高いアクセラレータや**GPU**の開発が進みます。
- 液冷（リキッドクーリング） や浸漬冷却（イマージョンクーリング） を含む、新しい冷却アプローチの採用が急速に広まります。
-

データセキュリティのさらなる強化

データとワークロードのセキュリティと隔離のレベル向上に対する需要が高まりました。その結果、ソブリンクラウドのデプロイに加え、ゼロトラストとコンフィデンシャルコンピューティングの採用を組み合わせた、強制力があり厳格なマルチテナンシーに注目が集まっています。

- ハードウェアが強制するマルチテナント隔離への需要が、サービスプロバイダー環境とエンタープライズ環境の両方で増加しています。
- 厳格な隔離をサポートし、実行環境のセキュリティを確保するために、ゼロトラストとコンフィデンシャルコンピューティングが採用されています。

スーパーコンピューターの復活

驚くべきことに現在、スーパーコンピューティング技術や、InfiniBand、SLURMのような高度なスケジューラーといったパラダイムが再興しています。

クラウドネイティブのワークロードは、主にスケールアウトやマルチコア処理のために設計されていますが、AIワークロードはこれとは大きく異なります。AIワークロードは、多数のGPUサーバーを、統合メモリを持つ単一のスーパーコンピューターに変えることを要求する場合があります。これは、古典的なスーパーコンピューティングセンターにおけるMPI（Message Passing Interface）ワークロードと同様に、RDMAや超高性能なネットワーキングを必要とします。

Mirantis k0rdent AI プラットフォームの概要

Mirantis k0rdent AI は、大規模なAIインフラストラクチャのための完全なプラットフォームであり、k0rdentのプロバイダーおよびテンプレートフレームワークを活用し構成されたものです。

これにより、企業やサービスプロバイダーは、GPUおよびCPUリソースへの制御されたアクセス、RDMAネットワーキング、高性能ストレージ、および高性能ネットワーキングを必要とするAIおよびMLワークロード向けに、大規模なマルチテナント対応のソブリンインフラストラクチャソリューションをデプロイし、管理することが可能になります。

また、**Everywhere Inference** PaaS や NVIDIA AI Enterprise ソフトウェア エコシステムを含む、選択可能な**AI**プラットフォームサービスの提供も可能にします。

最も高いレベルでは、Mirantis k0rdent AIには次のすべてのコンポーネントが含まれます。

コンポーネント	サブコンポーネント
Mirantis k0rdent Enterprise	Kubernetesクラスタ管理、アプリケーションサービス管理、オブザーバビリティ、セキュアなソフトウェアサプライチェーン、オペレーターポータル
Mirantis k0rdent Virtualization	KubeVirt、OpenStack（Mirantis OpenStack for Kubernetes）
Mirantis k0rdent AI Factory Operator	ベアメタルマネージャー、ファブリックマネージャー、DPUマネージャー
場所を問わないAI推論	オンデマンド推論エンジン PaaS
サードパーティ ソフトウェア	SLURM、KubeFlow
GPU ベンダーソフトウェア スタック	NVIDIA、AMD、Intel

Mirantisは、Mirantis k0rdent AIプラットフォームを活用したAI Factory 構築のためのリファレンスアーキテクチャを提案します。その詳細を次に示します。

Mirantis k0rdent AI プラットフォームの価値提案

> ***Mirantis k0rdent AI** は、AIとMLインフラストラクチャを大規模に構築、運用、最適化するための、セキュアで構成可能、拡張可能 かつソブリンなプラットフォームを提供します。*

信頼できる、構成可能な、ソブリンAIインフラストラクチャ

- ソブリン（主権対応）： オンプレミス、エッジ、またはハイブリッドクラウドなどインフラストラクチャ全体に対して完全な制御を維持できます。これにより、データレジデンシー（データ所在地の確保）、セキュリティ、および地域ごとの規制に対する準拠が保証されます。
- 構成可能なアーキテクチャ： ユーザーは、コンピューティング、ストレージ、GPU、ネットワーキングの各レイヤーにおいて、再利用可能なテンプレートから、特定の**AI**ワークロードのニーズに合わせてインフラストラクチャを組み立てることができます。

価値創出までの時間を加速

- **Mirantis k0rdent AI** のテンプレート化された宣言的モデルを使用した迅速なプロビジョニングにより、AIワークロードをハードウェア設置から数日以内にデプロイできます。
- モデルやサービスのプロトタイプ作成、イテレーション、デプロイを加速させ、AI開発のライフサイクルを劇的に短縮します。

大規模なマルチテナンシーと分離

- **GPU**、**VM**、**Kubernetes**の各レイヤーでの隔離を含む、厳格なマルチテナンシーをサポートします。
- **GPU-as-a-Service**のシナリオにおいて、チーム、部門、テナント間で、コンピューティングリソースを安全かつ効率的に共有することを可能にします。

統合コントロールプレーンによるシームレスなライフサイクルマネジメント

- ベアメタル、仮想化インフラストラクチャ、Kubernetes、およびAIサービスのフルスタックなライフサイクルマネジメントを実現します。
- 宣言的なテンプレートにより、トレーニングから推論ワークロードに至るまで、クラスタや環境全体で一貫性と再現性を確保します。
- **k0rdent**は、単一の管理画面を通じて、クラスタ、サービス、コストパフォーマンス分析などフルスタック管理を提供します。

- **k0rdent Observability and FinOps (KOF)** は、GPU、ネットワーク、ストレージ、サービスパフォーマンスのリアルタイムと履歴の分析を含む一元的なモニタリングを提供します。
- コストレポート、リソース使用状況の追跡、課金システムとの統合により、財務の透明性と効率性をサポートします。

幅広いエコシステム互換性

- **NVIDIA**、**AMD**、**Intel** のAIアクセラレータに対する組み込みのサポートを提供します。
- オープンスタンダードを活用し、AI/MLツール、オブザーバビリティ、CI/CD、セキュリティなど、厳選された統合ソリューションをk0rdentカタログより提供します。

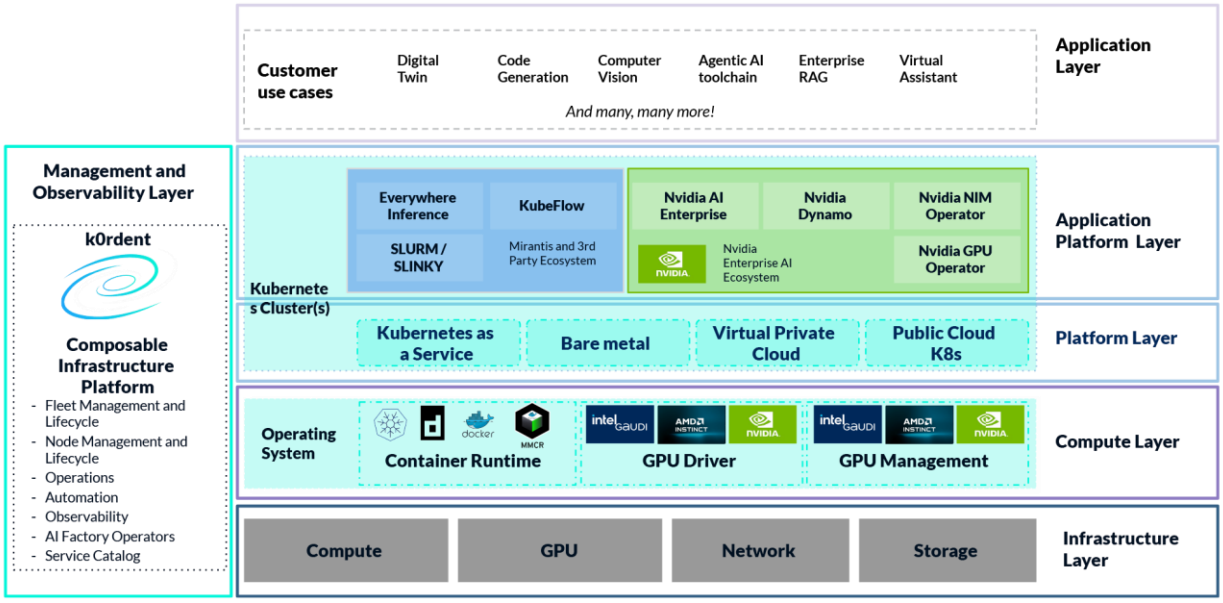
デプロイモデルの柔軟性

- 次に挙げる環境全体で、複数のAIワークロードタイプ（トレーニング、ファインチューニング、推論）をサポートします。
    - 占有または共有のベアメタル（物理サーバー）
    - 仮想化環境（KubeVirt/OpenStack）
    - パブリッククラウドまたはハイブリッド/マルチクラウド
    - エッジロケーション（拠点）

主な機能とベネフィット

機能	ベネフィット
厳格なマルチテナンシー	テナント間でのリソースの完全な分離を実現
ベアメタルのライフサイクル マネジメント	処理能力の迅速な展開と廃止が可能
モニタリングと オブザーバビリティ	ネットワーク、ストレージ、GPU、コンピューティングなど、すべてのインフラストラクチャの稼働状態に関するリアルタイムおよび過去データの洞察をフィードバック。推論サービスの測定データ収集もサポート。
高度なネットワーク	完全なノンブロッキング ネットワークにより、ワークロード、AI/ML処理、およびGPU間通信のレイテンシを低減。高度なネットワークセキュリティ機能の提供によりマルチテナンシーを強化。
Kubernetes-as-a-Service	物理サーバーから仮想マシンまで、すべてのワークロードを展開するための単一で統一されたメカニズムを提供。
テンプレート化されたデプロイメント	基盤となるインフラストラクチャサービスと、その上で動作するワークロードに対して、再現性とモジュール性を備えたレシピを定義し、設計と運用を簡素化。
監査とロギング	コンプライアンスとガバナンスのための長期的な履歴監査ログ
FinOpsレポート作成	請求および財務システムとの統合の容易さ
セルフサービスポータル	オペレーターとエンドユーザーにとっての使いやすさ
組み合わせ可能なAIオプション	トレーニングと推論のための異なるAI PaaSの選択
高度なAI最適化	GPU分割、ビンパッキング、高度なスケジューラーなど

Mirantis AI Factory リファレンスアーキテクチャ



コンセプト概要図- 簡略化バージョン

管理およびオブザーバビリティ レイヤー

Mirantis k0rdent Enterprise

k0rdentは、分散コンテナ管理やその他の機能を提供するコンポーザブル（組み合わせ可能）なインフラストラクチャ プラットフォームであり、以下の主要な機能を含んでいます。

- マルチクラウドおよびハイブリッドインフラストラクチャのサポート： Kubernetes Cluster API を活用し、**AWS EC2**、**AWS EKS**、**Azure Compute**、**Azure AKS**、**vSphere**、**OpenStack**を含む多様な環境全体で、クラスタのプロビジョニング（設定・準備）と管理を行います。
    - 追加のインフラストラクチャ プロバイダーに対応するための拡張性をサポートするように設計されています。
    - 物理サーバーの包括的なプロビジョニングとライフサイクルマネジメントを提供します。
- セキュリティとコンプライアンスの自動化：一元化されたポリシーの適用と宣言的な自動化を実装し、すべての環境でセキュリティとコンプライアンスを確保します。
- テンプレートモデル：多様なインフラストラクチャ全体で、Kubernetesクラスタとサービスのデプロイメントと管理を効率化します。このモデルはモジュール性、再利用性、宣言型構成を提供し、プラットフォーム エンジニアが複雑な環境を効率的に定義・管理することを可能にします。

- サービスと状態管理：k0rdentは、Kubernetesクラスタ内のサービスのデプロイメントと構成を管理するために ServiceTemplates を採用しています。これらのテンプレートは、サービスを定義するための標準化され、再利用可能な方法を提供し、異なる環境間での一貫したデプロイメントを容易にします。（詳細はk0rdentカタログを参照）

コアコンポーネントとアーキテクチャ

- **k0rdent Cluster Manager (KCM)**： プロビジョニングや更新を含む、Kubernetesクラスタのデプロイメントとライフサイクルマネジメントを取り扱います。
- **k0rdent State Manager (KSM)**：クラスタ内における不可欠なサービスやポリシーのインストールとライフサイクルを管理します。
- **k0rdent Observability and FinOps (KOF)**: モニタリング、イベントおよびログ管理、そしてコスト最適化のためのツールを提供します。

KCM (k0rdent Cluster Management)	**KSM** (k0rdent State Management)	**KOF** (k0rdent Observability & FinOps)
KCM はクラスタのライフサイクルを処理します。 クラスタとは何か ● **CAPI**（**Cluster API**） およびインフラストラクチャ プロバイダーを介した、**Kubernetes**（**k8s**）クラスタのライフサイクルマネジメントとプロビジョニング（設定・準備）。 ● ClusterTemplates（クラスタ テンプレート）を用いた、クラスタの **CRUD**（作成、読み取り、更新、削除）、アップグレード、およびバージョニングの処理。 ● 大規模なハイブリッド／マルチクラウド環境におけるクラスタのプロビジョニングを可能にする、インフラストラクチャとの統合。	KSMはサービスとランタイムの状態を管理します。 クラスタ上で何が実行されるか ● クラスタ上でのアプリケーションとサービスのデプロイメントとライフサイクルを管理。 ● **ServiceTemplates**（Helmのような仕組み）を使用し、クラスタ間での一貫したデプロイメントを実現。 ● KCMによってプロビジョニングされたクラスタ上で実行され、クラスタのセットアップとは独立して動作。	KOFは、可視性とコスト管理を提供します。 クラスタのパフォーマンスとコストはどうなっているか ● 管理対象のすべてのクラスタからのメトリクス、ログ、トレースを集約。 ● トラブルシューティング、SLO（サービスレベル目標）、およびコスト最適化のための大規模なオブザーバビリティを実現。 ● シングルペインオブグラスを提供し、アラート、可視化、FinOps（財務オペレーション） のシステムへデータを供給。

k0rdent のオブザーバビリティとFinOps

KOFは、メトリクス、ログ、トレースを活用し、k0rdentが管理するクラスタとサービス全体で統合されたオブザーバビリティとコスト管理を可能にします。

これは、一元化された画面を通じて、一元的な可視化、アラート、および**FinOps**（財務オペレーション）機能により実現するものです。

メトリクス管理
- 集中型メトリクス収集
- マルチクラスタ集計
- 長期保存
- クエリフェデレーション

コスト管理：
- OpenCost 統合
- クラウドコスト追跡
- リソース使用状況監視
- コスト配分とレポート作成

ログ管理:
- ログ集約
- 集中型ストレージ
- 検索と分析
- ログ保持ポリシー

可視化:
- 集中型Grafana
- 地域別Grafanaインスタンス
- カスタムダッシュボード
- アラート管理

Mirantis k0rdent AI Factory Operator

Mirantis k0rdent AI プラットフォームには、**Kubernetes** オペレーターとして開発された、いくつかの重要な新機能が含まれます。これらは「**AI Factory**」を大規模に運用するために必要な機能を補完するように設計されており、次のすべてを網羅しています。

AI Factory Operator	目的	開発状況	注記
ベアメタル オペレーター	ベアメタルのフルライフサイクルマネジメント	ベータ版	先行する取り組み（Mirantis Container Cloud および Mirantis OpenStack for Kubernetes）に基づき
ネットワーク ファブリック オペレーター	AIスイッチファブリックの構成（例：InfiniBandおよびEthernet）	開発中	
DPUオペレーター	DPUワークロード向けライフサイクルマネジメント	設計中	高度なハードマルチテナンシーによるネットワーク構築
ブロックストレージ オペレーター	高度なブロックストレージ構成とプロビジョニング	設計中	
仮想化オペレーター	KubeVirtインフラストラクチャのライフサイクルマネジメント（KubeVirt上に構築されたk8sクラスタのデプロイメントも含む）	GA	Mirantis k0rdent 仮想化の一部

*注：> 各オペレーターはプラグ可能なモデル（着脱可能な設計）に基づいており、複数のアーキテクチャへの拡張をサポートしています。（例：**NVIDIA BlueField DPU**や**AMD Pensando DPU**など）*

k0rdent カタログ

k0rdentアプリケーションカタログ は、多様なインフラストラクチャにわたるKubernetesベースのサービスのデプロイと管理を効率化するために設計された、検証済みで本番環境対応の統合ソリューションを厳選して提供するリポジトリです。このカタログは、プラットフォーム エンジニアに対して事前設定済みのテンプレートや **Helm** チャートの一元化されたコレクションを提供します。これにより、内部開発者プラットフォーム（**IDP**）の迅速な構築を容易にし、AI や機械学習アプリケーションを含む最新のワークロードを強力にサポートします。

本カタログは、次のような幅広いサービスカテゴリーを網羅しています。

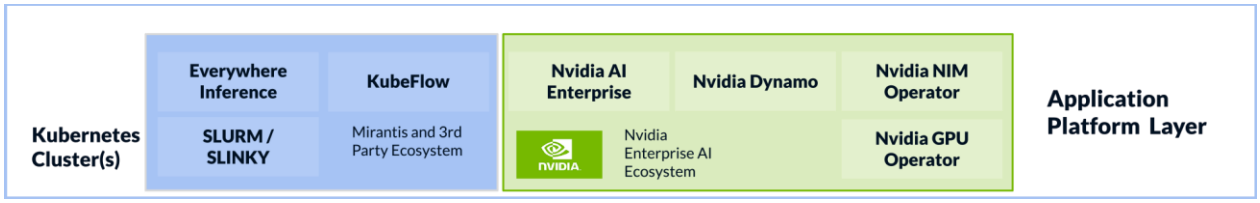
- **AI/**機械学習：機械学習（ML）や深層学習（DL）の開発から本番運用までを支えるClearML などのツール。
- **CI/CD**：宣言的で GitOps スタイルの継続的デリバリーを実現するArgoCDなどのソリューション。
- モニタリングとオブザーバビリティ：リソースの使用状況やパフォーマンス分析を行うための **Kube Prometheus Stack** や **cAdvisor** などのツール。
- セキュリティとコンプライアンス：Kubernetes ネイティブなポリシー管理を可能にする **Kyverno** などの統合機能。
- ネットワークとロードバランシング：ネットワークとセキュリティを担う Calico や、負荷分散（ロードバランシング）を行う **Ingress-NGINX** などのサービス。
- ストレージソリューション：ストレージの自動プロビジョニング（割り当て）を行うための **Amazon EBS CSI** や **Azure Disk CSI** などのドライバー。

アプリケーションレイヤー

アプリケーション層は、開発者やデータサイエンティストがコンテナ化されたワークロードを実行するための環境を提供します。これらのワークロードは、基盤となるプラットフォームサービスやインフラストラクチャが提供する機能を最大限に活用できるように設計されています。

この層による抽象化（複雑な内部構造を隠すこと）により、ユーザーは下位レベルにあるシステムコンポーネントの複雑な管理に煩わされることなく、自らのアプリケーションロジック、データ処理、そしてモデル開発に集中することが可能になります。コンテナ技術を活用することで、アプリケーション層は、さまざまなAIおよび機械学習タスクにおいて、可搬性、一貫性、効率的なリソース利用が保証されます。

アプリケーションプラットフォームレイヤー (PaaS または AIaaS)



アプリケーションプラットフォーム層は、AIワークロードの開発と実行を円滑にするために、適応性の高い**PaaS**（**Platform-as-a-Service**）サービスを提供します。ここでは、容易な拡張が可能で柔軟なPaaS機能が提供されます。

この層では、**k0rdent**の**ServiceTemplate**システムを活用しています。これにより、前述の図にある各アプリケーションワークロードを提供するKubernetes クラスタのデプロイを、「クッキー型」のような再現性のあるレシピとして定義することが可能になります。

- データサイエンティスト、機械学習エンジニア、アプリケーション開発者のためにインフラとツールを提供し、複雑な仕組みを抽象化、簡素化します。
- マネージドな推論環境（Managed Inference）、統合されたデータ管理、およびハードウェアアクセラレーション（処理の高速化）などの機能を備えています。
- 効率的で拡張性の高い AI/ML 環境を構築することで、新しい AI/ML 技術への適応性を確保し、将来の変化にも対応可能な基盤を提供します。

Mirantisとサードパーティーエコシステム

Mirantisは「Mirantis k0rdent AI」を通じて、拡大し続けるアプリケーションプラットフォームソリューション群へのアクセスとサポートを提供します。これにより、ビジネスニーズに最適なAI PaaSソリューションを自在に構成することが可能になります。

また、**k0rdent**カタログを利用することで、Mirantis製品およびサードパーティ製のツールの双方に簡単にアクセスできます。これらのツールは、Mirantis、または各サードパーティソフトウェアプロバイダーによって直接サポートされています。

場所を問わない推論基盤

“場所を問わない推論基盤” とは、エッジ、オンプレミス、クラウド、およびハイブリッドインフラストラクチャといったあらゆる環境において、機械学習モデルの超低遅延、高スループット、かつスケーラブルな展開を実現するために設計された、包括的なAI推論プラットフォームです。

このソリューションは、生成**AI**、物体検知、**AIoT**システムなど、リアルタイムな処理が求められるアプリケーションにおいて、特にその真価を発揮します。

スマートルーティング技術

- 低遅延と通信コスト削減の両立：データを最も近いデータセンターへルーティング（経路制御）し、遅延の最小化とトラフィック費用の低減を実現。
- リージョン指定の遵守：特定の地域（リージョン）内に留める必要があるワークロードを、指定された地域へ確実にルーティング。

柔軟性と拡張性

- マルチテナンシーマルチリージョン対応：ユーザー独自のクラウド、Mirantisのクラウド、またはオンプレミスのいずれでも運用可能。
- 秒単位のスケーリング：分単位ではなく「秒単位」で処理能力を拡張。パフォーマンスが極めて重要なAIアプリケーションに最適。
- 多様なモデルのサポート：あらゆるオープンソースモデルやカスタムモデルに対応し、モデルのライフサイクルマネジメントも容易。
- 柔軟なコンピューティング調整：ニーズに応じて計算リソース（コンピューティング）を自在に増減可能。

モニタリングとオブザーバビリティ

- リアルタイムのコスト追跡：リアルタイムでのコスト把握と、秒単位の課金（Billing）機能を提供。
- 詳細な監視：パフォーマンス分析やログ管理のための詳細なモニタリング機能。

NVIDIA Enterprise エコシステム

NVIDIA Enterprise AI Stack は、ハードウェアアクセラレーション（ハードウェアによる高速化）を活用したエンドツーエンドのプラットフォームです。企業が AI ワークロードを、データセンター、エッジ、クラウドのあらゆる環境において、効率的、安全、かつ大規模にデプロイできるよう設計されています。

このスタックは、**NVIDIA GPU**、**AI** ソフトウェア、ツール、ライブラリ、およびフレームワークを組み合わせ、検証済みでサポート可能な、パフォーマンスが最適化された構成として提供されます。

詳細は、付録の「NVIDIA エコシステム」を参照してください。

プラットフォームレイヤー

プラットフォーム層は k0rdent の機能を活用し、**Kubernetes** および仮想化プラットフォームのデプロイ、プロビジョニング（設定）、およびライフサイクルマネジメントに対する包括的な自動化を提供します。

これらの機能は、特化した **Kubernetes** コントローラーまたは、仮想化などの特化したアプリケーションとして実装されています。これらはすべて標準的な **k0rdent ClusterTemplates** および **ServiceTemplates** を通じて定義されます。

- プロビジョニング、スケーリング、アップデート、セキュリティパッチの適用、および廃棄（デコミッショニング）までを網羅。
- ベアメタルサーバー、仮想マシン、プライベートクラウド、およびパブリッククラウドサービスにわたる、ライフサイクルマネジメントの自動化。
- **AI**および機械学習のワークロード開発において、インフラ管理の複雑さを抽象化し簡略化。
- 一貫性の確保、設定ミスの削減、および運用効率の向上を実現。

KaaS - Kubernetes as a Service

Kubernetesクラスタの自動プロビジョニングは、幅広いターゲットインフラストラクチャソリューションをサポートしています。

- テンプレートベース：システム管理者が、独自の要件に合わせたカスタムテンプレートを定義できる柔軟性を提供します。
- シンプルなクラスタ定義：利用者や運用担当者は、サイズ情報（必要な処理能力の規模）を指定するだけでクラスタをデプロイできる、簡素化された操作体験を実現しています。

ベアメタル

ベアメタルのコンピューティングノードのプロビジョニングと管理を行い、宣言的なベアメタルプロビジョニングをサポートします。このベアメタルソリューションは、物理サーバーのライフサイクル全体をカバーし、以下の主要な機能を提供します。

サポートされるベアメタル ライフサイクルのフェーズ	
検出	イントロスペクションによる新規ハードウェアの自動登録。
プロビジョニング	ハードウェアへの**OS**イメージの起動および展開。
プロビジョニング	**OS**起動前（**Pre-boot**）および起動後（**Post-boot**）のカスタマイズのサポート。
モニタリング	ホストの状態、エラー、および健全性の追跡。
廃棄	撤去作業およびリソースクリーンアップの自動化。

主要な機能

- マシン イントロスペクション（構成の自動把握）
    - システム仕様（CPU、ディスク、メモリ、ネットワークカード、BIOS）を自動検出します。
    - ファームウェアやPCIデバイスを特定し、ネットワークインターフェースの状態や通信速度を把握します。
- 電源制御
    - **BMC**（管理用コントローラー）の認証情報を利用し、**Redfish**（推奨）や **IPMI** などの標準規格を用いて遠隔から電源操作を行います。
    - 必要に応じて、ベンダー独自の特殊なプロトコルへの拡張も可能です。
- ネットワーク起動
    - **PXE/iPXE** を活用したネットワーク経由の起動をサポートします。
    - **UEFI**と署名済みカーネルによるセキュアブート（安全な起動）や、Redfishの仮想メディアを用いた起動にも対応しています。
- マシンイメージのデプロイ
    - Redfish、IPMI、および仮想メディア機能を活用し、OSなどのイメージを効率的にハードウェアへインストールします。
- ローリングアップグレード
    - アップグレードはローリング方式（一台ずつ順番に実施）で行われます。これにより、システム全体を止めることなく更新が可能です。

仮想化

Mirantis k0rdent AIでは、仮想マシンベースのワークロードのプロビジョニングをサポートするため、2つの仮想化プラットフォームの選択肢を提供しています。

仮想化は、より高いレベルのテナント分離を可能にし、マルチテナンシーを強力にサポートします。これにより、ベアメタルと比較してリソース全体の利用効率を高めることができます。また、GPUの分割やパーティショニングにおいても、よりきめ細やかな設定が可能になります。

KubeVirt

KubeVirtは、**Kubernetes**ネイティブな手法を提供します。これにより、仮想マシンを、標準的な**Kubernetes**のワークロードと並んで「**Pod**」として管理することが可能になります。

OpenStack

Mirantis OpenStack for Kubernetesは、コンピューティング、ストレージ、およびネットワークリソースを管理するための完全な**IaaS**プラットフォームを提供し、データセンター全体の抽象化を実現します。

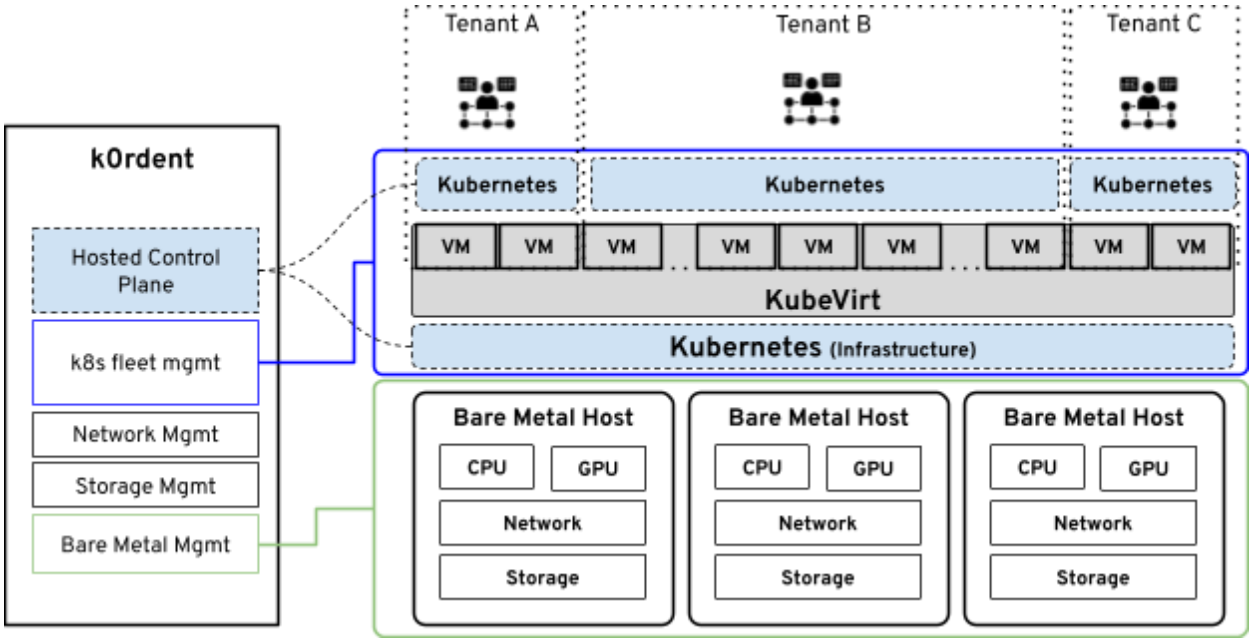
比較項目	**KubeVirt**	**OpenStack**
目的	**Kubernetes**によってオーケストレート（統合管理）されるワークロードとしての、Pod内での仮想マシン実行。	データセンター全体の抽象化を提供する、完全な**IaaS**プラットフォームの構築。
管理手法	**Kubernetes**ネイティブ**API**（CRD：カスタムリソース定義）による管理。	**OpenStack**固有の**API**による管理。
適した展開環境	エッジ、支店・拠点、**Kubernetes**環境に隣接する場所。	大規模データセンター。
ターゲットとなるワークロード	クラウドレディ / クラウドネイティブなワークロード。	**VM**ファースト（仮想マシン主体）、パブリッククラウドに代わるプライベートクラウド構築。
成熟度	実績のあるKVM技術に基づいた、新しいオーケストレーション手法。	成熟していて、大規模環境での検証済み。

ネストされた Kubernetes クラスタ

k0rdent、**KubeVirt**、および k0smotron を組み合わせることで、仮想化された Kubernetes クラスタの完全なエンドツーエンド管理が可能になります。

このソリューションにより、運用担当者は以下のメリットを備えた専用の **Kubernetes** クラスタをオンデマンドで提供できるようになります。

- 最小限のオーバーヘッド：システム負荷を抑えた効率的な動作
- 効果的なテナント分離：ユーザーごとの安全な切り分け
- リソース競合の管理：計算リソースの奪い合いを防止



このソリューションは次のメリットを提供します。

- 専用クラスタの提供
  - 単なる名前空間の分割ではなく、テナントごとに専用クラスタをプロビジョニング（構築）します。これにより、セキュリティと効率性が確保され、設定の競合や近隣ノードからの干渉を防ぐことができます。
- テナントの分離
  - **Kubernetes**層と**OS**層の両方で完全な分離を実現し、強力なセキュリティを担保します。
- ホスト型コントロールプレーン
  - k0smotronのモデルにより、Kubernetesの管理機能（コントロールプレーン）をワークロードから完全に分離して配置できます。これにより、個別に管理ノードを用意するリソースのオーバーヘッドを削減できます。
- リソース利用の最適化とビンパッキング
  - 各クラスタが必要な分だけ仮想マシン（VM）リソースを割り当てるため、物理インフラのリソースを効率的に使い切るようスケジュールできます。これにより、より効率的なビンパッキング（詰め込みの最適化）が可能になります。

クラウドプロバイダー：パブリッククラウドおよびプライベートクラウド

パブリッククラウドおよびプライベートクラウドサービスへの Kubernetes クラスタのプロビジョニングに加え、EKS や GKE といったパブリッククラウド独自の **Kubernetes** サービスの管理も行います。クラウドプロバイダーは以下の機能をサポートしています。

- 長期稼働ワークロード **(Long-running Workloads)**
  - 長期間運用される Kubernetes クラスタの、ローリングアップデートやメンテナンスを含むライフサイクルマネジメントをサポート。
- オンデマンドプロビジョニング
  - すべての必要なサービスを備えた新しいクラスタを数分で構築。一時的な負荷増大への対応や、QA（品質保証）・テスト用の一時的なワークロードに最適。
- 拡張可能なプロバイダーモデル
  - 主要なパブリッククラウドプロバイダーを幅広くサポート。さらに、独自のカスタムプロバイダーや API をサポートするための拡張も可能。

Mirantis k0rdent AI は、前述の EKS や GKE を含む、**Cluster-API （CAPI）** をリンク★対応のあらゆる Kubernetes サービスを管理するための機能を標準で備えています。

コンピュート層

OS、ドライバー、および関連するランタイムコンポーネントのプロビジョニングとライフサイクルマネジメントは、プロダクショングレード（本番環境レベル）の環境を維持する上で極めて重要な要素です。

OS

イメージベースのデプロイモデルを活用した、ホストOSのデプロイおよびライフサイクルマネジメント。Mirantis k0rdent AI は、以下のオペレーティングシステムライフサイクルマネジメント機能をサポートしています。

- プロビジョニング: OSの展開において、**iPXE** または **Redfish**仮想メディア をサポート（詳細はインフラストラクチャ層のセクションを参照）。
- **OS**ライフサイクルマネジメント: OSのライフサイクルマネジメントとして複数のオプションをサポート。詳細は次の通り。

サポートされるOSのライフサイクルマネジメント戦略

戦略	メカニズム	ユースケース	ライフサイクルアプローチ
イメージベースのホスト展開	Metal3 + Ironic	ホストベースのOS展開	**A:** 「Cordon & Drain（既存ノードの退避・分離）」メカニズムを利用した、ホストマシンの再デプロイ。 **B:** 長期稼働ホストのライフサイクルマネジメント（※Kubernetesオペレーターの展開が必要）。
イミュータブルOSイメージ	Kairos	**A:** エッジ展開 **B:** クラウド プロバイダ	**A / B** 共通**:** 「Cordon & Drain」メカニズムを利用した、ホストマシンの再デプロイ（丸ごと入れ替え）。

セキュアブートをサポート

Mirantis k0rdent AI は、信頼された起動状態を保証するためにセキュアブートの使用をサポートし、推奨しています。これをイミュータブルイメージと組み合わせることで、より高いレベルのセキュリティを提供し、改ざんのリスクを低減します。

k0rdent は **Kairos** と統合することでこの機能を実現しています。主要なすべての AI ハードウェアベンダー、GPU、および **SmartNIC** を含む、完全なセキュアブートサポートを有効にすることが可能です。

GPUドライバーおよびオペレーターのプロビジョニング

GPU リソースの構成と割り当て、および **GPU** の共有や分割の管理を行います。

Mirantis k0rdent AI のテンプレートモデルを活用した **GPU** リソースの宣言的な構成により、標準化され、再現性の高いデプロイを実現します。

サポート対象のGPUおよびアクセラレータソリューション:

GPU / アクセラレーター	オペレーター	対応環境	認証済みサポート
NVIDIA	NVIDIA GPU Operator	k8s, OS	対応済み
AMD	AMD ROCm Operator	k8s, OS	準備中
Intel	Intel Gaudi Base Operator	k8s, OS	準備中

コンテナランタイム

Mirantis AI Factory ソリューションは、**CRI**（**Container Runtime Interface**）に対応したすべての主要なコンテナランタイムをサポートしています。これには次が含まれます。

ランタイム	**k0s** サポート状況	認証済みサポート
Containerd	同梱済み	対応済み
CRIO	テスト済み・構成可能	対応済み
Docker	テスト済み・構成可能	対応済み
Mirantis Container Runtime (MCR)	テスト済み・構成可能	対応済み

インフラストラクチャ層

アーキテクチャにおいて、ハードウェア（コンピューティング、ストレージ、ネットワーク）および AI アクセラレーター（GPU、TPU、LPU など）のデプロイ、管理、および有効化を直接担当する部分を指します。

Mirantis k0rdent AI は、宣言的なテンプレートモデルに基づき、これら基盤要素の自動構成を提供します。

主な機能は次の通りです。

- **GPU**管理：**GPU**ベンダーの管理ツールと統合し、プロビジョニング、構成、運用管理をサポートします。これには、**GPU**のフラクショナル（分割利用）のサポートも含まれます。

- ストレージ管理：各種ストレージソリューションと連携し、ストレージボリュームのプロビジョニングを行います。また、ネットワークファブリックやプラットフォームコンポーネントとの整合性を確保します。
- ネットワークファブリック管理：**RDMA** や**Ethernet**を含む、基盤となるネットワークファブリックの自動構成および管理を行います。
- コンピューティング管理：コンピューティングインフラおよび**OS**の自動検出、プロビジョニング、そしてライフサイクルマネジメントを行います。

コンピューティング

コンピューティングインフラのプロビジョニングとライフサイクルマネジメントは、ベアメタルオペレーターによって制御されます（詳細はプラットフォーム層セクションの「ベアメタル」を参照）。コンピューティングのライフサイクルには、オペレーティングシステム（OS）のデプロイとライフサイクルマネジメントが含まれます。

GPU

k0rdent は、GPU 共有モデルの構成、および適切な GPU ドライバーとオペレーターのデプロイ機能を提供します。これらは、以下のテンプレートの組み合わせを通じて設定されます。

- プロバイダ テンプレート：ベンダー独自のコントローラーと連携するための、ベンダー固有オペレーターをデプロイ・管理します。
- クラスタ テンプレート：Kubernetes クラスタの構成を行い、クラスタに使用するマシンイメージを選択します。
- ホスト テンプレート：GPU 共有オプションをサポートするための、ホスト固有の構成を定義します。
- サービス テンプレート ：Kubernetes クラスタに対して、適切な GPU オペレーター ソリューションをデプロイします。

フラクショナル GPU のプロビジョニング戦略

GPU ベンダーは、さまざまなフラクショナル GPU 共有オプションを提供しています。各アプローチには一長一短があり、ベンダーごとに固有の仕様があります。各ベンダーのアプローチに関する詳細は、「付録」に記載されています。

GPU共有技術の比較

カテゴリ	**NVIDIA** のアプローチ	**AMD** のアプローチ	**Intel** のアプローチ	備考 **/** ギャップ
ハードウェア仮想化 (SR-IOVベース)	**vGPU** (SR-IOV 仮想機能)	**MxGPU** (SR-IOV 仮想機能)	**GVT-g** (SR-IOV 仮想機能、最新GPUではEOL)	全ベンダーがSR-IOVベースの仮想化をサポート。ただし、IntelのGVT-gはディスクリートGPU（Xeシリーズ）への注力により非推奨となっており、現在は積極的な開発が行われていない。
フル**GPU**パススルー (PCIe割り当て)	PCIe パススルー / 直接 PCIe 割り当て	PCIe パススルー / 直接 PCIe 割り当て	PCIe パススルー (VFIO経由のディスクリートXe GPU用)	全ベンダー共通の標準的な手法。ハイパーバイザーのサポートと、GPUへの排他的アクセスが必要。
ハードウェア パーティショニング (GPU内分割)	**MIG** (Multi-Instance GPU)	直接的な対抗機能なし	直接的な対抗機能は現時点でなし (Xeアーキテクチャのロードマップで計画中)	現時点でこの機能を提供しているのはNVIDIAのMIGのみ。AMDとIntelにはまだGPU内部の分割機能はない。
ソフトウェアベースのプロセス共有	MPS (Multi-Process Service)	ROCmベースのソフトウェア共有 / マルチプロセスランタイム	Level Zero ランタイムによるマルチプロセス対応 (初期段階)	IntelはLevel ZeroランタイムとoneAPIを介してプロセス共有をサポートしているが、エコシステムはまだ初期段階であり成熟度が低い。

Kubernetes デバイスプラグイン + GPU共有プロジェクト	デバイスプラグイン、NVIDIA オペレーター、サードパーティ製共有プラグイン	ROCmを用いたサードパーティ製デバイスプラグイン / K8sコミュニティプロジェクト	Kubernetes用Intelデバイスプラグイン + 実験的な共有プロジェクト	全ベンダーがデバイスプラグインをサポート。Intelは Level Zero APIに注力しており、特に推論ワークロード向けに進化中。
新興のマルチプロセス **DRM**サポート (グラフィックス重視)	N/A (クローズドスタックのため直接サポートなし)	LinuxにおけるDRMベースのGPU共有 (実験的、グラフィックス重視)	DRM共有をサポート (i915ドライバー、グラフィックス重視)	AMDとIntelは、グラフィックスGPU共有においてオープンソースのDRM/Mesaスタックとの親和性が高い。NVIDIAは（クローズドなドライバーモデルのため）この領域には関与していない。

ストレージ

通常、GPUクラスタでは2つのティア（階層）のネットワークストレージが利用されます。モデルや学習セットを保存するための超高性能・高帯域な **NVMe-oF** と、一般的なユーザーアクセスやアーカイブ用の中性能ストレージです。

ティア1：超高性能

大規模なモデルや学習データセットは、通常、超低遅延かつ極めて高いスループットを持つ **NVMe-oF** 上に保存されます。これは、推論や（再）学習時におけるデータのロード時間を短縮するためです。

ティア2：中～高性能

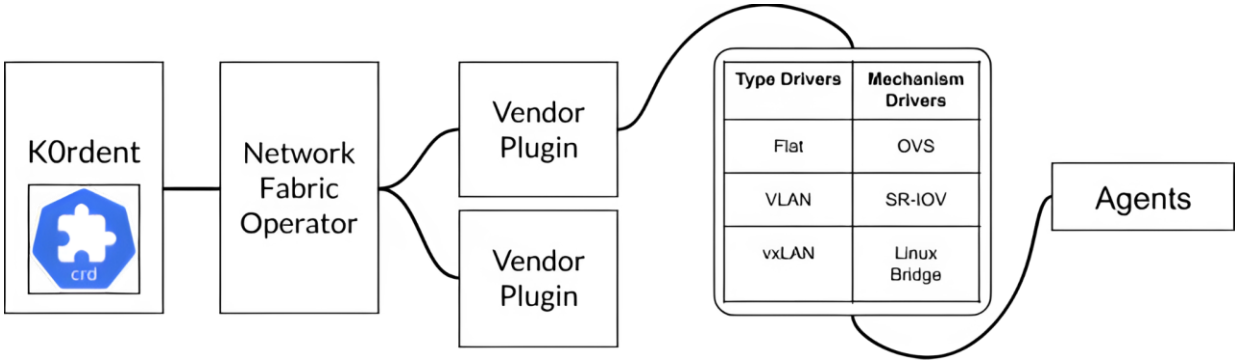
ユーザー用ストレージやアーカイブ、および同様のユースケース向けには、通常、**NFS**、オブジェクトストレージ、またはその他の中～高性能なネットワークストレージが提供されます。

ローカルストレージ

上記の2つのネットワークストレージに加え、多くのGPUサーバーは ローカルの**NVMe** を搭載しています。これは、GPUに迅速にロードする必要があるモデルやデータのキャッシュ、あるいは定期的なチェックポイントの保存において、サーバー単体での超高性能なパフォーマンスを実現するためです。

ネットワーク

k0rdent におけるネットワークの構成と管理は、専用のプラグインが可能なネットワークファブリック オペレーターを通じて行われます。これは、他の k0rdent コンポーネントと同様のテンプレート化されたアプローチをサポートしています。



ネットワーク オペレーターの概念図

ネットワークファブリック オペレーターは、次のネットワーク要素の構成をサポートします。

- **RDMA** ネットワーク
- **Ethernet** ネットワーク
- SmartNIC/DPU

ネットワークファブリック オペレーターの主な役割は、アンダーレイ（物理層）およびオーバーレイ（仮想層）の **L2**（データリンク層）と **L3**（ネットワーク層）の構成です。これは、主に次の役割を担う **DPU** オペレーターとは異なります。

- **DPU** のプロビジョニング
- エージェント・フレームワークのデプロイ
- **DPU** のリソースとしての登録
- DPU のモニタリング構成

注記: ネットワークファブリック オペレーターおよび DPU オペレーターは、現在活発に開発が進められている段階です。

ネットワークの概要

通常、Sovereign AI Factory には、特定の目的に応じた複数の異なるネットワークが存在します。

ネットワーク	目的	**L2**種別	備考
統合ネットワーク：アクセス＆ストレージ	インターネット接続、ワークロード間の通信、ストレージへのアクセス	Ethernet	主要な「ユーザー用」ネットワーク。ノンブロッキング（帯域が競合しない）設計を前提とする
RDMA ネットワーク	ワークロードの集約に向けた **GPU** 間の相互接続	Ethernet (RoCEv2) または Infiniband	GPU 用の主要なワークロード ネットワーク
コントロールプレーン ネットワーク	インバンド管理、および AI Factory 管理機能からの制御。	Ethernet	コントロールプレーンによる管理が可能
OOB (Out-of-Band) ネットワーク	ベアメタルのプロビジョニングやスイッチの管理アクセス用（低速）	Ethernet	

次にアクセスおよび RDMA 向けの高速ネットワークに関する推奨アプローチについて解説します。

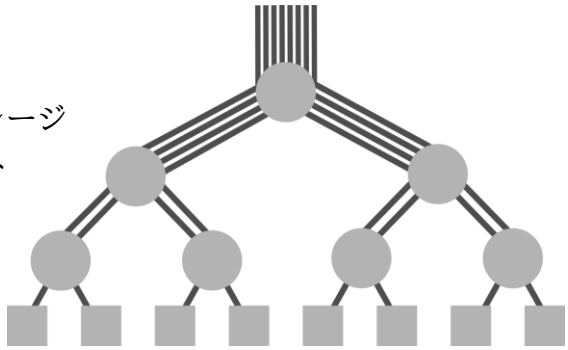
高速ネットワーク設計（ファットツリー・トポロジー）

GPU クラスタのネットワーク設計にはいくつかのアプローチがあり、コストとパフォーマンスの間には明確なトレードオフが存在します。しかし、一般的な出発点は、実績のある「ファットツリー」ネットワークトポロジー（2層または3層）を採用することです。

Mirantis では、後日大幅なアーキテクチャの再設計を強いることなく最大限の拡張性を確保できるよう、

可能であれば最初から**3**層構造で開始することを推奨しています。

3層ファットツリー トポロジーの主な利点と特徴は次の通りです。



ファットツリー ネットワークの主な特徴

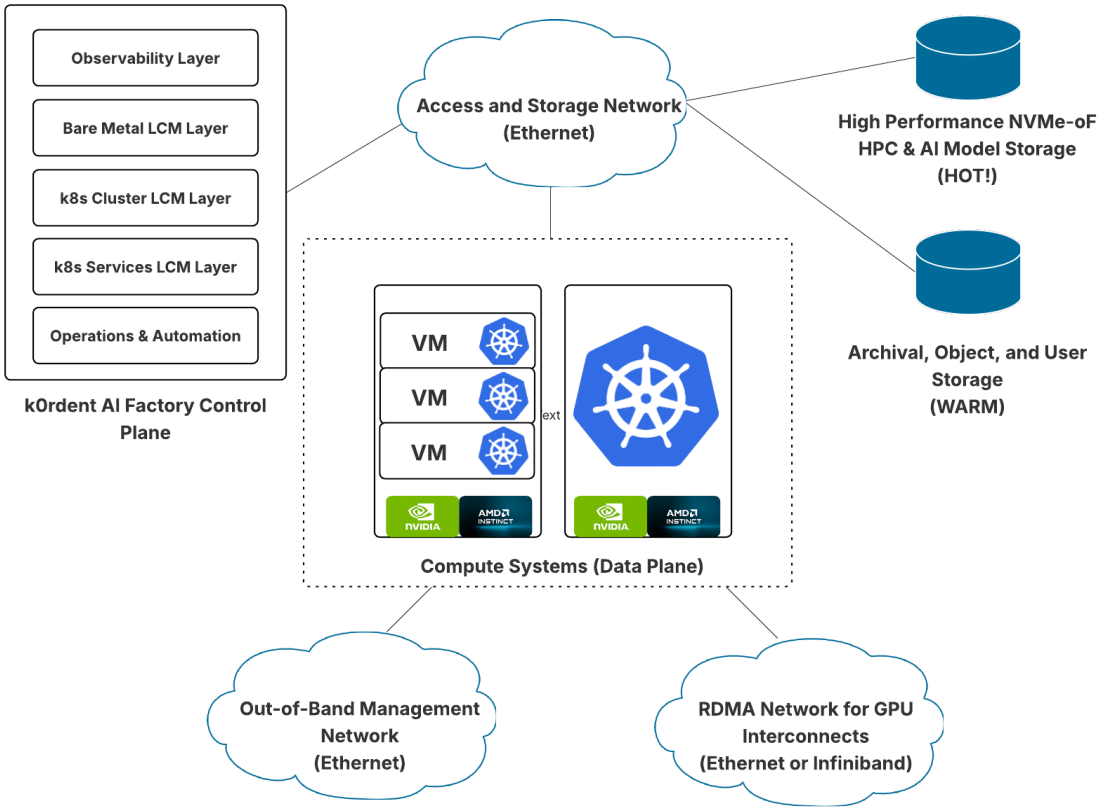
- 完全なノンブロッキング アーキテクチャ
  - システム間、GPU間、および高速ストレージ間において、最大限のイースト ウェスト（サーバー間）帯域幅を保証します。
- 拡張性
  - 階層構造により、ノードやスイッチを追加するだけで容易に拡張が可能です。クラスタの規模拡大に最適な構造です。
- 堅牢性
  - ノード間に複数の通信経路（マルチパス）が存在するため、ネットワークの一部で障害が発生しても通信が遮断されることはありません。
- シンプルなルーティング
  - すべてのトラフィックがコア層またはスパイン層を経由してルーティングされるため、ネットワーク構成が簡素化されます。
- 低コスト
  - ファットツリー ネットワークは、標準的な低コストのスイッチを用いて構築できるため、優れたコスト効率を実現します。

GPU クラスタにおけるメリット

- 高スループット
  - ネットワークのノンブロッキングな性質により、GPU 間の高帯域幅が確保されます。これは、並列処理や AI 学習において不可欠な要素です。
- 拡張性
  - パフォーマンスを低下させることなく、膨大な数の GPU を収容することが可能です。
- 管理の簡素化
  - 直感的な設計により、ネットワークの管理やメンテナンスが容易になります。

AI Factory 接続概念図

これまでの解説内容を以下の概念図に示します。各コンポーネントがどのように相互接続され、ネットワークのどこに配置されているかを確認できます。



GPUドライバーと管理

Mirantis AI Factory プラットフォームは、**NVIDIA**、**AMD**、**Intel** を含むすべての主要な GPU サプライヤーのドライバー、ランタイム、および管理機能をサポートしています。

各種ソフトウェアコンポーネントは、**k0rdent** が管理する **Kubernetes** コンテナを介して提供されます。各クラスタには、ベアメタルか仮想マシンかを問わず、適切なソフトウェア（ドライバー等）を導入するためのサービステンプレート（ServiceTemplates）が定義されています。

コンポーネント	NVIDIA GPUs (CUDA)	AMD GPUs (ROCm)	Intel Gaudi (Habana)
演算バックエンド	CUDA	ROCm/HIP	Intel Gaudi Software
ドライバー & ランタイム	NVIDIA GPU ドライバー, CUDA ランタイム	ROCm ランタイム / ドライバー	Habana ドライバー, Intel Gaudi ランタイム
Kubernetes サポート	NVIDIA GPU Operator	ROCm GPU Operator	Intel Gaudi Base Operator
CLI モニタリング	NVIDIA-smi	rocm-smi	hl-smi, hl_qual
サードパーティ統合用API	NVML	ROC-smi-lib	HLML
モニタリング	DCGM	ROCm Data Center Tools (RDC)	SynapseAI Mgmt
プロファイリング / 診断	Nsight Systems & Compute	ROCm Profiler & ROCgdb	Intel Gaudi Profiler
コンテナイメージ	NVIDIA NGC containers	ROCm Docker images	Habana Docker イメージ
統合モニタリング	k0rdent Observability & FinOps (KOF)	k0rdent Observability & FinOps (KOF)	k0rdent Observability & FinOps (KOF)
デプロイ例	DGX, AWS P4/P5, Azure ND-series	AMD MI250/MI300 (Frontier)	AWS EC2 DL1, Habana専用サーバー

デリバリーおよび運用モデル

RACI は、プロジェクトにおける役割と責任を明確にするためのフレームワークであり、AI Factory の構築のような複雑な取り組みにおいて特に有効です。RACI は、**Responsible**（実行責任者）、**Accountable**（説明責任者）、**Consulted**（協働・助言者）、Informed（報告先）の頭文字をとったものです。

- **Responsible**: タスクを実際に遂行する。
- **Accountable**: タスクが確実に完了することに責任を持つ。
- **Consulted**: 入力情報や専門知識を提供する。
- **Informed**: 進捗状況について常に最新の情報の共有を受ける。

RACI マトリクスを適用することで、所有権が明確になり、曖昧さが排除され、多角的な AI 開発チーム間でのコラボレーションが向上します。

RACI ペルソナ

Mirantis AI Factory のデプロイメントにおいて、主要なペルソナ（役割）は次のとおりです。

- ハードウェア ベンダー：ネットワーク、計算リソース、およびストレージハードウェアのサプライヤー
- クラウド プロバイダー ： Sovereign AI Factoryを所有する組織体
- **k0rdent**：Mirantis k0rdent AI が依存する、中核となるコントロールプレーン管理および自動化プラットフォーム
    - **k0rdent** サブコンポーネント： K8s クラスタのライフサイクルマネジメント（KCM）、K8s サービスのライフサイクルマネジメント（KSM）、オブザーバビリティ（KOF）、仮想化（k0rdent Virtualization）、k0rdent ベアメタル ライフサイクルマネジメント（k0rdent Metal）
- サービス **(Services)**：Mirantis が提供する付加価値の高い、手厚い専門支援サービス
- オペレーター **(Operator)**：顧客のSovereign AI Factory運用チーム

RACI アクティビティ

以下の表では、RACI アクティビティを主要なグループ（ハードウェア、コントロールプレーン、**Kubernetes** およびサービス管理、**SLA/SLO**、インシデント対応）に分けて記載しています。

R (Responsible): 実行責任者（タスクを行う） / **A** (Accountable): 説明責任者（最終承認・完了責任） **C** (Consulted): 協働・助言者 / **I** (Informed): 報告先

アクティビティ	ハードウェアベンダー	クラウドプロバイダー	**k0rdent (**プラットフォーム**)**	**Mirantis** サービス	オペレーター **(**運用チーム**)**
【ハードウェア】					
ハードウェア展開		I			R, A
キャパシティ管理		I			R, A
修理・交換	R	I			A
プロビジョニング		I	R	C	R, A
【コントロールプレーン】					
デプロイ		I		R	R, A
管理					R, A
アップグレード				C	R, A
モニタリング					R, A
【K8sクラスタ **&** サービス管理】					
定義			R	C	A
デプロイ			R		A
アップグレード			R		A
モニタリング			R		A
【SLA / SLO / SLI & 運用手順】					
インフラストラクチャ		I		C	R, A

k0rdent AI コントロールプレーンPlane		I		R	R, A
ハードウェア保守契約		R, A			I
【インシデント対応】					
ベンダーへのエスカレーション					R, A
顧客への通知・コミュニケーション		R, A			R
Tier1サポート					R, A
Tier 2サポート				R	R, A
Tier 3 Support				R	R, A

付録 – NVIDIA エコシステム

Mirantis k0rdent AI によるAI Factory の構築

15 May 2025

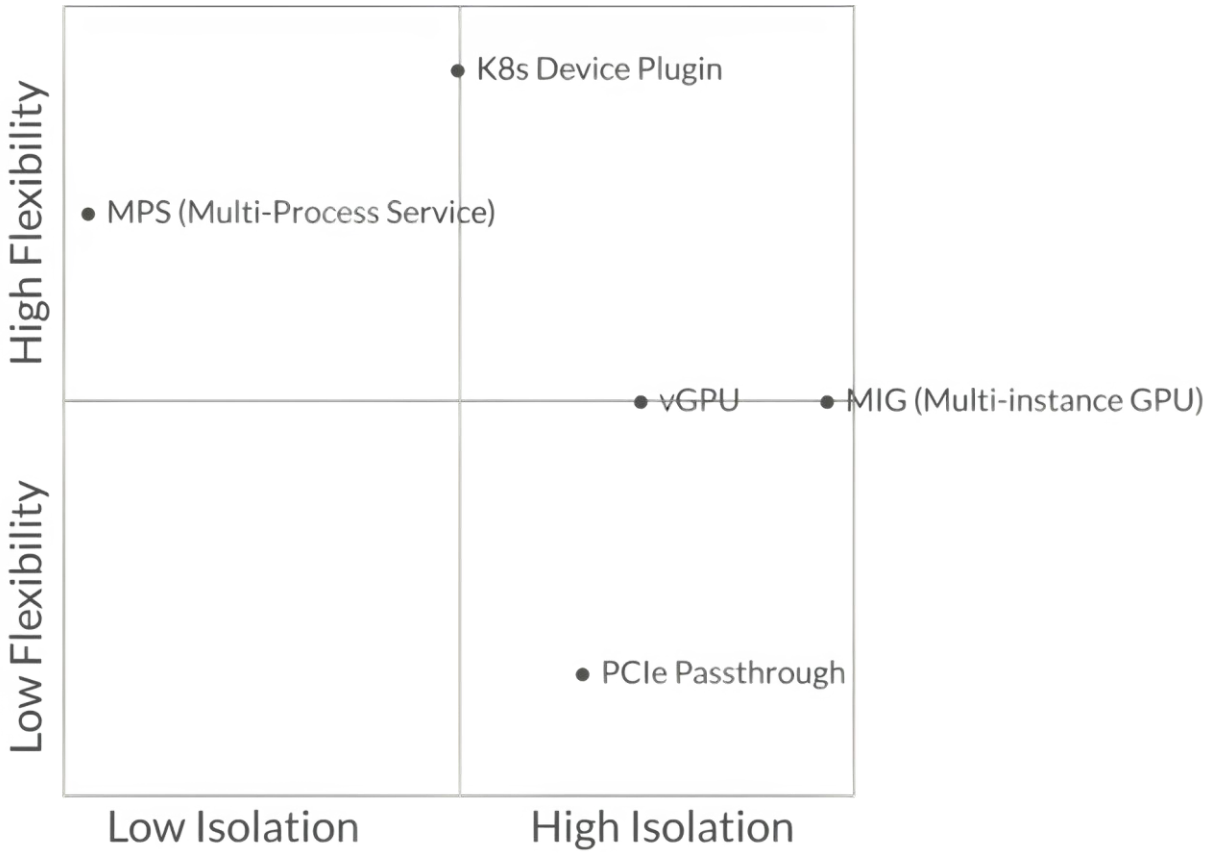
目次

NVIDIA フラクショナル GPU プロビジョニング戦略

NVIDIA

共有オプション	アプローチの概要	ワークロードの適合性	分離レベル
タイムスライス (MPS - Multi-Process Service)	NVIDIA MPS を利用し、演算リソースへのアクセスを時間分割することで、複数の CUDA アプリケーションで 1 つの GPU を共有します。	HPC ワークロード、小規模バッチの AI 推論、低遅延が求められるマルチテナント ワークロード。	低: プロセスが同じ **GPU** コンテキストを共有するため、プロセス間の分離は最小限です。
仮想 **GPU (vGPU)**	NVIDIA の仮想化ソフトウェアを使用して GPU を複数の仮想 GPU に分割し、それぞれに専用のメモリを割り当てます。	仮想デスクトップ (VDI)、AI/ML 学習・推論、および強力な分離が必要なグラフィックス集約型アプリ。	高: 各 **VM** やコンテナに専用の **vGPU** スライスが割り当てられ、ハイパーバイザー階層で強力に分離されます。
CUDA マルチインスタンス **GPU** (MIG)	A100 や H100 GPU を複数のインスタンスに分割。各インスタンスに専用の演算、メモリ、キャッシュ、帯域幅リソースを割り当てます。	AI 推論、HPC バッチジョブ、およびリソース予測可能性が求められるマルチテナント ML ワークロード。	高: ハードウェア強制による **MIG** インスタンス間の完全な分離。
デバイスプラグイン **+ Kubernetes** スケジューリング	デバイスプラグインやサードパーティ製オペレーター（NVIDIA Device Plugin 等）を使用し、GPU 全体または一部を Kubernetes でスケジュールします。	コンテナ環境での ML モデル学習や推論、共有クラスタ。	中: 共有メカニズムに依存。多くはソフトウェア制御であり、**vGPU** や **MIG** よりは弱くなります。
PCIe レベル・パススルー **(SR-IOV /** 直接 **PCIe** 割り当て**)**	物理 GPU 全体、または仮想機能（SR-IOV）を VM やコンテナに直接割り当てます。	高性能ワークロード、ML 学習、GPU 加速データベース、および低遅延が必要な用途。	極めて高: **VM** またはコンテナが **GPU/VF** を排他的に使用でき、ベアメタルに近い性能を発揮します。

共有オプション：分離性 vs. 柔軟性



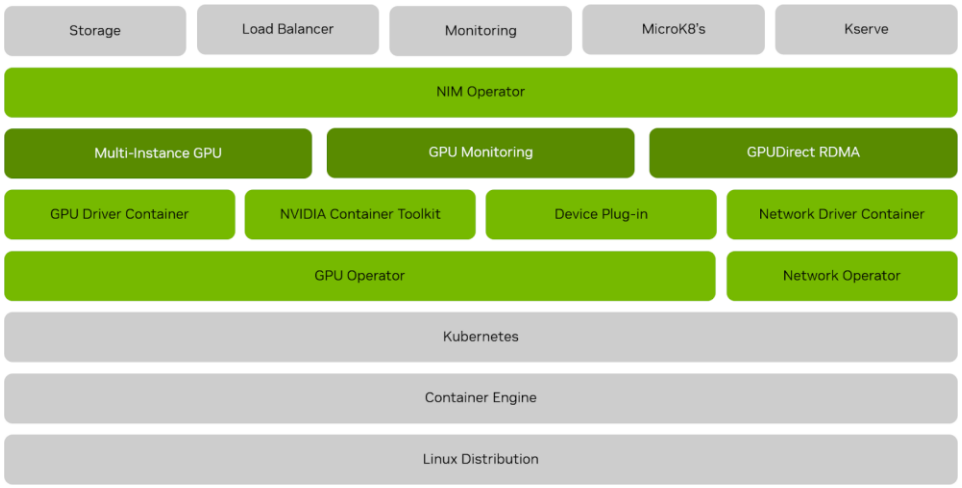
GPU共有オプションの長所と短所

共有オプション	長所	短所
MPS (Multi-Process Service)	● 複数プロセス実行時の高スループット ● 低遅延が求められる小規模ジョブに最適 ● サポート対象GPUでのセットアップが容易	● プロセス間の分離レベルが低い ● 干渉や「ノイジー ネイバー（うるさい隣人）」問題のリスクがある ● すべてのML/DLフレームワークに適しているわけではない
vGPU	● 強固なVMレベルの分離（VDI、グラフィックス、AI/MLに最適） ● 柔軟なユーザースペース・ツール群のサポート	● ライセンスが必要（NVIDIA vGPU ソフトウェア） ● 直接アクセスやMIGに比べるとパフォーマンスが劣る一部のML学習において効率が低下する場合がある
MIG (Multi-Instance GPU)	● 完全なハードウェアレベルの分離 ● インスタンスごとに予測可能なパフォーマンス ● 効率的なマルチテナント利用 ● 推論ワークロードに理想的	● A100、H100およびそれ以降のモデルのみ対応 ● アプリケーション側がMIGの制約を認識している必要がある ● 大規模バッチのML学習には不向き
K8s デバイスプラグイン + 共有プロジェクト	● コンテナ環境において非常に柔軟 ● 動的なGPU共有ポリシーをサポート ● Kubernetesエコシステムとの統合が容易	● Kubernetesエコシステムとの統合が容易 ● リソース競合のリスクがある ● 追加のチューニングが必要になる場合がある
PCIe パススルー (SR-IOV / 直接割り当て)	● 最大限の分離とベアメタルに近いパフォーマンス ● 高性能ワークロードに最適 ● 低遅延のニーズに対応	● 割り当てが固定（1つのVM/コンテナがGPU全体または仮想機能を専有） ● ワークロードがGPU集約型でない場合、リソース利用効率が悪くなる ● ハイパーバイザーとハードウェアのサポートが必須

NVIDIA Enterprise AI スタック

NVIDIA Enterprise AI Stack は、企業がデータセンター、エッジ、およびクラウドにわたって、AI ワークロードを効率的かつ安全に、そして大規模に展開できるように設計された、ハードウェア加速型のエンドツーエンド プラットフォームです。
これは、NVIDIA GPU、AI ソフトウェア、ツール、ライブラリ、およびフレームワークを組み合わせ、検証済みでサポート可能、かつパフォーマンスが最適化されたスタックとして統合したものです。

NVIDIA ソフトウェアスタック アーキテクチャ



https://docs.NVIDIA.com/ai-enterprise/reference-architecture/latest/software-stack.html

NVIDIA Enterprise AI Stack の主要レイヤー

1. ハードウェア基盤
   - **NVIDIA GPU**: A100、H100、L40S、L4、RTX、およびその他のモデル。
   - **DPU (BlueField)**: AIデータパイプラインのオフロード（負荷分散）とセキュリティ確保。
   - **Certified Systems**: OEMパートナー各社から提供される NVIDIA 認証サーバー。
   - **MIG (Multi-Instance GPU)**: GPUを分割し、セキュアなマルチテナントAIワークロードを実現。
   - **NVLink / NVSwitch**: GPU間を接続する超高速インターコネクト。

2. システムソフトウェア層

- NVIDIA AI Enterprise (AIE):
    - エンタープライズレベルのフレームワーク、ライブラリ、ツールを公式サポートとライセンス付きで提供するフラッグシップ スイート。
    - 主要コンポーネント
        - **NVIDIA AI Workbench**: AI開発環境の統合管理。
        - **NVIDIA AI Toolkit**: モデルの学習、最適化、推論用ツール。
        - **NVIDIA Triton Inference Server**: あらゆるモデルに対応する推論サーバー。
        - **NVIDIA TensorRT / TensorRT-LLM**: 推論高速化ランタイム（LLM対応版含む）。
        - **NVIDIA RAPIDS**: データサイエンスの高速化。
        - **NVIDIA NeMo**: 大規模言語モデル（LLM）開発用フレームワーク。
        - **NVIDIA Clara**: ヘルスケア向けAI。
        - **NVIDIA Morpheus**: サイバーセキュリティ向けAI。
        - **NVIDIA TAO Toolkit**: 転移学習（Transfer Learning）用ローコードツール。

    - 仮想化環境、ベアメタル、および Kubernetes 環境をサポート。

3. Layerインフラストラクチャ管理層

- **NVIDIA GPU Operator**: Kubernetes 上での GPU ドライバー、デバイスプラグイン、モニタリングのデプロイを自動化。
- NVIDIA AI Operator: Simplifies deployment and lifecycle management of AI frameworks on Kubernetes. **NVIDIA AI Operator**: Kubernetes 上での AI フレームワークのデプロイとライフサイクルマネジメントを簡素化。
- **NVIDIA AI Operator**: Kubernetes 上での AI フレームワークのデプロイとライフサイクルマネジメントを簡素化。

4. フレームワークおよびライブラリ

- フレームワーク: TensorFlow、PyTorch、ONNX Runtime、MXNet（TensorRT や cuDNN で最適化済み）。
- ライブラリ: cuBLAS、cuDNN、NCCL、DALI、RAPIDS など。
- 推論加速: TensorRT、Triton Inference Server。
- データ分析: RAPIDS（GPU 加速型データサイエンス）。

5. AI and ML Tools AI および ML ツール

- **NVIDIA NeMo Framework**: LLM の事前学習、ファインチューニング、デプロイ。
- **NVIDIA TAO Toolkit**: 特定ドメイン向け AI を構築するローコードツールキット。
- **NVIDIA Triton Inference Server**: 複数のフレームワークや精度（精度混合）に対応したスケーラブルな推論提供。
- **NVIDIA DeepStream**: ビデオ分析およびマルチセンサー フュージョン。

6. デプロイおよびオーケストレーション
   - **VMware vSphere**: vGPU サポートによる仮想化。
   - **Kubernetes** 統合: GPU および AI オペレーターを介した連携。
   - **NGC Helm** チャート: Kubernetes 対応の即時デプロイ。
   - 多様な環境への対応: ハイブリッド、マルチクラウド、およびエッジシナリオのサポート。

主な機能と長所

機能	長所
エンタープライズ　サポート (AIEライセンス経由)	SLA（サービス品質保証）に基づいたサポート、セキュリティ、および安定性の提供。
検証・認証済みインフラストラクチャ	検証済みのサーバーおよびソフトウェアスタックにより、デプロイのリスクを軽減。
フルスタック アクセラレーション	データの取り込み（インジェクション）からAI推論まで、すべての工程をGPUで最適化。
柔軟なデプロイ環境	仮想マシン（VM）、コンテナ、ベアメタルの各環境をサポート。
AI モデル ズー (NGC)	最適化済みで即座にデプロイ可能なAIモデルやワークフローへのアクセス。
統合されたデータサイエンス & AI	データの準備、学習、推論のための統合されたスタック。

NVIDIA ネットワーキング オプション

NVIDIAは、超高速なGPU間相互接続、高スループットなストレージの読み書き、およびワークロード間通信における膨大なイースト・ウェスト（サーバー間）トラフィックに対応するため、幅広いネットワークオプションをサポートしています。

これらには、RDMAトラフィック向けの **InfiniBand** および **Ethernet**、通常のコンピューティングやストレージ向けの高スループット**Ethernet**、高度な機能を備えた **SmartNIC** や **SuperNIC**、さらにスパコンやAI特有のネットワークパターンを加速させるベンダー固有の機能が含まれます。

RDMA ネットワーキング

単一のGPUサーバー内では、主に **NVLink** が通信を担います。単一ラックを超えて拡張する場合、

NVIDIAの **NVLink Switch** を使用することで、最大72基のGPUを驚異的な速度で相互接続できます。

単一の NVLink Switch の範囲を超える場合は、GPU間の高速接続のために **InfiniBand** または **Ethernet** 上での **RDMA** を使用する必要があります。InfiniBand と Ethernet は全く異なるレイヤー2（L2）プロトコルであり、それぞれ異なるスイッチファブリックとソフトウェアスタックを必要とします。多くの場合、異なるネットワークカード（NIC）も必要になりますが、**NVIDIA BlueField-3 DPU** は必要に応じてどちらのL2プロトコルもサポートするように構成可能です。NVIDIAは、InfiniBand によるネイティブなRDMA、またはコンバージドイーサネット上の **RoCEv2** によるRDMAの両方をサポートしています。

NVIDIA InfiniBand スイッチ ファブリック

NVIDIAの InfiniBand スイッチは、GPUクラスタ、HPC（ハイパフォーマンス・コンピューティング）、およびAIワークロードに最適化された、超低遅延かつ極めて高いスループットを提供します。

Quantum-2 (NDR) InfiniBand スイッチ

- 帯域幅: 400 Gb/s (NDR)、近日中に 800 Gb/s (XDR) に対応。
- ポート数: 1スイッチあたり最大64ポート、拡張可能な構成。
- 遅延 **(Latency)**: 極めて低い（ポート間レイテンシ 約90ns）。
- 主な機能
    - **SHARPv3**: ネットワーク内計算（In-network computing）による高速化。
    - 高度な輻輳制御とアダプティブルーティング。
    - マルチテナント集約ツリーのサポート。

- ユースケース:
    - AI学習（DGX SuperPODなど）
    - HPC スーパーコンピューティング
    - GPU集約型の分散ワークロード

NVIDIA Ethernet スイッチ ファブリック

Spectrum スイッチは、汎用データセンター、AI、およびストレージに使用される高性能な Ethernet スイッチです。

NVIDIA Spectrum-4 Ethernet スイッチ

- 帯域幅: 最大 800 Gb/s の Ethernet ポート。
- 遅延 **(Latency)**: 非常に低い（約 300-400ns）。

- ポート密度: 高密度構成（1U サイズで最大 128 ポート）。
- 主な機能:
    - プログラマブル・パイプライン: P4 言語によるプログラミングが可能。
    - 高度なテレメトリ: 「What Just Happened®（WJH）」技術によるリアルタイムな障害・パケットロス解析。
    - RoCEv2 サポート: Ethernet 上での RDMA ワークロードに対応。

- ユースケース:
    - Ethernet を介した AI ワークロード（Meta 社の AI 研究など）。
    - データセンター・ネットワーキング、クラウドスケールの展開。
    - Ethernet ベースのストレージソリューション (NVMe-oF)。

NVIDIA SmartNICs and DPUs (SuperNICs)

NVIDIA は、**SmartNIC (ConnectX-7)** と **DPU (BlueField-3)**（別名「SuperNIC」）という2つの主要な NIC ソリューションを提供しています。主な違いはコストと機能にあり、BlueField-3 は非常に高価ですが、極めて高度な機能を備えています。BlueField-3 は、**Ethernet** ネットワークを用いながら **InfiniBand** に匹敵するパフォーマンスを実現することを目的としています。

サマリー:

- ConnectX-7 および BlueField-3 アダプターは、Ethernet と InfiniBand の両方のプロトコルをサポートしています。
- プロトコルの選択には手動設定が必要です。自動検出や自動切り替え機能はありません。
- 動作モードの決定には、トランシーバーやケーブルなどのハードウェア選択、およびファームウェアとソフトウェアの設定が重要な役割を果たします。

最適なパフォーマンスと互換性を得るためには、展開時に使用するプロトコルを計画・構成することが不可欠です。

BlueField-3 の動作モード

BlueField-3 DPU には複数の動作モードがあり、それぞれプロトコルの選択に影響を与えます。

- **NIC** モード: ARM コアを無効にし、標準的な NIC として機能します。
- **DPU** モード: ARM コアがアクティブになり、ネットワーク、ストレージ、セキュリティタスクのオフロードと加速が可能になります。
- ゼロトラスト・モード: DPU をサーバーから隔離し、侵害のリスクを低減します。

NIC モード

ホストシステム（サーバー）側から見て、従来のネットワークアダプターと同様に機能します。

- 機能
  - 内蔵の ARM コアは動作せず、ネットワークトラフィックの管理やオフロードは行いません。ホストが NIC リソースを直接制御します。消費電力が低く、システム構成がシンプルになります。

- ユースケース:
  - 高度な DPU 機能が不要なシナリオや、オフロード機能よりもシンプルさと低消費電力を優先する場合に適しています。

DPU モード (組み込みCPU機能の所有権 - ECPF)

BlueField-3 DPU のデフォルトモードです。内蔵の ARM コアが NIC リソースとデータパスを制御します。

- 機能
  - ホストとネットワーク間の全トラフィックは、ARM コアが管理する仮想スイッチを通過します。DPU は信頼されたエンティティとして動作し、ドライバーのロード、インターフェースのリセット、更新などを処理します。
- ユースケース
  - 高度なセキュリティ、隔離、オフロード機能が必要な環境。ゼロトラスト・アーキテクチャの実装やインフラ・ワークロードの高速化に最適です。

ゼロトラスト モード

DPU モードの拡張版で、ホストシステムによる DPU へのアクセスを制限するより厳格なモードです。

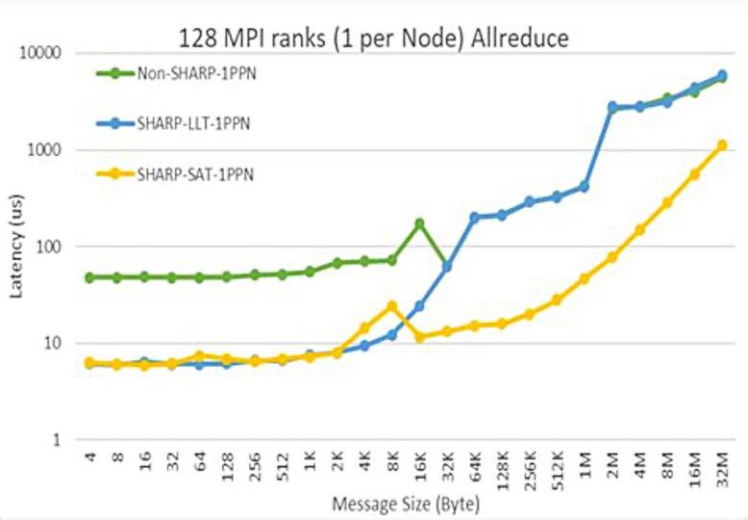
- 機能
  - ホストによるポート所有権の割り当て、ハードウェアカウンターへのアクセス、ファームウェアの書き換えなどを制限します。管理は ARM コアまたは BMC（ベースボード管理コントローラー）経由でのみ行われ、ホスト経由の侵害を防ぎます。

- ユースケース
  - マルチテナントのデータセンターや、極めて厳格なセキュリティと隔離が求められる環境。

各モードの構成方法に関する詳細な手順や、それぞれの設定がシステムに与える影響については、NVIDIAの公式ドキュメントを参照してください： **Modes of Operation - NVIDIA Docs**

NVIDIA ネットワーク拡張機能：SHARPv3

SHARPv3（Scalable Hierarchical Aggregation and Reduction Protocol version 3）は、NVIDIAが開発した独自のテクノロジーです。この技術は、集団通信操作（Collective Communicationの負荷をCPUやGPUからネットワークインフラ自体にオフロードすることで、ハイパフォーマンス・コンピューティング（HPC）やAIワークロードのパフォーマンスを劇的に向上させるよう設計されています。



AIおよびHPCワークロードにおけるメリット

SHARPv3の主な利点は以下の通りです。

- 遅延（レイテンシ）の低減
    - ネットワーク内でデータ集約（Aggregation）を行うことで、集団通信操作に伴う遅延を最小限に抑えます。
- スループットの向上
    - 通信タスクをネットワークにオフロードすることで、より高いスループットを実現し、データ処理やモデル学習を高速化します。
- リソースの最適化
    - CPUとGPUを通信タスクから解放し、本来の計算処理に専念させることで、システム全体の効率を向上させます。
- 拡張性
    - 複数の集約ツリーやマルチテナント運用へのサポートが強化されており、計算需要の増大に合わせて効果的にスケールさせることが可能です。

SHARPv3 の独自性

SHARPv3はNVIDIAのネットワーキング・エコシステムに密接に統合されており、オープン規格ではありません。その実装には、以下の特定のNVIDIA製ハードウェアおよびソフトウェアコンポーネントが必要です。

- **NVIDIA Quantum-2 InfiniBand** スイッチ: ネットワーク内でのリダクション（演算集約）およびアグリゲーションを実行するために必要なハードウェアを備えています。
- **NVIDIA Quantum-2 InfiniBand** スイッチ: ネットワーク内でのリダクション（演算集約）およびアグリゲーションを実行するために必要なハードウェアを備えています。
- **NVIDIA BlueField-3** データ処理ユニット **(DPU)**: ネットワーク内にプログラマブルな計算

リソースを提供することで、SHARPv3の機能をさらに強化します。
- **NVIDIA HPC-X** および **MLNX_OFED** ソフトウェアスタック: SHARPv3の機能を効果的に利用するためのドライバーとライブラリを提供します。

NVIDIA ソフトウェア エコシステムとの統合

SHARPv3は、NVIDIAの以下のソフトウェアツールとシームレスに連携するように設計されています。

- **NVIDIA NCCL (Collective Communication Library)**: AI学習で一般的に使用される集団通信操作を加速させるため、NCCLと統合されています。
- Open MPI: MPIベースのアプリケーションのパフォーマンスを向上させるために活用できます。

付録 - AMDエコシステム

Mirantis k0rdent AI による AI Factory の構築

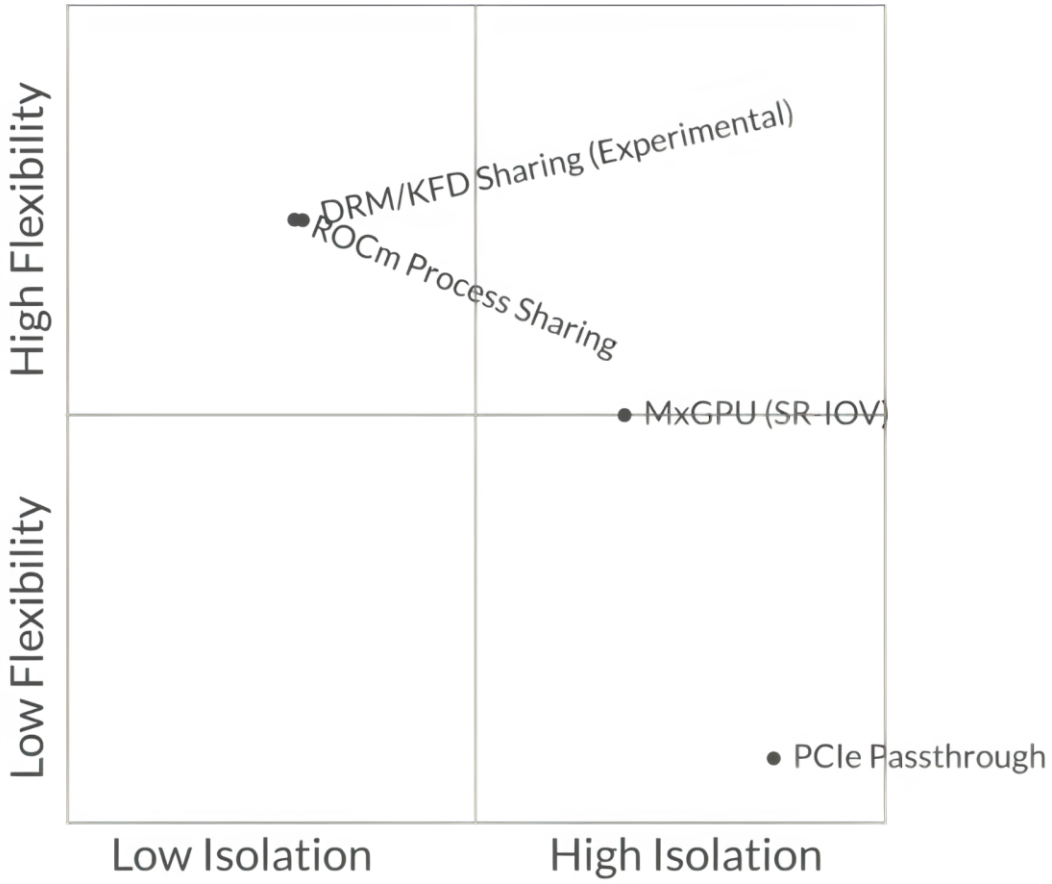
15 May 2025

目次

AMD 分散型GPU プロビジョニング戦略

共有オプション	アプローチの概要	ワークロードの適合性	分離レベル
MxGPU (Multiuser GPU - SR-IOV)	AMD独自のMxGPU技術。SR-IOVを利用して物理GPUから仮想機能（VF）を作成し、各VMに割り当てます。	仮想デスクトップ（VDI）、クラウドゲーミング、軽〜中規模の計算、リモート可視化。	高: PCIeレベルでのSR-IOVによる、ハードウェア強制の分離。
ROCm経由のパティショニング (ソフトウェアベースの共有)	AMD ROCmを使用したソフトウェア階層での共有。複数のプロセスやコンテナが同じGPUコンテキストを共有します。	HPCワークロード、AI推論、リソース共有が許容されるML学習。	低〜中: ソフトウェアによる分離。ハードウェア保証はなく、ランタイム等に依存。
パススルー (PCIe 直接割り当て)	ハイパーバイザーのPCIe割り当て機能により、GPU全体をVMまたはコンテナに直接パススルーします。	フルアクセスが必要な高性能ワークロード、ML学習、GPU加速DB、低遅延アプリ。	極めて高: 完全な分離。ベアメタルに近い性能でGPUを独占利用可能。
Linuxカーネル **DRM**によるマルチプロセス機能 (実験的)	Linux DRMおよびMesaスタックにおける新しい試み。基本的な分離を維持しつつ複数プロセスでGPUを共有します。	実験的なセットアップ、マルチテナント環境でのグラフィックス・ワークロード。	低〜中: 発展途上の技術であり、サポートは限定的。ベストエフォートな分離。

共有オプション：分離性 vs 柔軟性

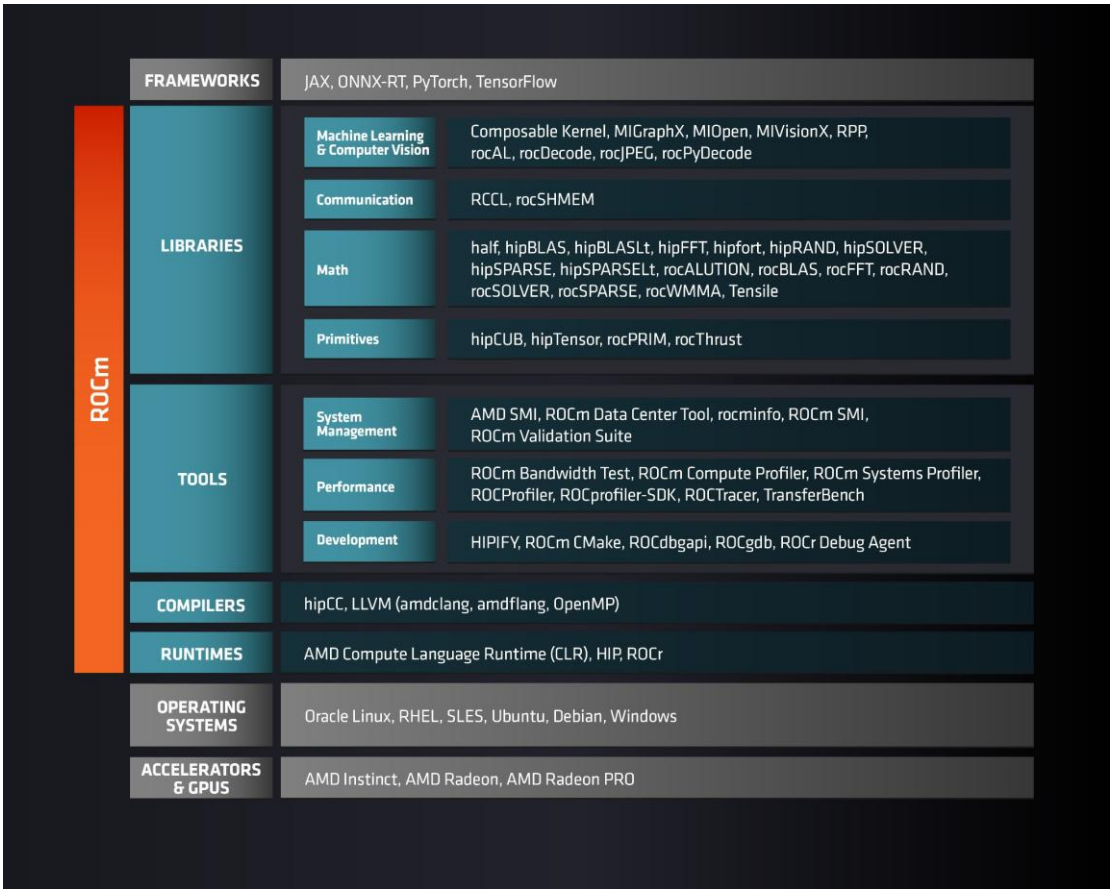


AMD GPU共有オプションの長所と短所

共有オプション	長所	短所
MxGPU (SR-IOV)	• ハードウェア強制の SR-IOV 分離 • VM ごとに専用の仮想 GPU 機能 (VF) を提供 • VDI、クラウドゲーム、リモートデスクトップ、軽計算に良好な性能 • 特定の Radeon Pro および MI シリーズ GPU でサポート	• 特定の AMD GPU でのみ利用可能 • 計算集約型や AI/ML ワークロードへのサポートが少ない • ライセンスが必要（Pro ドライバーが必要） • NVIDIA vGP

PCIe パススルー	• VM またはコンテナへの GPU の完全かつ直接的な割り当て • 最高の分離レベルとネイティブに近いパフォーマンス • AI/ML 学習、HPC、低遅延計算、レンダリングに最適	• GPU 共有不可（1つのワークロードに固定割り当て） • 適切なハードウェア /IOMMU サポートと設定が必要 • ワークロードがリソースを使い切らない場合、低利用率のリスクがある
ROCm ソフトウェア共有	• ROCm ランタイムによる柔軟なプロセスレベルの共有 • 複数プロセスでの GPU 共有が可能 • 厳格な分離より柔軟性が重要な HPC や AI/ML 推論に有効 • オープンソースの ROCm エコシステムと統合	• リソース競合や干渉の可能性がある • NVIDIA MPS と比較して、運用管理・監視ツールが発展途上 • ML/DL ワークロードにおいて NVIDIA ほど成熟していない
DRM/KFD ベース共有 **(**実験的**)**	• 基本的なプロセス共有のためのオープンソースドライバサポート • 複数プロセスの同時 GPU アクセスが可能 • Linux カーネルの DRM/KFD サブシステムを活用 • グラフィックスや実験的な計算リソース共有に適している	• 初期段階であり、計算用途の商用環境（プロダクション）レベルではない • ツールやエコシステムのサポートが限定的 • ベストエフォートな分離であり、ミッションクリティカルな AI/ML には不向き • 競合が発生するとパフォーマンスが低下する可能性がある

AMD ROCm ソフトウェアスタック アーキテクチャ



https://rocm.docs.amd.com/en/latest/what-is-rocm.html

AMD ネットワーキング オプション

AMD Pensando ネットワーキングは、他ベンダーと同様の高度なネットワーキング機能を提供しますが、特に **Ethernet**（**RDMA** 向けの **RoCEv2**） の活用と、主要なスイッチベンダーの既存 Ethernet スイッチファブリックとの統合に注力しています。Pensando SmartNIC は、他の AI ベンダーが提供する SmartNIC オプションとほぼ同等の機能を備えています。

RDMA ネットワーキング

単一の GPU サーバー内では、主に **AMD xGMI**がネットワーキングを担います。複数サーバー構成で実行する場合は、GPU 間の高速相互接続のために **Ethernet** 上の **RDMA**（**RoCEv2**） を使用する必要があります。

AMD スイッチファブリック オプション

AMD は、**RoCEv2**（**RDMA over Converged Ethernet**） をサポートする、あらゆる標準的な Ethernet スイッチファブリックを使用できます。AMD の Web サイトでは、Cisco および HP のオプションが推奨されています。

AMD DPUs (Data Processing Unit)

AMD の **Pensando DPU** は、データセンターにおけるネットワーク、ストレージ、およびセキュリティ機能をオフロードし強化するために設計された、高度でプログラム可能なネットワークアクセラレーターです。これらの DPU は **AMD EPYC CPU** とシームレスに統合され、クラウド、エンタープライズ、および AI ワークロード向けに最適化されています。

AMD Pensando DPUs の主な特徴

- プログラム可能なパケット処理: 最大 144 個のマッチ処理ユニット（MPU）を備えた **P4** プログラマブル パイプラインを活用し、カスタマイズ可能で効率的なパケット処理を実現します。
- **ARM** ベースのコントロールプレーン: 16 コアの ARM Cortex-A72 または Neoverse N1 プロセッサを搭載し、ホスト CPU からタスクをオフロードしてコントロールプレーンの操作を管理します。
- 高速ネットワーキング: デュアル 200 GbE または 400 GbE ポートを含む複数の構成をサポートし、高スループットのデータ転送を促進します。
- 高度なオフロード機能: 暗号化、圧縮、およびストレージ機能のハードウェアアクセラレーションを提供し、システム全体のパフォーマンスを向上させます。
- ソフトウェア エコシステム: **SSDK (Software-in-Silicon Development Kit)** を活用して、特定のワークロードに合わせたカスタム P4 プログラムの開発と展開が可能です。

最新の AMD DPU オプション

AMD の DPU における最新の進化には、AI、ハイパフォーマンス コンピューティング（HPC）、およびクラウドスケールのネットワーキングの厳しい要求を満たすように設計された **Pensando Salina 400** と **Pensando Pollara 400** があります。

AMD Pensando Salina 400 DPU

Salina 400 は、ハイパースケールおよび AI 駆動型環境におけるフロントエンド　ネットワークサービスに最適化された、AMD の第 3 世代 DPU です。

主な仕様

- 400 Gbps via dual 400GbE ports. ネットワークスループット: デュアル 400GbE ポート経由で最大 400 Gbps。
- プロセッサ: 16 コアの Arm Neoverse N1。
- メモリ: 最大 128 GB の DDR5 メモリをサポート。
- プログラマビリティ: カスタマイズ可能なパケット処理のための 232 個の P4 MPU を搭載。
- インターフェース: PCIe Gen5 x16.

主な機能

- 多機能なオフロード処理
    - ソフトウェア定義ネットワーク (SDN)、ファイアウォール、暗号化、ロードバランシング、ネットワークアドレス変換 (NAT)、およびストレージ・オフロードを処理します。
- 圧倒的なスケーラビリティ
    - 前世代の DPU と比較して、最大 2 倍のパフォーマンスと拡張性を実現しています。
- 完全な P4 プログラマビリティ
    - 完全に P4 言語でプログラミング可能であり、特定のニーズに合わせたネットワーク処理のカスタマイズが可能です。
- 最適化された設計
    - 低レイテンシ（遅延）、低ジッター、および低消費電力を実現するために最適化されています。

Salina 400 は、クラウドネットワーキング、高度なセキュリティ、およびストレージアクセラレーションを強化するために設計されており、現代のデータセンターにおける多目的なソリューションとなっています。

AMD Pensando Pollara 400 AI NIC

Pollara 400は、AIおよびHPC（ハイパフォーマンス コンピューティング）クラスタにおけるバックエンドの**GPU**間通信（**GPU-to-GPU**）に特化して設計された、400 Gbps対応のAIネットワークインターフェースカード（NIC）です。

主な仕様

- 最大帯域幅: 400 Gbps
- フォームファクタ: ハーフハイト ハーフレングス (HHHL)
- ホストインターフェース: PCIe Gen5 x16
- **Ethernet**インターフェース: QSFP112ポート（25/50/100/200/400 Gbpsの各構成をサポート）
- 管理機能: SMBus経由のMCTPをサポート

主な機能

- プログラム可能な RDMA トランスポート
    - 完全にプログラム可能な RDMA トランスポート機能を備え、ハードウェアベースの輻輳（ふくそう）制御を搭載しています。
- 高度なパケット制御技術
    - インテリジェント パケットスプレー（パケットの分散送信）、順序どおりの配信、選択的再送、および経路認識型の輻輳回避機能を備えています。
- UEC (Ultra Ethernet Consortium) 対応
    - UEC 準拠であり、次世代の AI ネットワーキング標準との互換性を保証します。
- AI ワークロードへの最適化
    - AI 学習および推論タスクに最適化されており、従来の RoCEv2 と比較して、メッセージ完了時間を最大 **6** 倍高速化します。

ユースケース

一般的に**Salina 400 DPU** はフロントエンド ネットワークサービス（外部通信や管理業務）に最適であり、一方の **Pollara 400 DPU** は GPU 間の相互接続や RDMA（高速な計算機間通信）向けに設計されています。

付録 - Intel エコシステム

Mirantis k0rdent AIによるAI Factory の構築

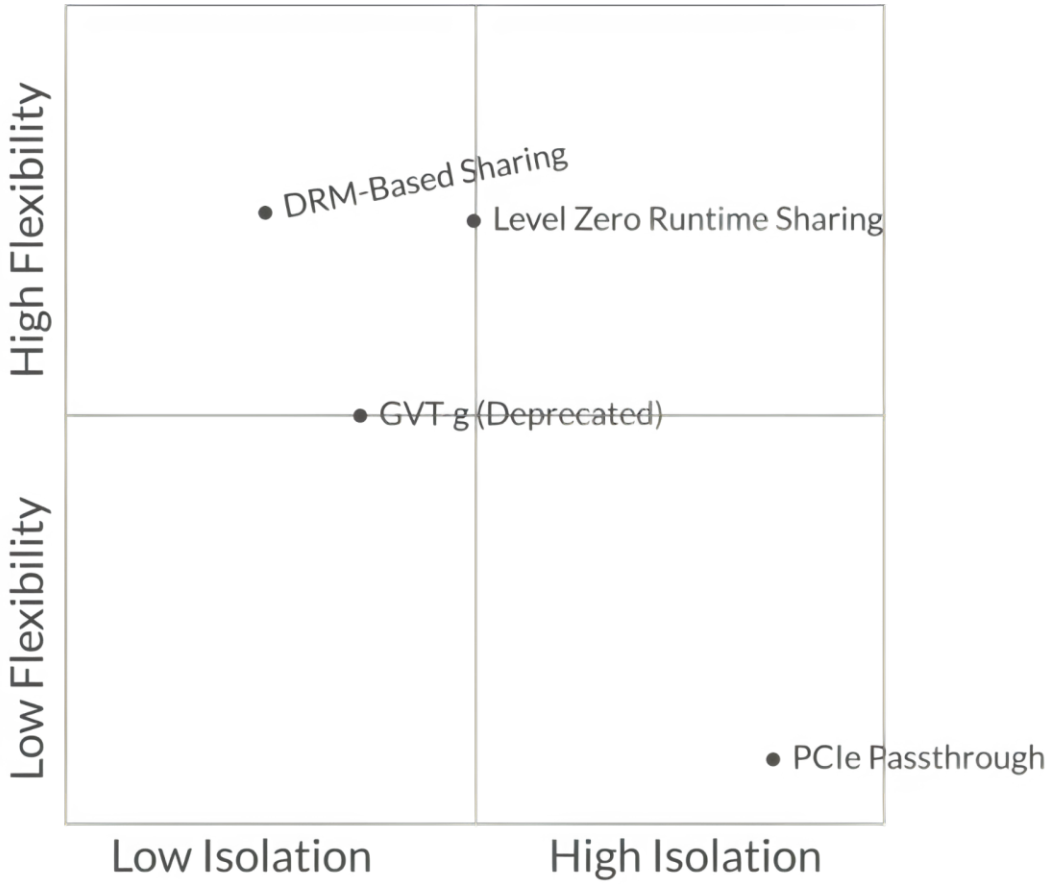
15 May 2025

目次

Intel 分散GPU プロビジョニング戦略

共有オプション	アプローチの概要	ワークロードの適合性	分離レベル
GVT-g (仮想化技術 **- SR-IOV** 風の共有**)**	仮想 GPU 機能を作成することで、統合 GPU を複数の VM で共有可能にするレガシー技術。	VDI、軽負荷のグラフィックス、リモートデスクトップ（※新世代 GPU では非推奨、Xe シリーズは非対応）。	中**: VM** 間のソフトウェアレベルの分離。ディスクリート **GPU** の **SR-IOV** よりは弱い。
PCIe パススルー **(VFIO** 直接割り当て**)**	VFIO を使用し、ディスクリート（単体）GPU 全体を VM またはコンテナに直接パススルー。	高性能グラフィックス、AI 推論、GPU への直接アクセスが必要な ML ワークロード。	極めて高**: VM/**コンテナが **GPU** をフル制御。ベアメタルに近い性能を発揮。
Level Zero ランタイムによるマルチプロセス共有 **(**初期段階**)**	Intel の **Level Zero API** と **oneAPI** ランタイムを使用。複数のアプリで GPU コンテキストを共有。	AI 推論、HPC、分析など、厳格な分離よりも共有効率が優先されるワークロード。	低～中**:** ソフトウェアレベルの共有。現時点ではハードウェア強制の保証はなし。
Intel GPU 用 **Kubernetes** デバイスプラグイン	Kubernetesプラグインにより、コンテナ間でIntel GPU（統合型および独立型）のスケジューリングが可能。多様なワークロードに対応。	コンテナ化された環境での AI 推論、分析、ビデオ処理。	中**:** ソフトウェアレベルの共有。分離の強さはオーケストレーターの設定に依存。
DRM ベースのマルチプロセス共有 **(i915** ドライバ**)**	Linux の i915 ドライバにより、DRM/Mesa スタックを介して複数のプロセスで統合 GPU を共有。	Linux 環境でのグラフィックス重視の処理（画面合成、デスクトップ共有、レンダリングなど）。	低～中**:** 厳格なハードウェア分離はなし。**DRM** スタックおよびユーザースペースドライバに依存。

共有オプション - 隔離 vs 柔軟性

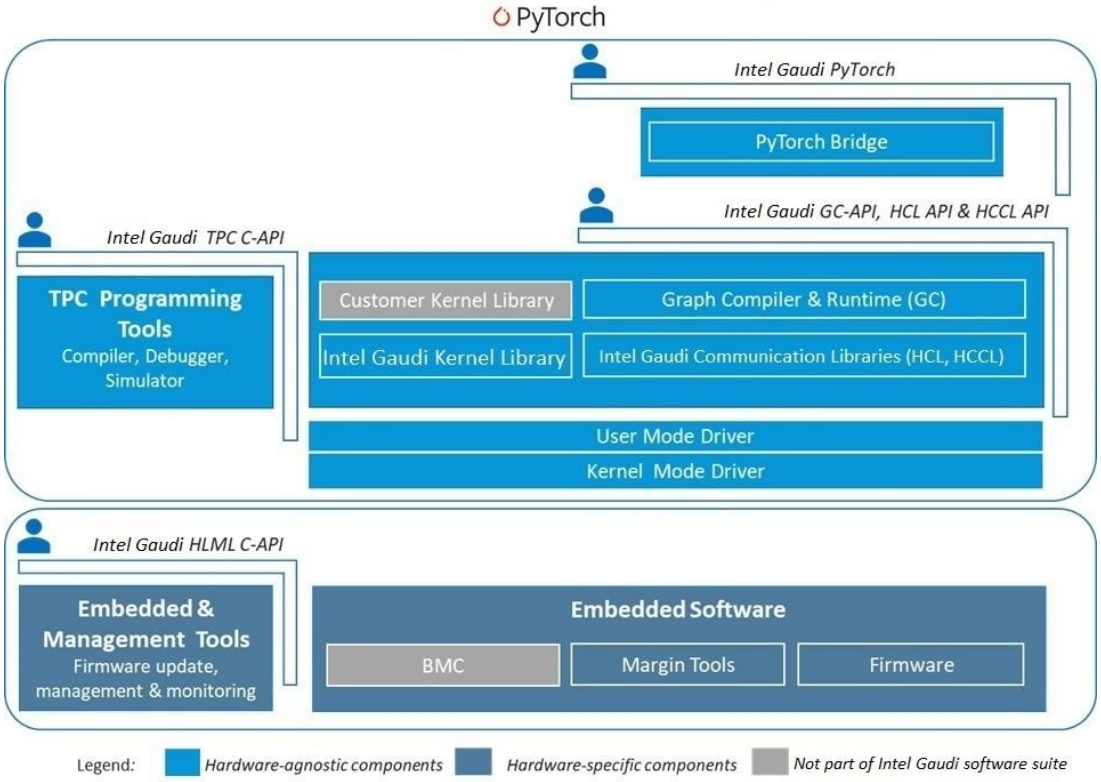


共有オプションの長所と短所

共有オプション	長所	短所
GVT-g (SR-IOV風の共有)	• 複数のVMで統合GPUを共有可能 • VDIや軽負荷のグラフィックスに最適 • データセンターやクライアントPCの既存Intel iGPUで動作	• 旧式（Deprecated）であり、Xe以降の単体GPUでは非サポート • ソフトウェアによる分離 • 計算集約型やAI/MLワークロードには不向き • 拡張性が限定的

PCIe パススルー (VFIO)	● 完全な分離とネイティブに近いGPUパフォーマンス ● 統合GPUと単体GPUの両方をサポート ● AI/ML、HPC、グラフィックス集約型ワークロードで良好に動作	● 共有不可。1つのGPUを1つのVM/コンテナが独占 ● リソース割り当てが固定され、柔軟性が低い ● ハイパーバイザーとIOMMUの設定が必要
Level Zero マルチプロセス共有	● Level Zero APIを介して複数プロセスでGPUを共有可能 ● Intel oneAPIエコシステムと統合 ● 共有が許容される推論やHPCワークロードに適している	● ソフトウェアレベルの分離のみ ● 初期段階であり、成熟度と広範なエコシステムサポートに欠ける ● プロセス間でのリソース競合の可能性がある ● NVIDIA MPSと比較してツールやオブザーバビリティが限定的
K8s デバイスプラグイン	● KubernetesネイティブなIntel GPUへのアクセスを提供 ● 統合GPUと単体GPUの両方をサポート ● コンテナでの推論、分析、ビデオ処理ワークロードに最適 ● 柔軟なリソースオーケストレーション	● 分離の強さはコンテナ境界とK8sの設定に依存 ● ハードウェア強制のパーティショニングはなし ● 高度なML学習やHPC用途としてはまだ発展途上
DRMベース共有 (i915ドライバー)	● オープンソースであり、Linuxカーネル（i915）に統合済み ● 統合GPUのマルチプロセス共有をサポート ● グラフィックス合成、レンダリング、Linuxデスクトップに最適 ● セットアップが容易	● 主にグラフィックス重視。計算集約型やAI/MLには不向き ● ハードウェア保証のないベストエフォートな分離 ● 高負荷時にはパフォーマンスが低下する可能性がある

Intel Gaudi ソフトウェアスタック アーキテクチャ



https://docs.habana.ai/en/latest/Gaudi_Overview/Intel_Gaudi_Software_Suite.html

Intel Gaudi ネットワーキング オプション

Intel Gaudi AI アクセラレータは、他の AI ベンダーとは大きく異なるネットワーキング アプローチを採用しています。各 Intel Gaudi アクセラレータには、ネットワーク機能（24 x 200GbE、双方向で 4.8Tbps）が標準で組み込まれています。
Intel Gaudi 3 AI アクセラレータでは、高性能な AI および HPC ワークロードに合わせてカスタマイズされた **RDMA**（**Remote Direct Memory Access**） ネットワーキングに大幅な拡張が導入されました。これらの独自技術の詳細は「Intel Gaudi 3 AI Accelerator White Paper」をご参照ください。次のセクションでその要点をまとめます。

Intel Gaudi RDMA 拡張機能

前述のとおりホワイトペーパー「Intel Gaudi 3 AI Accelerator White Paper」から、独自の RDMA 拡張機能について説明しているネットワーキング セクションの要点をまとめます。

拡張MPI集合操作サポート

- 従来の **RDMA**: 明示的なメモリ・アドレスを必要とするread/write操作に基づいています。
- **DNN**（深層学習）ワークロード: MPI スタイルの「Send/Receive」による集団通信（コレクティブ）を好みます。
- **Intel Gaudi 3** の実装: 送信側でハードウェアベースの Rendezvous handling を実装しており、以下を可能にします。
    - ソフトウェアの複雑さを軽減: 通信処理のオーバーヘッドを削減。
    - 低遅延とシングルパス・データ転送: データを一度の転送で効率的に送信。
    - コレクティブ・ロジックのオフロード: 集団通信のロジックを NIC ハードウェア（アクセラレータ内蔵）に直接オフロード。

ハードウェアへの集団通信オフロード

- rank/port調整へのCPU関与を低減します。
- わずか **300 KB** という小さなバッファサイズでフル帯域幅の利用が可能です。
- 効率的なマルチランク、マルチポートの集団通信操作を実現します。

時間ベースの輻輳（ふくそう）制御

- 一般的な ECN（明示的輻輳通知）だけでなく、**RTT**（往復時間）ベースのアルゴリズム（SWIFT®など）を実装しています。
- 標準的な RoCE 輻輳管理（RCM）よりも応答性が高く安定しています。
- 大規模なDNN（深層学習）クラスタにおいて、フローの整合性を高め、スループットのジッター（揺らぎ）を抑制します。

高度な負荷分散（パケットスプレー）

- パス認識型の動的負荷分散
    - 静的な ECMP（等コストマルチパス）を回避します。
    - 順序が入れ替わったパケットをインテリジェントに並べ替えます。
    - 高帯域幅を維持しつつ、テールレイテンシを低く抑えます。

メモリ効率に優れた信頼性のある接続 (RDMA RC)

- 大規模クラスタで問題となる **QP**（キューペア）数の爆発的増加を回避します。
- CCL（Collective Communication Library）を使用しQP スケーリングを効率的に管理します。
    - プロセス数に関係なく、対向ノードあたりわずか 4 つの QP で済みます。
    - メモリ使用量がプロセス数ではなくノード数に対して線形に増加するようになります。

ネットワーク内リダクション

- NIC（アクセラレータ内蔵）上でリダクション演算（合計、最小、最大）を実行します。
    - サポートデータ型：FP8, FP16, BF16, FP32。
- CPU の計算負荷を軽減し、通信と計算のより高度な並列処理を可能にします。

テンソル セマンティクス RDMA

- 標準的な RDMA は連続したバッファのみをサポートしますが、Gaudi 3 の NIC はテンソル・エンジンを導入し、テンソルおよびサブテンソルの概念をサポートします。
- 複雑な多次元配列にまたがる操作が発生する DNN ワークロードに最適化されています。

選択的確認応答（SACK）と選択的再送

- 標準的な RoCEv2 (InfiniBand スタイル) は「Go-Back-N」再送を使用するため、パケットロスが発生しやすい Ethernet 環境では非効率です。
- Gaudi 3 は 選択的確認応答 **(Selective ACK)** と 選択的再送 をサポートし、スループットの向上とテールレイテンシの低減を実現します。
- これにより、PFC（優先度ベースのフロー制御）を必要としない Ethernet のみの構成でも高い性能を発揮できます。

Intel Gaudi 3 の RDMA スタックは、InfiniBand や独自のスイッチハードウェアに依存することなく、AI のスケーラビリティ、パフォーマンス、および信頼性に特化した機能によって RoCEv2 を大幅に強化しています。

クラスタ アーキテクチャ構成例

標準的な Ethernet 機器のみを使用して、大規模な AI Factory を構築する例です。

- 512ノード（4096基のGPU）クラスタの構築例
    - サブクラスタ: 16ノード単位で構成。各サブクラスタに 3台の 800GbE スイッチを配置。
    - 全体接続: 32個のサブクラスタを 48台の 800GbE基幹スイッチで接続。
- 特徴
    - 標準的な **Ethernet** トポロジー: InfiniBand や NVLink スイッチは不要です。
    - **3**階層ファットツリー・トポロジー: 市販の（Off-the-shelf）Ethernet ハードウェアを使用して、完全にバランスの取れたファブリックを構築可能です。

機能	メリット
統合された NIC + 計算ユニット	レイテンシ（遅延）を最小限に抑え、同期処理へのホスト **CPU** の関与を排除。
ハードウェアへの集団通信オフロード	スループットの向上と、**CPU** 使用率の低減を実現。
テンソル認識型 **RDMA**	テンソルから連続バッファへのデータ変換に伴う複雑さを解消。
タイムリーな輻輳制御	ECN（明示的輻輳通知）のみの方法よりも、安定したスループットを維持。
ネットワーク内リダクション	集団通信のオフロードにより、**AI** 学習（トレーニング）を加速。
RoCE のスケーラビリティ	数千ノード規模にわたる、効率的な接続スケーリングが可能。
標準的な Ethernet ファブリック	高価な独自スイッチファブリックが不要。

付録 - 仮想KaaS

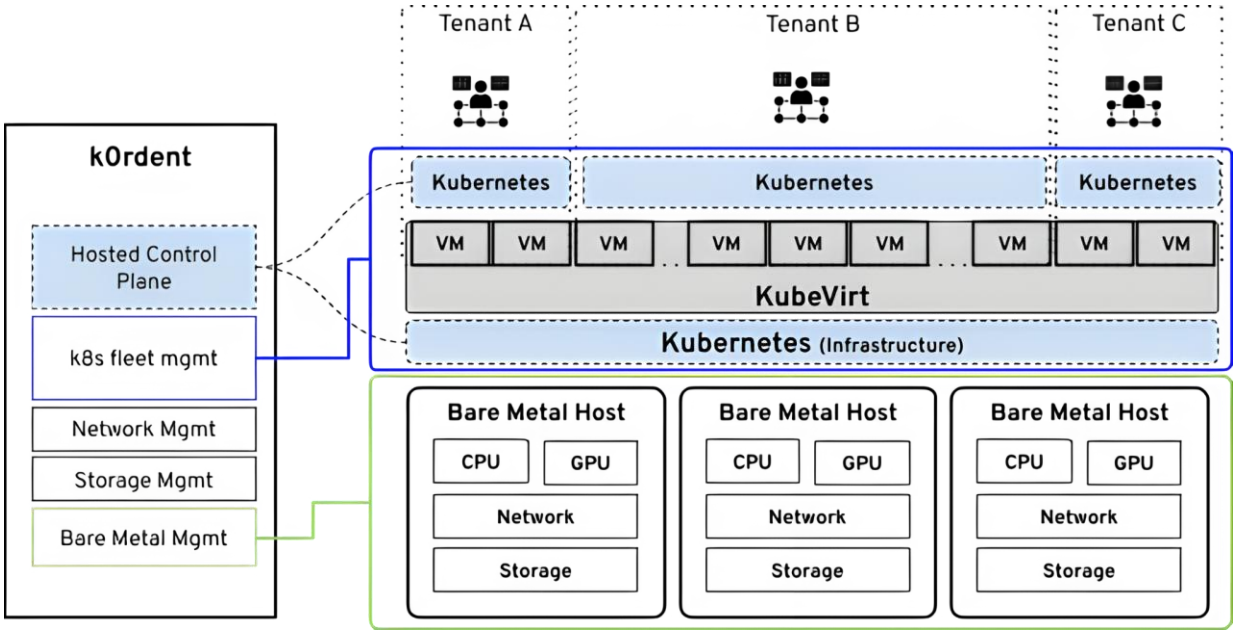
k0rdent AI によるAI Factory の構築

28 May 2025

目次

強力な分離を備えたマルチテナント Kubernetes as a Service (KaaS)

このセクションでは、スタックの全レイヤーにおいて厳格な分離を実現する、マルチテナント向けの **Kubernetes as a Service (KaaS)** 展開の詳細な概要を説明します。
このソリューションにより、サービスプロバイダーや企業は、大規模なベアメタル計算ノードを効率的に活用しながら、マルチテナント KaaS を提供できるようになります。複雑なネットワーキングと分離をサポートしつつ、テナントが自らの K8s クラスタへフルアクセスすることを可能にし、クラスタレベルで独自の **CNI**（**Container Network Interface**） を選択できる柔軟性も維持しています。



最終的な成果
エンドユーザー（テナント）およびサービスプロバイダーにとっての最終的な成果は、主要なサービスがあらかじめデプロイ・設定され、即座にワークロードを展開できる状態の **Kubernetes** クラスタが得られることです。提供されるクラスタには、以下の特性が備わっています。

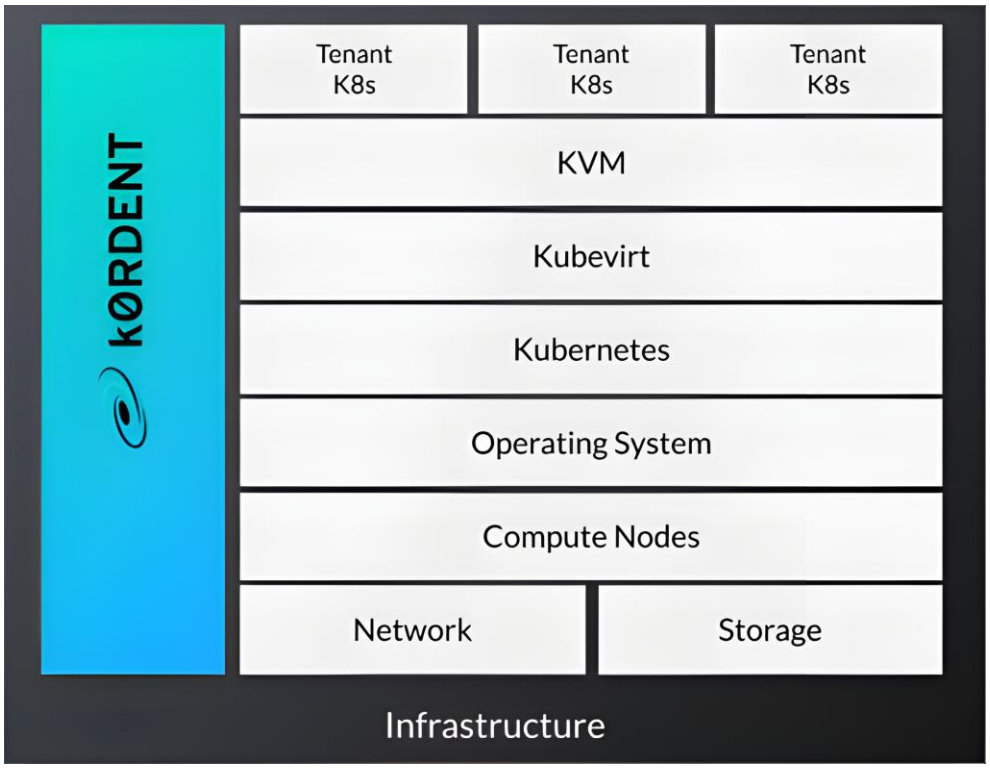
- ワーカーノードとコントロールプレーンの厳格な分離
  - Kubernetes のワーカーノード（計算リソース）とコントロールプレーン（管理機能）が完全に切り離されています。
- 完全ホスト・管理型の Kubernetes コントロールプレーン
  - コントロールプレーンはサービスプロバイダー側で完全にホストおよび管理されるため、テナントは管理負荷から解放されます。

- ネットワークインターフェース・レベルでの厳格な隔離
    - テナントは自身に割り当てられたネットワークインターフェースのみにアクセス可能です。また、それらのインターフェースは、そのテナント用に指定された VLAN にのみ接続できます。
- 隔離されたイミュータブル（不変）なワーカー用 OS

- 万が一、テナントがホスト OS を侵害（コンプライズ）したとしても、上位のネットワークやハイパーバイザー ホストには到達できない構造になっています。

ソリューションの概要

Mirantisは、**k0rdent** を活用し、テナント間の分離を備えたオンデマンドの **KaaS**（**Kubernetes as a Service**）クラスタを提供する、マルチティア（多階層）サービスを実現します。



k0rdent による多階層自動管理
k0rdent は、次の各レイヤーを一貫してデプロイおよび管理するための自動化機能を提供します。

- ベアメタル計算ホストとオペレーティングシステム
    - ベアメタルの起動から、ベースとなる OS のインストールまでを管理します。
- ホスト管理レイヤーとしてのベアメタル Kubernetes クラスタ

    - インフラ全体を制御するための土台となる、ベアメタル上の管理用 Kubernetes クラスタです。
- KubeVirt によってオーケストレーションされる仮想化レイヤー
    - Kubernetes 上で仮想マシン（VM）を動かす技術「KubeVirt」を用い、物理リソースを論理的に切り分けます。
- 隔離されたテナント・クラスタ
    - テナント（ユーザー）専用のクラスタです。ワーカーノードは **VM** として提供され、コントロールプレーンはホスト側で一括管理（Hosted Control Plane）されます。

計算ノードのネットワークインターフェースカード（NIC）の仕様や要件に応じて、次の2つのネットワーク展開オプションが提供されます。

	NVIDIA BlueField-3 DPU 構成	標準的なネットワーキング構成
環境の概要	計算ノードに **BlueField-3 DPU** を搭載。	計算ノードに 標準的な **NIC**（通常のネットワークカード）を搭載。
結果	ホストから隔離されたネットワーキング	ホスト **OS** 内でのネットワーキング

BlueField-3 ソリューションの概要

BlueField-3 の構成

Bluefield-3 DPU は、ホスト分離のために DPU モードおよび ZeroTrust モードで構成され、k0rdent を使用してプロビジョニングされます。カードの物理的な接続設定は、ホストで利用可能な BlueField カードの数と環境の要件に応じて指定でき、単一ホスト上でEthernetと Infinibandの両方をサポートします。今回の構成では、両方のポートがEthernetスイッチに接続されています。

Bluefield-3 DPU は、k0rdent ベアメタルオペレーターにより、それら用に定義された特定のマシンテンプレートに基づいてカスタムイメージで初期化されます。

カードが初期化されると、次のように設定されます。

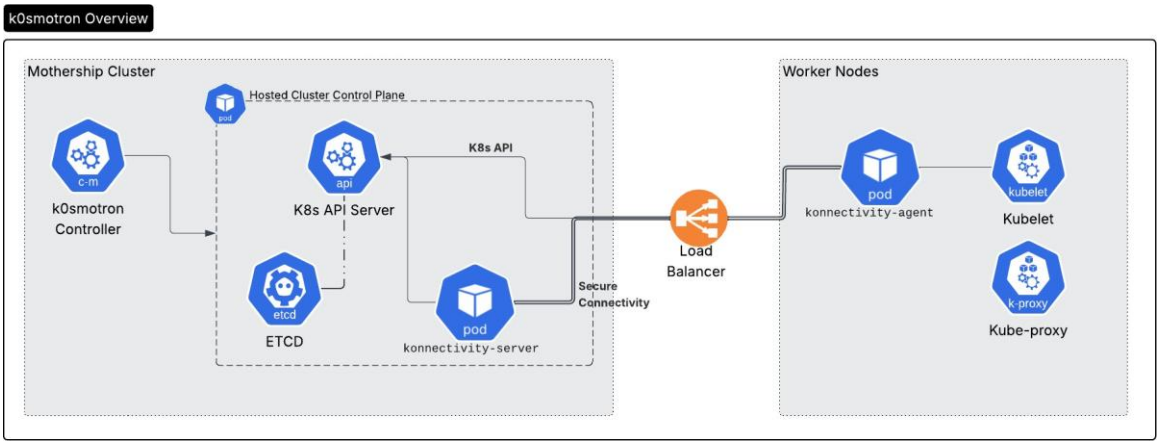
- redfishを介したZeroTrustを有効化
  （ '{"Oem": {"Nvidia": {"HostPrivilege": "Restricted"}}}' ）
- VirtIOエミュレーションとPCIスイッチエミュレーションを有効化し、ホストPCIe RShimインターフェースを無効化

Example Code Block

```
mlxconfig -d <device> set
   \ VIRTIO_NET_EMULATION_ENABLE=1
   \ VIRTIO_NET_EMULATION_NUM_PF=10
   \ VIRTIO_NET_EMULATION_NUM_VF=10
   \ PCI_SWITCH_EMULATION_ENABLE=1
   \ PCI_SWITCH_EMULATION_NUM_PORT=10
   \ RSHIM_ENABLE=0
```

k0rdentは、テンプレートに従って以下のプロセスを自動的に実行します。

- k0rdent Bare Metal Operator は、k0sワーカーを含むカスタムイメージでDPUを登録およびプロビジョニングするために使用されます。
- k8sクラスタは、k0smotronが提供するセキュアなコントロールプレーンを活用するホスト型コントロールプレーンテンプレートを使用してDPU上にプロビジョニングされ、ワーカーとコントロールプレーンの厳密な分離を保証します。
- DPFコンポーネントは、k0rdentサービス テンプレートを使用してDPUクラスタ上にデプロイされます。



ハイパーバイザーおよびベアメタルホストのプロビジョニング

ハイパーバイザーホストは、k0rdentベアメタルプロバイダーとk0smotronホスト型コントロールプレーンを使用してベアメタルホスト上にプロビジョニングされた、Kubernetesワーカーです。（これと同じパターンを、隔離されたネットワーキングを備えた専用Kubernetesクラスタとしてテナントに提供することも可能です。）その後、このKubernetesクラスタにKubeVirtがデプロイされ、仮想マシンのオーケストレーション層が提供されます。

デプロイメントの手順

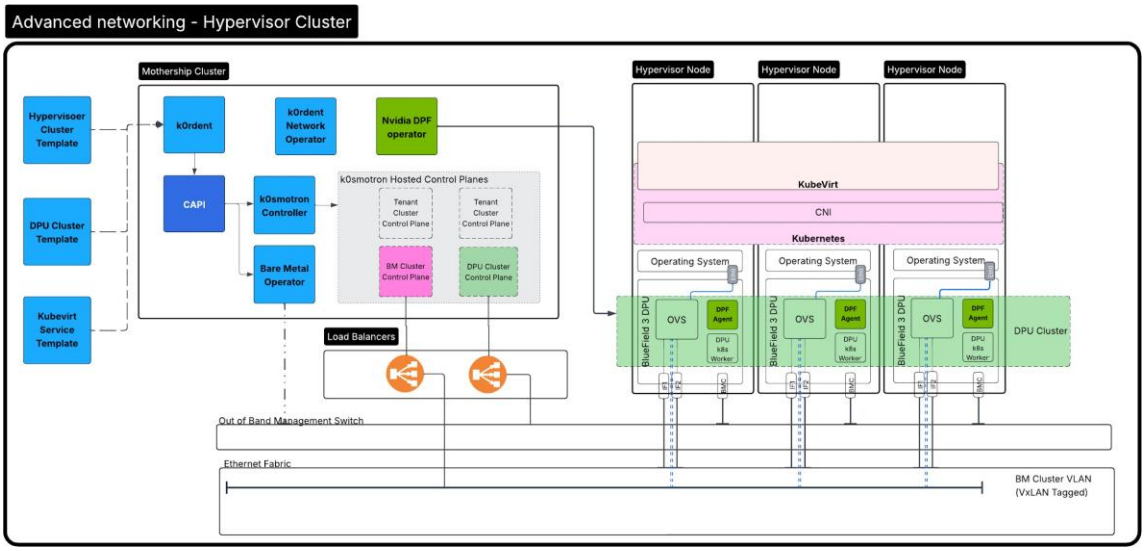
1. ネットワークの作成: k0rdentネットワーキング（テナント用にネットワークオブジェクトを作成）が、テナントネットワーク名、IPアドレス（IPAMサポートを含む）、およびVxLAN IDを定義します。
2. 管理インターフェース: k0rdent Network Operatorがホスト上のK8sクラスタ管理ポート用のポートを割り当て、OVS（Open vSwitch）を構成します。（一般的なVMデプロイメントでは、この手順は不要です。）
3. ノードディスカバリ: k0rdentベアメタル（IronicおよびMetal3を活用）がノードを照会し、関連する管理ネットワークポートの検出を含む、IOシステムおよびノード構成の詳細情報を取得します。
4. ノードプール: デプロイメントに使用できる状態のノードプールへノードを割り当てます。
5. クラスタのデプロイ: 使用するノードプールとネットワークを参照する、k0rdent Cluster Deploymentオブジェクトが作成されます。
6. ノードのプロビジョニング: オペレーティングシステムとKubernetesがk0rdentによってデプロイされます（CAPIおよびMetal3プロバイダーを活用）。
7. **GPU**オペレーター: Cluster Deploymentオブジェクトで定義されたサービス テンプレートにより、NVIDIA GPU Operatorがクラスタ上にデプロイされます。（セットアップに応じて、ここで構成オプションが適用されます。）
8. **KubeVirt**のデプロイ: Cluster Deployment オブジェクトで定義されたサービス テンプレートにより、KubeVirtがクラスタ上にデプロイされます。

ベアメタルホスト
ベアメタルホストのテンプレートには、ホスト構成が仮想化に対して適切に準備されることを保証するため、cloud-initを通じた特定の環境設定が含まれています。これらには以下の設定が含まれます（これらはIntelベースのホストを想定しています）。

- **IOMMU**の有効化: ホスト上の分離をより強固にするために有効化します。
  - intel_iommu=on iommu=pt
- **PCI**再割り当て（**PCI Reallocation**）の有効化: PCIアドレスの再利用をサポートするために有効化します。
  - pci=realloc
- nouveauモジュールの無効化とNVIDIA GPUドライバーのデプロイ
  - echo "blacklist nouveau" | sudo tee /etc/modprobe.d/blacklist-nouveau.conf
  - echo "options nouveau modeset=0" | sudo tee -a /etc/modprobe.d/blacklist-nouveau.conf

GPU オペレーター
ハイパーバイザーをデプロイする際、GPU オペレーター（GPU Operator）のサービステンプレートにおいて、VM パススルー（VFIO）または VM vGPU のどちらを使用するかを選択する必要があります。（単一のクラスタに両方をデプロイすることも可能ですが、その場合はホストレベルでの設定が必要になります。）
ここでの選択はCluster Deploymentオブジェクトで行われ、それによりデプロイメントの構成が正しくなり、ノードラベルが適切に設定されます。例として、VGPU を使用する場合は以下の通りです：

- ノードラベル: nvidia.com/gpu.workload.config=vm-vgpu
- サンドボックスの有効化（VM ワークロード）: sandboxWorkloads.enabled=true

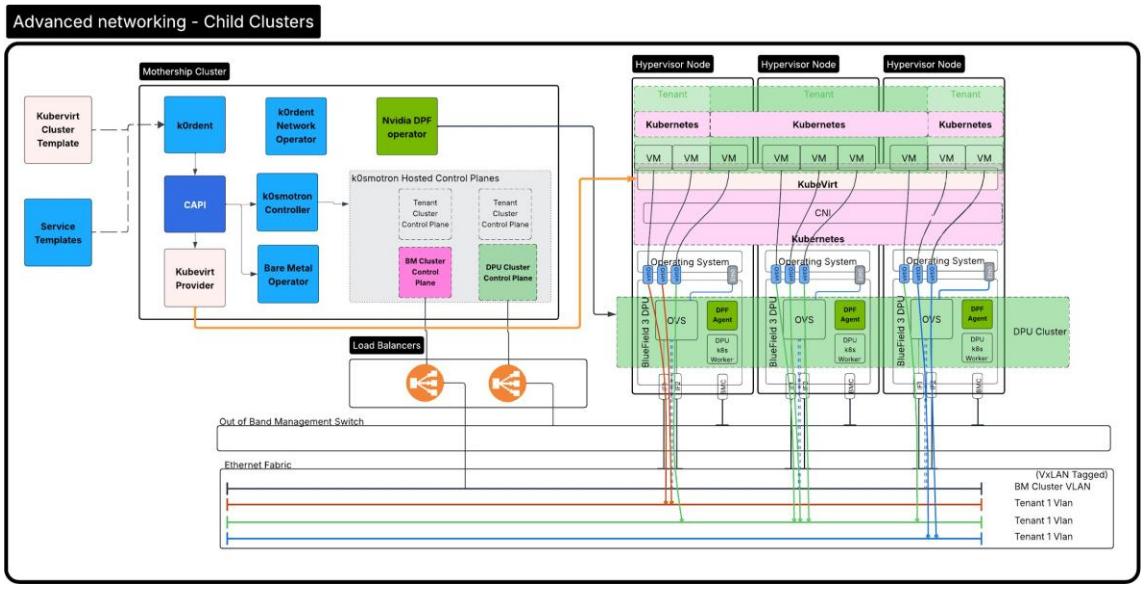


現時点で、ベアメタル上に構築され、KubeVirtとKVMが動作する、厳格な隔離を備えた完全管理型のKubernetesクラスタが完成し、テナント用の「チャイルド・クラスタ（子クラスタ）」をデプロイする準備が整いました。分かりやすくするために、ここではこれを「ハイパーバイザー・クラスター」と呼びますが、これも一つのチャイルド・クラスター（子クラスタ）です。「箱の中に箱がある」、あるいは「（世界を支える亀の下にはまた別の亀がいるという）turtles all the way down（無限後退）」のミームがここには当てはまります。

この環境は、GPUのパススルーをサポートするように構成されており、ワークロードの必要に応じてネットワークのSR-IOVもサポート可能です。

チャイルド クラスタ **(Child Clusters)**

チャイルド クラスタのデプロイはベアメタル・クラスタのデプロイと非常によく似ていますが、ベアメタル プロバイダの代わりに、KubeVirtプロバイダーをプロビジョナーとして使用する点が異なります。クラスタは、KubeVirtクラスタ・テンプレートを活用してクラスタをデプロイするCluster Deploymentオブジェクトを定義することで、宣言的に構築されます。



クラスタ テンプレート

使用されるチャイルド クラスタのテンプレートは、あらかじめ選択されたGPUプロビジョニング戦略（パススルーまたはvGPU）に応じて決まります。クラスタごとに固有のCluster Deploymentオブジェクトが作成され、それがk0rdentのKubeVirtプロバイダーおよびネットワークプロバイダーを活用してクラスタをプロビジョニングします。

システムが実行する手順は次の通りです。

1. ネットワーク作成: ネットワーク オペレーター（Network Operator）が、IPAMルール（またはユーザー割り当て）に基づいてネットワークを作成します。

2. ネットワーク フローの構成: ネットワーク・オペレーターがML2プラグイン・メカニズムを介してOVS（Open vSwitch）のネットワーク・フローを構成し、VirtIOポートをリンクして適切なネットワークにタグ付けするためのフローをセットアップします。

3. クラスタ デプロイ: その後、k0rdentが仮想マシン（VM）を作成し、割り当てられたポートにそれらを接続することでクラスタをデプロイします。

付録 - ホスト型コントロールプレーン

Mirantis k0rdent AIによるAI Factory の構築

28 May 2025

目次

ホスト型コントロールプレーン

Kubernetes ホスト型コントロールプレーン（「Control Plane as a Service」または「管理クラスター・ホスト型コントロールプレーン」とも呼ばれる）は、Kubernetes のコントロールプレーン・コンポーネント（API サーバー、コントローラーマネージャー、スケジューラーなど）を、専用の仮想マシンや物理ホスト上ではなく、「管理クラスタ」と呼ばれる別の Kubernetes クラスタ内でポッドとして実行する設計パターンです。

このパターンは、マルチクラスタ管理、SaaS 提供、および Kubernetes クラスタのコスト効率の高いスケーリングにおいて、ますます利用されるようになっています。

中核となる概念

従来の Kubernetes クラスタでは、各クラスタが独自のコントロールプレーンとワーカーノードを持つ自己完結型のユニットでした。ホスト型コントロールプレーン・アーキテクチャでは次のようになります。

- コントロールプレーン: 別の Kubernetes クラスター内部でワークロードとしてコンテナ上で動作します。
- ワーカーノード: コントロールプレーンから切り離され、セキュアなネットワークを介してリモートで接続します。

ビジネスおよび技術的メリット

1. 大規模なスケーラビリティ**:** 単一のプラットフォームから数百、数千のクラスターを管理できます。
2. リソースの効率化: コントロールプレーンが管理クラスター内の計算リソースを共有するため、クラスターごとのオーバーヘッドが削減されます。
3. 運用の簡素化: すべてのコントロールプレーンを標準的な Kubernetes ツール（kubectl、GitOps など）を使用して管理できます。
4. 運用の簡素化**:** すべてのコントロールプレーンを標準的な **Kubernetes** ツール（**kubectl**、**GitOps** など）を使用して管理できます。
5. マルチテナンシーの向上: 各テナントに隔離されたコントロールプレーンを提供しつつ、運用者は管理を中央集約化できます。
6. セキュリティとコンプライアンス: ポリシー適用と監査の中央強制が可能になります。また、コントロールプレーンは、信頼できない可能性のあるワーカーノードから隔離されます。

ユースケース

- **SaaS** プロバイダー: 顧客ごとに専用の Kubernetes クラスターを、最小限のオーバーヘッドで提供します。
- エッジ展開**:** 中央のコントロールプレーン・ホスティング環境から、数千のエッジ・ワーカーノードを管理します。
- **CI/CD** 環境**:** 一時的なテスト用クラスターを迅速に起動・削除します。
- 教育・トレーニング**:** トレーニング・ラボ用のクラスターを迅速かつ効率的にプロビジョニングします。

アーキテクチャの概要

管理クラスタ

- 多数のテナント/ユーザー・クラスターのコントロールプレーンをホストします。
- 各コントロールプレーンを、名前空間（隔離のための RBAC、リソースクォータ、セキュリティポリシーを含む）内のポッドセットとして実行します。

ホスト型コントロールプレーン

- 通常、次が含まれます:
    - kube-apiserver
    - kube-controller-manager
    - kube-scheduler
    - etcd（オプション：外部または共有の場合もあります）

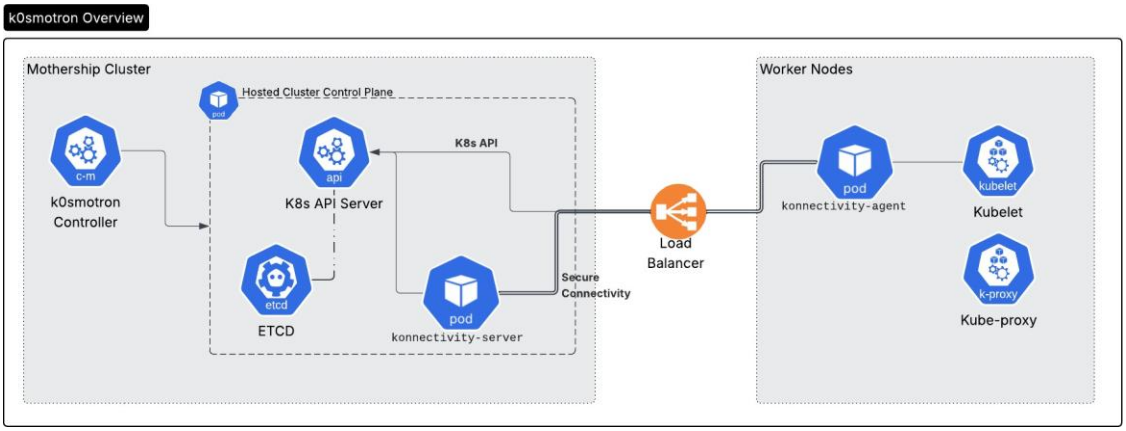
- リモートのワーカーノードが接続可能な API エンドポイントを公開します。

ワーカーノード

- クラウド上の仮想マシン（VM）、物理サーバー、またはエッジデバイス上で動作します。
- 管理クラスター内で動作しているそれぞれのコントロールプレーンと通信します。

k0smotron

k0smotron は、Mirantis によって開発されたオープンソースの Kubernetes 拡張機能です。これにより、「管理」Kubernetes クラスタ内において、k0s クラスタとして知られる Kubernetes クラスタのコントロールプレーンを、宣言的に管理することが可能になります。k0smotron は、コントロールプレーンを Kubernetes 内部のワークロードとして実行できるようにすることで、マルチクラスタのライフサイクルマネジメントを簡素化するように設計されています。



アーキテクチャの概要

1. ポッドとしての Kubernetes コントロールプレーン
    - **k0smotron** によるデプロイ: k0smotron は、（k0s を介して）Kubernetes コントロールプレーンを管理クラスター内のポッドとしてデプロイします。
    - ワーカーノードとの分離: これらのコントロールプレーンは、ターゲットクラスターのワーカーノードから完全に切り離されており、ワーカーノードはどこでも（ベアメタル、VM、クラウド）実行可能です。
    - 組み込みのディストリビューション: 必要なすべてのコンポーネント（etcd、kube-apiserver、scheduler、controller-manager）を含む k0s の組み込み Kubernetes ディストリビューションを使用します。

2. セキュリティと懸念の分離
    - 管理クラスタの役割: 管理クラスタ内ではコントロールプレーンのみが動作します。
    - セキュアな接続: （異なるクラウドやデータセンターにある）ワーカーノードは、セキュアなトンネルを介してコントロールプレーンに接続します。
    - **API** の公開: Kubernetes API は、クライアントアクセスや自動化のために ロードバランサーを通じて公開されます。

3. セキュアな接続性
    - 通信ネットワーク: コントロールプレーンとワーカーノードは、パブリックまたは

内部ネットワーク経由で通信します。
    - **Konnectivity** による保護: コントロールプレーンの接続性は、**Konnectivity** サービスを通じた SSL トンネル内で保護されます。

4. マルチテナンシーと隔離
    - 集約ホスティング: 単一の管理クラスター上で複数のコントロールプレーンをホストでき、それぞれが個別の名前空間に隔離されます。
    - ガバナンス: 分離とガバナンスのために、リソース制限（Resource Limits）を適用できます。

5. クラスタのライフサイクル
    - ライフサイクルマネジメント: CRD（カスタムリソース定義）とコントローラーロジックを介して、Kubernetes クラスターの作成、アップグレード、削除をサポートします。
    - 自動化統合: CNCF プロジェクトや CI/CD パイプラインと統合して自動化が可能です。
    - **CRD** ベースの構成:
        i. 宣言的プロビジョニング: Cluster などの CRD を使用してクラスターを定義し、宣言的なコントロールプレーンのプロビジョニングを可能にします。
        ii. **GitOps** のサポート: これらの定義を Git に保存することで、GitOps ワークフローをサポートします。

付録 - k0rdent アーキテクチャ

Mirantis k0rdent AI によるAI Factory の構築

28 May 2025

目次

k0rdent のアーキテクチャ

k0rdent のアーキテクチャは、Kubernetes の原則に基づいたクラスタ管理への「宣言的アプローチ」を採用しています。モジュール化され拡張性に優れたアーキテクチャにより、Cluster API (CAPI) やその他の Kubernetes コンポーネントといったサブコンポーネントと相互作用するための、再現可能なテンプレート駆動型のソリューションを提供します。

アーキテクチャの主要な原則は次の通りです。

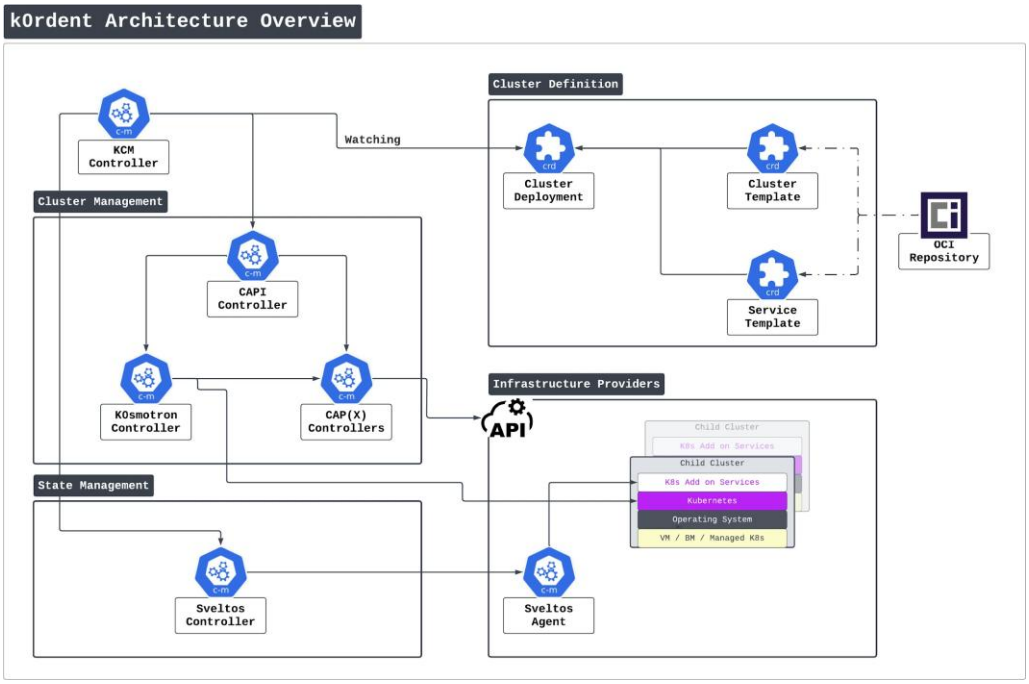
- Kubernetes コア原則の活用
- 高度に整合しつつも、疎結合なアーキテクチャ
- プラグイン可能で拡張性の高いアーキテクチャ
- 再現性のためのテンプレート駆動型アプローチ
- 標準化された API の採用
- 修正を加えていないアップストリーム コンポーネントの活用（例：Kubernetes Cluster API）
- ダウンストリームにおけるカスタムコンポーネントとの統合サポート

概要

k0rdent 管理クラスタ（k0rdent Management Cluster）は、複数のクラウドやインフラストラクチャ上にある複数のチャイルド・クラスタ（子クラスタ）のプロビジョニングとライフサイクルをオーケストレーションし、個別のインフラプロバイダーと直接やり取りする必要性を排除します。このようにインフラを抽象化することで、k0rdent は再利用性を促進し（例えば、特定のクラウド上で IDP を実装するための労力を削減）、実用的な標準化を促します。これにより、好みのクラウドやテクノロジーを利用しつつ、オープンソースのサブシステムやクラウド基盤といったコンポーネントの切り替えコストを最小限に抑えることができます。

k0rdent のアーキテクチャは、次のハイレベルなコンポーネントで構成されています。

- クラスタ管理 **(Cluster Management)**：クラスタの定義、プロビジョニング、および管理を行うためのツールとコントローラー。
- 状態管理 **(State Management)**：チャイルド クラスタとそのワークロードの状態を監視、更新、および管理するためのコントローラーとシステム。
- インフラストラクチャ・プロバイダー **(Infrastructure Providers)**: クラスタ用の仮想マシン、ネットワーク、ストレージといったリソースを、その背後で実際にプロビジョニングする役割を担うサービスと API。
- テンプレート **(Templates)**：管理対象のチャイルド クラスタ、またはその上で実行されるワークロードを定義します。これらのテンプレートをインスタンス化（実体化）することで、対応するリソースが作成されます。



管理クラスタ

管理クラスタ（「マザーシップ クラスタ」とも呼ばれます）は、k0rdent アーキテクチャの中核です。k0rdent を機能させるために必要なすべてのコントローラーをホストします。
これには次が含まれます：

- **k0rdent Cluster Manager (KCM)** コントローラー: KCM は、k0rdent の CAPI 関連機能のラッパーを提供し、以下をオーケストレーションします。
    - **Cluster API (CAPI)** コントローラー: 特定のインフラストラクチャ・プロバイダーと連携するように設計されています。例えば、ある CAPI コントローラーは AWS 上で動作する Kubernetes クラスタの作成とライフサイクルを管理し、別のコントローラーは Azure 上のクラスタを管理します。内部システムと統合するためにカスタム CAPI コントローラーを作成することも可能です。
    - **k0smotron** コントローラー: k0smotron は CAPI を拡張し、k0s Kubernetes 用のコントロールプレーンおよびワーカーノードのブートストラップ・プロバイダーなどの追加機能を提供します。また、「ホスト型コントロールプレーン（Hosted Control Plane）」、つまりホストとなる Kubernetes クラスタ（k0rdent をホストしているクラスタ自体も可能）上のポッドとして表現される k0s コントロールプレーンの作成をサポートするプロバイダーも提供します。さらに、k0smotron プロジェクトは CAPI 用の「RemoteMachine」インフラプロバイダーを提供しており、任意のプロトコル（SSH など）を介してリモート Linux サーバー（小規模なエッジデバイスを含む）上でのデプロイやクラスタ操作を可能にします。

- **k0rdent State Manager (KSM)** コントローラー: KSM は、クラスタ上のサービスやアプリケーションのライフサイクルマネジメント（デプロイ、スケーリング、更新、アップグレ

ード、削除）、およびそれらの継続的な状態管理を担当します。現在は KCM のコードベースの一部ですが、将来的に分割される可能性があります。以下をオーケストレーションします

- **Services Controller**:ームに機能を追加するサービスの組み合わせや、インフラプロバイダーの依存関係など、Kubernetes サービスの調整を担当します。例えば、依存関係を含む Nginx を一つの「サービス」としてパッケージ化できます。サービス用のアーティファクトはローカルまたは OCI リポジトリに保存され、Kubernetes CRD オブジェクトとして参照されます。

- **k0rdent Observability & FinOps (KOF)** コントローラー（上の図には描かれていません）： KOF は、k0rdent で管理される Kubernetes クラスタに対して、エンタープライズグレードのオブザーバビリティと FinOps（コスト管理）機能を提供します。統合された OpenTelemetry ベースのアーキテクチャを通じて、中央集約的なメトリクス、ロギング、およびコスト管理を可能にします。

これらの詳細については、後述の「役割と責任」のセクションで詳しく見ていきます。

クラスタ デプロイメント

クラスタ デプロイメントは、「チャイルド クラスタ」または「ワークロード クラスタ」とも呼ばれます。これはマネジメント クラスタによってプロビジョニングおよび管理される Kubernetes クラスタであり、開発者がアプリケーションやワークロードを実行する場所です。これらは「通常の」Kubernetes クラスタであり、管理用コンポーネントをホストすることはありません。クラスタは名前空間を介して管理クラスタから完全に隔離されているだけでなく、クラスタ同士も互いに隔離されており、マルチテナント環境の構築が可能です。
チャイルド クラスタは、イングレスコントローラー、モニタリング ツール、ロギングソリューションなどのカスタマイズされたアドオンを使用して、特定のユースケースに合わせて調整できます。また、ネットワークポリシー、ストレージクラス、セキュリティポリシーなどの特定の Kubernetes 構成を定義して、ユーザーのアプリケーションや環境に最適化することも可能です。

チャイルド クラスタは、イングレスコントローラー、モニタリング ツール、ロギングソリューションなどのカスタマイズされたアドオンを使用して、特定のユースケースに合わせて調整できます。また、ネットワークポリシー、ストレージクラス、セキュリティポリシーなどの特定の Kubernetes 構成を定義して、ユーザーのアプリケーションや環境に最適化することも可能です。

主な用語

- マザーシップ クラスタ **(Mothership Cluster)**: 管理およびオブザーバビリティコンポーネ

ントをホストし、全体のコントロールプレーンとなる Kubernetes クラスタ。

- チャイルド クラスタ **(Child Cluster)**: マザーシップから k0rdent によってデプロイおよび管理される Kubernetes クラスタ。
    - 注記: これらは「Cluster Deployment」オブジェクトとして定義されます。
- クラスタ テンプレート **(Cluster Templates)**: クラスタ デプロイメントのタイプに関する設定と構成を定義するテンプレート。通常、特定のインフラストラクチャ プロバイダーのタイプごとにスコープが設定されます。
- サービス テンプレート **(Service Templates)**: クラスタ上で実行される Kubernetes アプリケーションを定義するテンプレート。複雑な構成を含めることができ、デプロイされたクラスタからの情報を活用することも可能です。

テンプレート

プラットフォーム・エンジニアリングの哲学における重要な原則の一つは「Infrastructure as Code (IaC)」の活用ですが、k0rdent はテンプレートを使用することで、その概念を一歩先へと進めています。テンプレートは、クラスターの作成と管理に使用できるコンポーネントの再利用可能なテキスト定義です。テンプレートは、ユーザーや開発者が複雑なクラスターやコンポーネントをデプロイ・管理するための宣言的な手法を提供し、設定が必要なパラメータ数を大幅に削減します。

一般的に、k0rdent のテンプレートは次のような特徴を持ちます：

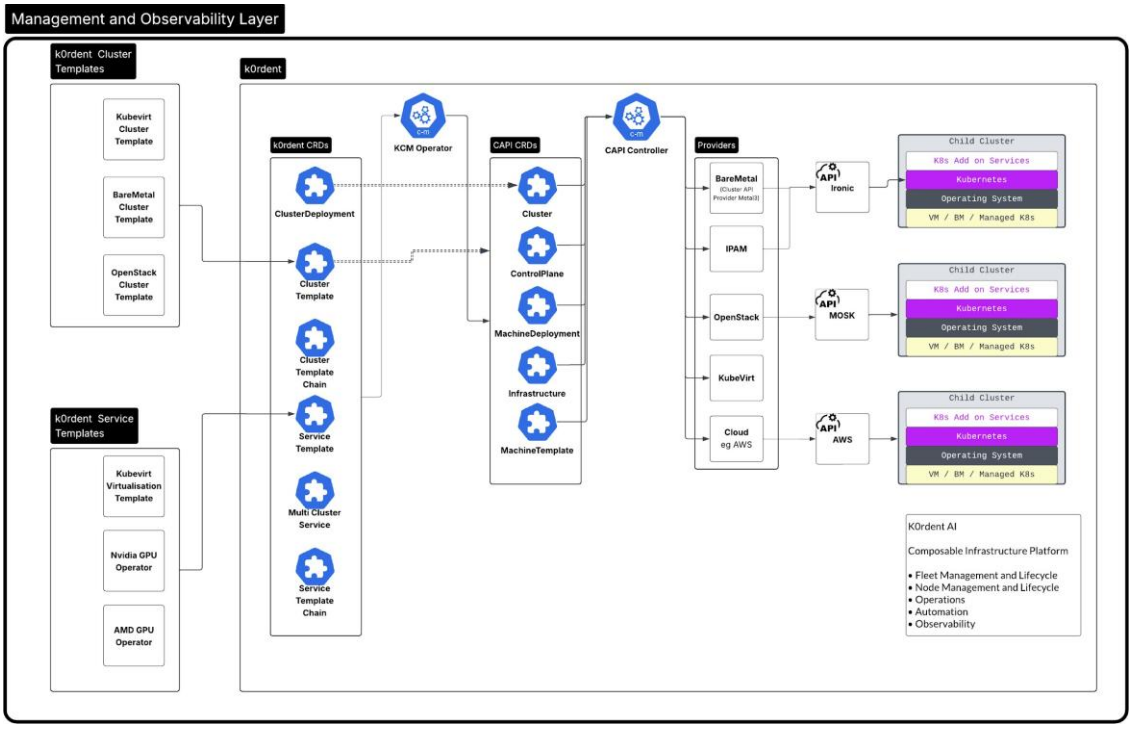
- **YAML** 形式: テンプレートはターゲットの状態を表現するための抽象化レイヤーとして YAML を使用しているため、人間が読みやすく、編集も容易です。
- ランタイム パラメータ化による多目的利用: プレースホルダー（変数）を使用することで、テンプレートを直接編集することなく、実行時にカスタマイズすることが可能です。
- クラスタ作成とアドオン管理の両方に使用: ユーザーは YAML を使用してクラスタを定義できるほか、イングレス・オペレーターやモニタリング・ツールなどのアドオンをそれらのクラスタに追加するように定義することもできます。
- スコープ制限が可能: k0rdent では、誰がどのテンプレートをデプロイできるかという制限を設定できます。例えば、プラットフォーム管理者（「役割と責任(Roles and Responsibilities))」セクションを参照）は、管理者以外のユーザーが特定のコントローラーセットをデプロイするテンプレートのみを実行できるように指定できます。

k0rdent で使用される主なテンプレートの種類には、次のものがあります

- **Cluster Templates**（クラスタ テンプレート）：ClusterTemplate オブジェクトは、クラスタが動作するクラウドやインフラストラクチャと連携してクラスタを定義します。これらは「イミュータブル（不変）」であるように設計されています。ClusterDeployment などの k0rdent オブジェクトによって呼び出され、個々のクラスタやクラスターグル

ープを作成・管理します。

- **Service Templates**（サービステンプレート）: ServiceTemplate オブジェクトは、クラスタ上で実行されるサービス、アドオン、およびワークロードを定義します。これらも同様に「イミュータブル」に設計されており、ClusterDeployment やその他の k0rdent オブジェクトによって呼び出されることで、IDP（内部開発プラットフォーム）やプラットフォームを一つの単位として宣言・管理できるようにします。



役割と責任

k0rdentは、階層的かつ補完的な役割と責任を持つ複数のグループによって使用されることを想定して設計されています。組織によって呼び名は異なるかもしれませんが、ここでは次のように定義します。

- プラットフォーム アーキテクト: ビジネスおよび技術的なステークホルダーに対し、IDP（内部開発プラットフォーム）の設計についてグローバルな責任を持つ個人またはチームです。特定のクラウドやインフラ、ワークロード、性能・コスト目標、セキュリティ・規制、運用要件への適合を見据えた設計を行います。k0rdentは、プラットフォーム・アーキテクトが再利用可能な「ClusterTemplate」と「ServiceTemplate」のセットを作成し、プラットフォームを抽象的に厳密定義することを可能にします。
- プラットフォーム リード:ビジネスおよび技術的なステークホルダーに対し、IDP（内部開発プラットフォーム）の設計についてグローバルな責任を持つ個人またはチームです。特定のクラウドやインフラ、ワークロード、性能・コスト目標、セキュリティ・規制、運用要件への適合

を見据えた設計を行います。k0rdentは、プラットフォーム・アーキテクトが再利用可能な「ClusterTemplate」と「ServiceTemplate」のセットを作成し、プラットフォームを抽象的に厳密定義することを可能にします。

- プラットフォーム・エンジニア:環境の日常的な管理を担当する個人またはチームです。プラットフォーム・リードから提供された（権限を与えられた）ClusterTemplateおよびServiceTemplateを使用し、さらに自身のアプリケーションに適したKubernetesクラスタへとカスタマイズするために、追加のServiceTemplateを作成することもあります。

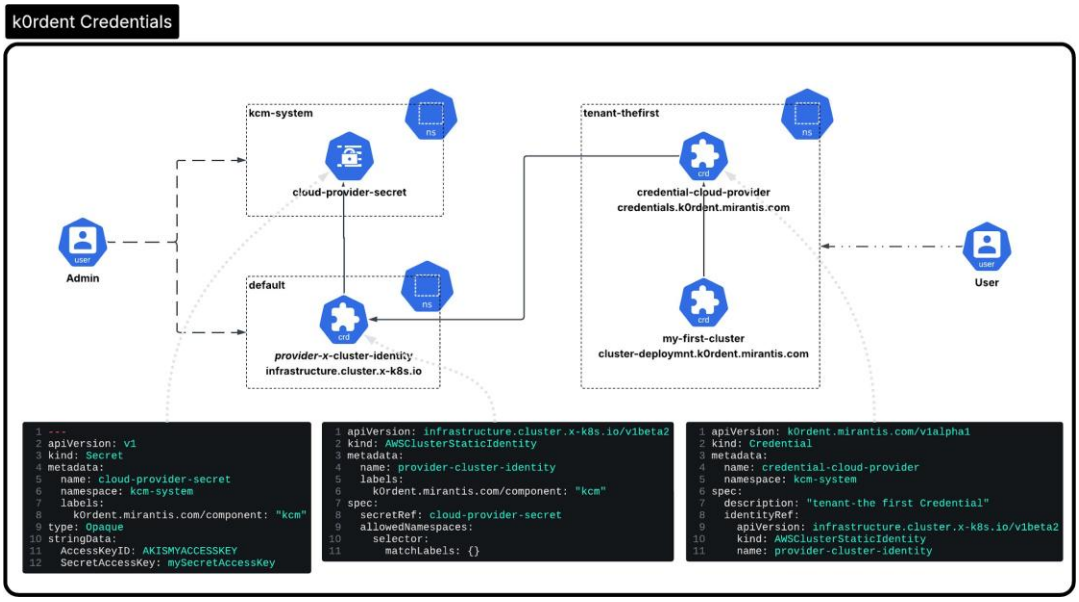
資格情報（Credential）

Kubernetesクラスタの作成と管理には、ターゲットとなるインフラ環境に対する適切な権限が必要ですが、AWSのアカウント情報をすべての開発者に配布することは望ましくありません。

この問題を解決するため、k0rdentでは、開発者が必要とするアクセス権を提供するCredentialオブジェクトを作成できます。仕組みは次の通りです：

1. The platform lead creates a provider-specific ClusterIdentity and Secret that include all of the information necessary to perform various actions. プラットフォーム・リードが、各種アクションの実行に必要なすべての情報を含む、プロバイダー固有のClusterIdentityおよびSecretを作成します。

   プラットフォーム・リードは、そのClusterIdentityを参照するCredentialオブジェクトを作成します。
2. 開発者は、このCredentialオブジェクトを参照します。これにより、開発者に資格情報を直接公開することなく、クラスタがこれらの資格情報（小文字のcredentials）にアクセスできるようになります。



Mirantis AI Factory の日本国内のお問い合わせはクリエーションライン株式会社まで。
お問い合わせはこちら。