

Dynamic Resource Allocation with AWS Neuron

Kenta Goto

Solutions Architect

2026/07



自己紹介

後藤 健汰

アマゾンウェブサービスジャパン
ソリューションアーキテクト

役割

- ✓ 様々な業界の顧客への技術支援
- ✓ コンテナ関連の技術支援

好きなこと

- ✓ Kubernetes
- ✓ Platform Engineering



@kennygt51



アジェンダ

1.AWS Trainium

2.Dynamic Resource Allocation

3.Trainium x Dynamic Resource Allocation

4.Demo

AWS Trainium



AWS Trainium2

AWS が自社開発した第 3 世代の生成 AI / ML アクセラレータ

1.3 / 5.2 PFLOPS **96** GB **2.9** TB/s
Dense / Sparse compute HBM 容量 HBM 帯域幅



2024 年一般提供開始

Trn2 : Trainium2 搭載インスタンス

- 最大 16 個の Trainium2 を搭載 (Trn2.48xlarge)
- 同等のインスタンスと比較し
30%-40% 高いコスト性能を実現
- 最大 **1.5 TB** の高速 HBM3 メモリ搭載
- **3,200 Gbps** のネットワーク帯域に対応
- Anthropic社 Claude モデルは Trainium2 上にデプロイ

Dynamic Resource Allocation



Device Plugin

GPU や NIC などの特定のハードウェアリソースを Kubernetes クラスター内で管理/利用可能にするための拡張機能

Device Plugins

Device plugins let you configure your cluster with support for devices or resources that require vendor-specific setup, such as GPUs, NICs, FPGAs, or non-volatile main memory.

FEATURE STATE: Kubernetes v1.26 [stable]

Kubernetes provides a device plugin framework that you can use to advertise system hardware resources to the Kubelet.

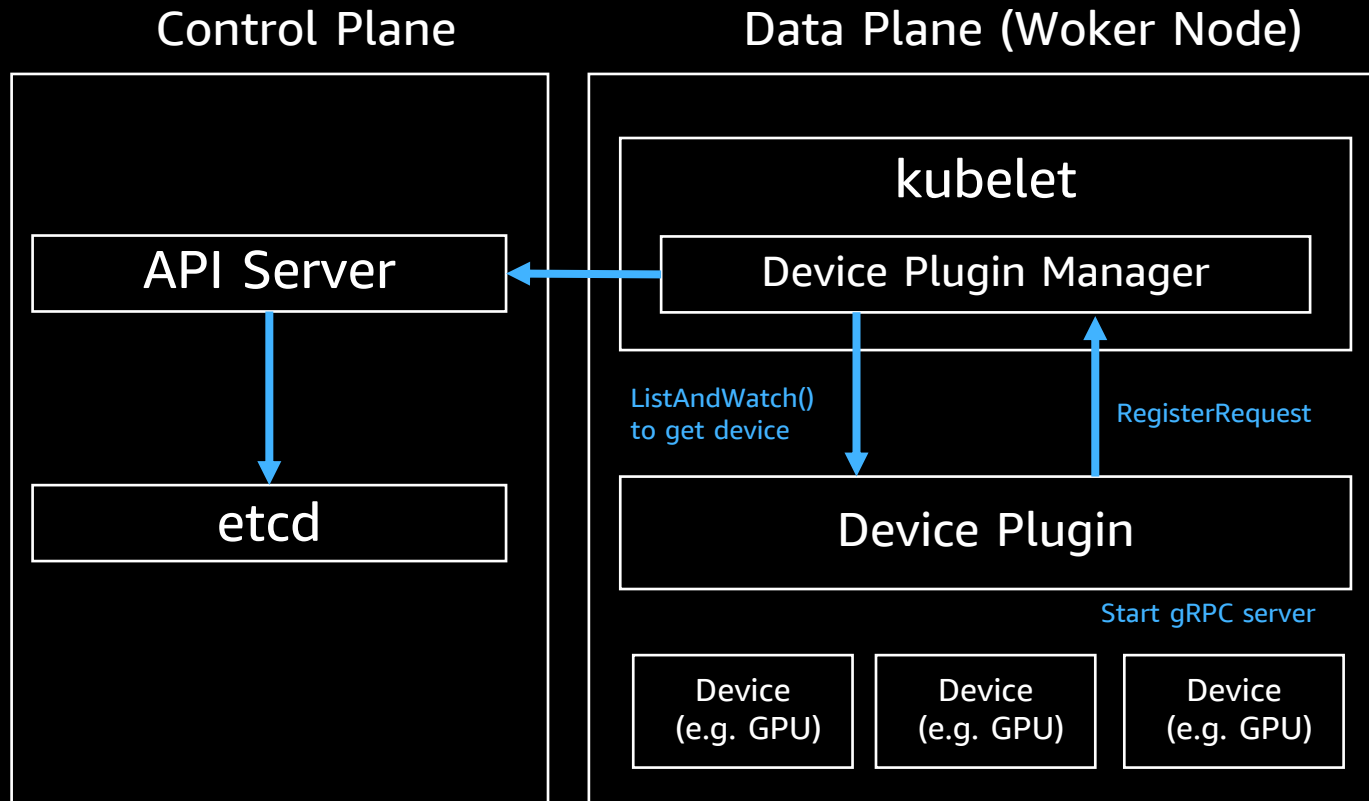
Instead of customizing the code for Kubernetes itself, vendors can implement a device plugin that you deploy either manually or as a DaemonSet. The targeted devices include GPUs, high-performance NICs, FPGAs, InfiniBand adapters, and other similar computing resources that may require vendor specific initialization and setup.

“Kubernetes は device plugin framework を提供しており、これを使用してシステムのハードウェアリソースを **kubelet にアドバタイズ**することができます”

“Kubernetes 自体のコードをカスタマイズする代わりに、ベンダーは手動または **DaemonSet** としてデプロイ可能な **device plugin** を実装できます”

<https://kubernetes.io/docs/concepts/extend-kubernetes/compute-storage-net/device-plugins/>

Device Plugin

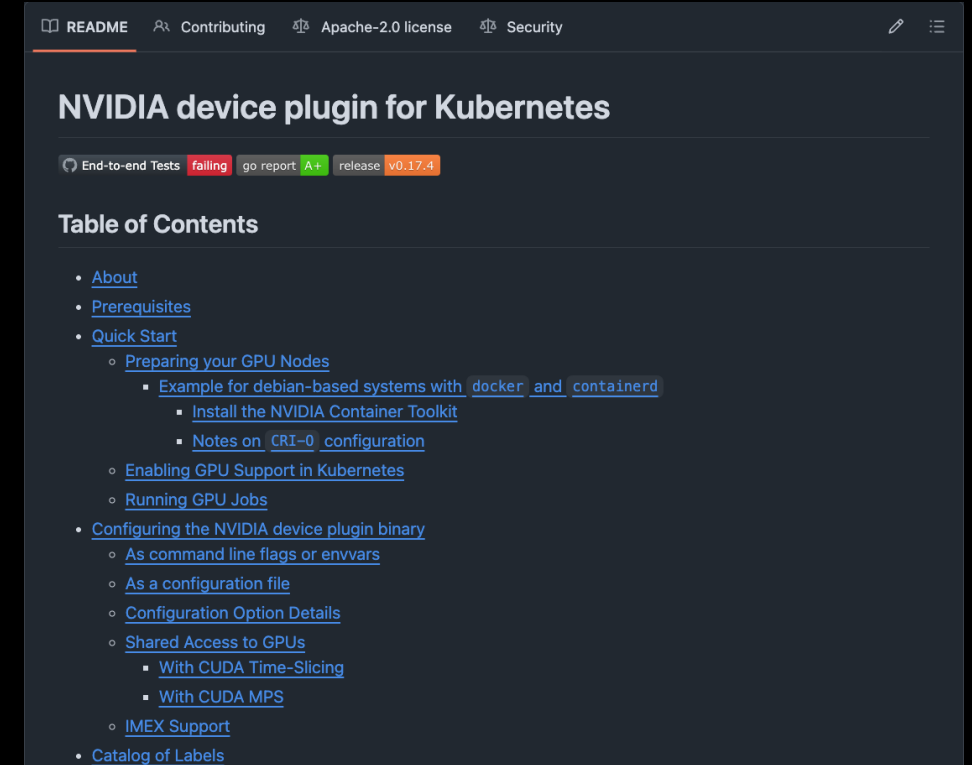


- ✓ Device Plugin は Worker Node 上で DaemonSet 起動し、gRPC インターフェースを通じて kubelet に自身を登録 (kubelet はどの種類のデバイスが利用可能かを認識)
- ✓ Device Plugin は kubelet に対して自身が管理するデバイスの一覧と状態を ListAndWatch() で逐次共有
- ✓ Pod 作成時に kubelet は Device Plugin の Allocate() を呼び出し、コンテナに割り当てるデバイスファイルのマウント情報や環境変数などを受け取る

<https://kubernetes.io/docs/concepts/extend-kubernetes/compute-storage-net/device-plugins/>

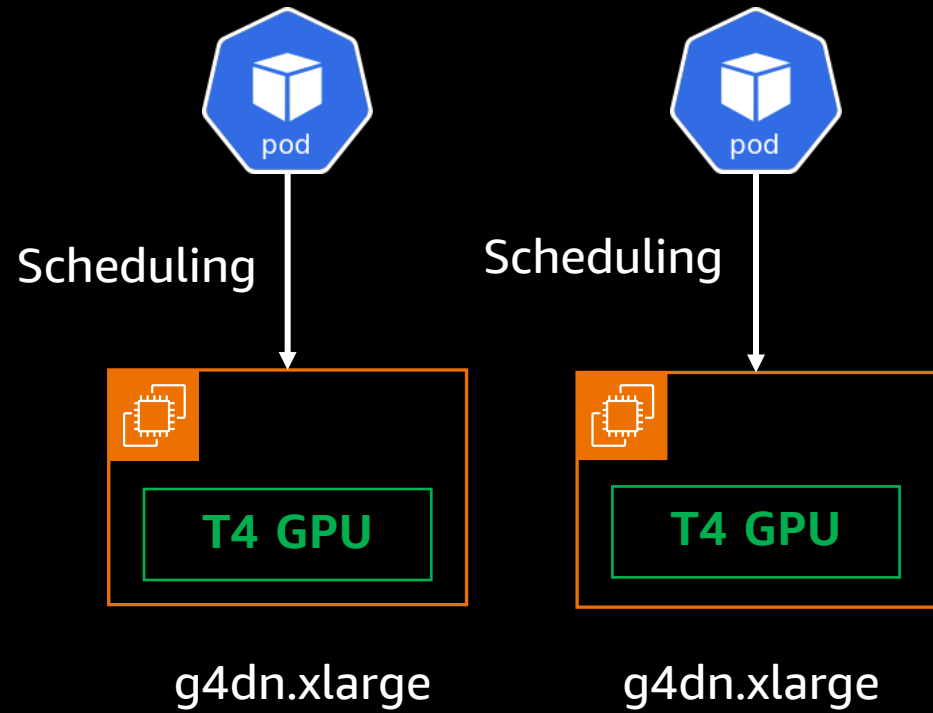
NVIDIA Device Plugin

- ✓ NVIDIA の GPU をコンテナで利用可能にするための Device Plugin
- ✓ Worker Node に DaemonSet としてインストールされ、ノード上の GPU の状態を監視して kubelet に通知
- ✓ Pod の作成時に GPU 使用のために必要なドライバライブラリをコンテナにマウントし、GPU 利用の準備を自動化
- ✓ これにより、ユーザーは Pod のリソース指定で `nvidia.com/gpu` を指定するだけで GPU を利用



<https://github.com/NVIDIA/k8s-device-plugin>

Device Plugin による一般的な GPU Allocation

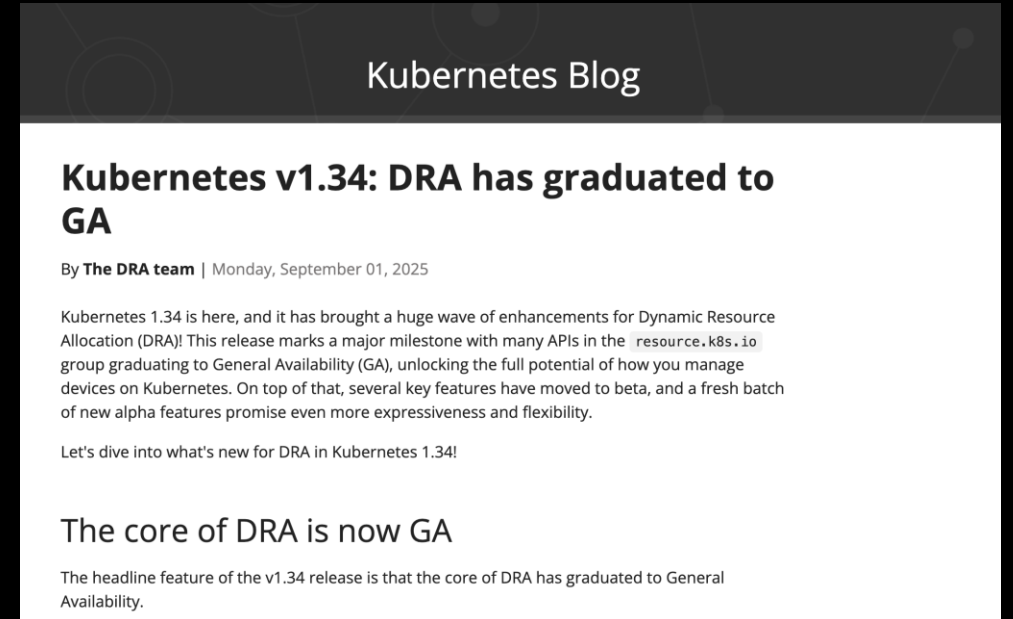


```
apiVersion: v1
kind: Pod
metadata:
  name: gpu-pod-example
spec:
  containers:
    - name: gpu-container
      image: nvidia/cuda:11.0-base
      resources:
        limits:
          nvidia.com/gpu: 1
```

GPU 要求を整数としてしか表現できず、
デバイス属性(GPU Type やメモリ量)やトポロジー (NVLinkで接続されているか等)が
スケジューラーの判断に入らない

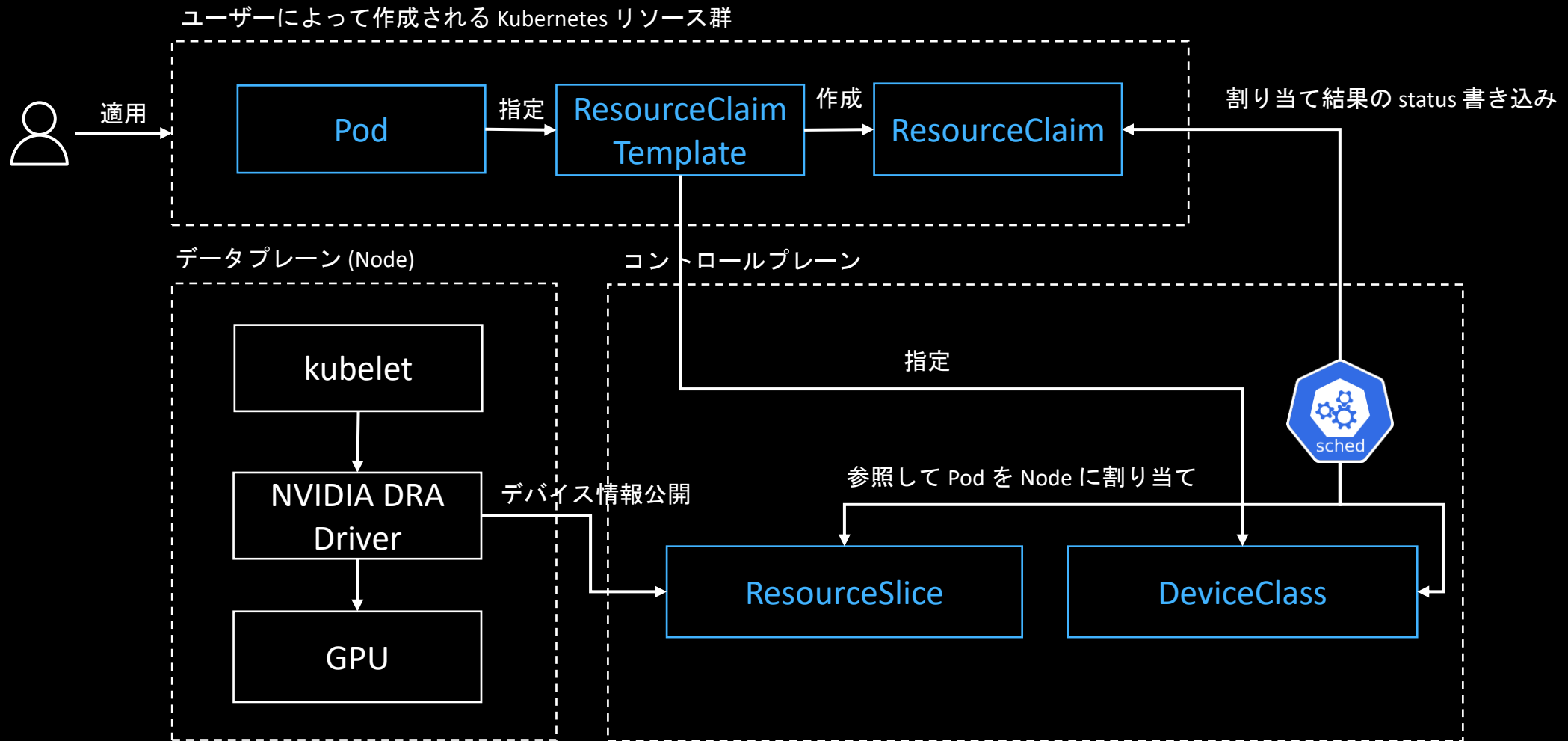
Dynamic Resource Allocation

- ✓ Kubernetes v1.34 で GA となった Kubernetes API で、**デバイスの割り当て (Allocation)** を **Kubernetes API のオブジェクトとして扱える**ようにする仕組み
- ✓ 「どんなデバイスを使いたいのか？」を Kubernetes オブジェクトとして表現し、それをもとにスケジューラーが「何を使わせるか」を決定する
- ✓ スケジューラーがデバイスの属性 (e.g. H200 かつ VRAM が 80GB 以上のデバイス) を元にスケジューリングしたり、NVLink で接続された GPU 同士を一緒に割り当てたり、複数のノードにまたがる GPU をひとつのリソースプールとして表現・割り当てできる



<https://kubernetes.io/blog/2025/09/01/kubernetes-v1-34-dra-updates/>

Dynamic Resource Allocation コンポーネント図



Dynamic Resource Allocation で提供されるリソース

Kubernetes リソース	概要
ResourceClaim	✓ ワークロードが必要な HW リソース (GPU など) を要求するためのリソース
ResourceClaimTemplate	✓ Podごとに新しい ResourceClaim を自動で作成するためのテンプレート
DeviceClass	✓ ワークロードがリクエストできるデバイスのカテゴリ（例：GPU）や特定のデバイス構成を定義 ✓ ResourceClaim / ResourceClaimTemplate から参照
ResourceSlice	✓ ノード上で利用可能なデバイスリソース（例：GPUやFPGAなど）のインスタンス情報を構造化して公開する ✓ 各デバイスのUUIDやモデル名など、詳細属性が含まれる

Step1: ユーザーが ResourceClaim を登録

- ✓ ResourceClaim は「どんなデバイスを、どんな条件で使いたい」を表すリソース
- ✓ Pod はこの Claim を参照し、スケジューラは Claim を満たすデバイスを選ぶ
- ✓ Claim は「要求」「申請」を意味しており、PersistentVolumeClaim (PVC) がストレージを要求するように、ResourceClaim はデバイスを要求する

```
apiVersion: resource.k8s.io/v1beta1
kind: ResourceClaim
metadata:
  name: single-gpu-claim
  namespace: default
spec:
  devices:
    requests:
      - name: gpu
        deviceClassName: gpu.nvidia.com # DeviceClass で対象デバイスを絞り込む
        selectors:                        # CEL 式でさらに条件を指定できる
          - cel:
              expression: device.capacity['nvidia.com'].memory.compareTo(quantity('16Gi'))
```

Step1: ユーザーが ResourceClaim を登録 (DeviceClass とは)

- ✓ DeviceClass は「どの種類のデバイスを対象にするか」を定義するクラスタスコープのリソース
- ✓ GPU、FPGA、NIC などの分類を指定し、DRA Driver が ResourceSlice で公開するデバイス集合から対象を絞り込むフィルタとして機能する
- ✓ 通常は DRA ドライバのベンダーやクラスタ管理者が用意し、ユーザーは ResourceClaim から名前で参照する

```
apiVersion: resource.k8s.io/v1beta1
kind: DeviceClass
metadata:
  name: gpu.nvidia.com
spec:
  selectors:
  - cel:
      expression: device.driver == 'gpu.nvidia.com' # NVIDIA ドライバが公開するデバイスに
```

Step 2: ユーザーが Pod を作成

- ✓ Pod からは `spec.resourceClaims` で Claim を参照し、コンテナ側で `claims` として使用を宣言

```
apiVersion: v1
kind: Pod
metadata:
  name: gpu-pod
spec:
  containers:
    - name: app
      image: nvidia/cuda:12.4.0-base-ubuntu22.04
      resources:
        claims:
          - name: gpu                # spec.resourceClaims のエントリ名を参照
  resourceClaims:
    - name: gpu
      resourceClaimName: single-gpu-claim
```

Trainium x Dynamic Resource Allocation



AWS Neuron の DRA サポート

- ✓ Neuron DRA ドライバーが EKS で利用可能に
 - ✓ Trainium インスタンス向けに、Kubernetes ネイティブのハードウェア対応スケジューリングを実現
- ✓ ML エンジニアの負担を軽減
 - ✓ デバイス数やトポロジー、ネットワーク構成といったインフラの詳細を意識せずにワークロードをデプロイできる
- ✓ インフラチームと ML チームの関心を分離
 - ✓ インフラチームが ResourceClaimTemplate を定義し、ML エンジニアはそれを参照するだけで正しいデバイス配置が得られる
- ✓ 分散トレーニング・長コンテキスト推論・分離推論にスケール
 - ✓ 複数ワークロードが同一ノードを効率的に共有でき、スケールアップ時のボトルネックを解消

AWS Neuron announces support for Dynamic Resource Allocation with Amazon EKS

Posted on: Mar 20, 2026

AWS announces the Neuron Dynamic Resource Allocation (DRA) driver for Amazon Elastic Kubernetes Service (EKS), bringing Kubernetes-native hardware-aware scheduling to AWS Trainium-based instances. The Neuron DRA driver publishes rich device attributes directly to the Kubernetes scheduler, enabling topology-aware placement decisions without custom scheduler extensions.

Deploying AI workloads on Kubernetes requires ML engineers to make infrastructure decisions that are not directly related to model development, such as determining device counts, understanding hardware and network topologies, and writing accelerator-specific manifests. This creates friction, slows iteration, and tightly couples workloads to underlying infrastructure. As use cases expand to distributed training, long-context inference, and disaggregated architectures, this complexity becomes a scaling bottleneck.

The Neuron DRA driver removes this burden by separating infrastructure concerns from ML workflows. Infrastructure teams define reusable ResourceClaimTemplates that capture device topology, allocation, and networking policies. ML engineers can simply reference these templates in their manifests, without needing to reason about hardware details. This enables consistent deployment across workload types while allowing per-workload configuration so multiple workloads can efficiently share the same nodes.

The Neuron DRA driver supports all AWS Trainium instance types and is available in all AWS Regions where AWS Trainium is available.

For documentation, sample templates, and implementation guides, visit [the Neuron DRA documentation](#).

Learn more:

- [Neuron EKS DRA templates](#)
- [Neuron EKS documentation](#)
- [Amazon EKS documentation](#)

<https://aws.amazon.com/about-aws/whats-new/2026/03/neuron-eks-dra-support/>



Neuron DRA Driver

AWS Trainium インスタンス向けの DRA Driver

- ✓ NeuronCore の様々な属性 (Device ID やトポロジー) を Kubernetes scheduler に直接公開する
- ✓ DRA driver と Neuron Device Plugin は共存できない点、2026/07 時点で EKS Auto Mode / Karpenter では非対応である点に留意

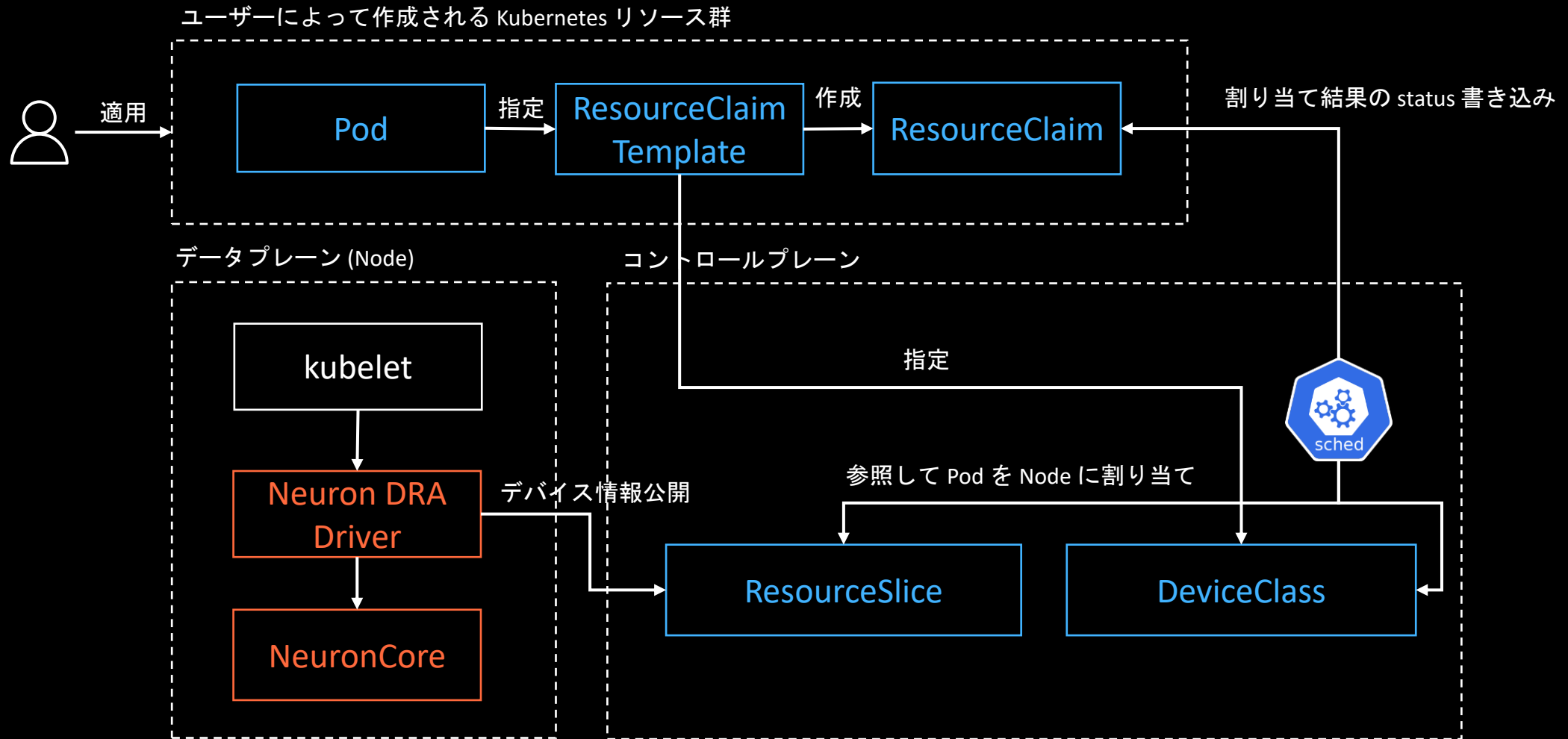
Helm Chart によるインストール (neuron-dra-driver DaemonSet として起動)

```
helm upgrade --install neuron-helm-chart oci://public.ecr.aws/neuron/neuron-helm-chart   
  --namespace neuron-dra-driver \  
  --create-namespace \  
  --set "devicePlugin.enabled=false" \  
  --set "npd.enabled=false" \  
  --set "draDriver.enabled=true"
```

※ <https://docs.aws.amazon.com/eks/latest/userguide/device-management-neuron.html>



Dynamic Resource Allocation コンポーネント図 (Neuron)



ユースケース

公式ドキュメント (※) で言及されているユースケース

Connected Neuron Devices

- ✓ トポロジを考慮して、近接する Neuron Devices のサブセットに Pod を割り当てる
- ✓ 従来 (Neuron Device Plugin) は Neuron Scheduler Extension が必要だったが、DRA ではネイティブに実現

Dynamic LNC config

- ✓ ワークロード単位で LNC を設定する
- ✓ ResourceClaim にてデフォルトの LNC を上書き (ノード単位での override が不要)

Four Node Inference on trn2u.48xlarge

- ✓ Trn2 UltraServer (Neuron Links で相互接続された 4 ノード構成) に対し、1 / 2 / 4 のサブセットを指定して割り当てる

※ <https://awsdocs-neuron.readthedocs-hosted.com/en/v2.27.0/containers/neuron-dra.html>



Demo



セッションをご視聴いただき有難うございました。
お近くのブーススタッフまで QR コードのご提示をお願いいたします。

Thank you for attending this session. Please show your QR code to the nearest booth staff.

Thank you!