# Hardware accelerators

## NPU

Alauda Build of NPU Operator

## About Alauda Build of Hami

About Alauda Build of Hami

## About Alauda Build of NVIDIA GPU Device Plugin

About Alauda Build of NVIDIA GPU Device Plugin

# NPU

## Alauda Build of NPU Operator

### Introduction

### Installation

Prerequisites

Procedure

FAQ

# Alauda Build of NPU Operator

## Introduction

## Installation

Prerequisites

Procedure

FAQ

# Introduction

Kubernetes provides access to special hardware resources (such as Ascend NPUs) through Device Plugins. However, multiple software components (such as drivers, container runtimes, or other libraries) are required to configure and manage a node having these hardware resources. Installation of these components is complex, difficult, and error-prone. The NPU operator uses the Operator Framework in Kubernetes to automatically manage all software components required for configuring Ascend devices. These components include the Ascend driver and firmware (which support the entire running process of clusters), as well as the MindCluster device plug-in (which supports cluster operations such as job scheduling, O&M monitoring, and fault recovery). By installing corresponding components, you can manage NPU resources, optimize workload scheduling, and containerize training and inference tasks, so that AI jobs can be deployed and run on NPU devices as containers.

For more details, refer to NPU Operator↗.

# Installation

## TOC



# Prerequisites

## Online Installation

1. **ACP version: v4.0 or later**
2. **Cluster administrator access to your ACP cluster**
3. **Ensure that the bash tool exists in the NPU node. Otherwise, the driver and firmware installation script may fail to be parsed.**

4. **Worker Node Operating System Requirements**

   - Worker nodes (or node groups) running **NPU workloads** must use one of the following operating systems (Arm architecture):

     - `openEuler 22.03 LTS`
     - `Ubuntu 22.04`

   - Worker nodes running **CPU workloads only** can use any operating system, as the NPU operator performs no configuration on nodes without NPU workloads.

5. **Supported NPU Hardware**

   - Nodes must use supported NPUs:

     - `Ascend 910B`
     - `Ascend 310P`

   - For detailed OS and hardware compatibility, see MindCluster Documentation ↗

6. **The `Alauda Build of Node Feature Discovery` cluster plugin must be installed.**

## Offline Installation

1. **Offline installation requires all online installation prerequisites plus additional preparation steps.**
2. **Prepare the driver and firmware package and the MindIO SDK package.** Download the following packages (if you do not need to install MindIO, then you do not need to download the MindIO package):

   - For the driver and firmware package, find the `config.json` file in the GitCode repository of the npu-driver-installer ↗, and download the package based on the version you want to choose, the NPU model and OS architecture of the corresponding node through the corresponding link provided.
   - For the MindIO SDK package, find the `config.json` file in the GitCode repository of the npu-node-provision ↗, and download the SDK package based on the NPU model and OS architecture of the corresponding node through the corresponding link provided.

3. **Save the ZIP file of the driver and firmware package to the `/tmp/driver_pkg/` path of the node where the offline installation is to be performed.**
4. **Save the ZIP file of the MindIO package to the `/opt/openFuyao/mindio/` path of the node where the offline installation is to be performed. (If you do not need to install MindIO, skip this step.)**
5. **Check whether the target node contains the following tools.**

   - For systems using Yum as the package manager, the following package needs to be installed: "jq wget unzip which net-tools pciutils gcc make kernel-devel-$(uname-r) kernel-headers-(uname-r) dkms".
   - For systems using apt-get as the package manager, the following package needs to be installed: "jq wget unzip debianutils net-tools pciutils gcc make dkms linux-headers-$(uname -r)".
   - For systems using DNF as the package manager, the following package needs to be installed: "jq wget unzip which net-tools pciutils gcc make kernel-devel-$(uname -r) kernel-headers-(uname-r) dkms".

# Procedure

## Downloading Cluster plugin

> **INFO**
>
> You can download the app named `Alauda Build of NPU Operator` and `Alauda Build of Node Feature Discovery` from the Marketplace on the Customer Portal website.

Note: The Volcano cluster plugin can be left uninstalled for now.

## Uploading the Cluster plugin

The platform provides the `violet` command-line tool for uploading packages downloaded from the Custom Portal Marketplace.

For details, see Upload Packages.

## Installing Alauda Build of NPU Operator

1. Apply the label `masterselector=dls-master-node` to all master nodes and the label `workerselector=dls-worker-node` to all worker nodes.

```
kubectl label nodes {master-node-id} masterselector=dls-master-node
kubectl label nodes {worker-node-id} workerselector=dls-worker-node
```

2. Go to the `Administrator` -> `Marketplace` -> `Cluster Plugin` page, switch to the target cluster, and then deploy the `Alauda Build of NPU Operator` Cluster plugin. Deployment form parameter description:

> **WARNING**
>
> If the components listed in the table below are already installed, be sure to disable the corresponding buttons during deployment.

> **TIP**
>
> Ascend Operator, NodeD, ClusterD, Resilience Controller, MindIO TFT, and MindIO ACP are not deployed by default. Please deploy them only when there is a clear need for them.

| Component | Default | Description |
| --- | --- | --- |
| Driver | Enabled | Whether to install driver and firmware. |
| Version | 24.1.RC3 | Driver and firmware version. You must select the version number from the repository directory npu-driver-installer↗. |
| Ascend Device Plugin | Enabled | Whether to install Ascend Device Plugin. |

| Component | Default | Description |
| --- | --- | --- |
| Ascend Docker Runtime | Enabled | Whether to install Ascend Docker Runtime. |
| NPU exporter | Enabled | Whether to install NPU exporter. |
| Ascend Operator | Disabled | Whether to install Ascend Operator. |
| NodeD | Disabled | Whether to install NodeD. |
| ClusterD | Disabled | Whether to install ClusterD. Need to install the Volcano cluster plugin first. |
| Resilience Controller | Disabled | Whether to install Resilience Controller. |
| MindIO TFT | Disabled | Whether to install MindIO TFT. |
| MindIO ACP | Disabled | Whether to install MindIO ACP. |

## Verification

1. First, you can see the status of "Installed" in the `Alauda Build of NPU Operator` Cluster plugin page.
2. Wait for the npu-driver pod to become running. Offline installation takes about 10 minutes, while online installation is much faster.

   ```
   kubectl -n kube-system get pod -w | grep npu-driver
   ```

3. Reboot all the NPU nodes.
4. Run the following command on the npu node.

   ```
   npu-smi info
   ```

   Make sure the display is working correctly.

5. Run the following command on the master node.

```
kubectl get npuclusterpolicy cluster
```

   Make sure the status of the `npuclusterpolicy` is Ready.

6. Check whether there are allocatable NPU resources on the NPU node in the control node of the business cluster. Run the following command:

```
kubectl get node  ${nodeName} -o=jsonpath='{.status.allocatable}'
# Example, the output contains: "huawei.com/Ascend310P":"1" (the specif
ic value depends on the number of NPU cards)
```

7. Run validation workload.

> **NOTE**
>
> Business applications must manually specify the `runtimeClassName` field as `ascend`.

   Create spec file:

```
key="huawei.com/Ascend310P" # For 310P
cat <<EOF > deploy-npu.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ascend-pytorch
spec:
  replicas: 1
  selector:
    matchLabels:
      service.cpaas.io/name: deployment-ascend-pytorch
  strategy:
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1
    type: RollingUpdate
  template:
    metadata:
      labels:
        service.cpaas.io/name: deployment-ascend-pytorch
    spec:
      affinity: {}
      containers:
        - args:
            - |
              sleep infinity
          command:
            - /bin/bash
            - -c
          image: ascendai/pytorch:ubuntu-python3.8-cann8.0.rc1.beta1-py
torch2.1.0
          imagePullPolicy: Always
          name: ascend-pytorch
          resources:
            limits:
              cpu: 500m
              $key: "1"
              memory: 2Gi
            requests:
              cpu: 500m
              memory: 2Gi
      runtimeClassName: ascend
EOF
```

Apply spec:

```
kubectl apply -f deploy-npu.yaml
```

```
kubectl exec -it  deploy/ascend-pytorch -- bash
```

Then run the following command in the container:

```
npu-smi info
```

Make sure the display is working correctly.

## Installing Monitor

If the NPU exporter component was deployed when installing the Alauda Build of NPU Operator, perform the following steps to create a monitoring panel.

1. Execute commands on the **control node** of the cluster.

```
cat << EOF | kubectl apply -f -
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  labels:
    prometheus: kube-prometheus
    serviceMonitorSelector: prometheus
  name: npu-exporter-ai
  namespace: monitoring
spec:
  endpoints:
  - interval: 10s
    path: /metrics
    port: http
    targetPort: 8082
  namespaceSelector:
    matchNames:
    - npu-exporter
  selector:
    matchLabels:
      app: npu-exporter-svc
EOF
```

2. You can import a Grafana dashboard JSON file by following Import Dashboard, which converts it into a monitoring dashboard for display. The JSON file is available in ascend-npu-dashboard↗.

**NOTE**

Tags in the Grafana dashboard JSON file cannot contain Chinese characters and need to be manually deleted. For examples:

```
{
  "tags": [
    "ascend",
    "昇腾"
  ]
}
```

After modification:

```
{
  "tags": [
    "ascend"
  ]
}
```

# FAQ

## What should I pay attention to when uninstalling Alauda Build of NPU Operator?

Even after Alauda Build of NPU Operator is uninstalled, the driver may still exist on the host machine. On the NPU node, execute the following command to uninstall the driver:

```
/usr/local/Ascend/driver/script/uninstall.sh
```

# About Alauda Build of Hami

Heterogeneous AI Computing Virtualization Middleware (HAMi), formerly known as k8s-vGPU-scheduler, is an "all-in-one" chart designed to manage Heterogeneous AI Computing Devices in a k8s cluster. It can provide the ability to share Heterogeneous AI devices among tasks.

**Note**

Because Alauda Build of Hami releases on a different cadence from Alauda Container Platform, the Alauda Build of Hami documentation is now available as a separate documentation set at Alauda Build of Hami↗.

# About Alauda Build of NVIDIA GPU Device Plugin

The NVIDIA device plugin for Kubernetes is a Daemonset that allows you to automatically:

- Expose the number of GPUs on each nodes of your cluster
- Keep track of the health of your GPUs
- Run GPU enabled containers in your Kubernetes cluster.

> **Note**
>
> Because Alauda Build of NVIDIA GPU Device Plugin releases on a different cadence from Alauda Container Platform, the Alauda Build of NVIDIA GPU Device Plugin documentation is now available as a separate documentation set at Alauda Build of NVIDIA GPU Device Plugin↗.