ALIBABA CLOUD

Alibaba Cloud

Container Service for Kubernetes

User Guide for Edge Container Service

Document Version: 20220706

Alibaba Cloud

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.

2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company or individual in any form or by any means without the prior written consent of Alibaba Cloud.

3. The content of this document may be changed because of product version upgrade, adjustment, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and an updated version of this document will be released through Alibaba Cloud-authorized channels from time to time. You should pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.

4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides this document based on the "status quo", "being defective", and "existing functions" of its products and services. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not take legal responsibility for any errors or lost profits incurred by any organization, company, or individual arising from download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, take responsibility for any indirect, consequential, punitive, contingent, special, or punitive damages, including lost profits arising from the use or trust in this document (even if Alibaba Cloud has been notified of the possibility of such a loss).

5. By law, all the contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of this document shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates.

6. Please directly contact Alibaba Cloud for any errors of this document.

Document conventions

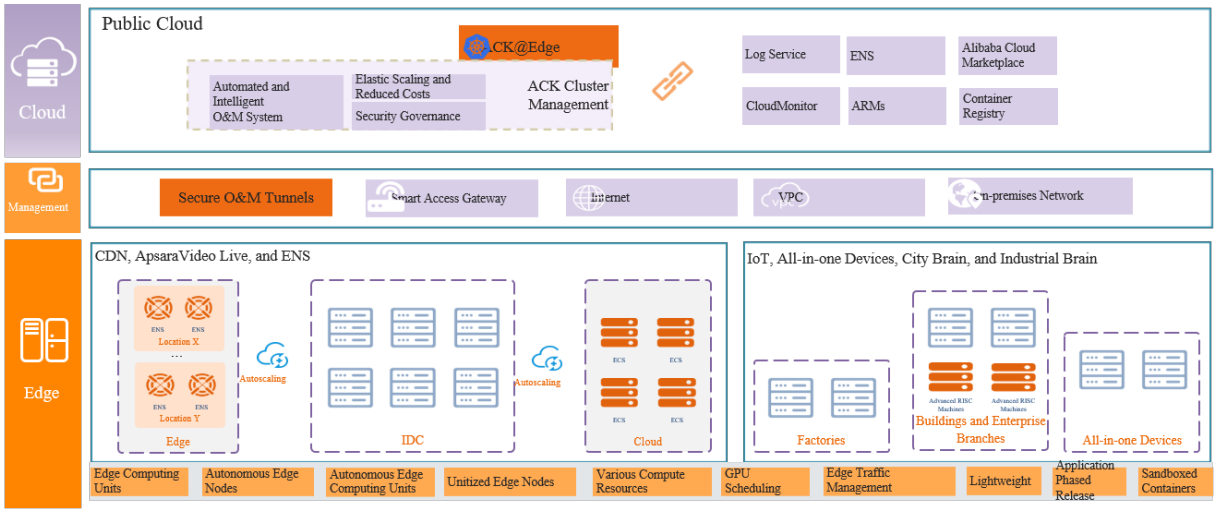
Style	Description	Example
Danger	A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	**Danger:** Resetting will result in the loss of user configuration data.
Warning	A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	**Warning:** Restarting will cause business interruption. About 10 minutes are required to restart an instance.
Notice	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	**Notice:** If the weight is set to 0, the server no longer receives new requests.
Note	A note indicates supplemental instructions, best practices, tips, and other content.	**Note:** You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click **Settings**> **Network**> **Set network type**.
Bold	Bold formatting is used for buttons , menus, page names, and other UI elements.	Click **OK**.
`Courier font`	Courier font is used for commands	Run the `cd /d C:/window` command to enter the Windows system folder.
Italic	Italic formatting is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	This format is used for an optional value, where only one item can be selected.	`ipconfig [-all|-t]`
{} or {a|b}	This format is used for a required value, where only one item can be selected.	`switch {active|stand}`

Table of Contents

1.ACK@Edge overview

Container Service for Kubernetes (ACK)@Edge (ACK@Edge) is a cloud-managed solution provided by Alibaba Cloud to coordinate cloud and edge computing. This topic describes the background and features of managed edge Kubernetes clusters.

Features



Managed edge Kubernetes clusters provide the following features to support the lifecycle management of containerized applications and resources in edge computing scenarios:

- Allows you to quickly create highly available edge Kubernetes clusters and provides the lifecycle management of edge Kubernetes clusters, such as scaling, upgrading, logging, and monitoring. You can perform the preceding operations in the ACK console.
- Supports access to various heterogeneous resources, such as data centers, IoT devices, and resources that are based on the x86 and ARM architectures. Hybrid scheduling of heterogeneous resources is also supported.
- Supports node autonomy and network autonomy to ensure the reliability of edge nodes and services in edge computing scenarios where the network connection is weak.
- Supports reverse tunneling for the management and maintenance of edge nodes.
- Provides cell-based management, cell-based deployment, and cell-based traffic management at the edge.

Related information

- Create a managed edge Kubernetes cluster
- Upgrade an edge cluster
- Expand an edge cluster

2.ACK@Edge billing

ACK@Edge is in public preview and is free for use.

3.Release notes on ACK@Edge

3.1. Release notes for ACK@Edge of Kubernetes 1.20

ACK@Edge is a cloud-managed solution provided by Container Service for Kubernetes (ACK). You can use ACK@Edge to achieve collaborative cloud-edge computing. This topic describes the release notes for ACK@Edge of Kubernetes 1.20.

Autonomy of edge nodes

The stability of components that are used to manage edge node autonomy is improved. The following section lists the major changes:

- **Health checks are enhanced**: The issue that **edge-hub** keeps sending heartbeats after kubelet is stopped is fixed.
- **Management of node certificates is enhanced**: If the **edge-hub** certificate is not deleted after a node is disconnected from a cluster, the **edge-hub** certificate is automatically updated after the node is connected to another cluster.
- **Traffic statistics on edge nodes are optimized**: You can view traffic statistics about requests on edge nodes by accessing the endpoint of **edge-hub**: http://127.0.0.1:10267/metrics.
- **Stability is improved**: The issue that a data race occurs when edge nodes process a large number of concurrent requests is fixed.

Proxy for nodes in the cloud

The cloud-hub component is available for nodes in the cloud. This component serves as a proxy to interact with the API server on behalf of other components in the cloud. The cloud-hub component supports Service Topology, which ensures that only CoreDNS is used to resolve domain names for requests from the cloud.

Cloud-edge O&M channel

The performance of cloud-edge O&M channels is optimized. The following section lists the major changes:

- **Request forwarding is improved**:
  - Requests that are destined for `{nodeName:Port}` can be forwarded from the cloud to edge nodes.
  - Requests that are destined for the `localhost endpoints` on edge nodes can be forwarded from the cloud to edge nodes. This requires you to specify the `localhost-proxy-ports` field in the **edge-tunnel-server-cfg** ConfigMap.
- **The configurations related to ports other than ports 10250 and 10255 are optimized**: You can specify HTTP ports in the `http-proxy-ports` field or specify HTTPS ports in the `https-proxy-ports` field of the **edge-tunnel-server-cfg** ConfigMap based on the type of the **endpoint** that is used by an edge node. The `dnat-ports-pair` field is retained. However, we recommend that you do not use dnat-ports-pair.
- Certificate management is improved for **edge-tunnel-server**. When the IP address of the **edge-**

tunnel-server-svc Service is changed, the **tls server** certificate of **edge-tunnel-server** is automatically updated. For example, the certificate is automatically updated when the edge-tunnel-server-svc Service is associated with a new Server Load Balancer (SLB) instance.

Network plug-ins of containers

The Flannel network plug-in is optimized. The following section lists the major changes:

- Flannel is updated to v0.13.1.3-6838863-aliyun.
- Flannel is optimized for intercommunication between the cloud and edge nodes when enhanced edge node pools are used or when the networks of the cloud and edge nodes are connected.

CoreDNS

CoreDNS is updated to v1.8.4. For more information, see CoreDNS.

Container runtimes

By default, containerd is used as the runtime and is updated to v1.4.8. For more information, see Release notes for containerd.

Add edge nodes to a cluster

The procedure for adding edge nodes to a cluster is optimized and new configurations are available. The new configurations allow you to connect Elastic Compute Service (ECS) instances to a cluster over an internal network. For more information, see Add an edge node.

3.2. Release notes for ACK@Edge of Kubernetes 1.18

ACK@Edge is a cloud-managed solution provided by Container Service for Kubernetes (ACK). You can use ACK@Edge to achieve collaborative cloud-edge computing. This topic lists the latest changes to ACK@Edge of Kubernetes 1.18.

Cloud-edge O&M channel and O&M monitoring

The cloud-edge O&M channel and O&M monitoring features are optimized:

- **tunnel-server** intercepts and handles the traffic of edge O&M and monitoring based on cluster DNS resolutions instead of the **iptables** rules of individual nodes.
- Monitoring components that depend on the cloud-edge O&M channel, such as **metrics-server** and **prometheus**, are no longer required to be deployed on the same node as **tunnel-server**.
- **tunnel-server** can be deployed on multiple pod replicas and support load balancing among all nodes.
- The `meta server` module is added to the cloud-edge O&M channel. This module is used to handle `Prometheus metrics` and `debug/pprof`. The endpoint of **tunnel-server** is `http://127.0.0.1:10265`. The endpoint of **edge-tunnel-agent** is `http://127.0.0.1:10266`. You can change the port in an endpoint by setting the --meta-port startup parameter of a component.

Autonomy of edge nodes

Edge caching, health checks, service endpoints, and traffic analysis are optimized. Edge traffic autonomy is enhanced. Access from edge applications to **kube-apiserver** in InCluster mode is enhanced. The following section describes the improvements:

- Traffic topology of Services at the edge is supported by **edge-hub** and is no longer dependent on Kubernetes feature gates.
- The endpoint of a `Service` at the edge is automatically changed by **edge-hub** to the public endpoint of **kube-apiserver** of the cluster. This allows applications at the edge to access the cluster in InCluster mode.
- CustomResourceDefinitions (CRDs) can be cached by **edge-hub**. For example, the `nodenetworkconfigurations` CRD can be cached. This CRD is used to store network information for Flannel.
- Health checks in the cloud are improved by **edge-hub**. During health checks, `Lease` heartbeats instead of `healthz` requests are sent.
- Port `10261` and port `10267` are listened on by **edge-hub**. Port `10261` is used to forward requests. Port `10267` is used to handle local requests sent to **edge-hub**, such as `liveness probes` , `metrics` , and `pprof` that are sent to `edge-hub` .
- The node_edge_hub_proxy_traffic_collector `metric` is supported by **edge-hub**. This metric shows the traffic generated when components of edge nodes such as **kubelet** and **kube-proxy** access Kubernetes resources, such as pods and Deployments.

Cell-based management at the edge

The Patch field is supported in cell-based management (based on the UnitedDeployment controller) at the edge. This field allows you to customize the configurations of each node pool. For example, you want to deploy nodes in different node pools in a deployment cell by using different local image repositories. In this case, you can specify an image address for each node pool by using the Patch field.

Add edge nodes to a cluster

Nodes that run the Ubuntu 20.04 operating system can be added to edge Kubernetes clusters.

Edge network

- The cloud-edge network built by using Flannel is optimized. **List operations and watch operations** are no longer performed on nodes. Instead, **list operations and watch operations** are performed on related CRDs. This reduces the traffic generated by these operations.
- Annotations about traffic management at the edge

- The following table describes the **keys** of annotations supported by Kubernetes 1.16 for traffic management at the edge.

Annotation Key	Annotation Value	Description
openyurt.io/topologyKeys	kubernetes.io/hostname	Specifies that the Service can be accessed by only the node on which the Service is deployed.
openyurt.io/topologyKeys	kubernetes.io/zone	Specifies that the Service can be accessed by only the nodes in the node pool where the Service is deployed.
None	None	Specifies that access to the Service is unlimited.

- In Kubernetes 1.18, the valid **values** of openyout.io/topologyKeys are modified. Valid values: `kubernetes.io/zone` and `openyurt.io/nodepool`. These values specify that the Service can be accessed by only nodes in the node pool where the Service is deployed. We recommend that you set the value to `openyurt.io/nodepool`.

3.3. Release notes on ACK@Edge of Kubernetes 1.16

ACK@Edge is a cloud-managed solution that is provided by Container Service for Kubernetes (ACK) to implement collaborative cloud-edge computing. This topic lists the latest changes to ACK@Edge of Kubernetes 1.16.

Kubernetes Core

The following describes the changes made to ACK@Edge of Kubernetes 1.16 in edge computing scenarios:

- Fixes the issue that **kubelet** fails to start when more than four records are stored in the *cpuacct.stat* file of a node.
- Kube-proxy supports the IP Virtual Server (IPVS) mode.
- You can use **kubelet** to configure the internal IP address of a node by specifying the name of a network interface controller (NIC).

For more information about the release notes on ACK, see Kubernetes 1.16 release notes.

Autonomy of edge nodes

ACK@Edge of Kubernetes 1.16 enhances the stability of edge node autonomy. The following lists the major changes:

- If the cached data is lost, clients receive the HTTP status code 404 instead of an empty string.
- The directory that is used to store the certificate of **edge-hub** changes from */etc/kubernetes/edge-hub* to */var/lib/edge-hub*.
- The certificate name of **edge-hub** changes from **edge-hub.kubeconfig** to **edge-hub.conf**,

bootstrap-edge-hub-current.conf --> bootstrap-hub.conf.

- An interface is added for **prometheus metrics**.
- The performance of **iptables** is improved. **iptables notrack** is added for IP addresses 127.0.0.1:10261 and 169.254.2.1:10261.

For more information, see Network autonomy of edge nodes.

Cloud-edge tunnels

ACK@Edge of Kubernetes 1.16 optimizes the performance of cloud-edge tunnels. The following lists the major changes:

- The tunneling protocol changes from TCP to **gRPC**. Compared with TCP, the size of data transmitted over **gRPC** tunnels is reduced by 40%.
- The **edge-tunnel-agent** component can automatically apply for and update certificates. This decouples the component from node certificates. In addition, the certificate of **edge-tunnel-agent** is stored in the */var/lib/edge-tunnel-agent/pki* directory.
- **prometheus metrics** are added.
- The label that is used to deploy the pod for **edge-tunnel-agent** is changed to `alibabacloud.com/is-edge-worker: "true"`.

For more information, see Cloud-edge tunneling.

Monitor components

ACK@Edge of Kubernetes 1.16 upgrades the **metrics-server** component and reinforces the security of this component. The following lists the major changes:

- **metrics-server** is upgraded from V0.2.1 to V0.3.8.
- ACK@Edge can be connected to Cloud Monitor by using tokens.

Cell-based management at the edge

ACK@Edge of Kubernetes 1.16 provides a new component **yurt-app-manager** and supports cell-based management at the edge. The following describes the main features of cell-based management at the edge:

- Manage nodes by node pool.
- Manage applications by using the UnitedDeployment controller.
- Configure a Service topology to expose a Service to only the node or node pool where the Service is deployed.

For more information about cell-based management at the edge, see Overview of edge node pools.

Enhanced node pools

Enhanced node pools are supported by ACK@Edge of Kubernetes 1.16. An enhanced node pool has the following features:

- Allows you to establish more stable and secure tunnels between the cloud and enhanced node pool.
- Allows applications in on-premises networks at the edge to communicate with applications in the cloud by using container networking.

For more information, see Create an enhanced edge node pool.

Container runtimes

The following lists the major changes to container runtimes in ACK@Edge of Kubernetes 1.16:

- The runC version of Advanced RISC Machine (ARM) and ARM64 is upgraded to 1.0.0-rc10.
- Cgroupfs cgroup driver is changed to Systemd cgroup driver.

CNI plug-in

ACK@Edge of Kubernetes 1.16 enhances the stability of the Container Network Interface (CNI) plug-in. This fixes the issue that pods with the same name in different namespaces may be allocated invalid IP addresses.

Add edge nodes to a cluster

ACK@Edge of Kubernetes 1.16 optimizes the procedure of adding edge nodes to a cluster and adds some parameters. The following lists the major changes:

- The procedure of adding edge nodes to a cluster is optimized and Classless Inter-Domain Routing (CIDR) conflict check is supported.
- The number of IP addresses that can be assigned to nodes is configurable.
- Parameters such as labels, nodeIface, annotations, and taints are added.
- The Linux 5.4 kernel released by Ubuntu is supported.

For more information, see Add an edge node.

API changes

You can call the node pool API operations to manage edge node pools. For more information, see Node pools.

4.Edge cluster management

4.1. Create a managed edge Kubernetes cluster

Managed edge Kubernetes clusters are designed to bring cloud computing to terminal devices at the edge. Managed edge Kubernetes clusters can be created, managed, and maintained in the Container Service for Kubernetes (ACK) console. ACK is a platform that integrates cloud computing with edge computing on top of the edge computing infrastructure. This topic describes how to create a managed edge Kubernetes cluster in the ACK console.

Prerequisites

ACK, Auto Scaling, and Resource Access Management (RAM) are activated.

ACK is activated in the ACK console. RAM is activated in the RAM console. Auto Scaling is activated in the Auto Scaling console.

> **Note**
>
> - Server Load Balancer (SLB) instances that are created along with an ACK cluster support only the pay-as-you-go billing method.
> - ACK clusters support only virtual private clouds (VPCs).
> - By default, each account has specific quotas on cloud resources that can be created. You cannot create clusters if the quota is reached. Make sure that you have sufficient resource quotas before you create a cluster. To request a quota increase, submit a ticket.
>   - By default, you can create at most 100 security groups with each account.
>   - By default, you can create at most 60 pay-as-you-go SLB instances with each account.
>   - By default, you can create at most 20 elastic IP addresses (EIPs) with each account.

Context

The sharp growth of smart devices connected to the Internet and the needs to deploy business and process data at edges have significant impacts on edge computing services. To meet these requirements, a variety of new edge computing services have emerged, such as edge intelligence, real-time edge computing, and edge analytics. Traditional cloud computing platforms provide computing and storage services in the cloud. However, this no longer meets the requirements of edge devices for time-efficient computing, larger storage capacity, and enhanced computing capacity. To meet these requirements, ACK provides managed edge Kubernetes clusters to coordinate services in the cloud and edges. A managed edge Kubernetes cluster is a standard, secure, and highly-available Kubernetes cluster deployed in the cloud. This type of cluster is integrated with features of Alibaba Cloud, such as virtualization, storage, networking, and security. This simplifies how to manage and maintain clusters and allows you to focus on your business development. In addition, quick access to a variety of heterogeneous edge computing services is supported at the edges. The cloud control center allows you to manage edge devices, such as IoT gateway devices, terminals, Content Delivery Network (CDN) resources, and data centers. The X86 and ARM architectures are supported. Managed edge Kubernetes clusters have been used in various fields, such as edge intelligence, intelligent buildings, intelligent

factories, audio and video live streaming, online education, and CDN.

Procedure

1.
2.
3.
4. Click the **Managed Edge Kubernetes** tab and configure the cluster.

   Configure basic settings of the cluster.

Parameter	Description
	Select **Standard edition** to create a managed edge Kubernetes cluster.
Region	
Resource Group	
Kubernetes Version	The supported Kubernetes versions are displayed.
Containerd Runtime	The supported containerd runtime are Docker, Containerd, and Sandboxed-Container. For more information about containerd, see Comparison of Docker, containerd, and Sandboxed-Container.
VPC	
VSwitch	
IP Addresses per Node	
Pod CIDR Block	The CIDR blocks specified by **Pod CIDR Block** and **Service CIDR** cannot overlap with the CIDR block of the VPC and the CIDR blocks of the existing ACK clusters in the VPC. You cannot modify the CIDR blocks after the cluster is created. The Service CIDR block cannot overlap with the Pod CIDR block. For more information about subnetting for ACK clusters, see Plan CIDR blocks for an ACK cluster.
Service CIDR	
Configure SNAT	
Access to API Server	**Note** Edge nodes interact with the API server in the cloud over the Internet. If you clear **Expose API Server with EIP**, the edge nodes cannot connect to the cluster in the cloud. As a result, the created cluster cannot be used in edge computing scenarios.
RDS Whitelist	

Parameter	Description
Security Group	
Deletion Protection	

Configure advanced settings of the cluster.

Parameter	Description
Labels	

5. Click **Next: Worker Configurations** to configure worker nodes.

> **Note** In a managed edge Kubernetes cluster, you must configure at least one worker node to deploy components.

Parameter	Description
Instance Type	**Note** To use advanced features such as logging, monitoring, and reverse tunneling, you must deploy the related components in the cloud. Therefore, you must create at least one Elastic Compute Service (ECS) instance as a worker node.
Selected Types	
Quantity	
System Disk	Configure the system disks of worker nodes. Standard SSDs and ultra disks are supported. **Note**
Mount Data Disk	
Logon Type	**Note** You must set the logon type if you select **Install CloudMonitor Agent on ECS Instance** or **Enable Log Service**.
Key Pair	

6. Click **Next: Component Configurations** to configure components.

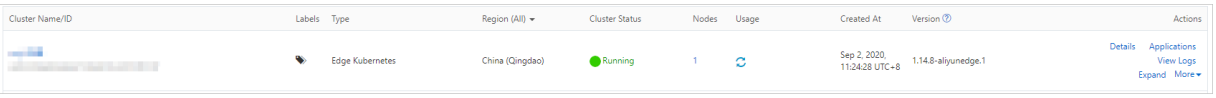
Parameter	Description
CloudMonitor Agent	Select whether to install the CloudMonitor agent. If you select **Install the CloudMonitor Agent on ECS Nodes**, you can view monitoring data about the nodes in the CloudMonitor console.
Log Service	

7. Click **Next: Confirm Order**.
8. Read Terms of Service, select the check box, and then click **Create Cluster**.

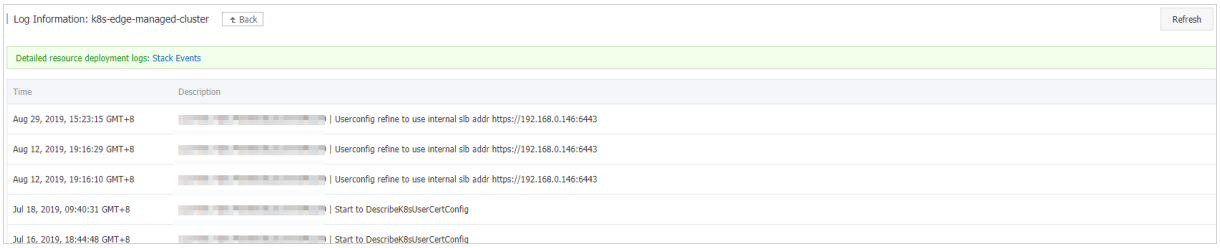
> **Note** It requires about 10 minutes to create a managed edge Kubernetes cluster.

Result

After the cluster is created, you can view the created cluster on the **Clusters** page in the ACK console.



Click **View Logs** in the Actions column. On the **Log Information** page, you can view the cluster log. To view detailed log information, click **Stack events**.



On the **Clusters** page, find the newly created cluster and click **Details** in the **Actions** column. On the details page of the cluster, you can click the **Basic Information** tab to view basic information about the cluster and click the **Connection Information** tab to view information about how to connect to the cluster.

The following information is displayed:

- **API Server Public Endpoint**: the IP address and port that the API server uses to provide services over the Internet. It allows you to manage the cluster by using kubectl or other tools on your terminal.
- **API Server Internal Endpoint**: the IP address and port that the API server uses to provide services within the cluster. The IP address belongs to the SLB instance that is bound to the cluster.
- **Testing Domain**: the domain name that is used to test Services. The suffix of the domain name is `<cluster_id>.<region_id>.alicontainer.com`.

> **Note** To remap the domain name, click **Rebind Domain Name**.

You can Connect to ACK clusters by using kubectl and run the `kubectl get node` command to view information about the nodes in the cluster.

```
Type "kubectl" to manage your kubenetes cluster [redacted]

shell@Alicloud:~$ kubectl get node
NAME                                STATUS   ROLES    AGE   VERSION
cn-beijing.i-2zehvxttbua2aw800uin   Ready    <none>   11m   v1.12.6-aliyun.1
shell@Alicloud:~$
```

Related information

- ACK@Edge overview
- Upgrade an edge cluster
- Add an edge node

4.2. Upgrade an edge cluster

This topic describes how to upgrade the Kubernetes version of your edge cluster in the Container Service for Kubernetes (ACK) console. The cluster upgrade process involves three phases: pre-check, master node upgrade, and worker node upgrade. When you upgrade a dedicated Kubernetes cluster, the serial number of the master node that is being upgraded is displayed. When the worker nodes are upgraded, the number of upgraded worker nodes and the total number of worker nodes are displayed.

Prerequisites

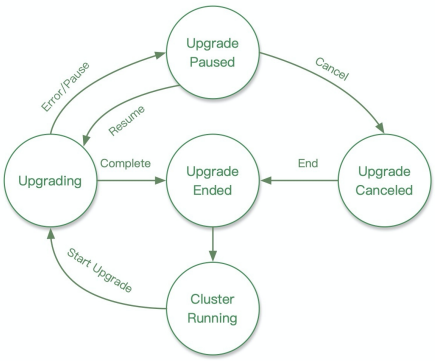
- An edge cluster is created. For more information, see Create a managed edge Kubernetes cluster.
- Docker is installed on your on-premises machine. For more information, see Install Docker.

Context

You can go to the **Clusters** page in the ACK console to check the Kubernetes version of your edge cluster and check whether a new version is available for upgrade.

How the upgrade works

The following content describes how the upgrade works and the steps that are involved in the upgrade process.



- Upgrade policy

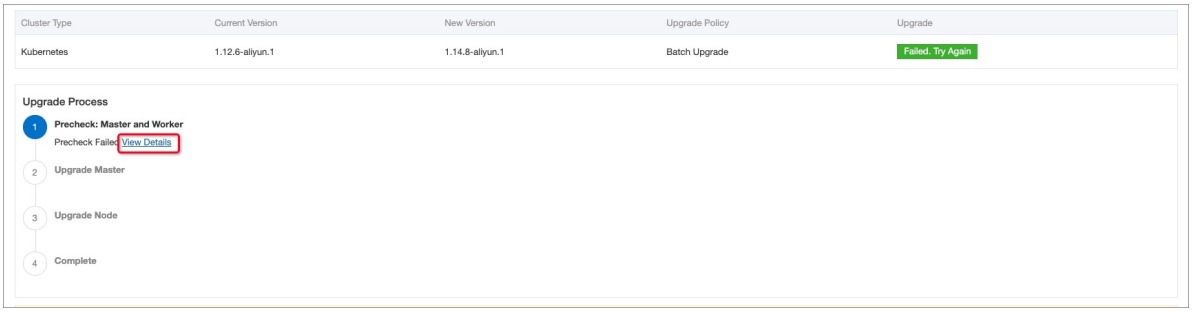
The upgrade policy defines how the upgrade is implemented. The default policy is batch upgrade. Batch upgrade is performed during the **worker node upgrade** phase to upgrade worker nodes in batches. The cluster is upgraded in batches:

- The first batch includes one node and the number of nodes increases by the power of 2 in batches thereafter. If you resume a paused upgrade, the first batch after the pause includes one node and the number of nodes increases by the power of 2 in batches thereafter.
- The maximum number of nodes in each batch does not exceed 10% of the total number of nodes.

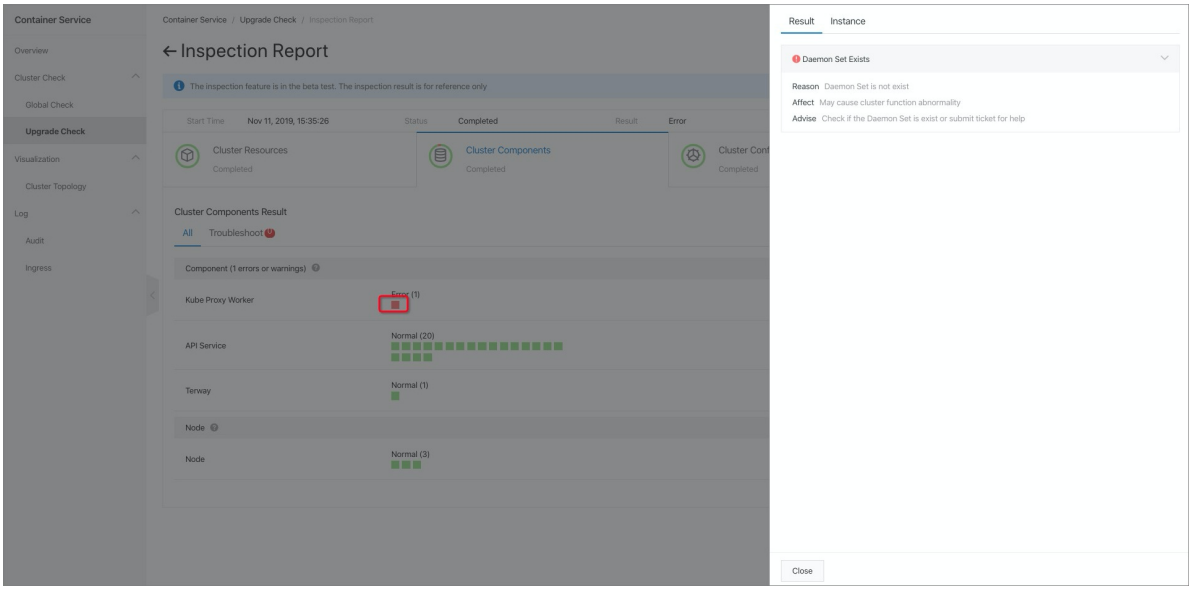
- Pre-check

  When you start the upgrade process, a pre-check is automatically run to detect possible issues that may affect the upgrade. The pre-check includes multiple check items to ensure that the upgrade can be completed without disruptions.

  If your cluster contains configuration errors or potential risks, the pre-check fails, as shown in the following figure.



  Click **View Details**. On the details page that appears, you can view the cause of the failure.



- Pause the upgrade

  You can pause the upgrade process at any time during the upgrade.

> **Note**
> - After you pause the upgrade, the upgrade will be completed on nodes where the upgrade has already started. The upgrade will not be performed on nodes where the upgrade has not started.
> - We recommend that you resume and complete a paused upgrade at your earliest convenience and do not perform operations on the cluster when the upgrade is paused.

After the upgrade is paused, you can click **Continue** to resume the upgrade process.

If an error occurs during the upgrade, the system automatically pauses the upgrade process. The cause of the error will be displayed at the bottom of the page.

- Cancel the upgrade

  After the upgrade is paused, you can click **Cancel** to cancel the upgrade.

> **Note**
> - After you cancel the upgrade, the upgrade will be completed on nodes where the upgrade has already started. The upgrade will not be performed on nodes where the upgrade has not started.
> - You cannot cancel the upgrade on the nodes where the upgrade has been completed.

Notes

- To upgrade a cluster, nodes in the cluster must have Internet access so that they can download the upgrade packages.
- Applications that run in the cluster are not interrupted during the upgrade. Applications that are strongly reliant on the API server may be interrupted temporarily.
- If you change the cluster configurations during the upgrade, for example, creating SWAP partitions, errors may occur during the upgrade process.
- The upgrade is performed in batches. You can pause the upgrade after a batch is upgraded. We recommend that you resume and complete a paused upgrade at your earliest convenience and do not perform operations on the cluster when the upgrade is paused. If the upgrade has been paused for more than **15 days**, it will be automatically canceled and related events and log information are deleted.
- During the upgrade process, do not modify the resources in the kube-upgrade namespace unless an error has occurred.
- If an error occurs during the upgrade, the upgrade will be paused. You need to troubleshoot the error and delete the failed pods in the kube-upgrade namespace. After the error is fixed, you can resume the upgrade. You can contact the Alibaba Cloud technical support team for assistance.

Preparations

> **Note** If the cluster that you want to upgrade is not deployed in the production environment, we recommend that you check whether the cluster meets the upgrade requirements before you start the upgrade in the production environment.

Before you upgrade a cluster, you must check the health status of the cluster to make sure that the cluster meets the upgrade requirements.

1.
2. Pre-upgrades cluster nodes.
   i.
   ii. On the **Clusters** page, find the cluster that you want to upgrade and click **Details** in the **Actions** column.
   iii. On the **details** page of the cluster, click the **Connection Information** tab. Click the **Public Access** tab and copy the content of the **kubeconfig** file of the cluster to the *$HOME/.kube/config* file in your on-premises machine.
   iv. Run the following command on your on-premises machine.

```
docker run -it -v ~/.kube:/root/.kube registry.cn-hangzhou.aliyuncs.com/edge-kubernetes/node-preprocess:v0.1.0 [Nodes in the cloud]
```

> **Note** You must specify nodes in the cloud in this command. This parameter is not required if your cluster does not have a node in the cloud.

3.
4. Find the cluster that you want to upgrade and choose **More > Cluster Check** in the **Actions** column.
5. In the left-side navigation pane of **Container Service Operation Center**, choose **Cluster Check > Upgrade Check**.
6. On the **Upgrade Check** page, click **Start**.
7. In the **Upgrade Check** panel, select the check box under **Warning** and click **Start**.
   After the upgrade check is complete, click **Details**.

   If the **result** shows **normal** in the report, it indicates that the cluster passes the check and you can perform upgrade operations.

If errors are found in the cluster, you must fix the errors before you can upgrade the cluster. You can also submit a ticket for assistance.

Procedure

1.
2.
3.
4. On the **Clusters** page, find the cluster that you want to upgrade and choose **More > Upgrade Cluster** in the **Actions** column.
5. In the Upgrade Cluster dialog box, click **Confirm**.

   You can view the progress of the upgrade.

After the upgrade is complete, you can go to the Clusters page and check the current Kubernetes version of your edge cluster.

Related information

- ACK@Edge overview
- Create a managed edge Kubernetes cluster

4.3. Upgrade components in an edge Kubernetes cluster

This topic describes how to upgrade the components in an edge Kubernetes cluster. This allows you to perform fine-grained version upgrades for the components in a cluster of the latest Kubernetes version.

Prerequisites

- An edge Kubernetes cluster is created. For more information, see Create a managed edge Kubernetes cluster.
- Docker is installed on your on-premises machine. For more information, see Install Docker.

Procedure

1.
2.
3. On the **Clusters** page, find the cluster that you want to manage and choose **More > Manage System Components** in the **Actions** column.
4.
5. (Optional)Upgrade the **edge-tunnel-server** and **edge-tunnel-agent** components.

   > **Notice** If the Kubernetes version of your cluster is 1.12.6-aliyunedge.1, you must perform the following steps to upgrade **edge-tunnel-server** and **edge-tunnel-agent**.

   i. Manually delete the DaemonSets, Deployments, and Services that are related to the **frps or frpc** component. To do this, perform the following steps:

      a. In the left-side navigation pane, click **Clusters**.

      b. On the **Clusters** page, click the name of the cluster that you want to manage or click **Details** in the **Actions** column.

      c. On the **details** page of the cluster, click the **Connection Information** tab. Click the **Public Access** tab and copy the content of the **kubeconfig** file to the *$HOME/.kube/config* file of your on-premises machine.

      d. Run the following command:

      ```
      docker run -v ~/.kube:/root/.kube registry.cn-hangzhou.aliyuncs.com/acs/edge-upgrade-addon:v1.0 tunnel
      ```

   ii. On the **Add-ons** page, find **edge-tunnel-server** and click **Upgrade** in the **Actions** column.

   iii. On the **Add-ons** page, find the **edge-tunnel-agent** component and click **Upgrade** in the **Actions** column.

Related information

- Upgrade an edge cluster

4.4. Expand an edge cluster

To expand an edge cluster, you can add edge nodes to the cluster in the Container Service for Kubernetes (ACK) console. In earlier versions, if you want to expand an edge cluster, you must purchase edge nodes and then add these nodes to the cluster. You can now purchase edge nodes on the cluster expansion page. Then, the purchased nodes are automatically added to the cluster. This topic describes how to expand an edge cluster by adding edge nodes of Edge Node Service (ENS) or by adding Elastic Compute Service (ECS) nodes.

Prerequisites

- A managed edge cluster is created.
- ENS is activated in the ENS console.

Procedure

1.
2.
3. On the **Clusters** page, find the managed edge cluster that you want to expand and click **Expand** in the **Actions** column.
4. On the **Expand** page, perform the following steps to expand the cluster.

   To expand an edge cluster, you can **add ENS nodes** or **add ECS nodes** in the ACK console.

   - **Add ENS nodes**

     To expand an edge cluster, you can add edge nodes to the cluster in the ACK console. Click the **Add ENS Node** tab, and then configure the edge nodes to be added.

Parameter	Description
Cluster Name	The name of the managed edge cluster.
Edge Node Type	Select the region where the edge nodes to be added are deployed.
Instance Type	Select the instance types of the edge nodes to be added. You can select multiple instance types.
Existing Worker Nodes	The number of existing worker nodes in the cluster.
Nodes to Add	Select the number of worker nodes to be added.
System Disk	Select the system disk for the nodes. The minimum size is 20 GiB. The default disk type is basic disk.
Data Disk	Specify whether to mount data disks.
Image	The default operating system is **centos_7_04_64_20G_alibase_20171211**.

Parameter	Description
Internet Bandwidth Billing Method	ENS is billed based on the fourth peak bandwidth of each month by default. If your monthly bandwidth usage is higher than 10 Gbit/s, we recommend that you contact the business manager to request the monthly 95th percentile billing method.
Password	Enter a password for logging on to the nodes.
Confirm Password	Enter the password again.
Billing Method	Only the subscription billing method is supported.
Duration	Select the subscription duration. You can select 1, 2, 3, 6, or 12 months.
Auto Renewal	Specify whether to automatically renew the subscription of the nodes.

- **Add ECS nodes**

  To expand an edge cluster, you can also add ECS nodes to the cluster in the ACK console. Click the **Add ECS Node** tab and configure the ECS nodes to be added. For more information about how to **add ECS nodes**, see Parameters for cluster expansion.

5. On the right side of the **Expand** page, click **Submit**.
6. On the **Confirm** page, select the check box after you read the terms of service, and click **OK**.

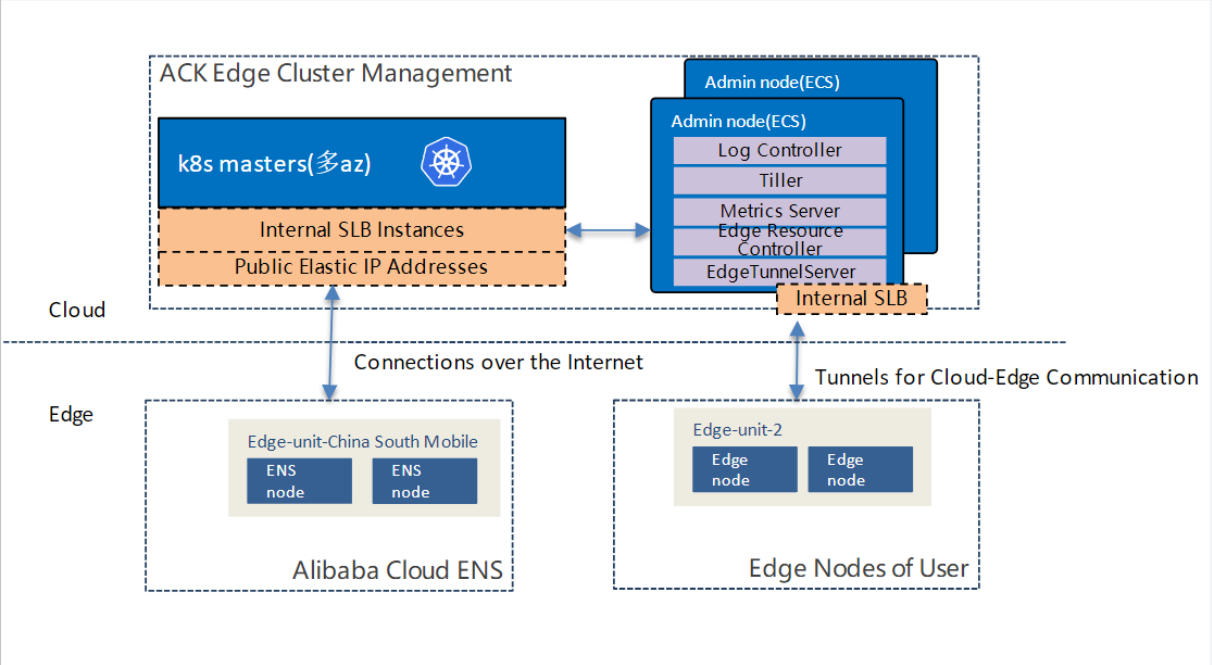
Result

Go to the **Clusters** page, you can find that the cluster changes to the **Scaling state**. It indicates that the expansion is in progress. When the cluster changes to the **Running state**, the expansion is complete.

4.5. ECS instances in an edge Kubernetes cluster

A managed edge Kubernetes cluster must contain at least one Elastic Compute Service (ECS) instance. This topic describes how ECS instances work and how to add or remove ECS instances.

Cloud management nodes in edge computing



When you create a managed edge cluster, ACK@Edge automatically creates at least one ECS instance as a cloud management node for the cluster. This ECS instance is used to run cloud management applications. You can deploy custom cloud management applications. The taint **node-role.alibabacloud.com/addon: Effect: NoSchedule** is automatically added to a cloud management node to prevent edge workloads from being scheduled to the node. By default, a management node of version 1.14.8-aliyunedge.1 or earlier has the following management applications:

- alibaba-log-controller: the Log Service controller.
- alicloud-monitor-controller: the CloudMonitor controller.
- metric-server: the server that is used to monitor the cluster.
- edge-tunnel-server: the server for the reverse O&M tunnel. It allows you to use the native Kubernetes API to access edge nodes, monitor containers, and use SSH to remotely run commands.

Deploy applications to cloud management nodes

If you want to deploy custom management applications on cloud management nodes, for example, to deploy different types of operators, you must specify a toleration that matches the preceding taint and configure node selectors. The following example describes how to set the parameters.

```
...
 nodeSelector:
   alibabacloud.com/is-edge-worker: 'false'
   beta.kubernetes.io/arch: amd64
   beta.kubernetes.io/os: linux
 tolerations:
   - effect: NoSchedule
     key: node-role.alibabacloud.com/addon
     operator: Exists
 ...
```

Add a cloud management node

To add a cloud management node, perform the following steps. ACK@Edge will support ECS-based auto scaling in later versions.

1. Purchase an ECS instance in the virtual private cloud (VPC) where the cluster is deployed.

   For more information about how to purchase an ECS instance, see Create an ECS instance.

   > **Note** Set the operating system to CentOS 7.6.

2. Call AttachInstances in OpenAPI Explorer to add the ECS instance to the cluster.

   The following example shows a request body:

```
{
  "password": "Helloxxxx!",
  "tags":[],
  "instances": [
    "i-uf65mbpn1x8xxxxxx"
  ]
}
```

Parameter	Description
password	The password that is used to log on to the ECS instance. The password must be 8 to 30 characters in length and must contain at least three of the following character types: uppercase letters, lowercase letters, digits, and special characters.
tags	The tags of the nodes. ○ key: the name of the tag ○ value: the value of the tag.
instances	An array of the existing instances.

   For more information about the API operation, see Add existing instances to a node pool.

Related information

- ACK@Edge overview
- Create a managed edge Kubernetes cluster

5.ACK@Edge Pro edition cluster

5.1. Introduction to professional edge Kubernetes clusters

Professional edge Kubernetes clusters offer higher reliability and security than standard edge Kubernetes clusters in large-scale production environments for enterprise users. In addition, professional edge Kubernetes clusters are covered by the service level agreement (SLA) that supports compensation clauses.

Professional edge Kubernetes clusters offer all benefits of standard edge Kubernetes clusters. For example, master nodes are highly available and managed by Container Service for Kubernetes (ACK). In addition, professional edge Kubernetes clusters provide higher reliability, security, and schedulability, and are covered by SLA terms for compensation. Professional edge Kubernetes clusters are suitable for enterprises that require higher security and stability for large-scale business deployed in a production environment.

Scenarios

- Internet enterprises. These enterprises deploy their business on a large scale and require business management with high stability, security, and observability.
- Big data computing enterprises. These enterprises deploy large-scale data computing services, high-performance data processing services, and other services with high elasticity. These services require clusters with high stability, high performance, and efficient computing capabilities.
- International enterprises that run their business in China. These enterprises prioritize security and services that provide SLAs with compensation clauses.
- Financial enterprises. These enterprises require SLAs with compensation clauses.

Features

- Managed master nodes with high reliability: etcd is a reliable store for disaster recovery and data restoration. etcd uses cold backups and hot backups to ensure data availability for professional edge Kubernetes clusters. Key metrics are collected for you to gain insights from control components. This allows you to detect potential risks.
- Clusters with higher security: By default, etcd uses encrypted disks in the control plane. In the data plane, kms-plugin is installed to encrypt Kubernetes Secrets. ACK provides security management for this type of cluster. The advanced security management feature allows you to inspect containers in the Running state and enable auto repairing.
- More intelligent pod scheduling: kube-scheduler is integrated to provide better pod scheduling capability. This allows you to schedule pods in bulk, set multiple scheduling algorithms, and schedule pods to NPU-accelerated nodes. This also enhances the pod scheduling capability in scenarios where large-scale data computing or high-performance data processing is required.
- SLA guarantees: Professional edge Kubernetes clusters are covered by the SLA that supports compensation clauses. A level of 99.95% uptime is guaranteed for the cluster API server.

Comparison

The following table compares professional edge Kubernetes clusters with standard edge Kubernetes clusters.

Category	Feature	ACK@Edge	
		Professional edge Kubernetes cluster	Standard edge Kubernetes cluster
Cluster size	N/A	Up to 1,000 nodes.	Up to 10 nodes for each newly created cluster. Existing standard edge Kubernetes clusters can be upgraded to professional edge Kubernetes clusters.
SLA	N/A	99.95% (supports compensation).	99.90% (does not support compensation).
API Server	Custom parameter settings	✓	✗
	Availability monitoring	✓	✗
ETCD	High-frequency cold backups, high-frequency hot backups, and geo-disaster recovery	✓	✗
	Observability metrics	✓	✗
Security management	The advanced security management feature that supports data encryption. For more information, see Use KMS to encrypt Kubernetes Secrets.	✓	✗

Supported regions

- Asia Pacific

Region	City	Region ID
China (Beijing)	Beijing	cn-beijing
China (Zhangjiakou)	Zhangjiakou	cn-zhangjiakou
China (Hohhot)	Hohhot	cn-huhehaote
China (Ulanqab)	Ulanqab	cn-wulanchabu

Region	City	Region ID
China (Hangzhou)	Hangzhou	cn-hangzhou
China (Shanghai)	Shanghai	cn-shanghai
China (Shenzhen)	Shenzhen	cn-shenzhen
China (Heyuan)	Heyuan	cn-heyuan
China (Chengdu)	Chengdu	cn-chengdu
China (Hong Kong)	Hong Kong	cn-hongkong
Japan (Tokyo)	Tokyo	ap-northeast-1
Singapore (Singapore)	Singapore	ap-southeast-1
Australia (Sydney)	Sydney	ap-southeast-2
Malaysia (Kuala Lumpur)	Kuala Lumpur	ap-southeast-3
Indonesia (Jakarta)	Jakarta	ap-southeast-5

- Europe & Americas

Region	City	Region ID
US (Silicon Valley)	Silicon Valley	us-west-1
US (Virginia)	Virginia	us-east-1
UK (London)	London	eu-west-1
Germany (Frankfurt)	Frankfurt	eu-central-1

- India

Region	City	Region ID
India (Mumbai)	Mumbai	ap-south-1

5.2. Create a professional edge Kubernetes cluster

Professional edge Kubernetes clusters offer higher reliability, security, and schedulability than standard edge Kubernetes clusters. In addition, professional edge Kubernetes clusters are covered by the service level agreement (SLA) that supports compensation clauses. This type of cluster is suitable for enterprise users who require high stability and security for large-scale workloads. This topic describes how to create a professional edge Kubernetes cluster in the Container Service for Kubernetes (ACK) console.

Prerequisites

Resource Access Management (RAM) is activated in the RAM console. Auto Scaling is activated in the Auto Scaling console.

> Note
>
> -
> -
> - By default, each account has specific quotas on cloud resources that can be created. You cannot create clusters if the quota is reached. Make sure that you have sufficient resource quotas before you create a cluster.
>   - For more information about the maximum numbers of clusters and nodes that can be created with each account, see Limits.
>
>   > Note
>
>   -
>   -
>   -

Procedure

1.
2.
3.
4. On the **Managed Edge Kubernetes** tab, configure the professional edge Kubernetes cluster.

i. Configure basic settings of the cluster.

Parameter	Description
	Select **Professional** to create a professional edge Kubernetes cluster.
Kubernetes Version	The Kubernetes versions that are supported by professional edge Kubernetes clusters are displayed.
	You can select at most three vSwitches that are deployed in different **zones**. If no vSwitch is available, click **Create VSwitch** to create one. For more information, see Work with vSwitches.
	Note Edge nodes interact with the API server in the cloud over the Internet. If you clear **Expose API Server with EIP**, the edge nodes cannot connect to the cluster in the cloud. As a result, the created cluster cannot be used in edge computing scenarios.

ii. Configure advanced settings of the cluster.

Parameter	Description
Secret Encryption	If you select **Select Key**, you can use a key that is created in the Key Management Service (KMS) console to encrypt Kubernetes Secrets. For more information, see Use KMS to encrypt Kubernetes Secrets.

5. Click **Next: Worker Configurations** to configure worker nodes.

> **Note** In a professional edge Kubernetes cluster, you must configure at least one worker node to deploy controllers.

Parameter	Description
	> **Note** To use advanced features such as logging, monitoring, and reverse tunneling, you must deploy the related components in the cloud. Therefore, you must create at least one Elastic Compute Service (ECS) instance as a worker node.
	> **Note** You must set the logon type if you select **Install CloudMonitor Agent on ECS Instance** or **Enable Log Service**.

6. Click **Next: Component Configurations** to configure components.

Parameter	Description
CloudMonitor Agent	Select whether to install the CloudMonitor agent. If you select **Install the CloudMonitor Agent on ECS Nodes**, you can view monitoring data about the nodes in the CloudMonitor console.

7. Click **Next: Confirm Order**.
8. Read **Terms of Service**, select the check box, and then click **Create Cluster**.

> **Note** It requires about 10 minutes to create a professional edge Kubernetes cluster that contains multiple nodes.

Result

- After the cluster is created, you can view the created cluster on the **Clusters** page in the ACK console.
- Click **View Logs** in the **Actions** column. On the **Log Information** page, you can view the cluster log. To view detailed log information, click **Stack events**.
- Click **Details** in the **Actions** column. On the details page of the cluster, click the **Basic Information** tab to view basic information about the cluster. You can also click the **Connection Information** tab to view information about how to connect to the cluster. The following information is displayed:

- **API Server Public Endpoint**: the IP address and port that the API server uses to provide services over the Internet. It allows you to manage the cluster by using kubectl or other tools on your terminal.
- **API Server Internal Endpoint**: the IP address and port that the API server uses to provide services within the cluster. The IP address belongs to the SLB instance that is bound to the cluster.
- **Testing Domain**: the domain name that is used to test Services. The suffix of the domain name is *<cluster_id>.<region_id>.alicontainer.com*.

> **Note** On the **Basic Information** tab, you can click **Rebind Domain Name** on the right side of **Testing Domain** to rebind the domain name.

5.3. CPU scheduling

5.3.1. Gang scheduling

Container Service for Kubernetes (ACK) provides the gang scheduling feature based on the new kube-scheduler framework. This feature provides a solution to job scheduling in the all-or-nothing scenario. This topic describes how to enable gang scheduling.

Prerequisites

- A professional managed Kubernetes cluster is created. For more information, see Create an ACK Pro cluster.

  > **Notice** Gang scheduling is available for only professional managed Kubernetes clusters. To enable gang scheduling for dedicated Kubernetes clusters, to add your account to the whitelist.

- The following table describes the system component versions that are required for topology-aware CPU scheduling.

Component	Required version
Kubernetes	V1.16 and later
Helm	V3.0 and later
Docker	19.03.5
Operating system	CentOS 7.6, CentOS 7.7, Ubuntu 16.04, Ubuntu 18.04, and Alibaba Cloud Linux 2.

Context

Gang scheduling is a scheduling algorithm that schedules all correlated processes to different processors in a parallel system and starts these processes simultaneously. Gang scheduling aims to start all correlated processes at the same time. This prevents the process group from being blocked when the system fails to start some processes. For example, if you submit a batch job that contains multiple tasks, either all of the tasks are scheduled or none of them is scheduled. Task scheduling in the all-or-nothing scenario is known as gang scheduling.

Kubernetes is widely used in online service orchestration. ACK wants to use Kubernetes as a platform for unified management of online services and offline jobs. This improves the resource utilization and performance of clusters. However, kube-scheduler cannot migrate specific offline workloads to Kubernetes clusters. For example, if a job requires all-or-nothing scheduling, all tasks of the job must be scheduled at the same time. If only some of the tasks are started, the started jobs must wait until all the remaining tasks are scheduled. If each submitted job contains unscheduled tasks, all submitted jobs remain in the Pending state and the cluster is deadlocked. To avoid this situation, you must enable gang scheduling for kube-scheduler.

Features

In ACK, a pod group is a group of pods that need to be scheduled at the same time. When you submit a job that requires all-or-nothing scheduling, you can add labels to pods. The labels specify the name of the pod group to which the job belongs and the minimum number of tasks that must be scheduled to run the job. kube-scheduler schedules tasks based on the minimum number of tasks that must be scheduled. The tasks are scheduled only when the cluster resources are sufficient to schedule the required number of tasks. Otherwise, the job remains in the Pending state.

How to enable gang scheduling

To enable gang scheduling, set min-available and name by adding labels to the pods.

```
labels:
    pod-group.scheduling.sigs.k8s.io/name: tf-smoke-gpu
    pod-group.scheduling.sigs.k8s.io/min-available: "3"
```

- name: Specifies the name of a pod group.
- min-available: Specifies the minimum number of pods that must be scheduled to run a job. Pods are scheduled only when the computing resources are sufficient to schedule the required number of pods.

> **Note** Pods in the same pod group must be assigned the same priority.

Examples

In this example, a distributed TensorFlow job is used to demonstrate how to enable gang scheduling. The ACK cluster that is used in this example has four GPUs.

1. Run the following command to use Arena of Kubeflow to deploy the environment in your ACK cluster to run the distributed TensorFlow job.

   > **Note** Arena is a subproject of Kubeflow. Kubeflow is an open source project for machine learning based on Kubernetes. Arena allows you to manage the lifecycle of machine learning jobs by using the command line interface or SDK. Lifecycle management includes environment setup, data preparation, model development, model training, and model prediction. This improves the working efficiency of data scientists.

```
git clone https://github.com/kubeflow/arena.git
kubectl create ns arena-system
kubectl create -f arena/kubernetes-artifacts/jobmon/jobmon-role.yaml
kubectl create -f arena/kubernetes-artifacts/tf-operator/tf-crd.yaml
kubectl create -f arena/kubernetes-artifacts/tf-operator/tf-operator.yaml
```

Run the following command to check whether the environment to run TensorFlow jobs is deployed. If the pods are in the Running state, it indicates that the environment is deployed.

```
kubectl  get pods -n arena-system
```

```
NAME                                READY   STATUS    RESTARTS   AGE
tf-job-dashboard-56cf48874f-gwlhv   1/1     Running   0          54s
tf-job-operator-66494d88fd-snm9m    1/1     Running   0          54s
```

2. Use the following template to submit a distributed TensorFlow job to the ACK cluster. The job runs on one parameter server (PS) pod and four worker pods. Each worker pod requires two GPUs.

```
apiVersion: "kubeflow.org/v1"
kind: "TFJob"
metadata:
  name: "tf-smoke-gpu"
spec:
  tfReplicaSpecs:
    PS:
      replicas: 1
      template:
        metadata:
          creationTimestamp: null
          labels:
            pod-group.scheduling.sigs.k8s.io/name: tf-smoke-gpu
            pod-group.scheduling.sigs.k8s.io/min-available: "5"
        spec:
          containers:
          - args:
            - python
            - tf_cnn_benchmarks.py
            - --batch_size=32
            - --model=resnet50
            - --variable_update=parameter_server
            - --flush_stdout=true
            - --num_gpus=1
            - --local_parameter_device=cpu
            - --device=cpu
            - --data_format=NHWC
            image: registry.cn-hangzhou.aliyuncs.com/kubeflow-images-public/tf-benchmar
ks-cpu:v20171202-bdab599-dirty-284af3
            name: tensorflow
            ports:
            - containerPort: 2222
              name: tfjob-port
            resources:
              limits:
                cpu: '1'
            workingDir: /opt/tf-benchmarks/scripts/tf_cnn_benchmarks
```

```
            workingDir: /opt/tf-benchmarks/scripts/tf_cnn_benchmarks
          restartPolicy: OnFailure
    Worker:
      replicas: 4
      template:
        metadata:
          creationTimestamp: null
          labels:
            pod-group.scheduling.sigs.k8s.io/name: tf-smoke-gpu
            pod-group.scheduling.sigs.k8s.io/min-available: "5"
        spec:
          containers:
          - args:
            - python
            - tf_cnn_benchmarks.py
            - --batch_size=32
            - --model=resnet50
            - --variable_update=parameter_server
            - --flush_stdout=true
            - --num_gpus=1
            - --local_parameter_device=cpu
            - --device=gpu
            - --data_format=NHWC
            image: registry.cn-hangzhou.aliyuncs.com/kubeflow-images-public/tf-benchmar
ks-gpu:v20171202-bdab599-dirty-284af3
            name: tensorflow
            ports:
            - containerPort: 2222
              name: tfjob-port
            resources:
              limits:
                nvidia.com/gpu: 2
            workingDir: /opt/tf-benchmarks/scripts/tf_cnn_benchmarks
          restartPolicy: OnFailure
```

- Submit the distributed TensorFlow job without enabling gang scheduling

  Run the following command to query the states of pods. Only two worker pods are running and the other worker pods are in the Pending state.

  ```
  kubectl get pods
  ```

  ```
  NAME                    READY   STATUS    RESTARTS   AGE
  tf-smoke-gpu-ps-0       1/1     Running   0          6m43s
  tf-smoke-gpu-worker-0   1/1     Running   0          6m43s
  tf-smoke-gpu-worker-1   1/1     Running   0          6m43s
  tf-smoke-gpu-worker-2   0/1     Pending   0          6m43s
  tf-smoke-gpu-worker-3   0/1     Pending   0          6m43s
  ```

  Run the following command to query log data of the running worker pods. The returned log data indicates that the running worker pods are waiting for the system to start the pending worker pods. The GPU resources occupied by the running worker pods are not in use.

  ```
  kubectl logs -f tf-smoke-gpu-worker-0
  ```

```
INFO|2020-05-19T07:02:18|/opt/launcher.py|27| 2020-05-19 07:02:18.199696: I tensorflo
w/core/distributed_runtime/master.cc:221] CreateSession still waiting for response fr
om worker: /job:worker/replica:0/task:3
INFO|2020-05-19T07:02:28|/opt/launcher.py|27| 2020-05-19 07:02:28.199798: I tensorflo
w/core/distributed_runtime/master.cc:221] CreateSession still waiting for response fr
om worker: /job:worker/replica:0/task:2
```

- Submit the distributed TensorFlow job with gang scheduling enabled

  Run the following command to query the states of pods. The computing resources in the cluster are insufficient to schedule the minimum number of pods. Therefore, the pod group cannot be scheduled and all pods are in the Pending state.

  ```
  kubectl get pods
  ```

  ```
  NAME                    READY   STATUS    RESTARTS   AGE
  tf-smoke-gpu-ps-0       0/1     Pending   0          43s
  tf-smoke-gpu-worker-0   0/1     Pending   0          43s
  tf-smoke-gpu-worker-1   0/1     Pending   0          43s
  tf-smoke-gpu-worker-2   0/1     Pending   0          43s
  tf-smoke-gpu-worker-3   0/1     Pending   0          43s
  ```

  After four GPUs are allocated to the cluster, the computing resources in the cluster are sufficient to schedule the minimum number of pods. After the pod group is scheduled, the four worker pods start to run. Run the following command to query the states of pods:

  ```
  kubectl get pods
  ```

  ```
  NAME                    READY   STATUS    RESTARTS   AGE
  tf-smoke-gpu-ps-0       1/1     Running   0          3m16s
  tf-smoke-gpu-worker-0   1/1     Running   0          3m16s
  tf-smoke-gpu-worker-1   1/1     Running   0          3m16s
  tf-smoke-gpu-worker-2   1/1     Running   0          3m16s
  tf-smoke-gpu-worker-3   1/1     Running   0          3m16s
  ```

  Run the following command to query log data of a running worker pod. The following output indicates that the tasks have been started.

  ```
  kubectl logs -f tf-smoke-gpu-worker-0
  ```

  ```
  INFO|2020-05-19T07:15:24|/opt/launcher.py|27| Running warm up
  INFO|2020-05-19T07:21:04|/opt/launcher.py|27| Done warm up
  INFO|2020-05-19T07:21:04|/opt/launcher.py|27| Step  Img/sec loss
  INFO|2020-05-19T07:21:05|/opt/launcher.py|27| 1 images/sec: 31.6 +/- 0.0 (jitter = 0.
  0) 8.318
  INFO|2020-05-19T07:21:15|/opt/launcher.py|27| 10  images/sec: 31.1 +/- 0.4 (jitter =
  0.7) 8.343
  INFO|2020-05-19T07:21:25|/opt/launcher.py|27| 20  images/sec: 31.5 +/- 0.3 (jitter =
  0.7) 8.142
  ```

5.4. Use KMS to encrypt Kubernetes Secrets

In a Container Service for Kubernetes (ACK) edge Pro cluster, you can use a key that is created by using Key Management Service (KMS) to encrypt Kubernetes Secrets. This topic describes how to use a key that is managed by KMS to encrypt Secrets for an ACK edge Pro cluster.

Prerequisites

- A customer master key (CMK) is created in the KMS console. For more information, see Create a CMK.

  > **Note** ACK edge Pro clusters support only CMKs of the Aliyun_AES_256 type.

- After you enable Secret encryption, do not use the KMS API or the KMS console to disable or delete the CMK that is used to encrypt and decrypt Secrets, or create a schedule to delete the CMK. Otherwise, the API server becomes unavailable and cannot retrieve Secrets and service accounts. As a result, service interruptions occur.
- Your Alibaba Cloud account is authorized to assume the AliyunCSManagedSecurityRole role. If your Alibaba Cloud account is not authorized to assume the AliyunCSManagedSecurityRole role, the system prompts you to complete the authorization first when you enable Secret encryption for a new ACK edge Pro cluster or an existing ACK edge Pro cluster.
- If you log on to the ACK console with a Resource Access Management (RAM) user or RAM role, make sure that the RAM user or RAM role is attached with the AliyunKMSCryptoAdminAccess policy. For more information, see Attach a RAM policy to a RAM user or RAM role.
- You are charged by KMS for key management and API calls (on a per 10,000 calls basis). After Secret encryption is enabled for an ACK Pro cluster, kube-apiserver must call the encryption and decryption API operations of KMS to perform read and write operations on Secrets. In most cases, a large number of read and write operations on Secrets are required during the lifecycle of service accounts, which may incur a large amount of fees in API calls. This situation intensifies when your cluster contains a large number of service accounts or Secrets. We recommend that you keep a sufficient account balance. If you are not familiar with the pricing rules or your account balance is insufficient, you can disable Secret encryption for the cluster. For more information, see Disable Secret encryption for an existing ACK edge Pro cluster. If your account has been overdue for more than seven days, you cannot manage the cluster. For more information about KMS billing, see Billing of KMS.

Context

Kubernetes Secrets are used to store and manage sensitive data, such as passwords to applications, Transport Layer Security (TLS) certificates, and credentials to download Docker images. Kubernetes stores Secrets in the etcd of a cluster. For more information about Kubernetes Secretes, see Secrets.

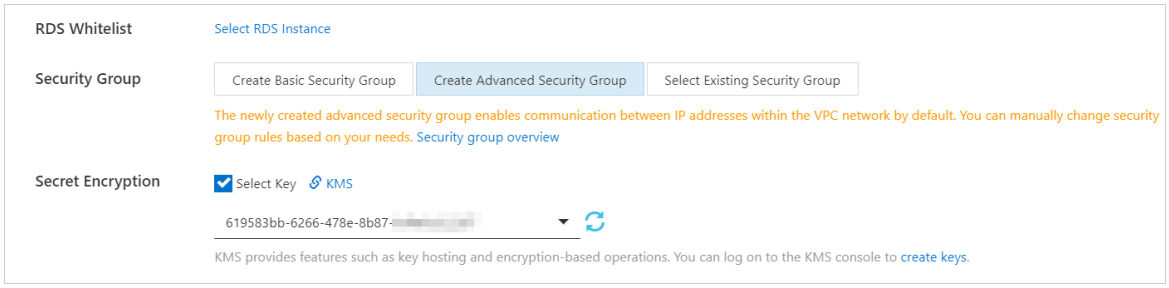
You can use keys that are created in KMS to encrypt Secrets in ACK edge clusters. KMS uses envelop encryption to encrypt and decrypt Secrets that are stored in etcd based on the KMS encryption provider. For more information about envelope encryption, see What is envelope encryption? The following content explains how to encrypt and decrypt Kubernetes Secrets:

- When you use a Kubernetes Secret to encrypt and store a password, the Kubernetes API server of your cluster generates a random data encryption key (DEK) to encrypt the Secret. Then, the API server sends the DEK to KMS. KMS uses the key that you specify to encrypt the DEK and returns the encrypted DEK to the API server. The API server then stores the encrypted Secret and DEK in etcd.

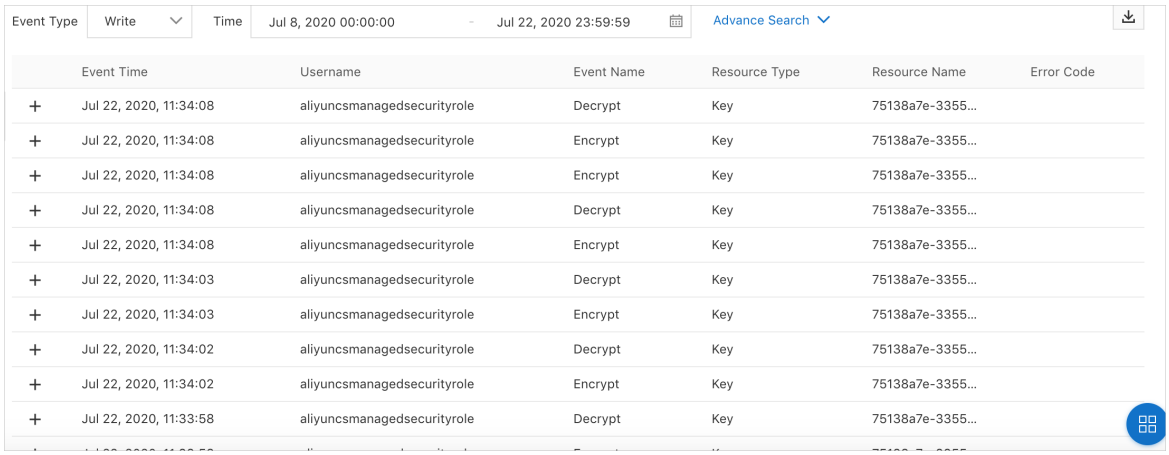
- When you decrypt the Kubernetes Secret, the system calls the Decrypt API operation of KMS to decrypt the DEK. Then, the system uses the decrypted DEK to decrypt the Kubernetes Secret and returns the password.

Enable Secrete encryption when you create an ACK edge Pro cluster

1.
2.
3.
4. Click the **Managed Edge Kubernetes** tab.
5. On the **Managed Edge Kubernetes** tab, find **Secret Encryption**, select **Select Key**, and then select a key from the drop-down list. For more information about how to configure an ACK edge Pro cluster, see Create a professional edge Kubernetes cluster.



Log on to the ActionTrail console. In the left-side navigation pane, click **Event Detail Query**. On the **Event Detail Query** page, check for encryption and decryption operations that are performed by assuming the aliyuncsmanagedsecurityrole role. If these operations exist, the Secret encryption feature is enabled.



Enable Secret encryption for an existing ACK edge Pro cluster

1.
2.
3. On the **Clusters** page, click the name of the ACK edge Pro cluster for which you want to enable Secret encryption.
4. On the details page of the cluster, click the **Basic Information** tab. In the **Basic Information** section, turn on **Secret Encryption**.

> **Note** If you log on to the ACK console with a RAM user, make sure that the RAM user is assigned one of the following role-based access control (RBAC) roles: the administrator role or O&M engineer role. For more information, see Assign RBAC roles to RAM users or RAM roles.

5. In the **Secret Encryption** dialog box, select an existing key. Click **OK**.

   If no key is available, click **create keys** to create a key in the KMS console. For more information, see Create a CMK.

   If the status of the cluster changes from **Updating** to **Running**, the Secret encryption feature is enabled for the cluster.

Disable Secret encryption for an existing ACK edge Pro cluster

1.
2.
3. On the **Clusters** page, click the name of the ACK edge Pro cluster for which you want to disable Secret encryption.
4. On the details page of the cluster, click the **Basic Information** tab. In the **Basic Information** section, turn off **Secret Encryption**.

   > **Note** If you log on to the ACK console with a RAM user, make sure that the RAM user is assigned one of the following RBAC roles: the administrator role or O&M engineer role. For more information, see Assign RBAC roles to RAM users or RAM roles.

   If the status of the cluster changes from **Updating** to **Running**, the Secret encryption feature is disabled for the cluster.

5.5. Customize the settings of control plane components in professional edge Kubernetes clusters

You can customize the settings of control plane components in a professional edge Kubernetes cluster to meet production needs. You can customize the settings of managed components such as Kube API Server and Kube Controller Manager (KCM). This topic describes how to customize the settings of control plane components in a professional edge Kubernetes cluster.

Considerations

Before you customize the settings of a control plane component, take note of the following items:

- After you customize the settings of a component, the component is automatically restarted. We recommend that you customize the settings during off-peak hours.
- After you customize the settings, the changes overwrite the default settings of the professional edge Kubernetes cluster.
- To ensure the stability of the control plane component, you are allowed to customize only some of the settings.
- Make sure that the values of the customized parameters are valid and complete. Otherwise, the

component may fail to be restarted. For more information about the parameters, see kube-apiserver and kube-controller-manager.

Customize the settings of a control plane component in a professional edge Kubernetes cluster

1.
2.
3.
4.

The following example shows how to customize the settings of Kube API Server:

5. In the **Core Components** section, find the component and click the icon.
6. In the **kube-apiserver Parameters** dialog box, set the parameters and click **OK**.

> **Note** Make sure that the specified values are valid and complete. You can customize only the settings of Kube API Server and KCM in professional edge Kubernetes clusters. For more information about the valid format and values of component parameters, see kube-apiserver and kube-controller-manager. Select the Kubernetes version based on the practical situation.

Default settings

The default settings are overwritten after you customize values for component parameters. You can reset the parameters to the default settings in the following table as needed.

Kubernetes version	Component	Parameter	Default value
1.16	kube-apiserver	ServiceNodePortRange	30000-32767
		EnableAdmissionPlugins	• If PodSecurityPolicy is enabled, the default value is `NodeRestriction,PodSecurityPolicy`. • If PodSecurityPolicy is disabled, the default value is `NodeRestriction`.
	kube-controller-manager	HorizontalPodAutoscalerSyncPeriod	15s

6.Cell-based management at the edge

6.1. Overview of cell-based management at the edge

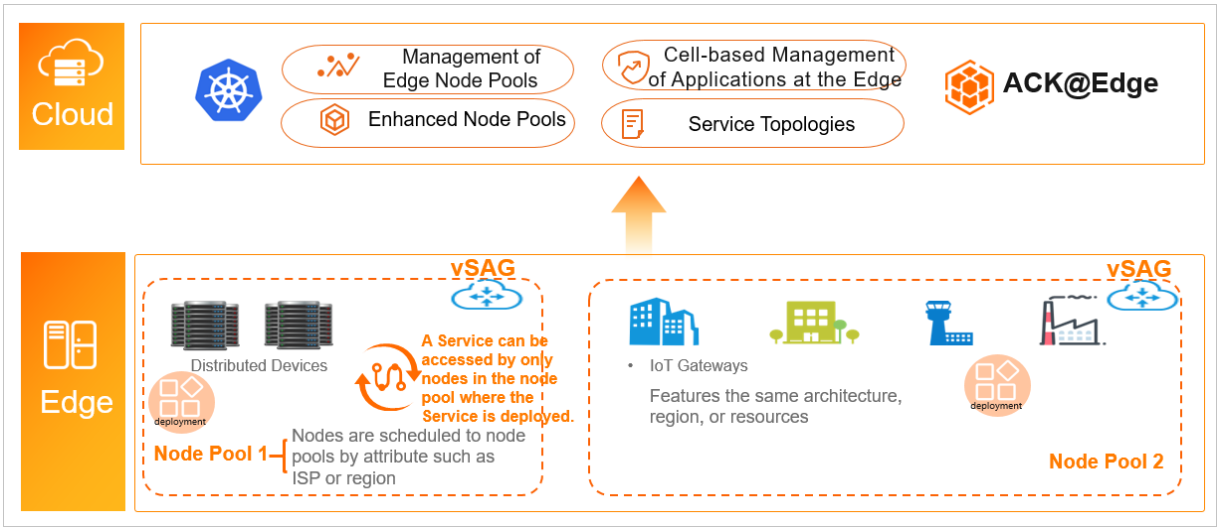
In edge computing, nodes that belong to different groups are isolated from each other in one way or another. For example, these nodes may be disconnected, may not share the same resources, may have heterogeneous resources, or may run applications that are independently deployed. ACK@Edge provides a cell-based management solution for edge computing scenarios. This topic describes how to implement cell-based management at the edge.

Traditional management

- In edge computing, edge nodes are classified into groups by zone, region, or other logical attribute such as CPU architecture, Internet service provider (ISP), or cloud service provider.
- Same applications or images may be deployed to different node pools.
- The backend endpoints of Kubernetes-native Services are arbitrarily distributed across nodes. Consequently, when Service requests are distributed to nodes across groups, these requests may fail to reach the nodes or may not be answered promptly.

Cell-based management at the edge

ACK@Edge provides a solution to solve these issues, as shown in the following figure.



- Node cell: You can create node pools to manage and maintain hosts in different regions.
- Application cell: You can deploy workloads to different node pools. This way, you can manage the number of pods and the image version of containers by node pool.
- Traffic cell: You can configure a Service topology to limit access to Service endpoints. For example, you can expose an application on an edge node to only the current node or other nodes in the same edge node pool.

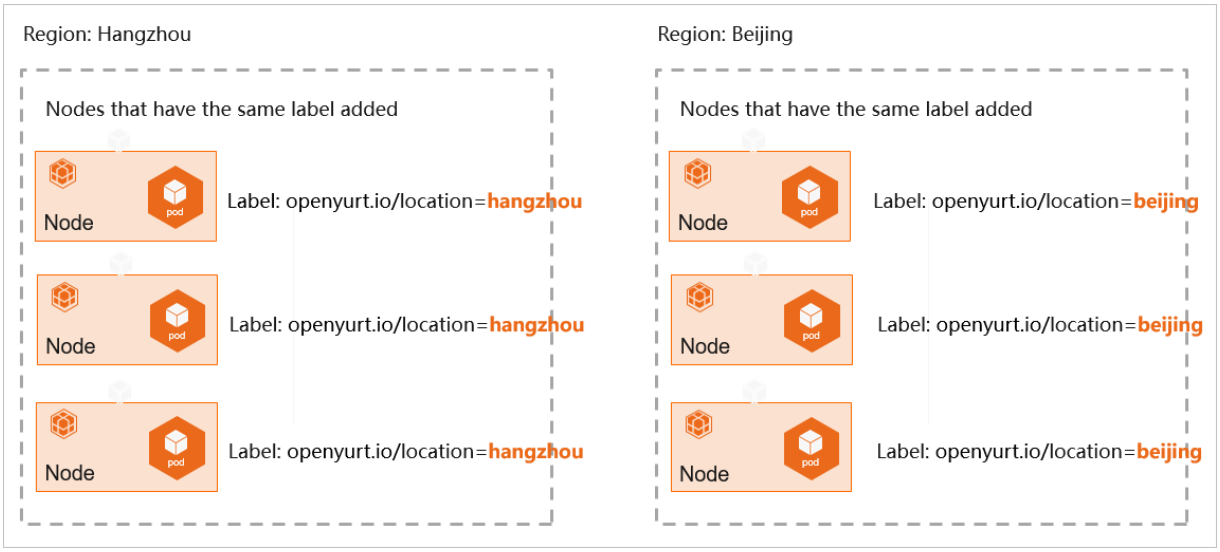
6.2. Manage edge node pools

6.2.1. Overview of edge node pools

The **yurt-app-manager** component provided by ACK@Edge allows you to manage edge node pools in edge computing scenarios. Edge nodes are distributed to different edge node pools based on specified attributes. This way, you can centrally manage and maintain edge nodes that are deployed in different regions by edge node pool. This topic describes edge node pools and how edge nodes are managed by edge node pool.

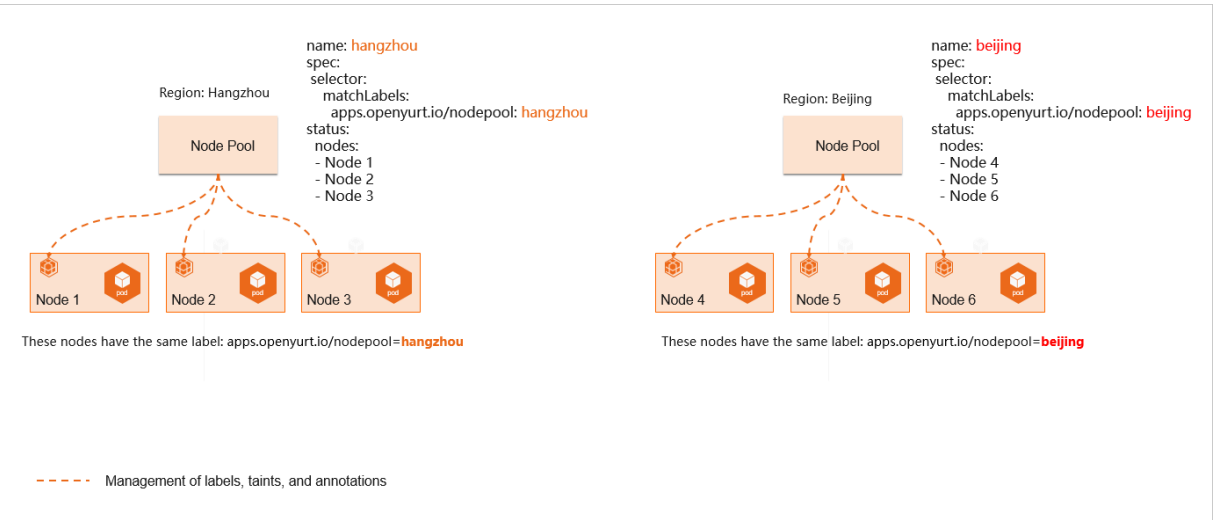
Traditional node management

In edge computing scenarios, edge nodes can be classified by different attributes such as CPU architecture, Internet service provider (ISP), and cloud service provider. Traditionally, labels are used to classify and manage nodes. However, as the numbers of nodes and labels increase, it becomes more complex to manage and maintain nodes. The following figure shows the traditional way of node management.



Edge node pools

Edge node pools allow you to classify nodes from a different dimension. You can centrally manage and maintain edge nodes that are deployed in different regions by edge node pool, as shown in the following figure.



For more information, see Overview of cell-based management at the edge.

Related information

- Create an edge node pool
- Add nodes to an edge node pool
- Create an enhanced edge node pool

6.2.2. Create an edge node pool

An edge node pool manages a group of nodes in a cluster, including the labels and taints of the nodes. This topic describes how to create an edge node pool in the Container Service for Kubernetes (ACK) console.

Prerequisites

- A managed edge Kubernetes cluster is created. For more information, see Create a managed edge Kubernetes cluster.
- The version of your cluster is 1.16 or later.

Procedure

1.
2.
3.
4.
5. On the **Node Pools** page, click **Create Edge Node Pool (Beta)** in the upper-right corner of the page.
6. In the **Create Edge Node Pool (Beta)** dialog box, set the parameters and click **Submit**.

Parameter	Description
Name	The name of the edge node pool.
Containerd Runtime	The supported containerd runtime are Docker, Containerd, and Sandboxed-Container. For more information about containerd, see Comparison of Docker, containerd, and Sandboxed-Container.
Coordination Network between Cloud and Edge	You can select Basic or Enhanced. For more information, see Create an enhanced edge node pool.
Maximum Nodes	The maximum number of nodes that can be added to the edge node pool.
Node Label	You can add labels to the nodes in the edge node pool.
Taints	You can add taints to the nodes in the edge node pool.

After the edge node pool is created, you can view information about the created node pool in the node pool list.

Related information

- Overview of edge node pools
- Create a managed edge Kubernetes cluster
- Add nodes to an edge node pool

6.2.3. Add nodes to an edge node pool

You can add worker nodes to an edge node pool that you created. Make sure that these worker nodes can communicate with the API server. This topic describes how to add nodes to an edge node pool.

Prerequisites

- An edge node pool is created. For more information, see Create an edge node pool.
- The version of your cluster is 1.16 or later.

Notice When you use a managed edge Kubernetes cluster, take note of the following limits:

- You can add only nodes that run CentOS 7.4, CentOS 7.6, or Ubuntu 18.04.
- Only Edge Node Service (ENS) instances with at least 2 cores and 4 GB of memory can be automatically added to the cluster. In addition, the ENS instances must be in the Running state and run CentOS 7.4 or 7.6.
- If the version of the cluster is 1.14.8-aliyunedge.1 or later, Advanced RISC Machine (ARM) nodes that run CentOS 7.4 or ARM64 nodes that run Ubuntu 18.04 can be added to the cluster.

Add nodes to an edge node pool

1.
2.
3.
4.
5. On the **Node Pools** page, find the edge node pool to which you want to add nodes and click **Add Existing Node** in the **Actions** column.

   Follow the steps that are described in the Add an edge node topic to add nodes. The topic also describes how to add ENS instances in auto and manual modes. For more information, see Add an edge node.

   After you add nodes to the edge node pool, you can click **Details** in the **Actions** column to view the nodes that you added.

Related information

- Add an edge node

6.2.4. Create an enhanced edge node pool

Enhanced cloud-edge networking is developed based on the Software Defined Network (SDN) solution of ACK@Edge. An edge node can connect to a Cloud Connect Network (CCN) instance through the nearest access point in the global transmission network of Alibaba Cloud. The CCN instance can communicate with virtual private clouds (VPCs) that are connected to the same Cloud Enterprise Network (CEN) instance. This connects the cloud and edge. This topic describes how an enhanced edge node pool works and how to create an enhanced edge node pool.

Prerequisites

- A CEN instance and a CCN instance are created. For more information, see Create a CEN instance and Create a CCN instance.
- Networks are planned to ensure that the CIDR block assigned to edge nodes does not conflict with the CIDR blocks of VPCs.

Context

Edge node pools support two types of modes for collaborative cloud-edge networking: **basic** and **enhanced**.

- **Basic**: The cloud and edge are connected through Internet connections. Applications in edge node pools cannot access VPCs in the cloud.
- **Enhanced**: This mode is based on the SDN solution of ACK@Edge. The cloud and edge are connected through high-speed and secure connections. Applications in edge node pools can access VPCs in the cloud. This mode outperforms the **basic** mode in terms of network quality and security.

Description	Basic	Enhanced
Cloud-edge networking	Establish Internet connections.	Create CCN instances.
Whether edge nodes can access VPCs	No.	Yes.

Description	Basic	Enhanced
Network quality	Low.	High. Edge nodes can connect to CCN instances through the nearest access points.
Security	Low.	High. Connections between the cloud and edge are encrypted.
Cost-effectiveness	Low.	High.
Scenarios	Workloads that are deployed at the edge and are not strongly reliant on cloud computing.	Applicable scenarios: • Workloads that require intercommunication between the cloud and edge. • Latency-sensitive workloads that require high network quality. • Workloads that require high network security.

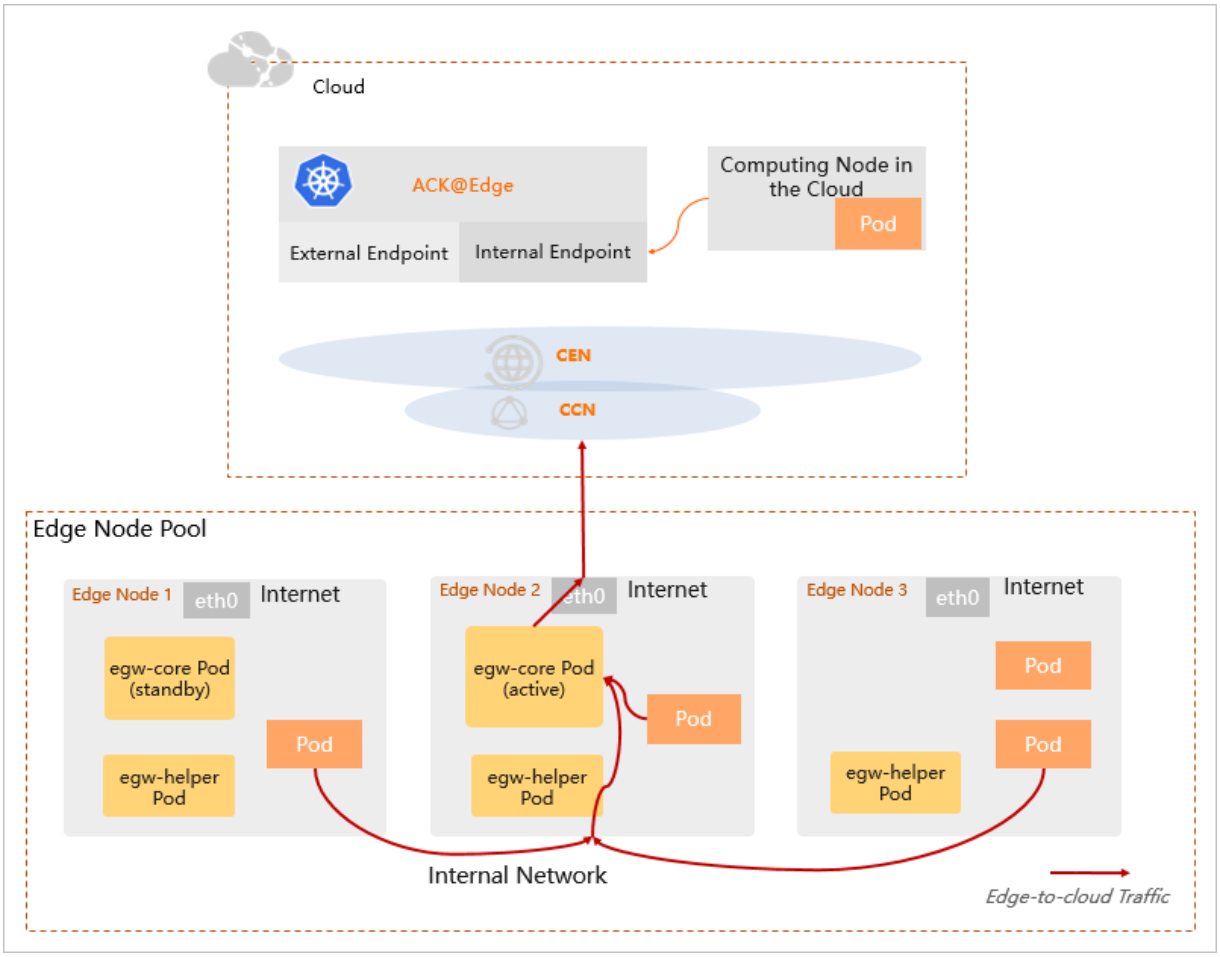
Limits

- The enhanced mode enables mutual communication between pods at the edge and pods in the cloud, and between pods at the edge and VPCs in the cloud. In addition, edge nodes can access VPCs and pods in the cloud. However, VPCs and pods in the cloud cannot access edge nodes. To enable a VPC to access edge nodes, you must use an elastic IP address (EIP).
- Each enhanced edge node pool must contain at least two AMD64 nodes.
- In an enhanced edge node pool, the gateway components are installed in pods on edge nodes. These pods support only the Flannel network plug-in. Host networking is not supported.
- When you create a basic or enhanced edge node pool, you must specify the maximum number of nodes that the edge node pool supports. This value is saved in an annotation of the NodePool object and cannot be modified. We recommend that you set the size of your edge node pool to a proper value.
- The metadata of an enhanced edge node pool is saved in the annotations of the NodePool object. You must not modify or delete these annotations. Otherwise, the enhanced edge node pool may fail to work in enhanced mode. For more information, see Annotations for an edge node pool.
- The `openyurt.io/desired-nodepool` label specifies the node pool to which a node belongs. You cannot move a node from an enhanced edge node pool to another enhanced edge node pool by modifying this label. To move the node, you must remove the node from the current node pool and then add it to another enhanced edge node pool. Otherwise, the node cannot work in enhanced mode. For more information, see Remove edge nodes.

How an enhanced edge node pool works

Enhanced cloud-edge networking is based on the SDN solution of ACK@Edge and the global network infrastructure of Alibaba Cloud. It enables reliable and secure communication between the cloud and edge. After you create an enhanced edge node pool and add edge nodes to the node pool, the gateway components are automatically installed in pods on edge nodes. The gateway components enable edge nodes to connect to CCN instances through the nearest access points. The CCN instances can communicate with the VPCs that are connected to the same CEN instance. This connects the cloud and edge. In enhanced mode, data exchanged between the cloud and edge is encrypted. Data is transmitted over the internal network of Alibaba Cloud. This ensures the efficiency and security of data transmission. In addition, edge nodes can access services that are deployed in VPCs.

When you create an enhanced edge node pool, the following components are deployed for the node pool: **edge-gateway-core(egw-core)** and **edge-gateway-helper(egw-helper)**. **edge-gateway-core** is the key component of an enhanced gateway. This component is deployed in the node pool as a Deployment. The Deployment creates and manages two pods in the node pool: One serves as the primary pod while the other serves as the secondary pod. The two pods are deployed on different nodes to ensure high availability. **edge-gateway-helper** is a component that synchronizes routes among nodes. This component is deployed on each node as a DaemonSet. It is used to configure routing information for nodes.



Create an enhanced edge node pool

1.
2.
3.

4.
5. On the **Node Pools** page, click **Create Edge Node Pool (Beta)** in the upper-right corner of the page.
6. On the **Create Edge Node Pool (Beta)** dialog box, set the required parameters. For more information, see Create an edge node pool.
   - Set **Coordination Network between Cloud and Edge** to **Enhanced**.
   - **CEN Instance**:
     - To use a CEN instance that belongs to your account, select **Use CEN Instance of Current Account** and then select the CEN instance.
     - To use a CEN instance that belongs to another account, you must first acquire the permissions to connect the VPC of the current cluster and the CCN instance to the CEN instance that you want to use. For more information, see Manage network instances and Attach a network instance. Then, select **Use CEN Instance of Other Accounts** and enter the UID of the account that owns the CEN instance and the ID of the CEN instance.
   - **CCN Instance**: Select the CCN instance that you have created.
7. Click **Submit**.
8. After the edge node pool is created, add at least two nodes to the node pool. For more information, see Add nodes to an edge node pool.

Terms

- CEN allows you to establish private connections between VPCs in different regions and between VPCs and data centers. This way, network resources are interconnected on a global scale.
- CCN is a matrix of distributed access gateways. You can connect on-premises resources to Alibaba Cloud by connecting CCN instances to CEN instances.

Annotations for an edge node pool

Annotation	Description
nodepool.openyurt.io/max-nodes	Specifies the maximum number of nodes that the enhanced edge node pool supports. This annotation is applicable to only non-default edge node pools.
nodepool.openyurt.io/pod-cidrs	Specifies the pod CIDR blocks that are assigned to the enhanced edge node pool. This annotation is applicable to only non-default edge node pools.
nodepool.openyurt.io/cen-id	Specifies the ID of the CEN instance for the enhanced edge node pool.
nodepool.openyurt.io/ccn-id	Specifies the ID of the CCN instance for the enhanced edge node pool.
nodepool.openyurt.io/ccn-region	Specifies the region of the CCN instance for the enhanced edge node pool. Only the China (Shanghai) region is supported in China.

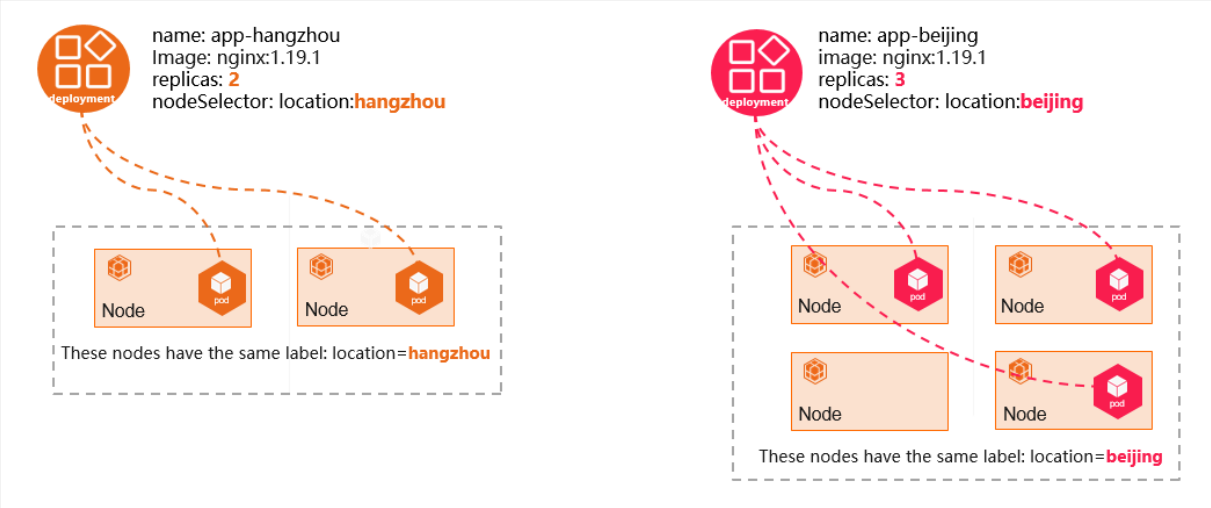
Annotation	Description
nodepool.openyurt.io/is-default	Specifies whether to set the enhanced edge node pool as a default edge node pool.

6.3. Use the UnitedDeployment controller to deploy applications

In edge computing scenarios, you can use the UnitedDeployment controller to deploy applications to different node pools. This way, you can centrally manage the number of pods and the image version of containers by node pool. This topic describes how to use the UnitedDeployment controller to deploy applications.

Context

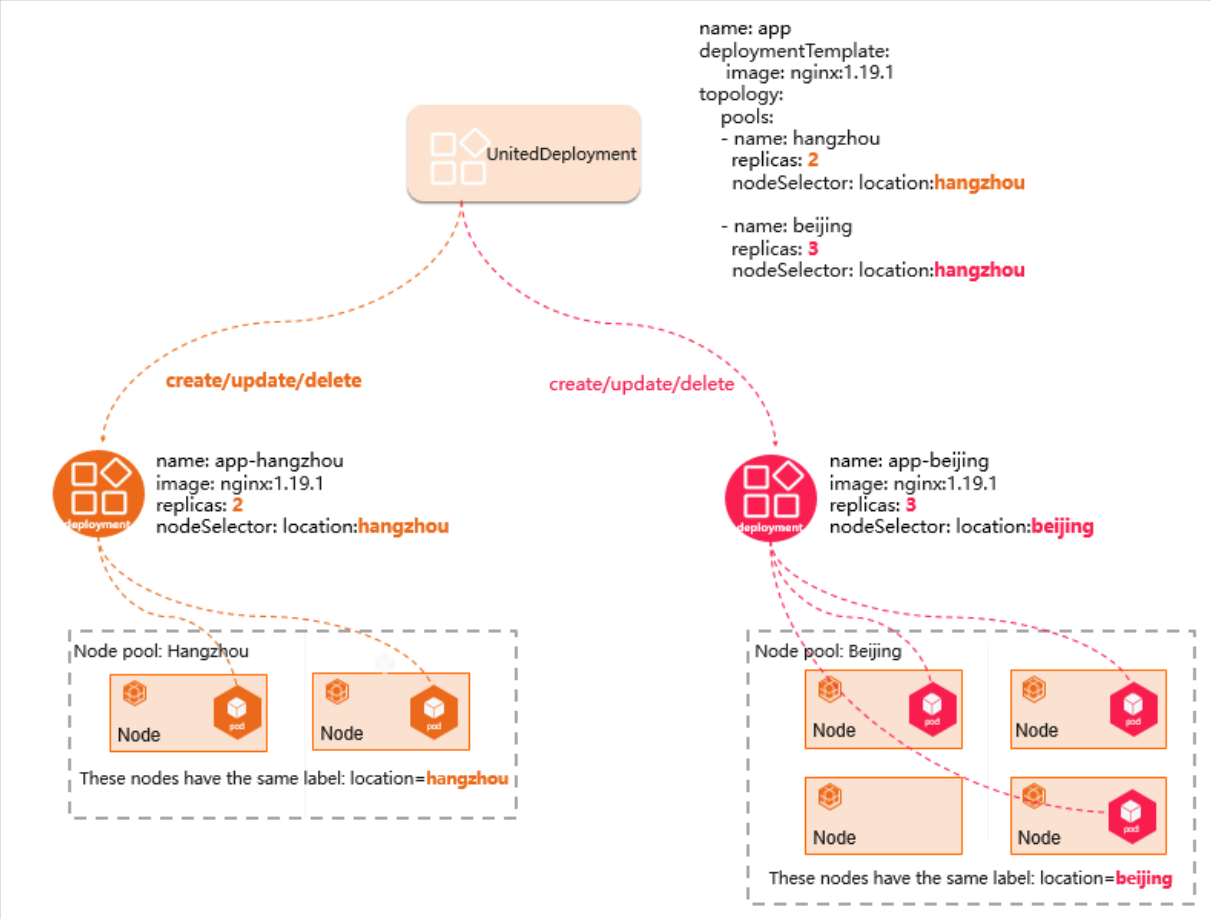
In edge computing scenarios, computing nodes may be deployed across regions, and same applications may run on nodes in different regions. Deployment is used as an example in this topic. Traditionally, you add the same label to the nodes that are deployed in the same region and create multiple Deployments. These Deployments match different labels through node selectors and are deployed in different regions to meet your business requirements.



Application management and maintenance become more complex with the increasing number of regions and differentiated requirements for applications in different regions. The following lists the main challenges:

- When a new image version is released, you must modify the image version for each Deployment.
- You must customize naming conventions to identify Deployments that belong to the same application.
- Deployments that belong to the same application are configured in the same way, except for the name, node selector, and replicated pods.

The UnitedDeployment controller is a feature provided by ACK@Edge. This feature allows you to centrally manage Deployments from a different dimension. For example, you can create, update, and delete multiple Deployments at a time.



The UnitedDeployment controller provides a template to define applications. This template allows you to deploy workloads in different regions and define each region as a node pool. The UnitedDeployment controller supports two types of workloads: StatefulSet and Deployment. The UnitedDeployment controller creates Deployments or StatefulSets based on the configurations of node pools. You can specify the number of replicated pods for each type of workload. UnitedDeployment enables automatic management and maintenance of multiple Deployments or StatefulSets within individual node pools. In addition, you can create differentiated configurations for these Deployments or StatefulSets, such as the name, node selector, and replicated pods.

Create a UnitedDeployment

Create a UnitedDeployment to deploy Deployments.

The following YAML template is an example:

```
apiVersion: apps.openyurt.io/v1alpha1
kind: UnitedDeployment
metadata:
  name: example
  namespace: default
spec:
  revisionHistoryLimit: 5
  selector:
    matchLabels:
      app: example
  workloadTemplate:
    deploymentTemplate:
```

```
      metadata:
        creationTimestamp: null
        labels:
          app: example
      spec:
        selector:
          matchLabels:
            app: example
        template:
          metadata:
            creationTimestamp: null
            labels:
              app: example
          spec:
            containers:
            - image: nginx:1.19.3
              imagePullPolicy: Always
              name: nginx
            dnsPolicy: ClusterFirst
            restartPolicy: Always
  topology:
    pools:
    - name: cloud
      nodeSelectorTerm:
        matchExpressions:
        - key: apps.openyurt.io/nodepool
          operator: In
          values:
          - cloud
      replicas: 2
    - name: edge
      nodeSelectorTerm:
        matchExpressions:
        - key: apps.openyurt.io/nodepool
          operator: In
          values:
          - edge
      replicas: 2
      tolerations:
      - effect: NoSchedule
        key: apps.openyurt.io/taints
        operator: Exists
```

The following table describes the fields in the YAML template.

Field	Description
spec.workloadTemplate	Indicates the workload template. Only the `deploymentTemplate` and `statefulSetTemplate` templates are supported.
spec.topology.pools	Specifies multiple node pools.

Field	Description
spec.topology.pools[*].name	The name of the node pool.
spec.topology.pools[*].nodeSelectorTerm	Specifies node affinity for the node pool. Set the key to `apps.openyurt.io/nodepool` and set the value to the name of the node pool.
spec.topology.pools[*].tolerations	Sets tolerance rules for the node pool.
spec.topology.pools[*].replicas	The number of pods in each node pool.

Use the UnitedDeployment controller to manage pods

- Upgrade pods: You can modify the `spec.template.workloadTemplate.deploymentTemplate` field to trigger pod upgrades. The UnitedDeployment controller updates the workload template for all node pools. Then, the node pool controller upgrades the pods in the node pools.
- Scale replicated pods for multiple node pools: You can modify the `spec.topology.pools` field to change the number of replicated pods for multiple node pools. Then, the replicated pods in the node pools are scaled based on the configuration.

6.4. Configure a Service topology

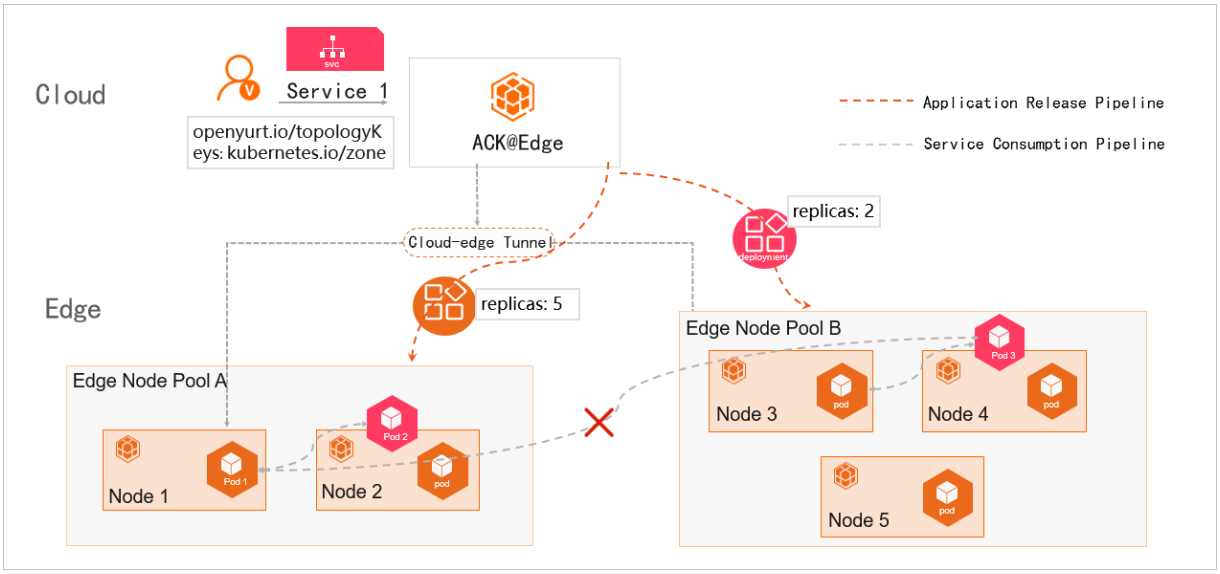
The backend endpoints of Kubernetes-native Services are arbitrarily distributed across nodes. Consequently, when Service requests are distributed to nodes across groups, these requests may fail to reach the nodes or may not be answered promptly. You can configure a Service topology to expose an application on an edge node to only the current node or other nodes in the same edge node pool. This topic describes how a Service topology works and how to configure a Service topology.

Context

In edge computing, edge nodes are classified into groups by zone, region, or other logical attribute such as CPU architecture, Internet service provider (ISP), or cloud service provider. Nodes in different groups are isolated from each other in one way or another. For example, these nodes may be disconnected, may not share the same resources, may have heterogeneous resources, or may run applications that are independently deployed.

How a Service topology works

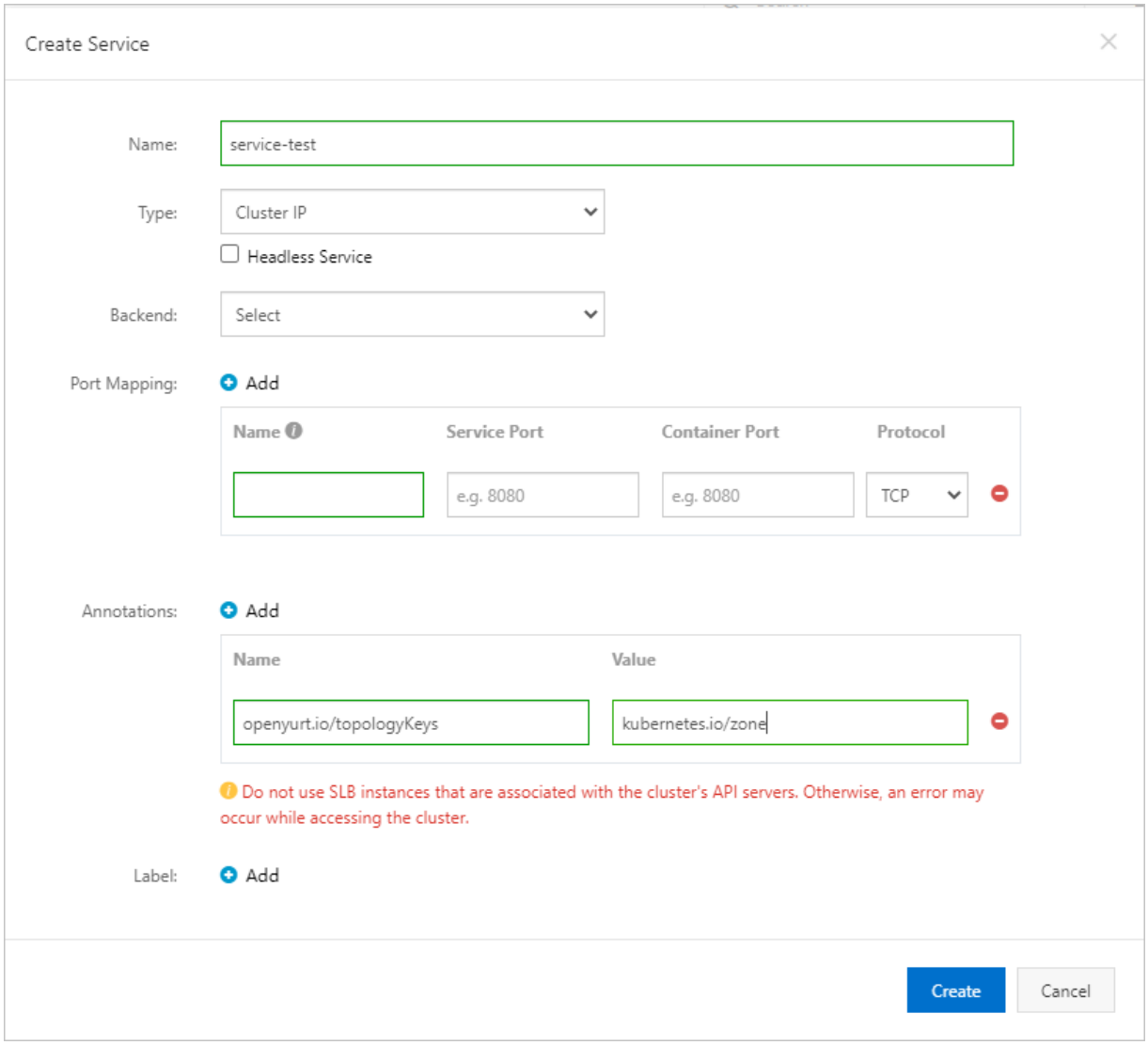
To solve the preceding issues, ACK@Edge provides a feature to manage the topology of endpoints of Kubernetes-native Services. You can configure a Service topology to specify how endpoints of a Service are accessed. For example, you can configure a Service topology to expose an application on an edge node to only the current node or other nodes in the same edge node pool. The following figure shows how a Service topology works.



- Service 1 is associated with Pod 2 and Pod 3. `annotation:"openyurt.io/topologyKeys: kubernetes.io/zone"` specifies the node pool that is allowed to access Service 1.
- Pod 2 is deployed on Node 2 and Pod 3 is deployed on Node 3. Node 2 belongs to Node Pool A and Node 3 belongs to Node Pool B.
- Pod 3 and Pod 1 do not belong to the same node pool. As a result, when Pod 1 accesses Service 1, the traffic is forwarded to only Pod 2. The traffic cannot be forwarded to Pod 3.

Method 1: Configure a Service topology in the ACK console

To create a Service that can be accessed by only the node pool where the Service is deployed, you need only to add an annotation to the Service. For example, you can set **Name** to `openyurt.io/topologyKeys` and **Value** to `kubernetes.io/zone`. For more information about how to create a Service, see Manage Services.



Method 2: Configure a Service topology by using the CLI

You can use the command-line interface (CLI) to configure a Service topology in the following ways:

- Create a Service that uses the topological domain of a specific node pool. Sample YAML template:

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    openyurt.io/topologyKeys: kubernetes.io/zone
  name: my-service-nodepool
  namespace: default
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 8080
  selector:
    app: nginx
  sessionAffinity: None
  type: ClusterIP
```

- Run the following command to configure the Service topology. The Service uses the topological domain of the specified node pool.

```
kubectl annotate service xxx openyurt.io/topologyKeys='kubernetes.io/zone'
```

Annotations

You can add annotations to a Kubernetes-native Service to configure a Service topology. The annotations are described in the following table.

Annotation key	Annotation value	Description
openyurt.io/topologyKeys	kubernetes.io/hostname	Specifies that the Service can be accessed by only the node where the Service is deployed.
openyurt.io/topologyKeys	kubernetes.io/zone	Specifies that the Service can be accessed by only the nodes in the node pool where the Service is deployed.
-	-	Specifies that access to the Service is unlimited.

7.Node management

7.1. Add an edge node

This topic describes how to add an edge node as a worker node to an existing Container Service for Kubernetes (ACK) edge cluster in the ACK console. Make sure that the node to be added can communicate with the Kubernetes API server of the cluster. You can add Elastic Compute Service (ECS) instances, Edge Node Service (ENS) instances, and on-premises servers to an ACK edge cluster.

Prerequisites

- An ACK edge cluster is created. For more information, see Create a managed edge Kubernetes cluster.
- ENS is activated and an edge service is created.

Limits

- Make sure that you have a sufficient node quota in the cluster. To add more nodes, to apply for a quota increase. For more information about the quota limits related to ACK edge clusters, see Limits.
- Only ENS nodes with at least 2 vCPUs and 4 GB of memory can be automatically added to an ACK edge cluster. In addition, the ENS nodes must be in the Running state and run CentOS 7.4 or 7.6.
- If you want to manually add nodes, the nodes must run operating systems that are listed in the following table.

OS architecture	OS version	Kernel version	Kubernetes version
AMD64/X86_64	CentOS 7.4, CentOS 7.5, CentOS 7.6, CentOS 7.7, CentOS 7.8, and CentOS 7.9	3.10.X	1.12.6-aliyunedge.1 and later
AMD64/X86_64	CentOS 8.0 and CentOS 8.2	4.18.X	1.18.8-aliyunedge.1 and later
AMD64/X86_64	Ubuntu 16.04	4.4.X	1.18.8-aliyunedge.1 and later
AMD64/X86_64	Ubuntu 18.04	4.15.X	1.12.6-aliyunedge.1 and later
AMD64/X86_64	Ubuntu 18.04	5.4.X	1.16.9-aliyunedge.1 and later
AMD64/X86_64	Ubuntu 18.04	5.11.X	1.18.8-aliyunedge.1 and later
AMD64/X86_64	Ubuntu 20.04	5.4.X	1.18.8-aliyunedge.1 and later
AMD64/X86_64	AliyunLinux 2.1903	4.19.X	1.20.11-aliyunedge.1 and later

OS architecture	OS version	Kernel version	Kubernetes version
AMD64/X86_64	AliyunLinux 3	5.10.X	1.20.11-aliyunedge.1 and later
ARM64	CentOS 8.0	4.19.X	1.14.8-aliyunedge.1 and later
ARM64	Ubuntu 18.04	4.9.X	1.14.8-aliyunedge.1 and later
ARM64	Ubuntu 18.04	4.19.X	1.14.8-aliyunedge.1 and later

Add a node

1.
2. Add an existing node. You can use one of the following methods:
   - Method 1:
     a. In the left-side navigation pane of the ACK console, click **Clusters**.
     b. On the Clusters page, find the cluster that you want to manage and choose **More > Add Existing Node** in the **Actions** column.
     c. On the **Node Pools** page, find the node pool that you want to manage and choose **More > Add Existing Node** in the **Actions** column.
   - Method 2:
     a. In the left-side navigation pane of the ACK console, click **Clusters**.
     b. On the **Clusters** page, find the cluster that you want to manage and click its name or click **Details** in the **Actions** column.
     c. In the left-side navigation pane of the details page, choose **Nodes > Node Pools**.
     d. On the **Node Pools** page, find the node pool that you want to manage and choose **More > Add Existing Node** in the **Actions** column.
3. On the page that appears, you can select **Manual** to manually add existing instances.

   Select the Manual mode.

   > **Note** In Manual mode, you can add ECS instances, ENS instances, and on-premises servers.

   i. Click **Next Step**.
   ii. On the Specify Instance Information wizard page, set the parameters that are used to add the node. For more information, see Parameters.

   > **Note** The default value of **Script Validity Period** is 1 hour. If you want to use the script for more than 1 hour, you can set the validity period to meet your requirement. If you set **Script Validity Period** to 0 hours, the script is permanently valid.

   iii. After the configuration is completed, click **Next Step**.

iv. On the **Complete** wizard page, click **Copy** to copy the script to the edge node that you want to add. Then, execute the script on the node.

If the following result is returned, the edge node is added to the cluster.

```
I0410 10:54:25.801554   19419 join-node.go:241] [join-node] Config the kubelet service configuration successfully.
I0410 10:54:25.801590   19419 join-node.go:246] [join-node] Adding edge hub static yaml
I0410 10:54:25.801662   19419 join-node.go:279] [join-node] Add edge hub static yaml is ok
I0410 10:54:25.801666   19419 join-node.go:384] [join-node] Start to joining node to cluster.
I0410 10:54:27.338166   19419 join-node.go:393] [join-node] Join node to cluster successfully.
I0410 10:54:27.338214   19419 install.go:151] [install-edgehub] Checking edgehub status
I0410 10:54:37.357405   19419 install.go:156] [install-edgehub] Edgehub is ok
I0410 10:54:37.357421   19419 install.go:86] [install-edgehub] Reconfiguring the kubelet configuration files.
I0410 10:54:37.364387   19419 install.go:103] [install-edgehub] Reconfigure the kubelet configuration files successfull
y.
I0410 10:54:37.364400   19419 install.go:104] [install-edgehub] Restarting the kubelet.
I0410 10:54:52.626540   19419 install.go:127] [install-edgehub] Restart the kubelet successfully.
I0410 10:54:52.626613   19419 postcheck.go:77] [post-check] Checking docker status
I0410 10:54:52.629194   19419 postcheck.go:86] [post-check] docker is ok
I0410 10:54:52.629208   19419 postcheck.go:92] [post-check] Checking kubelet status
I0410 10:54:52.631661   19419 postcheck.go:100] [post-check] Kubelet is ok
I0410 10:54:52.631671   19419 postcheck.go:106] [post-check] Checking edgehub status
I0410 10:54:52.642345   19419 postcheck.go:113] [post-check] Edgehub is ok
I0410 10:54:52.642356   19419 postcheck.go:129] [post-check] Checking addon kube-proxy status.
I0410 10:54:52.683227   19419 postcheck.go:133] [post-check] kube-proxy is OK.
I0410 10:54:52.683243   19419 postcheck.go:129] [post-check] Checking addon flannel status.
I0410 10:54:52.724501   19419 postcheck.go:133] [post-check] flannel is OK.
I0410 10:54:52.724518   19419 postcheck.go:129] [post-check] Checking addon coredns status.
I0410 10:54:52.764745   19419 postcheck.go:133] [post-check] coredns is OK.
I0410 10:54:52.764763   19419 postcheck.go:165] [post-check] Callback to the OpenAPI.
I0410 10:54:53.014706   19419 postcheck.go:178] [post-check] Callback to the OpenAPI successfully.
I0410 10:54:53.014760   19419 postcheck.go:66] This node joined into the cluster successfully.
```

Parameters

Parameter	Description	Default
flannelIface	The name of the network interface controller (NIC) that is used by the Flannel plug-in.	The name of the NIC that is specified in the default route entry of the node
enableIptables	Specifies whether to enable `iptables`.	`false`
quiet	Specifies whether to answer all questions with `yes` when you add nodes.	`false`
manageRuntime	Specifies whether to use edgeadm to install and manage the runtime.	`false`

Parameter	Description	Default
nodeNameOverride	The name of the node.	• `""`. This is the default value. This value indicates that the hostname is used as the node name. • `"*"`. This value indicates that a random string that contains six characters is used as the node name. • `"*.XXX"`. This value indicates that a random string that is appended with a suffix is used as the node name. The random string contains six characters.
allowedClusterAddons	The list of components to be installed. By default, this parameter is empty. This indicates that no component is installed. For a regular node, set this parameter to ["kube-proxy","flannel","coredns"].	`[]`
gpuVersion	Specifies whether the node to be added is a GPU-accelerated node. By default, this parameter is empty. Supported GPU models are Nvidia_Tesla_T4, Nvidia_Tesla_P4, Nvidia_Tesla_P100, and Nvidia_Tesla_V100. If other GPU models need to be supported, .	`""`. This is the default value. This value indicates that the node to be added is not a GPU-accelerated node.
inDedicatedNetwork	Specifies whether an Express Connect circuit is used to add the node to the ACK edge cluster.	`false`
labels	Specifies the labels to be added to the node.	`{}`
annotations	Specifies the annotations to be added to the node configurations.	`{}`

Parameter	Description	Default

nodeIface	This parameter specifies the following information:
	• This parameter specifies that kubelet retrieves the node IP address from the specified network interface. If you do not set this parameter, kubelet attempts to retrieve the node IP address in the following order:
	  ◦ Searches */etc/hosts* for the node whose name is the same as the specified hostname.
	  ◦ Retrieves the IP address of the network interface that is specified in the default route entry of the node.
	• This parameter specifies the name of the NIC that is used by Flannel. In this case, this parameter is equivalent to the flannelIface parameter. This parameter will replace the flannelIface parameter in the future.
		`""`

7.2. Configure node autonomy

This topic describes how to set the autonomy attribute for edge nodes. If an edge node is autonomous, applications run as expected on the edge node even if the edge node is disconnected from the cloud. This ensures that applications are not removed or migrated to other edge nodes in the case of network errors.

Prerequisites

- Create a managed edge Kubernetes cluster
- Add an edge node

Context

You can enable or disable node autonomy in the Container Service for Kubernetes (ACK) console.

- If an autonomous edge node is disconnected from the cloud, ACK does not migrate applications on this node to other nodes, and the applications are automatically restored. Node autonomy is applicable to edge computing scenarios where the network connection is weak.
- If a non-autonomous edge node is disconnected from the cloud, the node fails to send heartbeats to the nodes in the cloud. As a result, the state of the node is changed to **NotReady** and the applications on the node are removed or migrated to other nodes after a specific time period.

Procedure

1.
2.
3. On the **Clusters** page, find the cluster that you want to manage, and click the cluster name or click **Details** in the **Actions** column.
4.
5. On the **Nodes** page, find the node that you want to manage and choose **More > Node Autonomy Setting** in the **Actions** column.

   > **Note** The **Node Autonomy Setting** option is available for only edge nodes.

6. In the **Node Autonomy Setting** dialog box, click **OK**.

   > **Note** By default, edge nodes are not autonomous when they are added to the cluster. You can follow the preceding steps to enable or disable node autonomy.

Related information

- ACK@Edge overview
- Add an edge node

7.3. Remove edge nodes

You can remove edge nodes from a managed edge Kubernetes cluster. This topic describes how to remove edge nodes.

Prerequisites

- Create a managed edge Kubernetes cluster
- Connect to ACK clusters by using kubectl

Context

- When you remove an edge node, the pods that run on the node are migrated to other nodes. This may cause service interruption. We recommend that you remove nodes during off-peak hours.
- Unknown errors may occur when you remove nodes. We recommend that you back up data on these nodes before you remove them.
- During the removal process, nodes remain in the Unschedulable state.
- Only worker nodes can be removed. Master nodes cannot be removed.
- A managed edge Kubernetes cluster can contain two types of nodes: cloud nodes and edge nodes. You can remove both types of nodes at a time.
- You must retain at least one cloud node in a managed edge Kubernetes cluster.
- We recommend that you remove nodes in the Container Service for Kubernetes (ACK) console. If you run the `kubectl delete node` command to remove nodes, take note of the following limits:

- For cloud nodes:
  - The removed nodes cannot be added to other clusters.
  - After you delete a cluster, the Elastic Compute Service (ECS) instances where the removed nodes are deployed are automatically released.
- For edge nodes: You must run the Reset command in Edgeadm to reset the removed nodes before you can add them to other clusters.

Procedure

1.
2.
3.
4.
5. On the **Nodes** page, find the node that you want to remove and choose **More > Remove** in the **Actions** column.

   > **Note** To remove multiple nodes at a time, select the nodes that you want to remove on the **Nodes** page and click **Batch Remove**.

6. Optional. In the **Remove Node** dialog box, you can select **Release ECS Instance** and **Drain the Node** if all of the selected nodes are cloud nodes. Then, click **OK**.

   > **Note** If one or more edge nodes are selected, the **Release ECS Instance** and **Drain the Node** check boxes are unavailable in the **Remove Node** dialog box.

   - Release ECS Instance:
     - Select this option to release only pay-as-you-go ECS instances.
     - Subscription ECS instances are automatically released after the subscription expires.
     - If you do not select Release ECS Instance, the system continues to bill the ECS instances where the removed nodes are deployed.
   - Drain the Node: Select this option to migrate pods that run on the removed nodes to other nodes in the cluster. If you select this option, make sure that the other nodes in the cluster have sufficient resources for these pods. You can also run the `kubectl drain node-name` command to migrate pods that run on the removed nodes to other nodes in the cluster.

     > **Note** The value of *node-name* must be in the format of *your-region-name.node-id*. For example, the value can be *cn-hangzhou.i-xxx*.
     >
     > - *your-region-name* specifies the region where the cluster is deployed.
     > - *node-id* specifies the ID of the ECS instance where the node to be removed is deployed.

8.Application management

8.1. Network autonomy of edge nodes

The network autonomy of edge nodes ensures that applications are automatically reconnected to each other after they recover from exceptions. This topic describes the network autonomy of edge nodes.

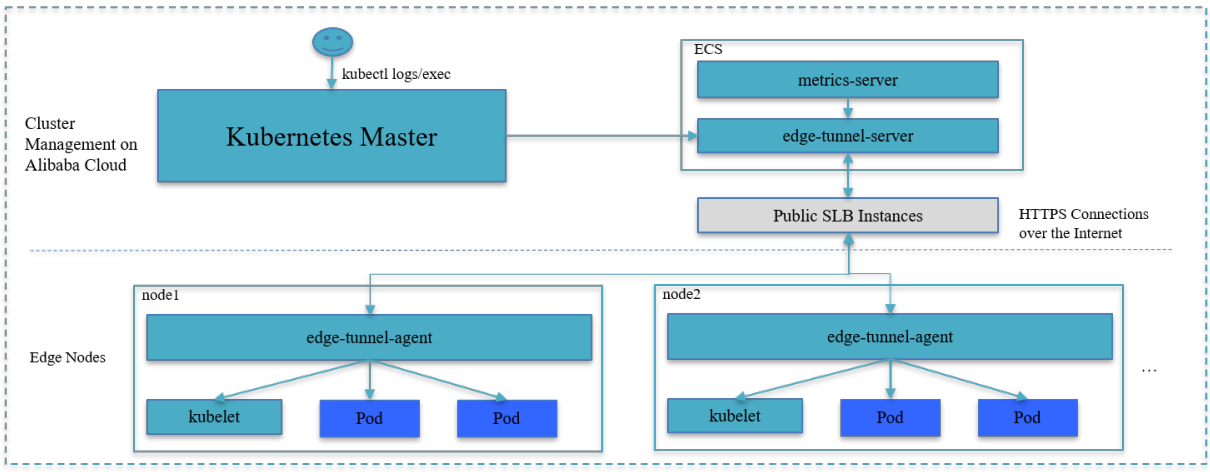
Background

By default, an edge node becomes network-autonomous after it is connected to a cluster in the cloud.

- For a pod that runs on a network-autonomous edge node, the pod IP address is bound to the pod name. The IP address of the pod remains unchanged even if the application that runs on the pod is restarted, or the node to which the pod belongs is restarted. In addition, the name of the node is bound to the MAC address of VXLAN Tunnel Endpoint (VTEP). In this example, VTEP is the virtual network interface controller (NIC) flannel.1. The MAC address of VTEP remains unchanged even if containers that use Flannel are restarted, or the node is restarted.
- Edge nodes may have exceptions or disconnect from the controllers in the cloud. In this case, if the edge nodes are network-autonomous, communication within an edge node or across edge nodes can be automatically restored after the applications or nodes are restarted. This applies to cross-node communication in edge computing scenarios where the network connection is weak.

Features

- By default, applications are deployed on network-autonomous nodes in both host network mode and non-host network mode.
- The network autonomy of edge nodes ensures that applications are automatically reconnected to each other across nodes after the applications recover from exceptions. The following figure shows how network-autonomous edge nodes work.



> **Note** If a pod is recreated or migrated to another node, the IP address of the pod is changed.

8.2. Cloud-edge tunneling

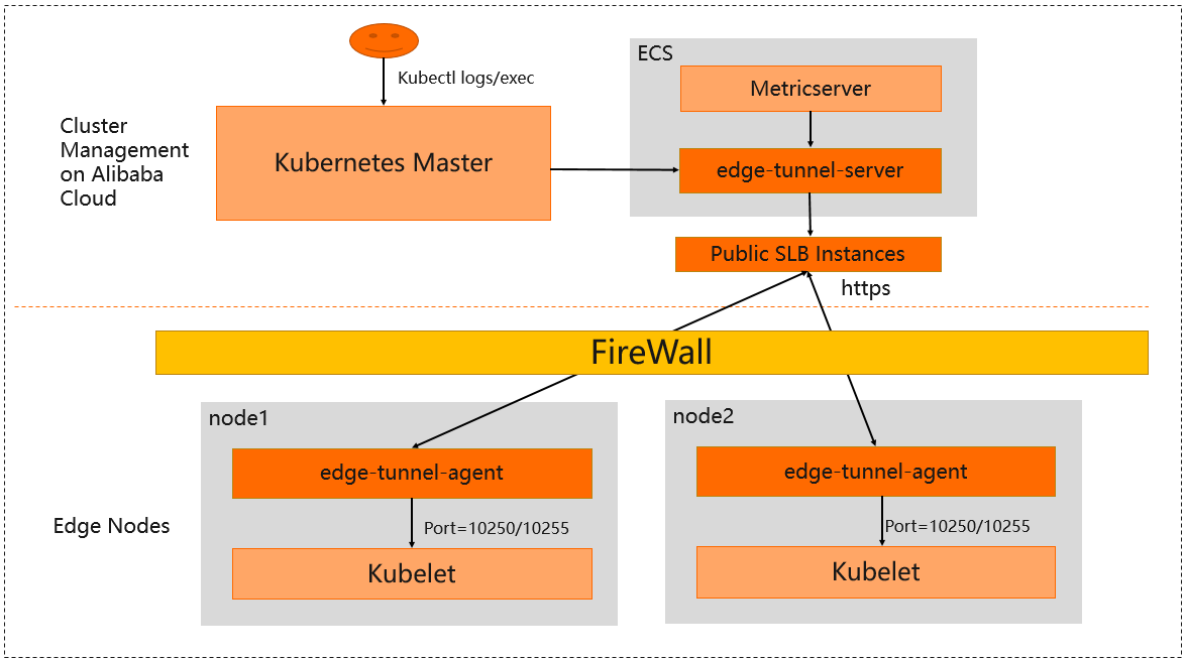
By default, Container Service for Kubernetes (ACK) deploys the **edge-tunnel-server and edge-tunnel-agent** components after you create an ACK edge cluster. These components are used to establish tunnels from the cloud to the edge. After the tunnels are established, you can access edge nodes from the cloud. This improves user experience. This topic describes the features of the components that are related to cloud-edge tunnels and how to extend the monitoring capabilities of edge nodes.

Background information

- In a Kubernetes cluster, the controller components in the cloud must run commands and pass maintenance requests to kubelet on edge nodes. The monitoring component metrics-server must collect monitoring data from edge nodes to the cloud. If the edge nodes of an ACK edge cluster are deployed in an internal network, you cannot directly access the edge nodes from the cloud.
- The **edge-tunnel-server** component is deployed as a Deployment on nodes in the cloud. The **edge-tunnel-agent** component is deployed as a DaemonSet on each edge node.
- When you perform O&M operations by running Kubernetes commands, such as kubectl logs and kubectl exec, or commands of metrics-server, the requests are sent to port 10250 and port 10255 on kubelet on edge nodes.

Description

- When you create an ACK edge cluster, you must create at least one Elastic Compute Service (ECS) instance to deploy the **edge-tunnel-server** component.
- To establish secure and encrypted tunnels over the Internet, the system creates a Server Load Balancer (SLB) instance for the Service that is created by **edge-tunnel-server**. The **edge-tunnel-agent** component on an edge node establishes a tunnel to edge-tunnel-server through the SLB instance.
- When components, such as kube-apiserver and metrics-server, attempt to access port 10250 and port 10255 on edge nodes from the cloud, the requests are automatically forwarded to edge-tunnel-server, without the need to modify the components.
- The following figure shows how cloud-edge tunneling works.



> **Note**
>
> - When edge nodes are disconnected from the cloud or the network connection is weak, the tunnels may fail to work as normal.
> - If you delete or stop the SLB instance through which the tunnels are established, the tunnels cannot work as normal.
> - For ACK edge clusters of early Kubernetes versions, such as 1.16.9-aliyunedge.1, you must deploy components, such as metrics-server, on the ECS node where edge-tunnel-server is deployed. Otherwise, the components cannot access edge nodes. For ACK edge clusters of 1.18.8-aliyunedge.1 or later, you can deploy components, such as metrics-server, and edge-tunnel-server on different ECS nodes.

Configure access to ports other than 10250 and 10255 on edge nodes

When you migrate your business to the cloud, the monitoring system of your business is also migrated to the cloud. To ensure a seamless migration and collect monitoring data from edge nodes to the cloud, you must configure access to ports other than 10250 and 10255 on edge nodes from the cloud. In this example, ports 9051 and 9052 on edge nodes are used.

> **Note**
>
> In this example, edge-tunnel-server listens on port 9051 over HTTP and listens on port 9052 over HTTPS.

Clusters of 1.18.8-aliyunedge.1

In clusters of 1.18.8-aliyunedge.1, only HTTP is supported when components collect monitoring data from edge nodes to the cloud through ports other than 10250 and 10255. You must modify the value of the `dnat-ports-pair` field in the `edge-tunnel-server-cfg` ConfigMap in the kube-system namespace. Set the value in the following format: port number=10264.

To enable components to access port 9051 on edge nodes from the cloud, use the following configuration:

```
cat <<EOF | kubectl apply -f
apiVersion: v1
data:
  dnat-ports-pair: '9051=10264'
kind: ConfigMap
metadata:
  name: edge-tunnel-server-cfg
  namespace: kube-system
EOF
```

Clusters of 1.20.11-aliyunedge.1

In clusters of 1.20.11-aliyunedge.1, HTTP and HTTPS are supported when components collect monitoring data from edge nodes to the cloud through ports other than 10250 and 10255. Components can also access the localhost endpoints on edge nodes from the cloud.

- To enable access to ports other than 10250 and 10255 over HTTP, configure the `http-proxy-ports`

field in the `edge-tunnel-server-cfg` ConfigMap in the kube-system namespace. Set the value in the following format: port 1, port 2.

- To enable access to ports other than 10250 and 10255 over HTTPS, configure the `https-proxy-ports` field in the `edge-tunnel-server-cfg` ConfigMap in the kube-system namespace. Set the value in the following format: port 1, port 2.
- To enable access to localhost endpoints on edge nodes, configure the `localhost-proxy-ports` field in the `edge-tunnel-server-cfg` ConfigMap in the kube-system namespace. Default value: 10250, 10255, 10266, 10267. You can add more ports.

To enable components to collect monitoring data from edge nodes to the cloud through ports 9051 and 9052 and to access localhost endpoints on edge nodes, for example, https://127.0.0.1:8080, use the following configuration:

```
cat <<EOF | kubectl apply -f
apiVersion: v1
data:
  http-proxy-ports: "9051"
  https-proxy-ports: "9052, 8080"
  localhost-proxy-ports: "10250, 10255, 10266, 10267, 8080"
kind: ConfigMap
metadata:
  name: edge-tunnel-server-cfg
  namespace: kube-system
EOF
```

8.3. Use LVM to manage local storage

Professional edge Kubernetes clusters allow you to use Logical Volume Manager (LVM) to manage local storage. LVM can automatically manage the lifecycle of logical volumes and schedule volumes based on the storage capacity of nodes. To use LVM to manage local storage as persistent volumes (PVs) and persistent volume claims (PVCs), you need only to define the topological relationship of local disks on the nodes. This topic describes how to use LVM to manage local storage in professional edge Kubernetes clusters.

Prerequisites

Local disks are available in cluster nodes.

Install node-resource-manager and csi-local-plugin

1.
2.
3.
4. On the **Add-ons** page, select **node-resource-manager** and **csi-local-plugin** in the Storage section and click **Install** to install the components.
5. In the **Note** dialog box, click **OK**.

Configure the VolumeGroup

Note To ensure data security, the components do not delete VolumeGroups or physical volumes. Before you can redefine a VolumeGroup, you must delete the existing VolumeGroup.

1. Use the following YAML template to configure a ConfigMap that defines the topology for the VolumeGroup:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: node-resource-topo
  namespace: kube-system
data:
  volumegroup: |-
    volumegroup:
    - name: volumegroup1
      key: kubernetes.io/storagetype
      operator: In
      value: lvm
      topology:
        type: device
        devices:
        - /dev/sdb1
        - /dev/sdb2
        - /dev/sdc
```

The following table describes the parameters.

Parameter	Description
name	The name of the VolumeGroup.
key	Specifies the key that is used to match the key of the label on the nodes.
operator	Specifies the `operator that is used in the label selector`. Valid values: ○ In: A match is found only when the `value` is the same as the `value` of the label that has the specified key. ○ NotIn: A match is found only when the `value` is different from the `value` of the label that has the specified key. ○ Exists: A match is found when the node has a label that has the specified key. ○ DoesNotExist: A match is found when the node does not have a label that has the specified key.
value	Specifies the value that is used to match the `value` of the label that has the specified key.
topology	Specifies the topology of devices on the node. `topology.devices` specifies the paths of local disks on the node. The specified disks are added to the VolumeGroup.

2. Add labels to the nodes.
   - Add a custom label to storage nodes based on the label that is specified in Step 1. This allows you to select nodes that meet the topological requirements. The label that is specified in Step 1 is as follows: `kubernetes.io/storagetype=lvm`.
   - Add the `alibabacloud.com/edge-enable-localstorage='true'` label to storage nodes. This allows the local storage manager to schedule the nodes.

   The **node-resource-manager** component on the node automatically creates a physical volume based on the preceding configurations and adds the physical volume to the VolumeGroup.

Use LVM to manage local storage

Use the following YAML file to define a PVC that specifies the StorageClass. Run the `kubectl apply -f ****.yaml` command to create the PVC. One PVC corresponds to one logical volume on the node. After the pod is created, the logical volume is mounted on the pod.

> **Note** In this example, the default `storageClassName` is `csi-local-lvm`.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: lvm-pvc-test
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 50Mi
  storageClassName: csi-local-lvm
---
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    k8s-app: local-test
  name: local-test
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      k8s-app: local-test
  template:
    metadata:
      labels:
        k8s-app: local-test
    spec:
      hostNetwork: true
      containers:
      - image: nginx:1.15.7-alpine
        imagePullPolicy: IfNotPresent
        name: nginx
        resources: {}
        volumeMounts:
          - name: local-pvc
            mountPath: /data
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      tolerations:
      - operator: Exists
      nodeSelector:
        alibabacloud.com/is-edge-worker: "true"
      volumes:
      - name: local-pvc
        persistentVolumeClaim:
          claimName: lvm-pvc-test
```

Run the following command to check whether the logical volume is mounted:

```
kubectl exec -it local-test-564dfcf6dc-qhfsf sh
/ # ls /data
```

Expected output:

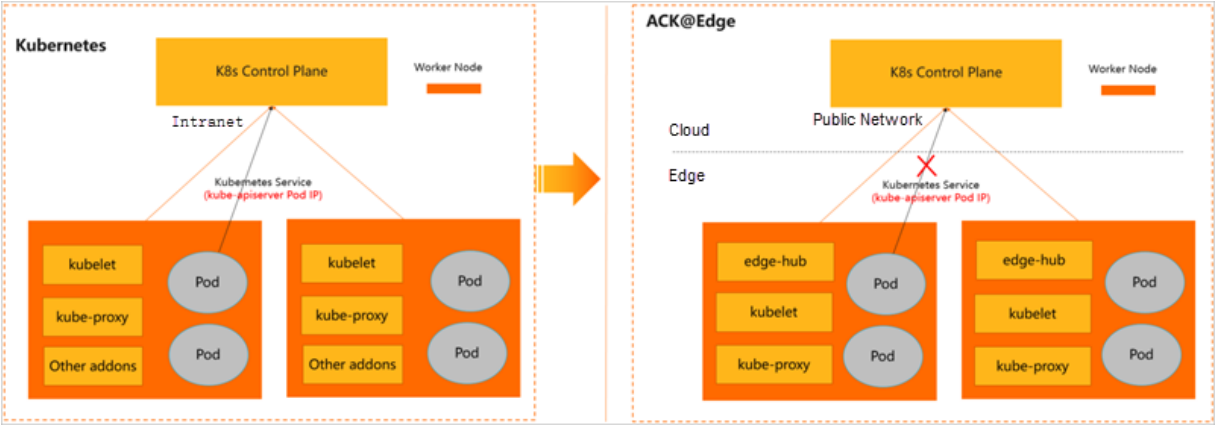
```
lost+found
```

The output indicates that the logical volume is mounted on the pod.

8.4. Run application pods that use InClusterConfig at the edge without making pod-facing changes

ACK@Edge is the first cloud-native edge computing service that coordinates workloads in the cloud and at the edge based on a non-intrusive approach. The service allows you to deploy application pods at the edge and set these pods to use InClusterConfig to access the Kubernetes API server without making pod-facing changes. This topic describes how to run application pods that use InClusterConfig at the edge.

Context



The following issues arise when you want to deploy application pods in an open source Kubernetes cluster to the edge and set these pods to use InClusterConfig to access the Kubernetes API server:

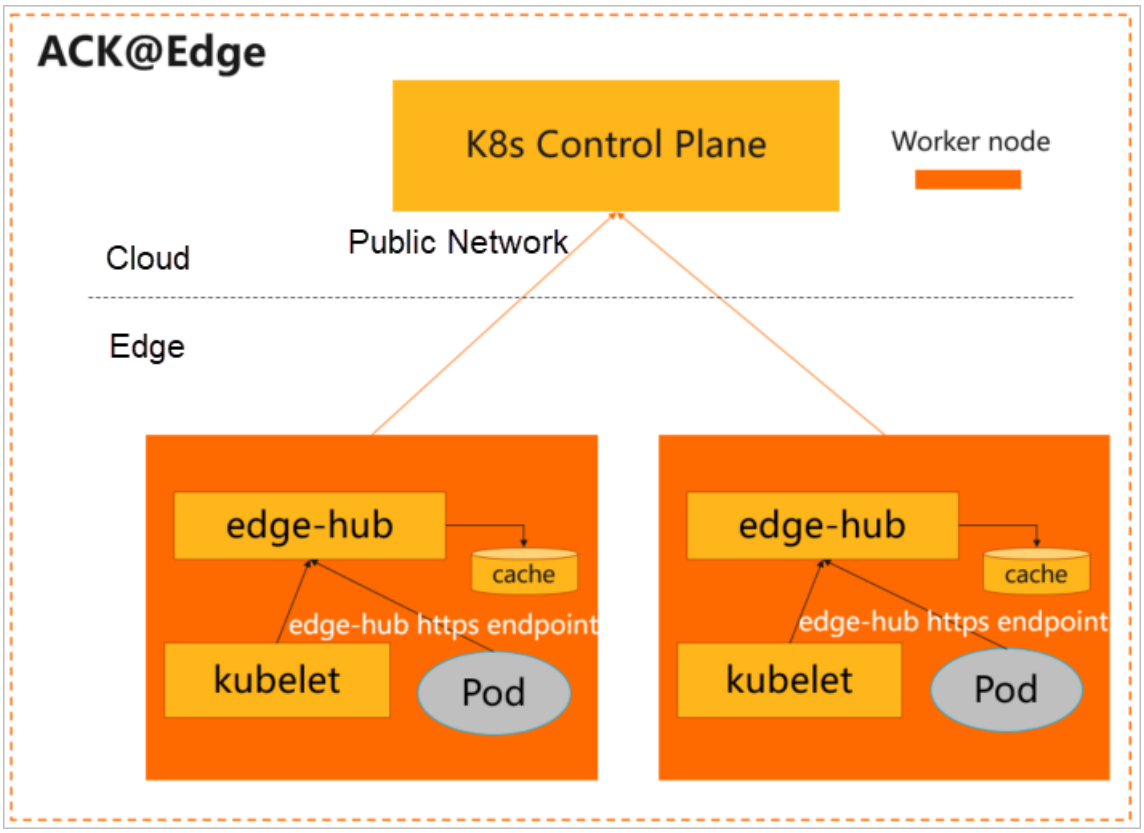
- Issue 1: Application pods access the Kubernetes API server through the addresses in InClusterConfig. The default load balancing rules (iptables/ipvs) configured on the node forward external requests to the application pods of the Kubernetes API server through their IP addresses. However, the pods at the edge and the Kubernetes API server in the cloud belong to different networks. Therefore, the pods at the edge cannot access the IP addresses of pods in the cloud. As a result, the application pods at the edge cannot use InClusterConfig to access the Kubernetes API server.
- Issue 2: After Issue 1 is resolved, if the application pods are restarted due to network jitters in the cloud, the pods at the edge cannot retrieve workload configurations from the Kubernetes API server. This affects the restart of application pods.

For more information about how to access the API from a pod, see Accessing the API from a Pod.

Solutions

You can enable the edge-hub of ACK@Edge on edge nodes to resolve the preceding issues based on a non-intrusive approach. Then, you can set application pods at the edge to use InClusterConfig to access the Kubernetes API server without making pod-facing changes. Take note of the following details:

- The endpoints of pods deployed at the edge are automatically changed from environment variables (KUBERNETES_SERVICE_HOST and KUBERNETES_SERVICE_PORT) to the HTTPS endpoint of edge-hub (`KUBERNETES_SERVICE_HOST=169.254.2.1` and `KUBERNETES_SERVICE_PORT=10268`) without the awareness of the application pods. This way, the application pods can use InClusterConfig to access the Kubernetes API server through edge-hub. This resolves the first issue.
- You must enable the caching feature of edge-hub. This way, application pods can retrieve data from the local cache when they are restarted. This resolves the second issue.



For more information about how to enable the caching feature of edge-hub, see Enable the caching feature of edge-hub.

Enable the caching feature of edge-hub

> **Note**
> - We recommend that you do not enable caching for pods that receive a large number of list or watch requests because data is cached on local disks.
> - You must restart the pods after the caching feature is enabled on the pods.

1. Obtain the User-Agent header.

   The User-Agent header can be found in the startup command of the application pod.

```
apiVersion: v1
kind: Pod
metadata:
  name: edge-app-pod
spec:
  containers:
    - name: "edge-app"
      image: "xxx/edge-app-amd64:1.18.8"
      command:
        - /bin/sh
        - -ec
        - |
          # The User-Agent header is found in the startup command: edge-app
          /usr/local/bin/edge-app --v=2
```

You can also find the User-Agent header in the edge-hub log, for example, `{User-Agent} watch {resource)`. The following command line is an example.

```
I0820 07:50:18.899015       1 util.go:221] edge-app get services: /api/v1/services/xxx
with status code 200, spent 21.035061152ms
```

2. Enable the caching feature of edge-hub.

   To enable the caching feature of edge-hub, add the User-Agent header included in the request destined for application pods to the cache_agents field in the edge-hub-cfg ConfigMap.

   The following YAML template provides an example:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: edge-hub-cfg
  namespace: kube-system
data:
  # This caches data when the edge-app pod whose User-Agent header is edge-app accesses
the Kubernetes API server.
  # Restart the application pod after caching is enabled.
  cache_agents: "edge-app"   # Separate multiple components with commas (,).
```

3. Check whether data returned by the application pod is cached.

   Check whether cache data exists in the */etc/kubernetes/cache/{User-Agent}* directory on the node that hosts the application pod.

9.Observability

9.1. Use Log Service to collect log data from containers of ACK edge clusters

Container Service for Kubernetes (ACK)@Edge is integrated with Log Service. When you create an ACK edge cluster, you can enable Log Service to collect log data from containers of the ACK edge cluster. Kubernetes writes log data to the standard output and text files. This topic describes how to collect log data from containers of an ACK edge cluster by using Log Service.

Step 1: Install Logtail

When you create an edge cluster, select **Enable Log service** to install Logtail. You can also install Logtail for an existing edge cluster.

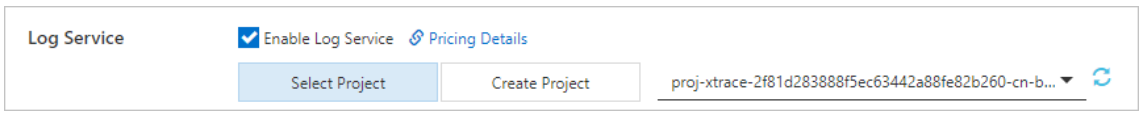
Install Logtail when you create a cluster

1.
2.
3. In the upper-right corner of the **Clusters** page, click **Create Kubernetes Cluster**.

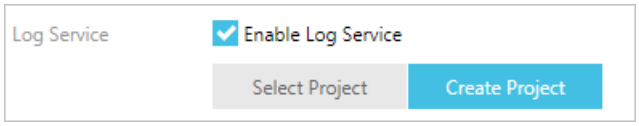
   In this example, only the steps to enable Log Service are described. For more information about how to create an ACK cluster, see Create a managed edge Kubernetes cluster.
4. On the **Component Configurations** wizard page, select **Enable Log Service** to install Logtail in the cluster to be created.

   If you select the **Enable Log Service** check box, the system prompts you to create a Log Service project. For more information about Log Service projects, see Project. You can use one of the following methods to create a Log Service project:

   - Click **Select Project** and select an existing project to manage the collected log.

   - Click **Create Project**. Then, a project named `k8s-log-{ClusterID}` is automatically created to manage the collected log. ClusterID indicates the unique ID of the edge cluster to be created.

5. After you set the parameters, click **Create Cluster** in the lower-right corner. In the message that appears, click **OK**.

   After the edge cluster is created, you can find the cluster that has Logtail enabled on the Clusters page.

Install Logtail for an existing cluster:

1.
2.
3.
4. In the left-side navigation pane of the details page, choose **Operations > Add-ons**. In the **Logs and Monitoring** section, find logtail-ds.

   > Note
   > - If the version of your edge cluster is later than 1.18.8-aliyunedge.1, the Log Service component is logtail-ds.
   > - If the version of your edge cluster is 1.18.8-aliyunedge.1 or earlier, the Log Service component consists of alibaba-log-controller and logtail-ds-docker.

5. Click **Install** next to the Log Service component.
6. In the **Install** message, click **OK**.

   If an earlier version of the Log Service component is installed, click **Upgrade** next to the component.

Step 2: Configure Log Service when you create an application

You can configure Log Service to collect log data from containers of an edge cluster when you create an application.

Step 3: View log data

The following example shows how to view the log data of a Tomcat application that is created by using the wizard in the console. The Log data of the Tomcat application is stored in Log Service. You can log on to the Log Service console to view log data collected from the containers. To view the log data, perform the following steps:

10.Windows containers

10.1. Add a Windows node to an ACK edge cluster

Container Service for Kubernetes (ACK) allows you to use a node pool to manage multiple nodes in a cluster as a group. For example, you can centrally manage the labels and taints of the nodes in a node pool. This topic describes how to add an existing Windows node to an ACK edge cluster.

Prerequisites

- An ACK edge cluster is created. For more information, see Create a managed edge Kubernetes cluster.
- The Windows license of the Windows node is valid.

Limits

- Make sure that you have a sufficient node quota in the cluster. To add more nodes, to apply for a quota increase. For more information about the quota limits related to ACK edge clusters, see Limits.
- Only Windows Server 2019 is supported.
- You can add both Windows nodes and Linux nodes to an ACK edge cluster. For more information about how to add a Linux node to an ACK edge cluster, see Add an edge node.
- You can deploy only workloads that support the HostNetwork mode on Windows nodes.

Step 1: Enable the Containers feature

Launch Windows PowerShell on the Windows node that you want to add to the cluster and run the following command to enable the Containers feature. For more information about how to launch Windows PowerShell, see Installing Windows PowerShell.

```
Install-WindowsFeature Containers
```

Expected output:

```
Success   Restart Needed   Exit Code        Feature Result
True      Yes              SuccessRest...   {Containers}
WARNING: You must restart this server to finish the installation process.
```

The output shows that you must restart the Windows node to finish the installation process.

Step 2: Add a Windows node

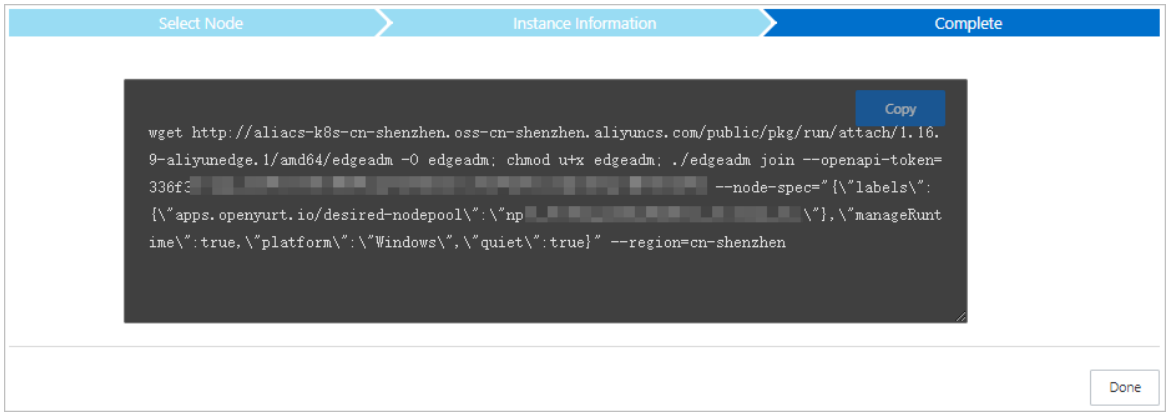
1.
2.
3.
4.
5. On the **Node Pools** page, find the node pool to which you want to add a Windows node and

choose **More > Add Existing Node** in the **Actions** column.

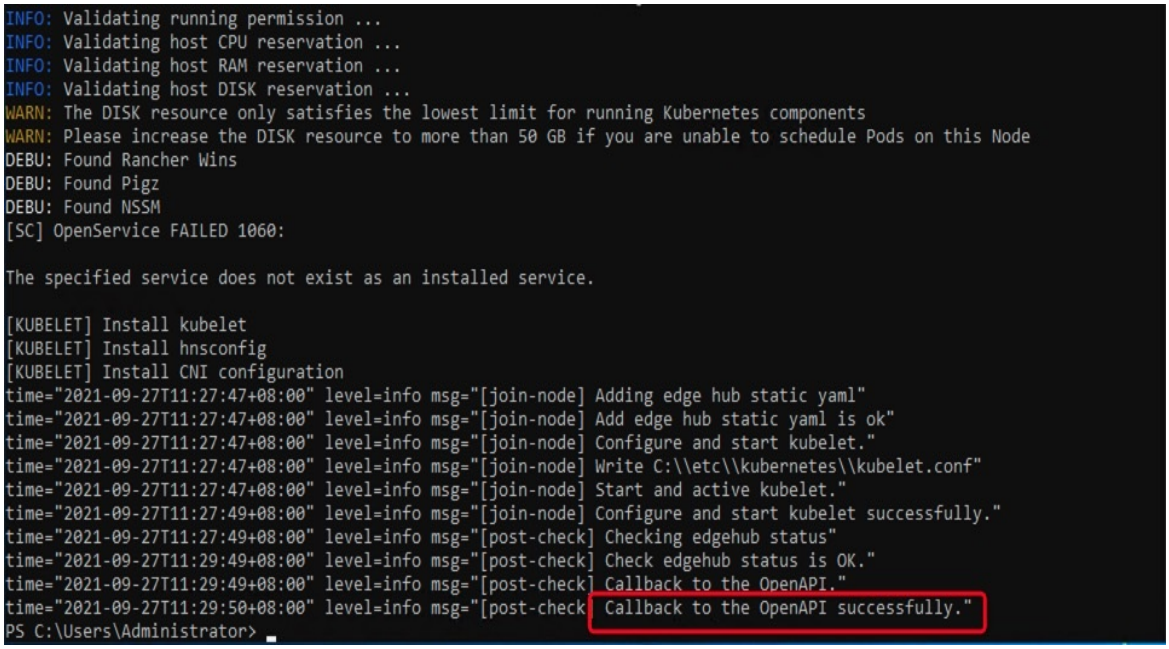
6. On the **Select Existing ECS Instance** wizard page, set **Mode** to **Manual** and select the Elastic Compute Service (ECS) instance that you want to add to the cluster from the ECS instances list.
7. Click **Next Step** to go to the **Specify Instance Information** wizard page.

Parameter	Description	Example
Cluster ID/Name	Information about the cluster to which you want to add the node. This parameter is automatically set.	c593a437a5e754c65876c3f47a8bd**** / testcluster
Script Validity Period	The default value of **Script Validity Period** is 1 hour. If you want to use the script for more than 1 hour, set the validity period to a proper value. If you set **Script Validity Period** to 0, the script is permanently valid.	1
Architecture	The CPU architecture of the node that you want to add. Select AMD64/X86_64 if you want to add a Windows node.	AMD64/X86_64
Specifications	The configurations of the node that you want to add. Refer to the sample configurations when you add a Windows node. For more information about the configuration parameters, see Parameters.	`{ "quiet": true, "platform": "Windows" }`

8. Click **Next Step**. On the **Complete** wizard page, click **Copy** to copy the script. Log on to the Windows node that you want to add and launch Windows PowerShell to run the script.



The message in the following figure appears after the node is added to the cluster.



9. On the **Complete** wizard page, click **Done**.

Step 3 (optional) : Add the Windows node again or remove the Windows node

If you want to remove the Windows node or add the node again when the system fails to add the node, perform the following steps:

1. Run the following command on the Windows node to clear the node data:

```
Start-BitsTransfer -Source http://aliacs-k8s-cn-hangzhou.oss-cn-hangzhou.aliyuncs.com/public/pkg/run/attach/{cluster_version}/windows/edgeadm -Destination edgeadm.exe; ./edgeadm.exe reset
```

> **Note** Replace `{cluster_version}` with the Kubernetes version of the cluster. Example: 1.18.8-aliyunedge.1.

2. Add the Windows node again or remove the Windows node.
   - You must clear the data on the Windows node before you add the node again. For more information about how to add a Windows node, see Step 2: Add a Windows node.
   - You must clear the data on the Windows node before you remove the node. For more information about how to remove a Windows node, see Remove a node.

10.2. Create an application on a Windows node

This topic describes how to use an orchestration template to create a Web application on a Windows node. A Deployment is created to provision pods for the application.

Prerequisites

A Windows node is added to the edge Kubernetes cluster. For more information about how to add a Windows node to an edge Kubernetes cluster, see Add a Windows node to an ACK edge cluster.

Context

In an orchestration template, you must define the resource objects that are required for running an application and configure mechanisms such as label selectors to manage the resource objects that make up the application.

Limits

You can deploy only workloads that support the HostNetwork mode on Windows nodes.

Procedure

1.
2.
3.
4.
5. On the **Deployments** page, click **Create from YAML** in the upper-right corner.
6. Set the parameters and click **Create**.
   - **Namespace**: Select the namespace to which the resource objects belong. By default, the default namespace is selected. Most resources are scoped to namespaces, except for underlying computing resources, such as nodes and persistent volumes (PVs).
   - **Sample Template**: ACK provides YAML templates of various resource types. This simplifies the deployment of resource objects. You can also create a custom template based on YAML syntax to define the resources that you want to create.
   - **Add Deployment**: This feature allows you to define a YAML template.
   - **Use Existing Template**: You can import an existing template to the configuration page.
   - **Save As**: You can save the template that you have configured.

   The following template shows how to deploy a Web application on a Windows node. You can use this template to quickly create a Web application.

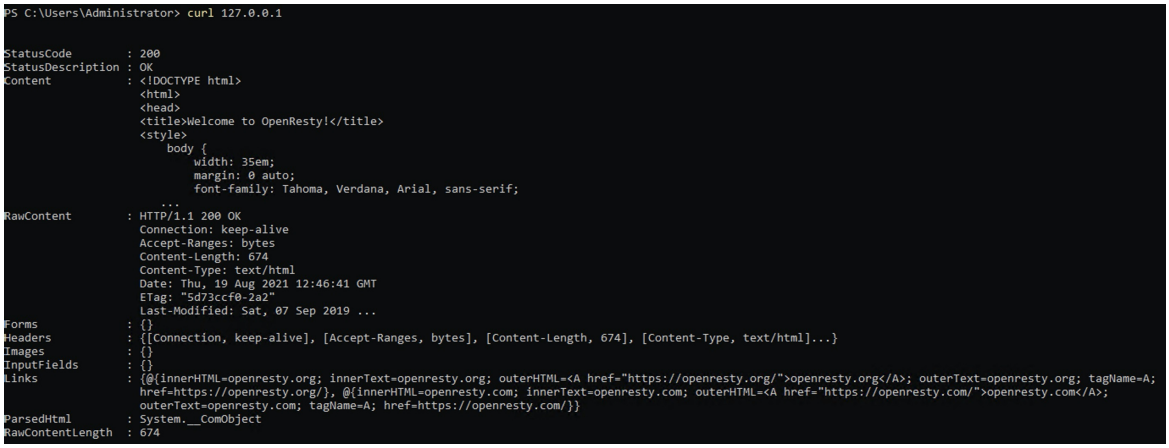
```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: web-windows
  name: web-windows
spec:
  selector:
    matchLabels:
      app: web-windows
  template:
    metadata:
      labels:
        app: web-windows
    spec:
      restartPolicy: Always
      hostNetwork: true
      terminationGracePeriodSeconds: 30
      tolerations:
      - key: os
        value: windows
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: kubernetes.io/os
                operator: In
                values:
                - windows
      containers:
        - image: registry.cn-hangzhou.aliyuncs.com/acs/sample-web-windows:v1.0.1
          name: windows
          ports:
          - containerPort: 80
            protocol: TCP
```

7. Click **Create**. A message that indicates the deployment status appears. After the Deployment is created, you can find the Deployment on the **Deployments** page.
8. Log on to the Windows node, launch Windows PowerShell, and run the following command to access the Web application.

   For more information about how to launch Windows PowerShell, see Installing Windows PowerShell.

   ```
   curl 127.0.0.1
   ```

   Expected output:


PS C:\Users\Administrator> curl 127.0.0.1
StatusCode : 200
StatusDescription : OK
Content : <!DOCTYPE html>
<html>
<head>
<title>Welcome to OpenResty!</title>
<style>
body {
width: 35em;
margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif;
...
RawContent : HTTP/1.1 200 OK
Connection: keep-alive
Accept-Ranges: bytes
Content-Length: 674
Content-Type: text/html
Date: Thu, 19 Aug 2021 12:46:41 GMT
ETag: "5d73ccf0-2a2"
Last-Modified: Sat, 07 Sep 2019 ...
Forms : {}
Headers : {[Connection, keep-alive], [Accept-Ranges, bytes], [Content-Length, 674], [Content-Type, text/html]...}
Images : {}
InputFields : {}
Links : {@{innerHTML=openresty.org; innerText=openresty.org; outerHTML=<A href="https://openresty.org/">openresty.org</A>; outerText=openresty.org; tagName=A;
href=https://openresty.org/}, @{innerHTML=openresty.com; innerText=openresty.com; outerHTML=<A href="https://openresty.com/">openresty.com</A>;
outerText=openresty.com; tagName=A; href=https://openresty.com/}}
ParsedHtml : System.__ComObject
RawContentLength : 674