ALIBABA CLOUD

Alibaba Cloud

Elastic Container Instance

Kubernetes

Document Version: 20201123

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.Kubernetes and ECI

Alibaba Cloud Elastic Container Instance (ECI) provides a basic environment for running pods of Kubernetes. However, you still need to use Kubernetes to configure features such as business dependencies, load balancing, auto scaling, and scheduling. This topic describes how to connect ECI to Kubernetes clusters and use ECI to run pods.

Connect ECI to Kubernetes

ECI provides a hierarchical solution for Kubernetes. In this solution, ECI schedules and manages underlying pods. On the Platform as a Service (PaaS) layer, Kubernetes manages workloads, such as deployments, services, StatefulSets, and cron jobs.

ECI can manage the underlying infrastructure of pods and provide resources for pods as needed. In this case, Kubernetes does not need to deploy or start pods or pay attention to the resources of the underlying virtual machines (VMs).

You can use ECI in Kubernetes in one of the following ways:

- Use only ECI to run pods of a Kubernetes cluster. All pods of the Kubernetes cluster are run in ECI. With ECI, you do not need to maintain underlying virtual machines (VM) or plan the capacity of the cluster.
- Use both ECI and traditional servers to run pods of a Kubernetes cluster. The hybrid use of ECI and traditional servers improves the resource utilization and scaling efficiency of the Kubernetes cluster and reduces the costs of auto scaling and jobs.

Use only ECI to run pods of a Kubernetes cluster based on Alibaba Cloud Serverless Kubernetes

You can run all pods of a Kubernetes cluster in ECI. The maintenance-free feature of ECI frees you from the maintenance work. Kubernetes only needs to manage workloads to ensure that your business applications run stably.

Alibaba Cloud Serverless Kubernetes allows you to create a serverless Kubernetes cluster built on ECI. For more information, see Serverless Kubernetes and Use ECI in Serverless Kubernetes.

We recommend that you create a serverless Kubernetes cluster in Serverless Kubernetes. Serverless Kubernetes provides a maintenance-free and cost-effective Kubernetes environment for you to run online or offline business applications and perform simulation, development, and testing.

Use both ECI and traditional servers to run pods of a Kubernetes cluster

If you have created a Kubernetes cluster, you can connect ECI to the cluster through virtual nodes. This improves resource utilization and scaling efficiency of the Kubernetes cluster and reduces operating costs. When the traffic of some long-running workloads increases, you can schedule the traffic to ECI, which shortens the time required to perform a scale-out, reduces the scale-out costs, and improves the utilization rate of existing resources.

When the traffic decreases, the Kubernetes cluster can release the pods deployed in ECI to reduce costs.

You can connect ECI to an existing Kubernetes cluster in one of the following ways based on the type of the cluster:

- To connect ECI to a managed Kubernetes cluster in Alibaba Cloud Container Service for Kubernetes, add virtual nodes in the Container Service console. For more information, see Managed Kubernetes and Use ECI in Container Service for Kubernetes.
- To connect ECI to a Kubernetes cluster created on Elastic Compute Service (ECS) instances, install Virtual Kubelet in the cluster. For more information, see Use ECI in a user-created Kubernetes cluster through Virtual Kubelet.
- To connect ECI to a Kubernetes cluster deployed in an on-premises data center or other clouds, install Virtual Kubelet in the cluster. Virtual Kubelet allows you to use the scalable resources of Alibaba Cloud in the Kubernetes cluster. For more information, contact your Alibaba Cloud architects.

2.Migrate existing Kubernetes applications to ECI

You can use Elastic Container Instance (ECI) with Alibaba Cloud Serverless Kubernetes and Alibaba Cloud Container Service for Kubernetes. This topic describes the adjustment required for migrating existing applications to ECI, and the issues that you may encounter during the migration.

Prerequisites

You understand basic concepts of Kubernetes or have used a Kubernetes-based container orchestration service on a private or public cloud before.

Preparations

- Learn about the process of using ECI in Kubernetes. For more information, see Use ECI in Serverless Kubernetes and Use ECI in Container Service for Kubernetes.
- You do not need to create an ECI in advance. You only need to create a serverless Kubernetes cluster or install the ack-virtual-node add-on in an existing managed Kubernetes cluster.

Manage Kubernetes clusters and ECIs

- You can use the Container Service console to manage serverless Kubernetes clusters and managed Kubernetes clusters.
- You can use Alibaba Cloud Cloud Shell to manage Kubernetes clusters. For more information, see Use kubectl on Cloud Shell to manage Kubernetes clusters.
- You can use the kubectl client to connect to a remote Kubernetes cluster from your local device. For more information, see Connect to Kubernetes clusters through kubectl.

View created ECIs

- Log on to the ECI console. In the upper-left corner, select the target region from the drop-down list. The created ECIs appear. If a blank page appears, you need to apply for the permission to view ECIs.
- Log on to the Container Service console. In the left-side navigation pane, choose **Applications** > **Pods**. On the page that appears, select the target cluster and namespace from the drop-down lists in the upper-left corner. The created pods appear. Pods that are scheduled to the virtual-kubelet node are ECIs. Click **Details** in the Actions column of an ECI to view its details.

Limits on application migration

ECIs are not scheduled to the virtual-kubelet node in a **centralized** manner because this node is a virtual node used by ECI to interact with Kubernetes. Instead, ECIs are scattered in the whole resource pool of Alibaba Cloud.

Due to security issues of the Internet and limits of a virtual node, ECI does not support host-related features and DaemonSets. The following table lists the features that are not supported by ECI currently.

Feature	Description	Alternative

Feature	Description	Alternative
HostPath	Allows you to mount a file from the host to a container.	Use an emptyDir volume or Apsara File Storage NAS.
HostNetwork	Allows you to map a host port to a container.	Create services of the LoadBalancer type.
DaemonSet	Allows you to deploy a static pod on the host of a container.	Deploy multiple images in a pod by using sidecar containers.
Privileged permissions	Allows you to grant privileged permissions to a container.	Use securityContext to grant permissions to a pod.
Service of the NodePort type	Allows you to map a host port to a container.	Create services of the LoadBalancer type.

Remarks on application migration

- Serverless Kubernetes can share an image repository with Kubernetes clusters. You can upload your container images to this image repository in advance. We recommend that you use VPC addresses of images to accelerate image pulling, which are in the format of registry-vpc.xxx.
- Serverless Kubernetes clusters and virtual nodes support common controllers such as the deployment, ReplicaSet, job, CronJob, and StatefulSet controllers. Theoretically, applications of these types can run directly in Serverless Kubernetes clusters and on virtual nodes
- Serverless Kubernetes clusters and virtual nodes use PrivateZone to expose services. We recommend that you enable the PrivateZone service for a serverless Kubernetes cluster when you create the cluster.
- Serverless Kubernetes clusters and virtual nodes support services of the LoadBalancer type. You can change the type field of a service to LoadBalancer to migrate the service. For more information, see Access services by using SLB.

```
apiVersion: v1
kind: Service
metadata:
 labels:
  app: nginx
 name: nginx
 namespace: default
spec:
 externalTrafficPolicy: Cluster
 ports:
 - port: 80
  protocol: TCP
  targetPort: 80
 selector:
  app: nginx
 sessionAffinity: None
 type: LoadBalancer
```

3.Use ECI in Serverless Kubernetes

This topic describes how to use Elastic Container Instance (ECI) in Serverless Kubernetes. Serverless Kubernetes integrates Kubernetes clusters with Alibaba Cloud services, such as ECI. You can quickly create a serverless Kubernetes cluster and use ECI in the cluster.

Serverless Kubernetes overview

ECI is compatible with Kubernetes. Based on ECI, Alibaba Cloud provides the Serverless Kubernetes service for you to create Kubernetes clusters that are maintenance-free and fully managed.

Serverless Kubernetes is a serverless platform that is optimized for running containers. This platform provides powerful capabilities for managing Kubernetes clusters and handling workloads, such as deployments, StatefulSets, jobs, and cron jobs. This platform allows you to abstract the architecture of applications and components and eliminates the need of server management, such as server creation, infrastructure management, O&M, upgrade, and capacity planning. You can focus on applications, rather than the underlying resources, and only pay for the resources used by the applications. Serverless Kubernetes automatically scales in and out resources based on application types, and manages resources in a finer-grained way.

Serverless Kubernetes has the following benefits:

- Easy to use: You can create a serverless Kubernetes cluster within 1.5s and deploy an application in the cluster within 30s. You do not need to manage the Kubernetes cluster.
- Integration with ecosystems: You can use the native API operations of Kubernetes to benefit from the Kubernetes ecosystem. Serverless Kubernetes is seamlessly integrated with the Alibaba Cloud infrastructure and allows the communications between Kubernetes clusters and the existing applications in your Virtual Private Clouds (VPCs).
- Security isolation: Serverless Kubernetes provides powerful capabilities for strong pod isolation based on the elastic computing architecture.
- Billed in the pay-as-you-go mode: You are charged based on the CPU and memory used by pods. You can configure auto scaling policies for Serverless Kubernetes to automatically scale resources.

Scenarios

With the high portability and flexibility of containers and high scalability and isolation of the elastic computing architecture of Alibaba Cloud, Serverless Kubernetes can be used in a variety of scenarios. Serverless Kubernetes is an agile service that can be used in deployment of web applications and back-end services of mobile applications, multimedia processing, data processing, and continuous integration. It is especially suitable for processing batch jobs and burst traffic.

Get started with ECI and Serverless Kubernetes

For more information about how to use ECI in Serverless Kubernetes, see Get started with ECI or Serverless Kubernetes Quick Start.

For more information about Serverless Kubernetes, see Overview.

4.Use ECI in Container Service for Kubernetes

This topic describes how to connect Elastic Container Instance (ECI) to a Container Service for Kubernetes cluster by installing a virtual node. An ECI serves as a pod in the Kubernetes cluster. If your Kubernetes cluster needs a scale-out, you can create ECIs on demand on the virtual node without the need to plan the computing capacity for the node. The ECIs can communicate with the existing pods in the Kubernetes cluster.

Overview

For more information about how virtual nodes work, see How Virtual Kubelet works. Virtual nodes are charged based on pods in the pay-as-you-go mode. For more information, see Billing.

Install a virtual node

Container Service for Kubernetes cluster

You can deploy Virtual Kubelet in a Container Service for Kubernetes cluster by installing the ack-virtual-node add-on.

Deploy Virtual Kubelet

1. Log on to the Container Service console.
2. In the left-side navigation pane, choose **Marketplace** > App Catalog. On the App Catalog page that appears, search for **ack-virtual-node** and click the card of the add-on that is found.
3. On the **App Catalog - ack-virtual-node** page that appears, click the **Parameters** tab and set the parameters for deploying Virtual Kubelet.

Parameter	Description	How to obtain the value
ECI_ACCESS_KEY	The AccessKey ID of your Alibaba Cloud account.	For more information, see How can I obtain an AccessKey pair?
ECI_SECRET_KEY	The AccessKey secret of your Alibaba Cloud account.	For more information, see How can I obtain an AccessKey pair?
ALIYUN_CLUSTERID	The ID of the Kubernetes cluster where Virtual Kubelet is to be deployed.	Click the target cluster name on the Clusters page. On the **Basic Information** page that appears, copy the value of **Cluster ID** in the **Basic Information** section.
ECI_REGION	The ID of the region where the Kubernetes cluster resides.	Click the target cluster name on the Clusters page. On the **Basic Information** page that appears, copy the value of **Region** in the **Basic Information** section.

Parameter	Description	How to obtain the value
ECI_VPC	The Virtual Private Cloud (VPC) where the Kubernetes cluster resides.	Click the target cluster name on the Clusters page. On the **Basic Information** page that appears, copy the value of **VPC** in the Cluster Resources section.
ECI_VSWITCH	The default VSwitch to be used by Virtual Kubelet to create ECIs.	Click a node ID on the **Nodes** page. On the **Instance Details** page that appears, copy the value of **VSwitch** in the **Configuration Information** section.
ECI_SECURITY_GROUP	The default security group to be used by Virtual Kubelet to create ECIs.	Click a node ID on the **Nodes** page. Click **Security Groups** in the left-side navigation pane. Copy the value of **Security Group ID** for the corresponding security group on the **Security Groups** tab.

4. In the **Deploy** pane on the right, select the target cluster and click **Create**. The Namespace and Release Name parameters are automatically set to kube-system and ack-virtual-node, respectively, which cannot be changed.

Verify the installation

Check the node list

After the virtual node is installed, choose **Cluster** > Nodes in the left-side navigation pane. Verify that the virtual-kubelet node appears on the **Nodes** page.

You can run the kubectl command to view the information about nodes.

```
$ kubectl get node
NAME                 STATUS  ROLES   AGE  VERSION
cn-beijing.192.168.0.238  Ready   <none>  66d   v1.14.6-aliyun.1
cn-beijing.192.168.0.239  Ready   <none>  66d   v1.14.6-aliyun.1
cn-beijing.192.168.0.240  Ready   <none>  66d   v1.14.6-aliyun.
virtual-kubelet          Ready   agent    19d   v1.11.2
```

Schedule pods to the virtual node

After the virtual node is installed, you can schedule pods to the virtual node. Virtual Kubelet automatically creates the corresponding ECIs.

1. In the Container Service console, choose **Applications** > Deployments in the left-side navigation pane. On the Deployments page that appears, select the target cluster and namespace, and click **Create from Template** in the upper-right corner.

2. Select a sample template or Custom from the Sample Template drop-down list, and click **Create**. For example, you can create a deployment by using the following YAML file:

```
apiVersion: apps/v1beta2 # For versions earlier than 1.8.0, use apps/v1beta1.
kind: Deployment
metadata:
 name: nginx-deployment-basic
 labels:
  app: nginx
spec:
 replicas: 2
 selector:
  matchLabels:
   app: nginx
 template:
  metadata:
   labels:
    app: nginx
  spec:
   containers:
   - name: nginx
    image: nginx
    imagePullPolicy: Always
   nodeName: virtual-node-eci-0 # The name of the virtual node.
```

3. On the **Deployments** page, verify that the deployment appears in the deployment list, as shown in the following figure.

4. In the left-side navigation pane, choose **Applications** > **Pods**. If the page shown in the following figure appears, the pods are scheduled to the virtual node.

Schedule pods to a virtual node

If a virtual node exists in your Kubernetes cluster, you can schedule pods to the virtual node. Virtual Kubelet automatically creates the corresponding ECIs. You can schedule a pod to the virtual node in one of the following ways:

- Add a label to the namespace to which the pod belongs.
- Add a label to the pod.

For more information about how to use virtual nodes, see Virtual nodes.

5.Use ECI in a user-created Kubernetes cluster through Virtual Kubelet

This topic describes how to use Elastic Container Instance (ECI) in a user-created Kubernetes cluster on Alibaba Cloud through Virtual Kubelet.

Deploy Virtual Kubelet in a user-created Kubernetes cluster

You can deploy Virtual Kubelet in a user-created Kubernetes cluster by using a YAML file.

Before you begin

Obtain the information described in the following table before you deploy Virtual Kubelet.

Parameter	Description	How to obtain the value
ECI_ACCESS_KEY	The AccessKey ID of your Alibaba Cloud account.	For more information, see How can I obtain an AccessKey pair?
ECI_SECRET_KEY	The AccessKey secret of your Alibaba Cloud account.	For more information, see How can I obtain an AccessKey pair?
ALIYUN_CLUSTERID	The ID of the Kubernetes cluster where Virtual Kubelet is to be deployed.	You can specify a unique ID for the Kubernetes cluster.
ECI_REGION	The ID of the region where ECIs are to be created.	Log on to the ECI console and obtain the ID of the region. For example, cn-beijing represents the China (Beijing) region.
ECI_VPC	The Virtual Private Cloud (VPC) where ECIs are to be created.	Log on to the VPC console and obtain the ID of an appropriate VPC in the selected region.
ECI_VSWITCH	The default VSwitch to be used by Virtual Kubelet to create ECIs.	Log on to the VPC console. In the left-side navigation pane, click VSwitches. On the VSwitches page, obtain the ID of an appropriate VSwitch in the selected VPC.

Parameter	Description	How to obtain the value
ECI_SECURITY_GROUP	The default security group to be used by Virtual Kubelet to create ECIs.	Log on to the VPC console. On the VPCs page, click the ID of the selected VPC to go to the VPC Details page. In the Network Resources section of the page, click the number next to Security Groups. On the Security Groups page of the Elastic Compute Service (ECS) console, obtain the ID of an appropriate security group.

YAML file

```
$ cat deployment-vk.yaml
apiVersion: v1
kind: ServiceAccount
metadata:
 name: virtual-node-sa
 namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
 name: virtual-node-role-binding
roleRef:
 apiGroup: rbac.authorization.k8s.io
 kind: ClusterRole
 name: cluster-admin
subjects:
 - kind: ServiceAccount
   name: virtual-node-sa
   namespace: kube-system
---
apiVersion: apps/v1beta2
kind: Deployment
metadata:
 name: virtual-node-controller
 namespace: kube-system
 labels:
   app: virtual-node-controller
spec:
 replicas: 1
```

```
replicas: 1
selector:
 matchLabels:
  app: virtual-node-controller
template:
 metadata:
  labels:
   app: virtual-node-controller
 spec:
  serviceAccount: virtual-node-sa
  containers:
  - name: alicloud-virtual-kubelet
   image: registry.cn-hangzhou.aliyuncs.com/acs/virtual-nodes-eci:v1.0.0.1-aliyun
   imagePullPolicy: Always
   args: ["--provider", "alibabacloud"]
   env:
   - name: KUBELET_PORT
    value: "10250"
   - name: VKUBELET_POD_IP
    valueFrom:
     fieldRef:
      fieldPath: status.podIP
   - name: VKUBELET_TAINT_KEY
    value: "virtual-kubelet.io/provider"
   - name: VKUBELET_TAINT_VALUE
    value: "alibabacloud"
   - name: VKUBELET_TAINT_EFFECT
    value: "NoSchedule"
   - name: ECI_REGION
    value: ${aliyun_region_name}
   - name: ECI_VPC
    value: ${aliyun_vpc_id}
   - name: ECI_VSWITCH
    value: ${aliyun_vswitch_id}
   - name: ECI_SECURITY_GROUP
    value: ${aliyun_sg_id}
   - name: ECI_ACCESS_KEY
    value: ${aliyun_access_key}
   - name: ECI_SECRET_KEY
    value: ${aliyun_secret_key}
   - name: ALIYUN_CLUSTERID
```

```
      value: ${custom_define_cluster_id}
```

Replace the following content in the YAML file with the obtained information:

- `aliyun_region_name`
- `aliyun_vpc_id`
- `aliyun_vswitch_id`
- `aliyun_sg_id`
- `aliyun_access_key`
- `aliyun_secret_key`

Set the specified ID for your Kubernetes cluster and replace `custom_define_cluster_id` with the ID.

Deploy Virtual Kubelet

```
# View the information about nodes in the cluster before you deploy Virtual Kubelet.
[root@k8s-master01 ~]#
[root@k8s-master01 ~]# kubectl get node -o wide
NAME          STATUS ROLES   AGE VERSION INTERNAL-IP  EXTERNAL-IP OS-IMAGE           KERNEL-VERS
ION         CONTAINER-RUNTIME
k8s-master01  Ready  <none>  50d v1.14.2  192.168.0.15  <none>      CentOS Linux 7 (Core) 3.10.0-957
.21.3.el7.x86_64  docker://18.9.6
k8s-master02  Ready  <none>  50d v1.14.2  192.168.0.16  <none>      CentOS Linux 7 (Core) 3.10.0-957
.21.3.el7.x86_64  docker://18.9.6
k8s-master03  Ready  <none>  50d v1.14.2  192.168.0.17  <none>      CentOS Linux 7 (Core) 3.10.0-957
.21.3.el7.x86_64  docker://18.9.6
# Deploy Virtual Kubelet.
[root@k8s-master01 ~]# kubectl apply -f deployment-vk.yaml
serviceaccount/virtual-node-sa created
clusterrolebinding.rbac.authorization.k8s.io/virtual-node-role-binding created
deployment.apps/virtual-node-controller created
# Check whether Virtual Kubelet is deployed.
[root@k8s-master01 ~]# kubectl get deploy/virtual-node-controller -n kube-system
NAME                 READY  UP-TO-DATE  AVAILABLE  AGE
virtual-node-controller  1/1    1           1          161m
# View the information about nodes in the cluster after you deploy Virtual Kubelet.
[root@k8s-master01 ~]# kubectl get node -o wide
NAME          STATUS ROLES   AGE VERSION INTERNAL-IP  EXTERNAL-IP OS-IMAGE           KERNEL-VERS
ION         CONTAINER-RUNTIME
k8s-master01  Ready  <none>  50d v1.14.2  192.168.0.15  <none>      CentOS Linux 7 (Core) 3.10.0-95
7.21.3.el7.x86_64  docker://18.9.6
k8s-master02  Ready  <none>  50d v1.14.2  192.168.0.16  <none>      CentOS Linux 7 (Core) 3.10.0-95
7.21.3.el7.x86_64  docker://18.9.6
k8s-master03  Ready  <none>  50d v1.14.2  192.168.0.17  <none>      CentOS Linux 7 (Core) 3.10.0-95
7.21.3.el7.x86_64  docker://18.9.6
virtual-kubelet  Ready  agent  82m v1.11.2  172.30.176.3  <none>      <unknown>          <unknown>
<unknown>
```

Verify the installation

1. Create a pod on the specified virtual node.

```
$ cat test-pod.yaml
apiVersion: v1
kind: Pod
metadata:
 name: test-pod
spec:
 containers:
 - name: centos
  image: centos:latest
  args:
  - /bin/sh
  - -c
  - date; sleep 6000h
 nodeName: virtual-kubelet
# Create a pod.
[root@k8s-master01 ~]# kubectl create -f test-pod.yaml
pod/test-pod created
```

2. Check whether the pod is created.

```
[root@k8s-master01 ~]# kubectl get pod/test-pod -o wide
NAME    READY  STATUS   RESTARTS  AGE  IP           NODE           NOMINATED NODE  READINESS GA
TES
test-pod  1/1    Running  0         95s  192.168.6.165  virtual-kubelet  <none>          <none>
# Virtual Kubelet adds annotations to the pod running on the virtual-kubelet node.
[root@k8s-master01 ~]# kubectl  get pod -o yaml test-pod
apiVersion: v1
kind: Pod
metadata:
 annotations:
  ProviderCreate: done
  k8s.aliyun.com/eci-instance-id: eci-2zeaak7c7i6xb5uqnw4m
```

3. View the corresponding container group in the ECI console.

6.Pod annotations supported by ECI

When you schedule pods of a Kubernetes cluster to Elastic Container Instance (ECI) through a virtual node, you can add annotations to the pods to use the various features of ECI. When you add annotations, make sure that they comply with Kubernetes semantics. This topic describes the annotations supported by ECI.

> **Note** These annotations only take effect for the pods to be scheduled to a **virtual node**. They are invalid for the pods to be scheduled to Elastic Compute Service (ECS) instances.

Specify the specifications of an ECI

You can add the `k8s.aliyun.com/eci-use-specs` annotation to specify the specifications of the ECI to be created for running a pod. You can specify the specifications in a list and separate them with commas (,). Each element in the list represents a set of specifications. When instances of the specifications specified by an element are out of stock, the specifications specified by the next element are used.

You can specify each element in the list in one of the following ways.

1. Specify the CPU, memory, and GPU specifications.

- `${cpu}-${mem}Gi` : specifies the CPU and memory specifications. For example, 2-4Gi indicates that the ECI to be created has `2 vCPUs and 4 GB memory` .
- `ecigpu-${gpuType}-${gpuCount}` : specifies the GPU specifications. Example: ecigpu-P100-4. **ecigpu** specifies that a GPU instance is to be created. **P100** specifies the GPU type. **4** specifies the number of GPUs that you want to request for the ECI. The following table describes the supported values of gpuType and gpuCount and the mapped ECS instances types.

gpuType	gpuCount	ECS instance type
P4	1	ecs.gn5i-c2g1.large
P4	2	ecs.gn5i-c16g1.8xlarge
P4	4	ecs.gn5i-c28g1.14xlarge
V100	1	ecs.gn6v-c8g1.2xlarge
V100	4	ecs.gn6v-c8g1.8xlarge
V100	8	ecs.gn6v-c8g1.16xlarge
P100	1	ecs.gn5-c8g1.2xlarge
P100	2	ecs.gn5-c8g1.4xlarge
P100	4	ecs.gn5-c8g1.8xlarge
P100	8	ecs.gn5-c8g1.14xlarge

2. Specify the exact ECS instance type, for example, ecs.c6.xlarge. For more information about ECS instance types supported by ECI, see Instance families.

```
apiVersion: apps/v1beta2 # For versions earlier than 1.8.0, use apps/v1beta1.
kind: Deployment
metadata:
 name: vk-cos-use
 labels:
  app: cos
spec:
 replicas: 1
 selector:
  matchLabels:
   app: cos
 template:
  metadata:
    annotations:
      "k8s.aliyun.com/eci-use-specs": "64-512Gi, 2-4Gi,ecs.c6.xlarge,ecigpu-P100-4"
    labels:
      app: cos
  spec:
   containers:
   - name: u1
    image: "registry-vpc.cn-beijing.aliyuncs.com/lxx/cos-4g"
   nodeName: virtual-node-eci-0
```

Specify an image cache to accelerate pod creation

When you create a pod, you can use an image cache to accelerate pod creation. For more information, see Use an image cache CRD to accelerate pod creation.

Specify an existing image cache

When you create a deployment, you can specify an existing image cache to accelerate deployment creation.

```
apiVersion: apps/v1beta2 # For versions earlier than 1.8.0, use apps/v1beta1.
kind: Deployment
metadata:
 name: nginx-with-imagecache
spec:
 replicas: 1
 selector:
  matchLabels:
   app: nginx
 template:
  metadata:
   annotations:
     "k8s.aliyun.com/eci-image-snapshot-id": "${your_image_cache_id}"
   labels:
     app: nginx
  spec:
   containers:
   - name: nginx
    image: nginx:1.7.9
    imagePullPolicy: IfNotPresent
   nodeName: virtual-node-eci-0
```

Enable automatic image cache match

When you create a deployment, you can enable automatic image cache match. The system selects the optimal image cache from existing image caches to accelerate deployment creation.

```
apiVersion: apps/v1beta2 # For versions earlier than 1.8.0, use apps/v1beta1.
kind: Deployment
metadata:
 name: nginx-dynamic-image-cache
 labels:
  app: nginx-dynamic-image-cache
spec:
 replicas: 1
 selector:
  matchLabels:
   app: nginx
 template:
  metadata:
    annotations:
      "k8s.aliyun.com/eci-image-cache": "true"
    labels:
      app: nginx
  spec:
   containers:
   - name: nginx
    image: nginx:latest
    imagePullPolicy: IfNotPresent
   nodeName: virtual-node-eci-0
```

Associate an EIP with an ECI

You can associate an Elastic IP Address (EIP) with an ECI to enable Internet access for the ECI. For more information, see What are Elastic IP Addresses?

Enable automatic EIP creation

When you create a deployment, you can enable automatic EIP creation to create an EIP and associate the EIP with the ECI for running the deployment.

The settings of an EIP created in this way are as follows:

- Default bandwidth: **5 Mbit/s**. You can change the bandwidth through an annotation.
- Billing method: **pay-as-you-go**.

```
apiVersion: apps/v1beta2 # For versions earlier than 1.8.0, use apps/v1beta1.
kind: Deployment
metadata:
 name: cos-vk-resource-group-id
 labels:
  app: vk
spec:
 replicas: 1
 selector:
  matchLabels:
   app: nginx
 template:
  metadata:
    annotations:
      "k8s.aliyun.com/eci-with-eip": "true"
      "k8s.aliyun.com/eip-bandwidth": "10"
    labels:
      app: nginx
  spec:
   containers:
   - name: nginx
    image: nginx:latest
   nodeName: virtual-node-eci-0
```

Specify an existing EIP

When you create a deployment, you can select an existing EIP and associate the EIP with the ECI for running the deployment.

```
apiVersion: apps/v1beta2 # For versions earlier than 1.8.0, use apps/v1beta1.
kind: Deployment
metadata:
 name: cos-vk-resource-group-id
 labels:
  app: vk
spec:
 replicas: 1
 selector:
  matchLabels:
   app: nginx
 template:
  metadata:
    annotations:
      "k8s.aliyun.com/eci-eip-instanceid": "${your_eip_Instance_Id}"
    labels:
      app: nginx
  spec:
   containers:
   - name: nginx
    image: nginx:latest
   nodeName: virtual-node-eci-0
```

Specify one or more NTP servers

You can specify one or more NTP servers for an ECI through an annotation. For more information, see Set one or more NTP servers for pods.

```
apiVersion: apps/v1beta2 # For versions earlier than 1.8.0, use apps/v1beta1.
kind: Deployment
metadata:
 name: set-ngnix-ntp
 labels:
  app: vk
spec:
 replicas: 1
 selector:
  matchLabels:
   app: nginx
 template:
  metadata:
    annotations:
      "k8s.aliyun.com/eci-ntp-server": 100.100.5.1,100.100.5.2  # Specify the IP addresses of your NTP servers
.
    labels:
      app: nginx
  spec:
   containers:
   - name: nginx
    image: nginx:latest
   nodeName: virtual-node-eci-0
```

Specify the resource group to which an ECI belongs

You can specify the resource group to which an ECI belongs through an annotation.

```
apiVersion: apps/v1beta2 # For versions earlier than 1.8.0, use apps/v1beta1.
kind: Deployment
metadata:
 name: cos-vk-resource-group-id
 labels:
  app: vk
spec:
 replicas: 1
 selector:
  matchLabels:
   app: nginx
 template:
  metadata:
    annotations:
      "k8s.aliyun.com/eci-resource-group-id" : "${your_resource_group_id}"
    labels:
      app: nginx
  spec:
   containers:
   - name: nginx
    image: nginx:latest
   nodeName: virtual-node-eci-0
```

Assign a RAM role to an ECI

You can assign an existing Resource Access Management (RAM) role to an ECI to allow the ECI to access other Alibaba Cloud services.

```
apiVersion: apps/v1beta2 # For versions earlier than 1.8.0, use apps/v1beta1.
kind: Deployment
metadata:
 name: set-ram-role
 labels:
  app: vk
spec:
 replicas: 1
 selector:
  matchLabels:
   app: nginx
 template:
  metadata:
    annotations:
      "k8s.aliyun.com/eci-ram-role-name" : "${your_ram_role_name}"
    labels:
      app: nginx
  spec:
   containers:
   - name: nginx
    image: nginx:latest
   nodeName: virtual-node-eci-0
```

Specify the VSwitch used to create an ECI

When you deploy Virtual Kubelet, you can set the `ECI_VSWITCH` environment variable to specify the default VSwitch used to create ECIs. When you create deployments on the virtual node, the default VSwitch is used to create ECIs. To create an ECI by using another VSwitch in the same Virtual Private Cloud (VPC), you can specify the VSwitch through an annotation.

```
apiVersion: apps/v1beta2 # For versions earlier than 1.8.0, use apps/v1beta1.
kind: Deployment
metadata:
 name: set-vswitch
 labels:
  app: nginx
spec:
 replicas: 1
 selector:
  matchLabels:
   app: nginx
 template:
  metadata:
    annotations:
      "k8s.aliyun.com/eci-vswitch" : "${your_vsw_id}"
    labels:
      app: nginx
  spec:
   containers:
   - name: nginx
    image: nginx:latest
   nodeName: virtual-node-eci-0
```

Specify the security group used to create an ECI

When you deploy Virtual Kubelet, you can set the `ECI_SECURITY_GROUP` environment variable to specify the default security group used to create ECIs. When you create deployments on the virtual node, the default security group is used to create ECIs. To create an ECI by using another security group in the same VPC, you can specify the security group through an annotation.

```
apiVersion: apps/v1beta2 # For versions earlier than 1.8.0, use apps/v1beta1.
kind: Deployment
metadata:
 name: set-security-group
 labels:
  app: nginx
spec:
 replicas: 1
 selector:
  matchLabels:
   app: nginx
 template:
  metadata:
    annotations:
      "k8s.aliyun.com/eci-security-group" : "${your_security_group_id}"
    labels:
      app: nginx
  spec:
   containers:
   - name: nginx
    image: nginx:latest
   nodeName: virtual-node-eci-0
```

7.Install the virtual-kubelet-autoscaler add-on in a Container Service for Kubernetes cluster

Alibaba Cloud provides the virtual-kubelet-autoscaler add-on for Kubernetes. If pods fail to be scheduled to existing Elastic Compute Service (ECS) worker nodes due to insufficient resources, the virtual-kubelet-autoscaler add-on reschedules the pods to virtual nodes provided by Elastic Container Instance (ECI).

Prerequisites

- A managed Kubernetes cluster is created in Container Service for Kubernetes and a virtual node is deployed in the cluster. For more information, see Create a managed Kubernetes cluster and Virtual nodes.

Install the virtual-kubelet-autoscaler add-on from the application marketplace

1. Log on to the Container Service console.
2. In the left-side navigation pane, choose **Cluster** > Cluster. On the Clusters page that appears, find the cluster in which you want to install the virtual-kubelet-autoscaler add-on and take note of the name of the cluster.
3. In the left-side navigation pane, choose **Marketplace** > App Catalog. On the App Catalog page that appears, search for ack-virtual-kubelet-autoscaler and click the card of the add-on that is found.
4. In the **Deploy** pane on the right, select the cluster in which you want to install the virtual-kubelet-autoscaler add-on and click Create.

Verify that the add-on is installed

In the left-side navigation pane, choose **Applications** > Deployments. On the **Deployments** page that appears, select the kube-system namespace from the drop-down list. Verify that the virtual-kubelet-autoscaler application appears in the list.

After the add-on is installed, the managed Kubernetes cluster can schedule pods to the virtual node provided by ECI when resources on existing worker nodes are insufficient. For more information, see Schedule pods to virtual nodes through the virtual-kubelet-autoscaler add-on.

8.Run elastic workloads in a Container Service for Kubernetes cluster

This topic describes how to install the ack-kubernetes-elastic-workload add-on in a Container Service for Kubernetes cluster and run elastic workloads.

Prerequisites

- A Container Service for Kubernetes cluster is created. For more information, see Create a Kubernetes cluster.

Install the ack-kubernetes-elastic-workload add-on

1. Log on to the Container Service console.

2. In the left-side navigation pane, choose Cluster > Cluster. On the Clusters page, find the cluster in which you want to install the ack-kubernetes-elastic-workload add-on and take note of the name of the cluster.

3. In the left-side navigation pane, choose Marketplace > App Catalog. On the App Catalog page, search for ack-kubernetes-elastic-workload and click the card of the add-on that is found.

4. In the Deploy pane on the right, select the cluster in which you want to install the ack-kubernetes-elastic-workload add-on and click Create.

5. In the left-side navigation pane, choose Applications > Deployments. On the Deployments page, select the cluster where the add-on is installed and the kube-system namespace from the drop-down lists. Verify that the deployment named ack-kubernetes-elastic-workload-controller-manager appears in the list.

Run an elastic workload

Kubernetes schedules each workload and manages the lifecycle of workloads. To run an elastic workload, Kubernetes must be able to perform the following operations:

1. Change the scheduling policy when the number of replicas reaches a certain value.

2. Preferentially destroy certain pods during lifecycle management.

This section demonstrates how to define and run an elastic workload and explains how Kubernetes schedules the replicas of the elastic workload.

Create a deployment by using the following template:

```
apiVersion: apps/v1 # for versions before 1.8.0 use apps/v1beta1
kind: Deployment
metadata:
 name: nginx-deployment-basic
 labels:
  app: nginx
spec:
 replicas: 2
 selector:
  matchLabels:
   app: nginx
 template:
  metadata:
   labels:
    app: nginx
  spec:
  # nodeSelector:
  #   env: test-team
   containers:
   - name: nginx
    image: nginx:1.7.9 # replace it with your exactly <image_name:tags>
    ports:
    - containerPort: 80
```

Create an elastic workload by using the following template:

```
# Define an elastic workload.
apiVersion: autoscaling.alibabacloud.com/v1beta1
kind: ElasticWorkload
metadata:
 name: elasticworkload-sample
spec:
 sourceTarget:
  name: nginx-deployment-basic
  kind: Deployment
  apiVersion: apps/v1
  min: 2
  max: 4
 replicas: 6
 elasticUnit:
 - name: virtual-kubelet
  labels:
   virtual-kubelet: "true"
  annotations:
   virtual-kubelet: "true"
  nodeSelector:
   type: "virtual-kubelet"
  tolerations:
  - key: "virtual-kubelet.io/provider"
   operator: "Exists"
  # min: the minimum number of replicas that must be scheduled to the elastic unit. You can specify the mini
mum number and maximum number of replicas for each elastic unit.
  # max: 10
```

The method of using an elastic workload is similar to that of using Horizontal Pod Autoscaler (HPA). An elastic workload scales out pods on the virtual node connected to your cluster, which has no impact on existing workloads.

Typically, the definition of an elastic workload consists of the following information:

1. SourceTarget: defines the type of the source workload and the allowed number of replicas.

2. elasticUnit: defines the scheduling policies for elastic units in an array. If you need to specify multiple elastic units, define the parameters of each elastic unit in the same order as that in the template.

Based on the preceding definition, the source deployment can have two to four replicas. When the number of replicas does not exceed four, the replicas are scheduled in the source deployment. When more than four replicas are required, certain replicas are scheduled to the elastic unit virtual-kubelet. The elastic unit virtual-kubelet has its own scheduling policy, including the labels, annotations, nodeSelector, affinity, and tolerations.

Kubernetes monitors the source deployment and generates cloned replicas on the elastic unit based on the scheduling policy configured for the elastic unit. When the number of replicas of the elastic workload changes, Kubernetes dynamically schedules replicas to the source deployment and elastic unit.

After you run the elastic workload template, you can run a command to view the status of the elastic workload. The Desired Replicas parameter for each unit in the Status section indicates the number of replicas scheduled to the unit.

```
kubectl describe ew elasticworkload-sample  # same as kubectl get elasticworkload
Name:       elasticworkload-sample
Namespace:  default
Labels:     <none>
Annotations: <none>
API Version: autoscaling.alibabacloud.com/v1beta1
Kind:       ElasticWorkload
Metadata:
 Creation Timestamp: 2020-05-06T03:43:41Z
 Generation:       27
 Resource Version:  20635284
 Self Link:        /apis/autoscaling.alibabacloud.com/v1beta1/namespaces/default/elasticworkloads/elasticw
orkload-sample
 UID:              0e9205ff-38b8-43b7-9076-ffa130f26ef4
Spec:
 Elastic Unit:
  Annotations:
   Virtual - Kubelet: true
  Labels:
   Virtual - Kubelet: true
  Name:            demo
  Node Selector:
   Type: virtual-kubelet
  Tolerations:
   Key:     virtual-kubelet.io/provider
   Operator: Exists
 Replicas:    6
 Source Target:
  API Version: apps/v1
  Kind:       Deployment
  Max:        2
  Min:        0
  Name:       nginx-deployment-basic
Status:
```

```
  Elastic Units Status:
    Desired Replicas:  4
    Name:              nginx-deployment-basic-unit-virtual-kubelet
    Update Timestamp:  2020-05-07T12:38:27Z
  Replicas:            6
  Selector:            app=nginx
  Source Target:
    API Version:       apps/v1
    Desired Replicas:  2
    Kind:              Deployment
    Name:              nginx-deployment-basic
    Update Timestamp:  2020-05-07T12:38:27Z
Events:                <none>
```

After the elastic workload template is run, you can run a command to view the status of pods. Based on the command output, the source deployment is cloned to generate new deployments and pods. The number of pods for each deployment is allocated based on the definition of the elastic workload.

```
kubectl get pod -o wide
NAME                                          READY   STATUS    RESTARTS   AGE    IP             NODE                 NOMINATED NODE   RE
ADINESS GATES
nginx-deployment-basic-7ff9955f89-djxwv       1/1     Running   0          138m   172.20.1.151   cn-hangzhou.10.0.
5.212   <none>           <none>
nginx-deployment-basic-7ff9955f89-hrw2z       1/1     Running   0          138m   172.20.1.27    cn-hangzhou.10.0.5
.208    <none>           <none>
nginx-deployment-basic-unit-demo-8bb586568-4f8xt   1/1     Running   0          138m   10.1.76.63     virtual-node-
eci-1   <none>           <none>
nginx-deployment-basic-unit-demo-8bb586568-bl5pd   1/1     Running   0          138m   10.1.76.65     virtual-node-
eci-0   <none>           <none>
nginx-deployment-basic-unit-demo-8bb586568-ndbp8   1/1     Running   0          138m   10.1.76.64     virtual-node
-eci-0  <none>           <none>
nginx-deployment-basic-unit-demo-8bb586568-vx9jx   1/1     Running   0          138m   10.1.76.62     virtual-node-
eci-2   <none>           <none>
```

In addition, you can use the elastic workload with HPA, as shown in the following configuration file. HPA scales replicas of the elastic workload based on the metric of CPU usage, and dynamically adjusts the number of replicas on each unit. For example, if the number of replicas needs to be reduced from six to four, HPA preferentially destroys replicas on the elastic unit.

```
apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
 name: elastic-workload-demo
 namespace: default
spec:
 scaleTargetRef:
  apiVersion: autoscaling.alibabacloud.com/v1beta1
  kind: ElasticWorkload
  name: elasticworkload-sample
 minReplicas: 2
 maxReplicas: 10
 metrics:
 - type: Resource
  resource:
   name: cpu
   target:
    type: Utilization
    averageUtilization: 50
```

Note In conclusion, Kubernetes generates multiple deployments for an elastic workload by cloning the source deployment and overriding the scheduling policy of the source deployment. In this way, Kubernetes can change the scheduling policy when the number of replicas reaches a certain value. Based on the number of replicas required, Kubernetes can dynamically adjust the number of replicas in the source deployment and on the elastic unit. This allows Kubernetes to preferentially destroy certain pods during a scale-in.