容器服务

常见问题

腾讯云

版权所有：腾讯云计算（北京）有限责任公司

【版权声明】

©2013-2025 腾讯云版权所有

本文档（含所有文字、数据、图片等内容）完整的著作权归腾讯云计算（北京）有限责任公司单独所有，未经腾讯云事先明确书面许可，任何主体不得以任何形式复制、修改、使用、抄袭、传播本文档全部或部分内容。前述行为构成对腾讯云著作权的侵犯，腾讯云将依法采取措施追究法律责任。

【商标声明】

腾讯云

及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。未经腾讯云及有关权利人书面许可，任何主体不得以任何方式对前述商标进行使用、复制、修改、传播、抄录等行为，否则将构成对腾讯云及有关权利人商标权的侵犯，腾讯云将依法采取措施追究法律责任。

【服务声明】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。
您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100或95716。

文档目录

常见问题
TKE 标准集群
扩容缩容相关
集群相关
注册节点相关
如何选择 Containerd 和 Docker
自建 DNS 导致节点初始化异常
原生节点专用调度器相关
TKE Serverless 集群
TKE Serverless 集群相关
负载均衡相关
超级节点相关
镜像仓库相关
运维类
基础监控相关
日志采集相关
Prometheus 监控相关
服务类
Service/Ingress FAQ
镜像仓库类
远程终端类
事件类
资源管理类
Nginx Ingress FAQ

版权所有：腾讯云计算（北京）有限责任公司

常见问题
TKE 标准集群
扩容缩容相关

最近更新时间：2024-11-13 11:32:21

Cluster Autoscaler（CA）与基于监控指标的弹性伸缩的节点扩缩容有什么不同?

Cluster Autoscaler 确保集群中的所有 Pod 都可调度，不管具体的负载；而基于监控指标的节点弹性伸缩在自动扩缩时不关心 Pod，可能会添加一个没有任何 Pod 的节点，或者删除一个有一些系统关键 Pod 的节点，例如 kube-dns。Kubernetes 不鼓励这种自动缩容机制，故 Cluster Autoscaler 与基于监控指标的弹性伸缩的节点互相冲突，请不要同时启用。

CA 和伸缩组的对应关系是什么?

启用 CA 的集群会根据选择的节点配置，创建一个启动配置和绑定此启动配置的伸缩组。绑定后，将会在此伸缩组内进行扩缩容，扩容后的 CVM 自动加入集群。自动扩缩容的节点都是按量计费的。伸缩组的相关文档请参见 弹性伸缩文档 。

容器服务控制台手动添加的节点是否会 CA 缩容?

不会，CA 缩容的节点只限于伸缩组内的节点。在 容器服务控制台 添加的节点不会加入到伸缩组中。

弹性伸缩控制台是否可以添加或者移出云服务器?

不可以，不建议您在 弹性伸缩控制台 进行任何修改操作。

扩缩容会继承所选节点的哪些配置?

创建伸缩组时，需要选择集群内的一个节点作为参考来创建 启动配置 ，参考的节点配置包括：

- vCPU
- 内存
- 系统盘大小
- 数据盘大小
- 磁盘类型
- 带宽
- 带宽计费模式
- 是否分配公网IP
- 安全组
- 私有网络
- 子网

如何使用多个伸缩组?

根据服务的重要级别、类型等特点，您可以通过创建多个伸缩组，为伸缩组设置不同的 label，从而指定伸缩组扩容出节点的 label，来对服务进行分类。

扩缩容最大值可以设置为多少?

目前腾讯云用户每个可用区均有30个按量计费类型 CVM 配额，如果希望伸缩组有超过 30 台按量计费的 CVM，请 在线咨询 申请。
具体配额请参见您当前可用区的云服务器 实例数及配额 。另外弹性伸缩也有最大值的限制，其最大值为200。如果希望弹性伸缩超过最大值，请 在线咨询 申请。

集群启用缩容是否安全?

由于在缩容节点时会发生 Pod 重新调度的情况，所以服务必须可以容忍重新调度和短时的中断时再启用缩容。建议您为您的服务设置 PDB 。PDB 可以在任何时候指定一个处于运行状态的 Pod 集合副本的最小数量或者最小百分比。有了 PodDisruptionBudget，应用部署者可以保证同一时间内主动移除 Pod 的集群操作不会销毁过多 Pod，避免了因销毁过多 Pod 导致数据丢失、服务中断或者无法接受的服务降级等影响。

节点上有哪些类型的 Pod 时不会被缩容?

- 当您设置了严格的 PodDisruptionBudget 的 Pod 不满足 PDB 时，不会缩容。
- Kube-system 下的 Pod。
- 节点上有非 deployment，replica set，job，stateful set 等控制器创建的 Pod。
- Pod 有本地存储。
- Pod 不能被调度到其他节点上。

节点满足缩容条件后多长时间会触发缩容?

10分钟。

节点 Not Ready 后多长时间会触发缩容?

20分钟。

多长时间扫描一次是否需要扩缩容?

10秒。

需要多长时间才可以扩容出 CVM?

一般在10分钟内，相关弹性伸缩的说明文档请参见 弹性伸缩 。

为什么有 Unschedulable 的 Pod，却未进行扩容?

请确认以下原因：

- Pod 的请求资源是否过大。
- 是否设置了 node selector。
- 伸缩组的最大值是否已经达到。
- 账号余额是否充足（账号余额不足，弹性伸缩无法扩容），以及配额不足等其他原因，请参见 弹性伸缩故障处理 。

如何防止 Cluster Autoscaler 缩容特定节点?

```
# 可以在节点的annotations中设置如下信息
kubectl annotate node <nodename> cluster-autoscaler.kubernetes.io/scale-down-disabled=true
```

容器服务节点扩缩容事件如何查看?

您可在弹性伸缩控制台查询伸缩组的伸缩活动，也可查看 k8s 的事件。在以下三种资源上都会有对应的事件：

- kube-system/cluster-autoscaler-status config map

资源	事件
ScaledUpGroup	CA 触发扩容。
ScaleDownEmpty	CA 删除了一个没有运行 Pod 的节点。
ScaleDown	CA 缩容。

- node

资源	事件
ScaleDown	CA 缩容。
ScaleDownFailed	CA 缩容失败。

- pod

资源	事件
TriggeredScaleUp	CA 由于此 Pod 触发扩容。

NotTriggerScaleUp	CA 无法找到可扩容的伸缩组使得此 Pod 可调度。
ScaleDown	CA 尝试驱逐此 Pod 来缩容节点。

容器集群 kube-proxy 模式可以更改吗?

Kube-proxy 代理模式支持 iptables 和 ipvs，在创建集群的时候支持选择，集群创建完成后**不支持更换**。详情见 集群启用 IPVS 。

集群相关

最近更新时间：2024-10-24 10:54:52

创建集群常见问题

创建集群时，云服务器可以不选取公网 IP 么?

云服务器可以不选取公网 IP，无公网IP的云服务器只能拉取镜像仓库下我的镜像，不能拉取 dockerhub 以及第三方镜像。
无公网 IP，但有外网带宽的云服务器可以通过绑定弹性 IP 来访问 Internet。

创建集群时，选择所属网络的作用是什么?

选择的所属网络和子网，是集群内云服务器的所在子网，用户可以通过添加不同的云服务器到不同可用区的子网下进行跨可用区容灾。

创建集群支持什么类型的机型?

支持 CVM 提供的标准型、计算型、高 IO 型、GPU 型、黑石型中等机器，具体以 TKE 控制台展示的机型为主。

当前容器服务宿主机支持什么操作系统?

当前支持操作系统详情见 TKE 支持的 公共镜像列表 。

容器服务如何设置节点的自定义 Kubelet 参数?

- 该功能以白名单形式开放，您可以通过 提交工单 申请使用。
- 在 **新增节点** 页面、**添加已有节点**页面、新增节点池 页面设置节点的自定义 Kubelet 参数。如下图所示：

容器服务创建集群如何设置自定义 Kubernetes 组件参数?

具体操作步骤可参考 自定义 Kubernetes 组件启动参数 。

容器服务支持独立集群部署模式吗?

TKE 为您提供集群完全自主可控的 Master 独立部署模式。该模式下，Kubernetes 集群的 Master 和 Etcd 会部署在您购置的 CVM 上，并且您拥有 Kubernetes 集群的所有管理和操作权限。详情请参考 Master 独立部署模式 。

TKE Serverless 集群创建后支持修改可用区吗?

容器服务 TKE 的 Serverless 集群创建成功后，是不支持更换或者再添加其他可用区的。

TKE 集群支持获取客户端 ipv6 地址吗?

目前 TKE 集群不支持获取客户端 ipv6 地址。

TKE 集群可以导出配置吗?

TKE 容器服务器不支持将您现在的集群进行导出。

TKE 集群创建提示需完成节点异常检测 plus 参数设置

创建集群提示需要完成节点异常检测 plus 参数设置，需要您检查下是否有勾选 NodeProblemDetectorPlus 组件，如已勾选，则需要配置相关参数。组件详情见 NodeProblemDetectorPlus 说明 。

> **说明：**
> 如暂无法评估是否需要使用此组件，可以取消勾选，直接创建集群。后续如需使用相关功能，在组件管理中再次操作安装即可。

TKE 集群如何启用 IPVS?

具体操作可参考 集群启用 IPVS 。

容器服务如何查看集群访问凭证?

1. 登录 容器服务控制台 ，选择左侧导航栏中的**集群**，进入集群管理界面。
2. 单击需要连接的集群 ID/名称，进入集群详情页。
3. 选择左侧导航栏中的**基本信息**，即可在“基本信息”页面中查看“集群 APIServer 信息”模块中该集群的访问地址、外网/内网访问状态、Kubeconfig 访问凭证内容等信息。

容器服务怎么才能添加其他账号名下的服务器到集群?

目前不支持将其他账号下的云服务器添加到集群，当前仅支持添加同一 VPC 下的云服务器。

使用 K8S 官方 SDK 连接集群 API Server 报错 certificate verify failed: self signed certificate 如何处理?

可通过如下方案解决:

1. 新增跳过服务端校验的配置。
   若使用 nodejs sdk，需设置环境变量:

```
export NODE_TLS_REJECT_UNAUTHORIZED=“0”;
```

   若使用 python sdk，需在 kubeconfig 里设置:

```
clusters:
- cluster:
    insecure-skip-tls-verify: true
```

2. 编辑 Deployment default/kubernetes-proxy，在 .spec.template.spec.containers 中将 kubernetes-proxy 容器启动参数 `--CN=kubernetes` 修改为 `--CN=kubernetes-proxy` ，保存更新。

> **注意:**
> 此操作会重启 kubernetes-proxy pod，会导致已有连接断开，建议选择合适的时间操作。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  # ...
  name: kubernetes-proxy
  namespace: default
spec:
  # ...
  template:
    # ...
    spec:
      # ...
      containers:
      - args:
        - ./apiserver-proxy
        - start
        - --CN=kubernetes-proxy # 原为--CN=kubernetes
        - --port=443
        # ...
        name: kubernetes-proxy
```

集群网络常见问题

自定义 Webhook 开发时，针对集群网络需要注意什么?

在进行自定义 Webhook 开发时，不要拦截 kube-system namespace 下 Pod，否则会导致集群网络使用异常。

扩展云服务器常见问题

扩展云服务器有什么限制？

只能选择当前集群所在的地域，但可以选择不同的可用区，允许集群跨可用区部署。

云服务器的数量有限制么？

有，用户所有按量计费云服务器不能超过用户配额，具体详情请看 云服务器概览页 。

销毁云服务器常见问题

销毁云服务器后，该主机下部署的容器怎么办？

销毁云服务器时，该主机下的容器等资源也会随之销毁。若某服务的容器数量小于期望运行的容器数量时，集群将会在其他主机上启动容器，直到运行的容器数量等于期望运行容器数量为止。

注册节点相关

最近更新时间：2024-07-25 16:21:52

集群类

注册节点特性是否支持独立部署集群？

支持，您可以在托管形态和独立部署形态的集群中使用注册节点的特性。

使用注册节点特性，为什么要求集群内必须要存在云上节点？

由于注册节点所在网络与 VPC 网络的差异性，目前集群内部分系统组件必须运行在云上节点，所以当前要求集群内必须存在云上节点。

节点类

注册节点和云上节点在能力上有哪些差异？

注册节点和云上节点的功能差异，请参见 注册节点与云上节点能力对比 。

注册节点支持哪些操作系统？

为了保障注册节点的稳定性，当前注册节点的操作系统仅支持 TencentOS Server 3.1 和 TencentOS Server 2.4(TK4) 。

节点上由于 docker、containerd 相关软件导致添加节点失败，如何处理？

可以使用下载的脚本，执行以下清理的指令，再进行添加。

```
./add2tkectl-cls-m57oxxxp-np-xxxx clear
```

注册节点脚本安装过程中报错中断如何处理？

1. 报错提示 "nvidia nv_driver not installed"
   说明：NVIDIA-SMI 驱动未安装。
   解决方法：
   1.1 执行 clear 命令清理环境：

   ```
   ./add2tkectl-cls-m57oxxxp-np-xxxx clear
   ```

   1.2 参见 安装 NVIDIA Tesla 驱动 安装 NVIDIA 驱动，并执行 nvidia-smi 命令验证安装结果。
   1.3 重新执行注册节点安装命令。

   ```
   ./add2tkectl-cls-m57oxxxp-np-xxxx install
   ```

2. 报错提示 "Install gpu toolkit failed!"
   说明：nvidia toolkit 工具包安装失败。
   解决方法：
   2.1 执行 clear 命令清理环境：

   ```
   ./add2tkectl-cls-m57oxxxp-np-xxxx clear
   ```

   2.2 重新执行注册节点安装命令。

   ```
   ./add2tkectl-cls-m57oxxxp-np-xxxx install
   ```

3. 报错提示 "can not get nodes node-xxx gpu capacity after 60s"

说明：节点 gpu 容器能力初始化失败。
解决方法：

- 执行clear命令清理环境。

```
./add2tkectl-cls-m57oxxxp-np-xxxx clear
```

- 重新执行注册节点安装命令。

```
./add2tkectl-cls-m57oxxxp-np-xxxx install
```

网络、流量接入类

注册节点的容器如何对外暴露服务?

基于腾讯负载均衡 CLB，我们提供了注册节点的四层以及七层的流量接入方案，请参见 流量接入 。

> **注意：**
> 当前只有注册节点（专线版）支持接入 CLB，注册节点（公网版）暂时不支持 CLB。

运维类

注册节点的日志如何接入日志服务 CLS？

TKE 集群日志接入 CLS 后，默认支持集群中的注册节点，无需特殊配置。TKE 集群日志接入 CLS，请参见 日志采集 ，集群中的注册节点默认使用内网的方式（会占用专线带宽）进行日志投递。
如需要注册节点使用公网的方式投递日志，请按照如下步骤操作：

1. 修改 `kube-system` 命名空间下的 **externalnode-config** 配置文件（ConfigMap），通过更新 **clsPushMethod** 的值来配置注册节点日志投递的方式，有效值：
   - intranet：内网方式投递日志，默认值。
   - public：公网方式投递日志，按需修改，请确保注册节点有访问公网的能力。
2. 重建 `kube-system` 命名空间下 **tke-log-agent**（DaemonSet）所管理的 Pod，使上述配置生效。

注册节点如何接入 Prometheus 监控服务?

TKE 集群 Prometheus 监控服务，默认支持集群中的注册节点，无需特殊配置。TKE 接入 Prometheus 监控服务请参见 Prometheus 监控概述 。

> **注意：**
> 当前只有注册节点（专线版）支持接入 Prometheus 监控服务，注册节点（公网版）暂时不支持。

Cilium-Overlay 模式下如何创建 admission webhook?

集群创建时使用 Cilium-Overlay，如果在注册节点上部署了 admission webhook 组件，apiserver 访问 webhook 的 svc 会提示错误，例如：

```
Error from server (InternalError): error when creating "dev/manifests/pods/[redacted].yaml": Internal error occurred: failed calling webhook "[redacted]": failed to call webhook: Post "https://[redacted].default.svc:443/mutate-pods?timeout=2s": context deadline exceeded
```

这里的原因是 apiserver 托管在 tke meta 集群，并不在用户 vpc 的 overlay 网络中，无法通过 webhook 的 svc 访问到 webhook 的 pod ip，导致访问失败。
处理方式：将 webhook 的网络模式设置为 Hostnetwork，如下图所示：

```
    dnsPolicy: ClusterFirst
    hostNetwork: true
    restartPolicy: Always
    schedulerName: default-scheduler
```

如何选择 Containerd 和 Docker

最近更新时间：2024-10-09 15:31:21

如何选择 Containerd 和 Docker

如何选择运行时组件?

> **说明：**
> - Kubernetes 1.24通过 Dockershim 对 Docker 的支持已移除，新建节点的容器运行时请使用 Containerd，通过 Docker 构建的镜像可以继续使用。
> - Containerd 是更为稳定的运行时组件，支持 OCI 标准，但不支持 Docker API。

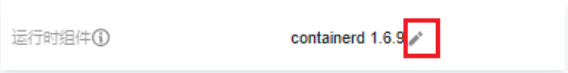
容器运行时（Container Runtime）是 Kubernetes（K8S）最重要的组件之一，负责管理镜像和容器的生命周期。Kubelet 通过 Container Runtime Interface (CRI) 与容器运行时交互，以管理镜像和容器。
TKE 支持用户选择 Containerd 和 Docker 作为运行时组件：

- Containerd：调用链更短，组件更少，更稳定，占用节点资源更少。建议选择 Containerd。
- Docker：在以下情况下，请选择 Docker 作为运行时组件：
  - 如需使用 Docker in Docker。
  - 如需在 TKE 节点使用 `docker build/push/save/load` 等命令。
  - 如需调用 Docker API。
  - 如需使用 Docker Compose 或 Docker Swarm。

如何修改运行时组件?

1. 登录 容器服务控制台，选择左侧导航栏中的**集群**。
2. 在集群管理列表，选择目标集群 ID，进入该集群基本信息页面。
3. 在集群信息中修改运行时组件。如下图所示：

> **说明：**
> 修改运行时组件及版本，只对集群内无节点池归属的增量节点生效，不会影响存量节点。



Containerd 和 Docker 组件常用命令是什么?

Containerd 不支持 docker API 和 docker CLI，但是可以通过 cri-tool 命令实现类似的功能。

镜像相关

镜像相关功能	Docker	Containerd
显示本地镜像列表	docker images	crictl images
下载镜像	docker pull	crictl pull
上传镜像	docker push	无
删除本地镜像	docker rmi	crictl rmi
查看镜像详情	docker inspect IMAGE-ID	crictl inspect IMAGE-ID

容器相关

容器相关功能	Docker	Containerd

显示容器列表	docker ps	crictl ps
创建容器	docker create	crictl create
启动容器	docker start	crictl start
停止容器	docker stop	crictl stop
删除容器	docker rm	crictl rm
查看容器详情	docker inspect	crictl inspect
attach	docker attach	crictl attach
exec	docker exec	crictl exec
logs	docker logs	crictl logs
stats	docker stats	crictl stats

POD 相关

POD 相关功能	Docker	Containerd
显示 POD 列表	无	crictl pods
查看 POD 详情	无	crictl inspectp
运行 POD	无	crictl runp
停止 POD	无	crictl stopp

调用链区别有哪些?

- Docker 作为 K8S 容器运行时，调用关系如下：
  `kubelet --> docker shim （在 kubelet 进程中） --> dockerd --> containerd`
- Containerd 作为 K8S 容器运行时，调用关系如下：
  `kubelet --> cri plugin（在 containerd 进程中） --> containerd`

> **说明：**
> 1. Docker 增加了 swarm cluster、docker build、docker API 等功能，但也会引入一些 bug，并且多了一层调用，包括 exec，preStop，ipv6，日志 stdout 的格式等。
> 2. Docker 运行时的 systemd 服务是 dockerd.service，而非 docker.service。

Stream 服务的区别

`kubectl exec/logs` 等命令需要 kubelet 在 apiserver 跟容器运行时之间建立流转发通道。

Stream 服务的原理是什么?

可以通过了解 `kubectl exec` 命令的原理来了解 CRI 的 Stream Service 是如何工作的：

1. dockershim 或 containerd 在启动后会监听某个端口，用以运行 Stream 服务。
2. 当执行 `kubectl exec` 等命令时，请求经过 kube-apiserver 找到 Pod 对应的节点，转发到 kubelet。
3. 此时 kubelet 会请求 CRI-Service（dockershim 或 containerd-cri）的 GetExec 接口，CRI-Server 会为本次请求生成一个随机的 Token，记录后和 CRI Stream server 监听端口组合成 URL，返回给 kubelet。
4. kubelet 将 kube-apiserver 发过来的 HTTP 请求升级为 websocket，并作为 apiserver 和 CRI Stream 之间的 proxy 转发数据。

如何在 Containerd 中使用并配置 Stream 服务?

Docker API 本身提供 stream 服务，Dockershim 位于 kubelet 内部有默认的配置 "127.0.0.1:0"。
Containerd 的 stream 服务需要单独配置：

```
[plugins.cri]
  stream_server_address = "127.0.0.1"
  stream_server_port = "0"
  enable_tls_streaming = false
```

K8S 1.11 前后版本配置区别是什么?

Containerd 的 stream 服务在 K8S 不同版本运行时场景下配置不同。

- 在 K8S 1.11 之前:
  Kubelet 不会做 stream proxy，只会做重定向。即 Kubelet 会将 containerd 暴露的 stream server 地址发送给 apiserver，并让 apiserver 直接访问 containerd 的 stream 服务。此时，您需要给 stream 服务转发器认证，用于安全防护。
- 在 K8S 1.11 之后:
  K8S 1.11 引入了 kubelet stream proxy，使 containerd stream 服务只需要监听本地地址即可。

容器 Exec 的区别

Docker 和 Containerd 在 Exec 的实现上略有区别，区别主要在 Execsync 的实现上，也即执行单条命令的情况。
因此 `kubectl exec` 命令在不指定参数 `-it` 时 和 pod lifecycle 中的 ExecProbe 在 Runtime 类型不同的节点上表现可能稍有不同。

kubectl exec 的区别

- `docker exec` 时会以当前 exec 首进程结束为 exec 结束的标志。
- `CRI exec` 会以当前 exec 中所有进程结束为本次 exec 结束。

例如 `kubectl exec <pod-id> -- bash -c "nohup sleep 10 &"` 命令，在 Runtime 是 Docker 的节点上会在两秒左右结束；而在 Containerd 的节点上需要等到 sleep 进程退出后才能结束。

pod exec probe 的区别

kubelet 在实现 exec probe 时使用了 CRI Runtime 的 ExecSync 接口，因此 exec probe 和 `kubectl exec # no -t -i` 的表现一致，也即:

- 在 Docker 节点上，exec probe 如果残留子进程仍会正常退出。
- 在 Containerd 节点上，exec probe 会等到 probe 中所有的进程退出再结束。

区别导致的影响主要出现在 pod lifecycle 的 postStartHook 和 preStopHook 中，如果在 hook 中使用 exec probe 并且出现残留子进程的情况，在 containerd 的节点上可能会遇到 Pod 长期卡在 containerCreating 状态。原因是 kubelet 在 syncPod 时会逐个容器拉起，并执行 probe，如果某个 probe 因上述原因阻塞住，会导致后续容器无法启动。
在 ExecProbe 中拉起子进程并退出父进程属于 K8S 中未定义的行为，具体表现可能会和运行时版本、种类相关，因此建议尽量不要在 probe 执行过于复杂的操作。

容器网络的区别

在正常情况下，Pod 中的容器会共享同一个 Network Namespace，因此 Pod 需要在创建 Sandbox 容器时将网络准备好。为了更好的说明区别，我们简单介绍下 Pod 网络初始化的流程:

1. kubelet 调用 CRI Runtime（dockershim 或 containerd）创建 Pod 的 Sandbox 容器。
2. CRI Runtime 调用底层 docker 或 containerd 创建 pause 容器（此时 pause 进程还没启动，但已经初始化 Network Namespace）。
3. CRI Runtime 调用 CNI 执行在 Network Namespace 中创建 veth 并加入 cbr0 网桥等网络初始化操作。
4. CRI Runtime 启动 pause 容器，pause 进程被拉起。
5. kubelet 继续调用 CRI Runtime 执行创建容器等后续操作。

Docker 和 containerd 在**创建 pause 容器并初始化 Network Namespace**和 **调用 CNI 初始化 veth** 这两步有区别。

创建 pause 容器

Containerd 是为了 kubernetes 设计的 CRI Runtime，没有独立的网络模块；但 Docker 在设计时带有自己的网络功能，因此 docker 在创建 Pause 容器时，会进行 Docker 特有的网络设置。该设置导致和 Containerd 最大的区别是在不使用 IPv6 的情况下，Docker 会将容器 Network Namespace 中内核参数 `net.ipv6.conf.all.disable_ipv6` 设置为1，也即关闭容器内的 ipv6 选项。
同样的 Pod 在 Docker 的节点上只开启了 IPv4，而在 Containerd 的节点上会同时开启 IPv4 和 IPv6。
同时开启 IPv4 和 IPv6 的情况中，DNS 解析可能会同时发出v4、v6两个版本的包。在某些情况，业务如果需要频繁进行 DNS 解析，可能会触发 DNS 解析库的 Bug（取决于 Pod 业务的实现时的依赖）。在 Containerd 节点上，可以通过给 Pod 添加 init container 来针对 Pod 关闭 IPv6 设置。代码如

下：

```
apiVersion: v1
kind: Pod
...
spec:
  initContainers:
  - image: busybox
    name: sysctl
    command: ["sysctl", "-w", "net.ipv6.conf.all.disable_ipv6=1"]
    securityContext:
      privileged: true
...
```

调用 CNI

两者在调用 CNI 上没有实质区别。

对比项	Docker	Containerd
谁负责调用 CNI	Kubelet 内部的 docker-shim	Containerd 内置的 cri-plugin

如何配置 CNI	Kubelet 参数 `--cni-bin-dir` 和 `--cni-conf-dir`	Containerd 配置文件（toml）：
		`[plugins.cri.cni]`
		`bin_dir = "/opt/cni/bin"`
		`conf_dir = "/etc/cni/net.d"`

容器日志的区别

对比项	Docker	Containerd
存储路径	如果 Docker 作为 K8S 容器运行时，容器日志的落盘将由 docker 来完成，保存在类似 `/var/lib/docker/containers/$CONTAINERID` 目录下。Kubelet 会在 `/var/log/pods` 和 `/var/log/containers` 下面建立软链接，指向 `/var/lib/docker/containers/$CONTAINERID` 该目录下的容器日志文件。	如果 Containerd 作为 K8S 容器运行时， 容器日志的落盘由 Kubelet 来完成，保存至 `/var/log/pods/$CONTAINER_NAME` 目录下，同时在 `/var/log/containers` 目录下创建软链接，指向日志文件。
存储大小	Pod 中的每个容器，docker 默认会保留 100MB*10 = 1G 日志	Pod 中的每个容器，containerd 默认会保留 10MB*5 = 50MB 日志

配置参数	在 docker 配置文件中指定：
	`"log-driver": "json-file",`
	`"log-opts": {"max-size": "100m","max-file": "5"}`
		方法一：在 kubelet 参数中指定：
		`--container-log-max-files=5`
		`--container-log-max-size="100Mi"`
		方法二：在 KubeletConfiguration 中指定：
		`"containerLogMaxSize": "100Mi",`
		`"containerLogMaxFiles": 5,`

把容器日志保存到数据盘	把数据盘挂载到 “data-root”（缺省是 `/var/lib/docker` ）即可。	创建一个软链接 `/var/log/pods` 指向数据盘挂载点下的某个目录。在 TKE 中选择“将容器和镜像存储在数据盘”，会自动创建软链接 `/var/log/pods` 。

自建 DNS 导致节点初始化异常

最近更新时间：2024-11-13 11:32:22

背景信息

在使用容器服务 TKE 的自定义镜像时，为了能够解析到业务内部的相关服务，用户在自定义镜像中修改了 DNS 的解析顺序或将腾讯官方的 DNS 解析地址完全替换为自建 DNS。

操作影响

上述情况可能会导致节点在注册进集群的过程中，无法解析到腾讯云的官方资源库，进而大概率出现节点初始化失败、网络、存储等相关组件功能异常等情况。

- 节点初始化：在节点初始化流程中可能会报错误信息 “Failed to resolve address, dns may be changed”。
- 网络组件：网络组件如 IPAMD 功能依赖腾讯云内网 DNS 解析，解析不到腾讯云的官方资源库可能会导致网络组件功能不可用。
- 存储组件：存储组件如 CBS-CSI 的 mount 或 unmount 失败。

> **说明：**
> 集群内的相关组件安装需要依赖腾讯云官方资源库，解析地址为：
>
> ```
> nameserver 183.60.83.19
> nameserver 183.60.82.98
> ```

解决措施

将腾讯云 DNS Nameserver 配置为自建 DNS 的上游

建议将 `/etc/resolv.conf` 配置中的 nameserver 添加到自建 DNS 服务器的上游，因为部分服务依赖腾讯云内部 DNS 解析，如果未将其设为自建 DNS 的上游，可能导致部分服务无法正常工作。本文以 BIND 9 为例修改配置文件，将上游 DNS 地址写入 forwarders 中，示例如下：

```
options {
        forwarders {
                183.60.83.19;
                183.60.82.98;
        };
        ...
```

完成上述操作后的 24 小时内，节点初始化流程的自动重试策略会不断尝试执行节点初始化工作直到解析到腾讯云官方资源库；超过 24 小时后，需要用户删除节点重新创建。若上述解决措施不生效，请 提交工单 来寻求帮助。

> **注意：**
> 自建 DNS Server 和请求源不在同个 Region，可能会导致部分不支持跨域访问的腾讯域名失效。

原生节点专用调度器相关

最近更新时间：2024-04-08 17:30:51

原生节点专用调度器功能原理详情请参见 产品介绍 。

通用问题

调度器只作用于原生节点是什么意思?

原生节点是 TKE 基于云原生场景发布的增强型节点，其中包含了大量容器场景的特殊优化。调度器的功能依赖于原生节点独有的特性，因此只在原生节点上生效。即使选择了普通节点，调度器的参数配置 CRD 也无法生效。

调度器的参数配置 CRD 是什么?

除了可以通过控制台操作外，原生节点专用调度器的参数配置也支持使用 YAML 配置。每个配置文件都保存在名为 clusternoderesourcepolicies 的资源对象中，您可以直接创建或修改该类型的资源对象。

使用 CRD 有什么注意事项?

自定义设置节点放大系数和水位线非常灵活，可以选择一个节点或节点池、一批节点或节点池，甚至整个集群配置相同的节点规格放大系数和水位线。您可以创建多个配置文件并应用到不同的节点上。如果一个节点被多个配置文件声明，并且这些文件声明的参数不同，会导致异常。

集群中既有普通节点又有原生节点会有什么样的效果?

不会影响正常的调度行为。

集群中只有部分原生节点配置了调度参数会有什么样的效果?

如果某些原生节点配置了调度器参数，则这些原生节点将按照配置的调度参数进行调度。其他没有配置调度参数的原生节点和普通节点将按照正常的调度规则进行调度。

调度器里的 CPU 利用率是如何计算的?

1. 获取当前的 CPU 时间（user，nice，system，idle，iowait，irq，softirq，steal，guest，guest_nice）和上一次获取的 CPU 时间。
2. 计算每个 CPU 时间的总和（total = user + nice + system + idle + iowait + irq + softirq + steal + guest + guest_nice）。
3. 计算自上次检查以来 CPU 时间的总增量（delta_total = total − last_total）。
4. 计算自上次检查以来 CPU 空闲时间的增量（delta_idle = idle − last_idle）。
5. 计算 CPU 利用率（cpu_percent = 100.0 * (1.0 − delta_idle / delta_total)）。

其中，last_total 和 last_idle 是上一次检查时的 CPU 时间和 CPU 空闲时间。

如何批量配置调度器参数?

每个配置文件保存在名为 clusternoderesourcepolicies 的资源对象里，您可以直接创建或修改该类型的资源对象。

如果两个调度器配置 CRD 作用于同一个原生节点会有什么后果?

CRD 和节点的关系类似于其他资源对象，例如 Deployment 和 Pod 之间、HPA 和 Pod 之间，都是通过 labelselector 来控制的。如果一个原生节点被两个 CRD 同时作用，那么这两个 CRD 会不断对同一个原生节点进行覆盖修改。

如果调度器 OOM 会有什么样的风险?

调度器 OOM 会触发 kubelet 的重启逻辑，导致本节点上的重启。在重启过程中，如果发生了调度或需要驱逐的行为，可能会短暂受到影响。重启完成后，调度器功能将正常工作。

驱逐线、调度线、运行时水位、低负载节点这些都是什么意思?

- **调度线**：即调度时的水位线，对应 CRD 中的 targetLoadThreshold，Pod 在调度时参考的水位线。
- **驱逐线**：即运行时的水位线，对应 CRD 中的 evictLoadThreshold，当节点的利用率达到该水位时，会触发驱逐动作。
- **低负载节点**：在发生驱逐时，为了保证成功率，调度器要求默认至少有三个节点满足低负载利用率。判断低负载节点的标准是节点的利用率低于 targetLoadThreshold。
  - 如果使用的是 Crane Scheduler 1.1.10版本，并且配置了驱逐停止水位，则至少需要三个原生节点的利用率低于驱逐停止水位才行。

- **驱逐停止水位线**：在发生驱逐动作时，当节点的利用率达到一定水平后，停止驱逐。默认值是 targetLoadThreshold。
  - 如果使用的是 Crane Scheduler 1.1.10版本，则为 evictTargetLoadThreshold。如果您没有设置该参数，则默认为 targetLoadThreshold。

节点放大系数

如何配置节点放大参数?

详情请参见 原生节点规格放大 。

为什么节点放大参数有限制?

节点的 CPU 放大系数在控制台上的最大限制为3，内存放大系数的最大限制为2。这是为了避免设置过大的放大系数导致节点不稳定。较大的放大系数会使节点调度更多的 Pod，增加 Pod 之间的资源竞争，从而增加节点的不稳定性风险。

如何设置更大的放大系数?

您可以通过 CRD 进行配置。每个配置文件都保存在名为 clusternoderesourcepolicies 的资源对象中，您可以直接创建或修改该类型的资源对象。

调度时水位

配置了调度时水位线会有什么样的现象?

例如：某个原生节点的调度时水位配置成了CPU：30%，内存80%。
效果：当一个 Pod 需要调度时，如果这个原生节点的 CPU 利用率大于30%，或者内存利用率大于80%，则该原生节点不是调度的目标节点。

利用率的计算是什么样的?

- 5分钟平均 CPU 利用率：节点过去5分钟平均 CPU 利用率超过设定阈值，不会调度 Pod 到该节点上。
- 1小时最大 CPU 利用率：节点过去1小时最大 CPU 利用率超过设定阈值，不会调度 Pod 到该节点上。
- 5分钟平均内存利用率：节点过去5分钟平均内存利用率超过设定阈值，不会调度 Pod 到该节点上。
- 1小时最大内存利用率：节点过去1小时最大内存利用率超过设定阈值，不会调度 Pod 到该节点上。

以上一个问题为例，此时 CPU 两个利用率的阈值都是30%；内存的两个利用率的阈值都是80%。

调度器是仅作用在预选阶段吗?

不是，调度器既作用在预选阶段，又作用于优选阶段。预选阶段对原生节点进行过滤，优选阶段对过滤后的节点进行打分排序。

没有配置调度时水位，那预选节点的打分就是0了吗?

不是，Kubernetes 默认调度器支持丰富的 调度器插件 ，原生节点专用调度器只是扩展了调度器的能力，不会覆盖默认调度器的预选优选策略。

Kubernetes 默认调度器和原生节点专用调度器的关系?

它们之间是交集关系，原生节点专用调度器扩展了部分调度能力，但不会覆盖默认调度器。两者会一起进行预选和优选。

如何查看调度器调度的日志?

您可以查看 Deployment:crane-scheduler(kube-system) 的日志。

如何持久化日志信息?

详情请参见 采集容器日志到 CLS 。

为什么使用了原生节点调度器，还是存在集群节点负载不均衡的问题?

原生节点调度器通过调度时的水位和运行时的水位来帮助集群实现负载均衡。

- 调度时的水位，可以让 Pod 在调度时考虑节点的真实负载，高负载的不调度，只选择低负载节点，可以让节点的负载相较更均衡。
- 运行时的水位，可以让节点在高负载时，发生驱逐动作，使集群每个节点更均衡。

从逻辑上，这两个水位都是帮助您的集群变得相对更均衡，但是集群整体负载均衡是一个系统性功能，例如：

- 业务波动：调度时可能均衡，但运行一段时间后，业务本身流量会波动，有波峰波谷，部分节点利用率低；利用率高的节点也没有触发重调度的动作。

- 节点问题：节点发生了下线、驱逐、重启等状态，导致 Pod 被驱逐调度，节点的利用率也发生了变化。

运行时水位

配置了运行时水位线会有什么样的现象？

例如：某个原生节点的运行时水位配置成了CPU：50%，内存90%。
效果：当一个 Pod 需要调度时，如果这个原生节点的 CPU 利用率大于50%，或者内存利用率大于90%，则该原生节点会开始发生驱逐，会驱逐节点上的特定 Pod。

发生驱逐的要求有哪些？

- 节点利用率触达了运行时水位。
- 有 Pod 可以被驱逐：需要添加 annotation：`descheduler.alpha.kubernetes.io/evictable: 'true'`。
- 有空间可以容纳 Pod：沿用社区要求，至少有三个原生节点的利用率低于调度时水位。
- 如果是 Crane Scheudler 1.1.10版本，且配置了驱逐停止水位，则是至少有三个原生节点的利用率低于驱逐停止时水位才行。

为什么触达了运行时水位，结果并没有 Pod 被驱逐？

可能原因有多种，除了上述问题外，如果您不是通过腾讯云可观测平台查看的监控，可能与监控系统的差异有关。

为什么驱逐出现延迟？

驱逐的延迟可能存在于多个步骤中：
- 目前驱逐沿用社区的逻辑，最小监控取值点为最近5分钟的利用率均值，每5分钟更新一次。
- 调度器本身会有检查间隔时间和执行动作时间。

驱逐的 Annotation 打在 Workload 还是 Pod 上面？

都可以。建议打在 Workload 的 metadata 上面，这样的动作不会触发 Pod 的重建，该 Workload 下的所有 Pod 都可以被驱逐。如果打在 Pod 上面或者 Workload 的 Pod 的模板上面，会触发 Pod 的重建。建议在测试环境中进行测试。

如何查看调度器驱逐的日志？

您可以查看 Deployment:crane-descheduler(kube-system) 的日志，例如一个驱逐的日志信息。如下图所示：

```
00]  [10.0.0.38 load:31.36 low:50.00 high:80.00] [10.0.0.218 load:3.14 low:100.00 high:100.00]    [10.0.0.163
load:3.47 low:100.00 high:100.00] [10.0.0.165 load:100.00 low:30.00 high:70.00] [10.0.0.168 load:4.85 low:100.
00 high:100.00]]
I0816 16:07:45.781434       1 reduceHighloadNode.go:246] Select cpu overload nodes: [10.0.0.165:100.00]
I0816 16:07:45.799139       1 reduceHighloadNode.go:302] node 10.0.0.165 need to evict... (cpu5minLoad 100.
00>evictThres 70.00, evictTargetThres:30.00, cpuOrigin:4.00)
I0816 16:07:45.810574       1 accessory.go:102] nonRemovablePods [default/ds-test-hotvalue-h4btz default/
guaranteed-58855b8dcf-k5sg8 default/kubernetes-proxy-7f65fbf97c-ppwrp kube-system/coredns-5dcdbb78f4-tn4xn
kube-system/csi-cbs-node-xf2h7 kube-system/ip-masq-agent-m7q6q kube-system/kube-proxy-k96zb kube-system/
l7-lb-controller-6bb5998bd8-xtkms kube-system/qos-agent-hkwqr kube-system/tke-bridge-agent-sw68x kube-system/
tke-cni-agent-zxwlv kube-system/tke-monitor-agent-pvxdm], bePods [], bPods [], gPods []
I0816 16:07:45.810593       1 reduceHighloadNode.go:310] no evictable pods on node 10.0.0.165, ignore...
I0816 16:07:45.810602       1 reduceHighloadNode.go:246] After Evict cpu overload nodes: [10.0.0.165:100.00]
I0816 16:08:45.811507       1 reduceHighloadNode.go:167] initNodeLoadStategy default {100 100 100 100 0 1 0.01
```

PDB 和驱逐之间的关系是什么？

PDB 的优先级高于驱逐，PDB 可以确保最小 Pod 可用性，Pod 可以被调度到不同的节点上。每个节点进行驱逐时，都会考虑到工作负载的最小可用数量，可以中断驱逐操作，以防止对业务造成影响。

为什么驱逐后，监控指标会大幅下降？

这里有两个原因：
1. 运行中的 Pod 由于资源共享，导致驱逐后的监控数值比实际情况要大。
2. 驱逐过程中的 Pod 利用率计算标准是基于驱逐时刻的，因此当节点的实际负载已经降低时，监控中会显示出多次驱逐的情况。

Pod 驱逐的优先级顺序是怎样的？

- 优先级：Priority > podQOS > podload 负载
- podload 负载是从 metricClient 中读取 Pod 的 CPU/Memory 资源使用量，使用量越高，优先级越高。

Pod 的驱逐动作是怎样的？

- 当满足驱逐条件时，如果 podLoad/node.Status.Capacity[res]*100 > 驱逐线，则不会驱逐该 Pod，否则开始驱逐操作。

> **说明：**
> Pod 占节点的资源用量，如果这个 Pod 的占用量直接超过了驱逐线，这个 Pod 就不会被驱逐。

- 如果驱逐成功，会从 nodeload 中减去该 Pod 的所有资源占用率，并在最近5分钟的资源负载中减去该 Pod 的资源占用率。
- 判断最近5分钟的资源负载是否已经小于等于目标值，如果是，则跳过该节点上其他 Pod 的驱逐操作。
- 如果该节点上的 Pod 已经驱逐完成，在节点上更新驱逐时间：`node.Annot[node.kubernetes.io/last-evicted-time] = time.Now()`

TKE Serverless 集群
TKE Serverless 集群相关

最近更新时间：2025-07-04 15:21:32

本文汇总了 TKE Serverless 的集群常见问题，介绍集群相关常见问题的出现原因及解决办法。

为什么 Pod 规格与填写的 Request/Limit 不一致?

在分配 Pod 资源量时，TKE Serverless 需要对工作负载设置的 Request 及 Limit 进行计算，自动判断 Pod 运行所需的资源量，并非按照设置的 Request 及 Limit 值进行资源分配。详情请参见 CPU Pod 规格计算方法 及 GPU Pod 规格计算方法，进一步了解如何通过 Request、Limit 自动计算指定资源规格。

如何新增或修改 TKE Serverless 集群的容器网络?

在创建集群时，需要选择一个 VPC 网络作为集群网络，同时指定一个子网作为容器网络，详情请参见 容器网络说明 。Serverless 集群的 Pod 会直接占用容器网络子网 IP。在使用集群过程中，如需新增或修改容器网络，需要通过新增/移出超级节点的操作来实现，具体操作请参考以下步骤。

步骤1：新建超级节点以新增容器网络

1. 登录容器服务控制台，选择左侧导航栏中的 集群 。
2. 单击需要修改容器网络的集群 ID，进入集群基本信息页。
3. 选择左侧的**超级节点**，在**超级节点**页面，单击**新建**。
4. 在**新建超级节点**页面，选择 IP 充足的容器网络，单击**确定**即可完成新建。

步骤2：移出超级节点以删除容器网络

> **注意：**
> 移出超级节点后必须保证 Serverless 集群至少存在一个超级节点，即若此时只存在一个超级节点，则无法执行移出操作。

移出超级节点之前，需要将此超级节点上的 Pod 全部驱逐到其他超级节点上（不包含 DaemonSet 管理的 Pod），完成驱逐后才可执行移出操作，否则会导致移出节点失败。具体操作请参考以下步骤。

1. 登录容器服务控制台 ，选择左侧导航栏中的 集群 。
2. 单击需要修改容器网络的集群 ID，进入集群基本信息页。

3. 选择左侧的**超级节点**，在**超级节点**页面选择节点名称右侧的**更多** > **驱逐**。如下图所示：

4. 在**驱逐节点**页面，确定节点信息，单击**确定**。驱逐后，该超级节点的状态会变更为“已封锁”，将不再向该节点上调度 Pod。

> **注意：**
> 驱逐会导致 Pod 重建，请谨慎操作。

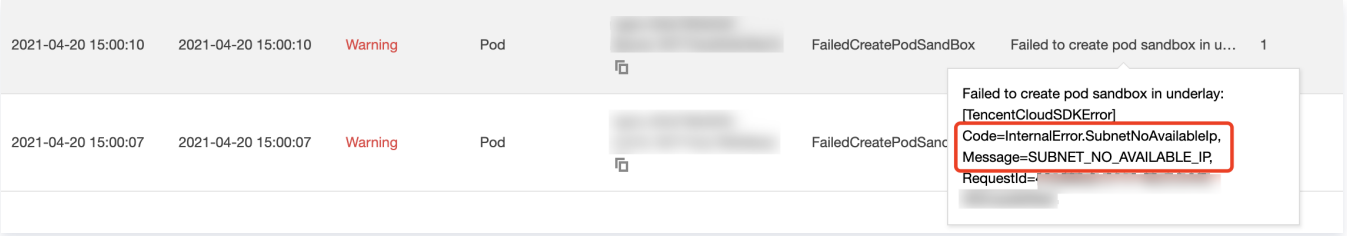
5. 在**超级节点**页面中，选择节点名称右侧的**移出**。
6. 在**删除节点**页面中，单击**确定**即可完成删除节点操作。

Pod 因子网 IP 耗尽调度失败时，该如何处理？

当 Pod 因容器网络的子网 IP 耗尽而发生调度失败时，会在节点日志中观察到如下图所示两种事件：

- 事件一：

- 事件二：



您可通过 容器服务控制台 或在命令行执行以下命令查询超级节点的 YAML。

```
kubectl get nodes -oyaml
```

返回结果如下所示：

```
spec:
  taints:
  - effect: NoSchedule
    key: node.kubernetes.io/network-unavailable
    timeAdded: "2021-04-20T07:00:16Z"
```

```
- lastHeartbeatTime: "2021-04-20T07:55:28Z"
    lastTransitionTime: "2021-04-20T07:00:16Z"
    message: eklet node has insufficient IP available of subnet subnet-bok73g4c
    reason: EKLetHasInsufficientSubnetIP
```

```
      status: "True"
      type: NetworkUnavailable
```

说明此时 Pod 是因容器网络的子网 IP 耗尽而发生调度失败。发生以上情况时，需要新建超级节点去新增子网，以此扩展集群 Pod 可用网段，新建超级节点的操作请参见 新建超级节点 。

TKE Serverless 安全组使用指引和说明有哪些?

在创建 Serverless 集群 Pod 时，若未指定安全组，则默认使用 default 安全组，您也可以通过 `Annotation eks.tke.cloud.tencent.com/security-group-id: 安全组 ID` 为 Pod 指定安全组，请确保同地域已存在该安全组 ID。关于此 Annotation 的详细说明请参加 Annotation 说明 。

如何设置容器终止消息?

Kubernetes 可以通过 terminationMessagePath 设置容器退出的消息来源，即当容器退出时，Kubernetes 将从容器的 terminationMessagePath 字段中指定的终止消息文件中检索终止消息，并使用此内容来填充容器的终止消息，消息默认值为：`/dev/termination-log` 。
此外，您还可以设置容器的 terminationMessagePolicy 字段，进一步自定义容器终止消息。该字段默认值为 `File` ，即仅从终止消息文件中检索终止消息。您可以根据需求设置为 FallbackToLogsOnError，即在容器因错误退出时，如果终止消息文件为空，则使用容器日志输出的最后一部分内容来作为终止消息。
代码示例如下：

```
apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: nginx
spec:
  containers:
  - image: nginx
    imagePullPolicy: Always
    name: nginx
    resources:
      limits:
        cpu: 500m
        memory: 1Gi
      requests:
        cpu: 250m
        memory: 256Mi
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: FallbackToLogsOnError
```

通过以上配置，当容器错误退出且消息文件为空时，Get Pod 会发现 stderr 的输出显示在 containerStatuses 中。

如何使用 Host 参数?

在使用 Serverless 集群时需要注意以下事项：
Serverless 集群虽然没有节点，但兼容 Hostpath、Hostnetwork: true、DnsPolicy: ClusterFirstWithHostNet 等与 Host 相关的参数。您在使用时请注意以下内容：

1. 因为没有节点，这些参数提供能力并不能完全与标准 k8s 对齐。
   例如，期望使用 Hostpath 共享数据，但调度到同一个超级节点上的两个 Pod 查看到的是不同子机的 Hostpath，而且 Pod 重建后，Hostpath 的文件也将同时删除。
2. 系统不会自动清理 Hostpath 目录下的文件，需要您手动清理 ，在 镜像复用的场景 下，Hostpath 的文体也会保留在系统盘上，占用系统盘空间。

如何挂载 CFS/NFS?

在 Serverless 集群中，支持使用腾讯云 文件存储 CFS ，也支持使用自建的文件存储 NFS 以 Volume 的形式挂载到 Pod 上，以实现持久化的数据存储。Pod 挂载 CFS/NFS 的 YAML 示例如下：

```
apiVersion: v1
```

```
kind: Pod
metadata:
  name: test-pd
spec:
  containers:
  - image: k8s.gcr.io/test-webserver
    name: test-container
    volumeMounts:
    - mountPath: /cache
      name: cache-volume
    volumes:
    - name: nfs
      nfs:
        path: /dir
        server: 127.0.0.1
```

- spec.volumes：设置数据卷名称、类型、数据卷的参数。
- spec.volumes.nfs：设置 NFS/CFS 盘。
- spec.containers.volumeMounts：设置数据卷在 Pod 的挂载点。

关于 Pod 挂载 Volume 的详细操作，请参见 存储卷的使用说明 。

如何通过镜像复用加快容器启动速度?

TKE Serverless 支持缓存容器镜像以便在下次用同样镜像启动容器时加快启动速度。

复用条件：

1. 对于同一工作负载的 Pod，如果缓存时间内在同一个可用区（Zone）有 Pod 创建且销毁过，新建的 Pod 默认不重复拉取相同的镜像。
2. 如果不同的工作负载（包括 Deployment、Statefulset、Job）的 Pod 想复用镜像，可以使用如下 annotation：

   ```
   eks.tke.cloud.tencent.com/cbs-reuse-key
   ```

   同一个用户账号下，有同一个 annotation value 的 Pod，缓存时间内会尽量复用启动镜像，建议 annotation value 填写镜像名：`eks.tke.cloud.tencent.com/cbs-reuse-key: "image-name"` 。

缓存时间：2小时。

如何解决复用镜像异常问题?

当启用复用镜像功能时，当创建 Pod 时，`$kubectl describe pod` 可能见到如下错误：

- `no space left on device: unknown`

  ```
  Warning FreeDiskSpaceFailed 26m eklet, eklet-subnet-xxx failed to garbage collect required amount of images.
  Wanted to free 4220828057 bytes, but freed 3889267064 bytes
  ```

恢复方法：

无需任何操作，等待若干分钟后 Pod 会自动 running。

原因：

- `no space left on device: unknown`

  Pod 默认复用系统盘时，系统盘内的原有镜像占满磁盘空间，导致磁盘当前没有足够的空间下载新镜像，所以报错 “no space left on device: unknown”。TKE Serverless 支持定时镜像回收机制，遇到磁盘空间占满时，会自动将系统盘中现有的多余镜像删除，给当前磁盘腾出可用空间。（耗时若干分钟）

  ```
  Warning FreeDiskSpaceFailed 26m eklet, eklet-subnet-xxx failed to garbage collect required amount of images.
  Wanted to free 4220828057 bytes, but freed 3889267064 bytes
  ```

- 该条日志说明当前 Pod 下载镜像需要4220828057空间，但目前只清空了3889267064空间的数据。产生该条 event 的原因为磁盘上有多个镜像，目前只清理了部分镜像，TKE Serverless 的定时镜像回收机制会继续清理镜像，直到能成功拉取到新镜像为止。

挂载自建的 nfs 时，事件报 Operation not permitted 如何处理?

如果您使用自建的 nfs 实现持久化存储时，连接时事件报 Operation not permitted。您需要修改自建 nfs 的 /etc/exports 文件，添加 /<path><ip-range>(rw,insecure) 参数，示例如下：

```
/data/  10.0.0.0/16(rw,insecure)
```

Pod 磁盘满了（ImageGCFailed）如何处理?

TKE Serverless 的 Pod 默认免费提供 20G 可用的系统盘空间，若系统盘空间满了，可以通过如下方式进行处理。

1. 清理未使用的容器镜像

如果使用空间达到 80%，TKE Serverless 后台会触发容器镜像的回收流程，尝试回收未使用的容器镜像来释放磁盘空间。如果未能释放任何空间，则会有一条事件提醒：ImageGCFailed: failed to garbage collect required amount of images， 提醒用户磁盘空间不足。
常见磁盘空间不足的原因有：

- 业务有大量临时输出。您可以通过 du 命令确认。
- 业务持有已删除的文件描述符，导致磁盘空间未释放。您可以通过 lsof 命令确认。

如果业务希望调整容器镜像回收的阈值，可以设置如下 annotation：

```
eks.tke.cloud.tencent.com/image-gc-high-threshold: "80"
```

2. 清理已退出的容器

如果业务原地升级过，或者容器异常退出过，已退出的容器仍会保留，直到磁盘空间达到 85% 时才会清理已退出的容器。清理阈值可以使用如下 Annotation 调整：

```
eks.tke.cloud.tencent.com/container-gc-threshold: "85"
```

如果已退出的容器不想被自动清理（例如需要退出的信息进一步排障的），可以通过如下 Annotation 关闭容器的自动清理，但副作用是磁盘空间无法自动释放：

```
eks.tke.cloud.tencent.com/must-keep-last-container: "true"
```

> **说明：**
> 此特性上线时间为 2021-09-15，故在此时间前创建的 Pod，并未带有此特性。

3. 重启磁盘用量高的 Pod

业务需要在容器的系统盘用量超过某个百分比后直接重启 Pod，可以通过 Annotation 配置：

```
eks.tke.cloud.tencent.com/pod-eviction-threshold: "85"
```

只重启 Pod，不会重建子机，退出和启动都会进行正常的 gracestop、prestop、健康检查。

> **说明：**
> 此特性上线时间在 2022-04-27，故在此时间前创建的 Pod，需要重建 Pod 来开启特性。

9100 端口问题

TKE Serverless 的 Pod 默认会通过 9100 端口对外暴露监控数据，用户可以执行以下命令访问 9100/metrics 获取数据：

- 获取全部指标：

```
curl -g "http://<pod-ip>:9100/metrics"
```

- 大集群建议去掉 ipvs 指标：

```
curl -g "http://<pod-ip>:9100/metrics?collect[]=ipvs"
```

如果业务本身需要监听 9100 端口，则可以在创建 Pod 时，通过使用 9100 之外的端口来收集监控数据，避免跟业务的 9100 端口冲突。配置方式如下：

```
eks.tke.cloud.tencent.com/metrics-port: "9110"
```

如果没有变更监控暴露的端口，业务直接监听 9100 端口，则在 TKE Serverless 新的网络方案里，将报错提醒用户 9100 端口已经被使用：

```
listen() to 0.0.0.0:9100, backlog 511 failed (1: Operation not permitted)
```

出现报错提醒时，需要为 Pod 添加一个 Annotation：`metrics-port`，以变更监控端口，再重建 Pod。

> **注意：**
> 如果 Pod 带有公网 eip，则需要设置安全组，注意 9100 端口问题，并放通需要的端口。

负载均衡相关

最近更新时间：2024-11-27 17:50:32

本文汇总了负载均衡相关常见问题，介绍与 Service/Ingress CLB 相关的各种常见问题的出现原因及解决办法。

本文档需要您：

- 熟悉 K8S 的 基本概念 。例如 Pod、工作负载/Workload、Service、Ingress 等。
- 熟悉 腾讯云控制台 容器服务之 TKE Serverless 集群的常规操作。
- 熟悉通过 kubectl 命令行工具操作 K8S 集群中的资源。

> **注意：**
> 您可通过多种方式操作 K8S 集群中的资源，本文档向您介绍如何通过腾讯云控制台进行操作，及如何通过 kubectl 命令行工具进行操作。

TKE Serverless 会为哪些 Ingress 创建 CLB 实例?

TKE Serverless 会为满足如下条件的 Ingress 创建 CLB 实例：

对 Ingress 资源的要求	其他说明
annotations 中含有如下键值对： kubernetes.io/ingress.class: qcloud	如果不希望 TKE Serverless 为 Ingress 创建 CLB 实例，例如希望使用 Nginx-ingress，则只需保证 annotations 中不包含前述键值对。

如何查看 TKE Serverless 为 Ingress 创建的 CLB 实例?

如果成功为 Ingress 创建 CLB 实例，TKE Serverless 会将 CLB 实例的 VIP 写入 Ingress 资源的 `status.loadBalancer.ingress` 中，并且将如下键值对写入 annotations 中：

```
kubernetes.io/ingress.qcloud-loadbalance-id: CLB 实例 ID
```

如需查看 TKE Serverless 为 Ingress 创建的 CLB 实例，具体步骤如下：

1. 登录容器服务控制台，选择左侧导航栏中的 集群 。
2. 在集群列表页面，选择集群 ID 进入集群管理页面。
3. 在集群管理页面，选择左侧**服务与路由** > **Ingress**。
4. 在 “Ingress” 页面，查看 CLB 实例 ID 及其 VIP。如下图所示：

TKE Serverless 会为哪些 Service 创建 CLB 实例?

TKE Serverless 会为满足如下条件的 Service 创建 CLB 实例：

K8S 版本	对 Service 资源的要求
所有 TKE Serverless 支持的 K8S 版本	spec.type 为 LoadBalancer
魔改版 K8S（kubectl version 返回的 Server GitVersion 带有 "eks.*" 或 "tke.*" 后缀）	spec.type 为 ClusterIP，并且 spec.clusterIP 的值不是None（即非 Headless 的 ClusterIP 类型的 Service）

非魔改版 K8S（kubectl version 返回的 Server GitVersion 不带 "eks.*" 或 "tke.*" 后缀）	spec.type 为 ClusterIP，并且明确指定 spec.clusterIP 为空字符串（""）

注意：
如果成功创建 CLB 实例，TKE Serverless 会将如下键值对写入 Service annotations 中：

```
service.kubernetes.io/loadbalance-id: CLB 实例 ID
```

如何查看 TKE Serverless 为 Service 创建的 CLB 实例?

如果成功为 Service 创建 CLB 实例，TKE Serverless 会将 CLB 实例的 VIP 写入 Service 资源的 `status.loadBalancer.ingress` 中，并且将如下键值对写入 annotations 中：

```
kubernetes.io/ingress.qcloud-loadbalance-id: CLB 实例 ID
```

如需查看 TKE Serverless 为 Service 创建的 CLB 实例，具体步骤如下：

1. 登录容器服务控制台，选择左侧导航栏中的 集群 。
2. 在集群列表页面，选择集群 ID 进入集群管理页面。
3. 在集群管理页面，选择左侧**服务与路由** > **Service**。
4. 在 **Service** 页面，查看 CLB 实例 ID 及其 VIP。如下图所示：

为什么 Service 的 ClusterIP 无效（无法正常访问）或没有 ClusterIP？

对于 spec.type 为 LoadBalancer 的 Service，目前 TKE Serverless 默认不分配 ClusterIP，或者分配的 ClusterIP 无效（无法正常访问）。如果用户同时需要使用 ClusterIP 访问 Service，可以通过在 annotations 中加入如下键值对指示 TKE Serverless 基于内网 CLB 实现 ClusterIP：

```
service.kubernetes.io/qcloud-clusterip-loadbalancer-subnetid: Service CIDR 子网 ID
```

Service CIDR 子网 ID 在创建集群时指定，为 `subnet-********` 字符串。您可在 CLB 基本信息页面中查看该子网 ID 信息。

注意：
只有使用魔改版 K8S（kubectl version 返回的 Server GitVersion带有 "eks.*" 或 "tke.*" 后缀）的 TKE Serverless 集群才支持该特性。对于早期创建的、使用非魔改版 K8S（kubectl version 返回的 Server GitVersion 不带 "eks.*" 或 "tke.*" 后缀）的 TKE Serverless 集群，您需要升级 K8S 版本后使用该特性。

如何指定 CLB 实例类型（公网或内网）？

您可以通过容器服务控制台 或通过 kubectl 命令行工具指定 CLB 实例类型：

通过容器服务控制台操作

- 对于 Ingress，通过“网络类型”选择“公网”或“内网”：

- 对于 Service，通过“服务访问方式”控制，其中“VPC内网访问”对应内网 CLB 实例：

通过 kubectl 命令行工具操作

- 默认创建的 CLB 实例是“公网”类型。
- 如需创建“内网”类型的 CLB 实例，则需为 Service 或 Ingress 加上相应的 annotation：

资源类型	需要在 annotations 中添加的键值对
Service	service.kubernetes.io/qcloud-loadbalancer-internal-subnetid: 子网 ID
Ingress	kubernetes.io/ingress.subnetId: 子网 ID

注意：

> 子网 ID是形如 subnet-******** 的字符串，并且该子网必须在创建集群时为 “集群网络”指定的 VPC 中，该 VPC 信息可在腾讯云控制台集群 “基本信息” 中查询到。

如何指定使用已有 CLB 实例?

您可以通过容器服务控制台 或通过 kubectl 命令行工具指定使用已有 CLB 实例:

通过容器服务控制台 操作

在创建 Service 或 Ingress 时，可以选择“使用已有” CLB 实例。对于 Service，还可以在 Service 创建之后，通过“更新访问方式”切换成“使用已有” CLB 实例。

通过 kubectl 命令行工具操作

在创建 Service/Ingress 或变更 Service 时，为 Service 或 Ingress 加上相应的 annotation 即可:

资源类型	需要在 annotations 中添加的键值对
Service	service.kubernetes.io/tke-existed-lbid: CLB 实例 ID
Ingress	kubernetes.io/ingress.existLbId: CLB 实例 ID

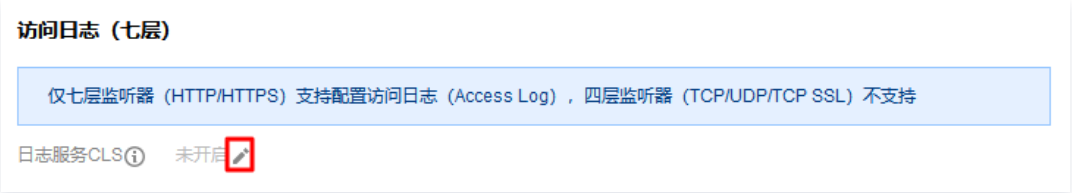
> **注意:**
> “已有 CLB 实例” 不能是 “TKE Serverless 为 Service 或 Ingress 创建的 CLB 实例”，并且 TKE Serverless 不支持多个 Service/Ingress 共用同一个已有 CLB 实例。

如何查看 CLB 实例的访问日志?

仅7层 CLB 实例支持配置访问日志，但 TKE Serverless 为 Ingress 创建的7层 CLB 实例默认不开启访问日志。如需开启 CLB 实例的访问日志，可在 CLB 实例的详情页面进行操作。具体步骤如下:

1. 登录容器服务控制台，选择左侧导航栏中的 集群 。
2. 在集群列表页面，选择集群 ID 进入集群管理页面。
3. 在集群管理页面，选择左侧**服务与路由** > **Ingress**。
4. 在 “Ingress” 页面，选择 CLB 实例 ID 进入 CLB 基本信息页面。如下图所示:

5. 在 CLB 基本信息页面的“访问日志（七层）”中，单击 ，在弹出窗口中进行开启。如下图所示：



为什么 TKE Serverless 没有为 Ingress 或 Service 创建 CLB 实例?

请参考 TKE Serverless 会为哪些 Ingress 创建 CLB 实例 及 TKE Serverless 会为哪些 Service 创建 CLB 实例 问题的回答，确认对应的资源是否具备对应的条件。如具备条件但未成功创建 CLB 实例，可通过 `kubectl describe` 命令查看“资源”相关事件。
通常 TKE Serverless 会输出相关的 Warning 类型事件。示例图如下，输出事件表明子网中已经无可用的 IP 资源，所以无法成功创建 CLB 实例。

```
[flyma@VM_227_154_centos ~]$
[flyma@VM_227_154_centos ~]$ kubectl describe svc -n demo normal-clusterip-nginx
Name:                     normal-clusterip-nginx
Namespace:                demo
Labels:                   <none>
Annotations:              service.kubernetes.io/qcloud-loadbalancer-internal-subnetid: subnet-j55z6ay7
Selector:                 k8s-app=nginx,qcloud-app=nginx
Type:                     ClusterIP
IP:
Port:                     tcp-80-80  80/TCP
TargetPort:               80/TCP
Endpoints:                10.0.24.70:80,10.0.24.81:80,10.1.0.174:80
Session Affinity:         None
Events:
  Type     Reason                    Age                From                    Message
  ----     ------                    ----               ----                    -------
  Warning  CreatingLoadBalancerFailed  46m                clb-service-controller  Failed to syncLbWithService (will retry): failed to ensure internal load balan
cer for 'service:demo/normal-clusterip-nginx', [TencentCloudSDKError] Code=ResourceInsufficient, Message=The number of IP in subnet subnet-j55z6ay7 is not enough.,
 RequestId=3c6f9d95-650d-4e98-8171-006fbb0d0f88
  Warning  CreatingLoadBalancerFailed  29m                clb-service-controller  Failed to syncLbWithService (will retry): failed to ensure internal load balan
cer for 'service:demo/normal-clusterip-nginx', [TencentCloudSDKError] Code=ResourceInsufficient, Message=The number of IP in subnet subnet-j55z6ay7 is not enough.,
 RequestId=8567bab4-3f57-4f3d-9e99-f34d8c28d5e8
```

如何在多个 Service 中使用相同的负载均衡?

TKE Serverless 集群默认多个 Service 不可共用同一个 CLB 实例。如果您希望 Service 复用其他 Service 占用的 CLB，请添加此 annotation 并将 value 填写为 "true"，例如 `service.kubernetes.io/qcloud-share-existed-lb: true` 。

为什么访问 CLB VIP 时失败?

请您按照以下步骤进行分析：

查看 CLB 实例类型

1. 在 “Service” 或 “Ingress” 页面，选择 CLB 实例 ID 进入 CLB 基本信息页面。如下图所示：

2. 您可在 CLB 基本信息页面中查看上述 CLB 实例的“实例类型”。

确认访问 CLB VIP 的环境是否正常

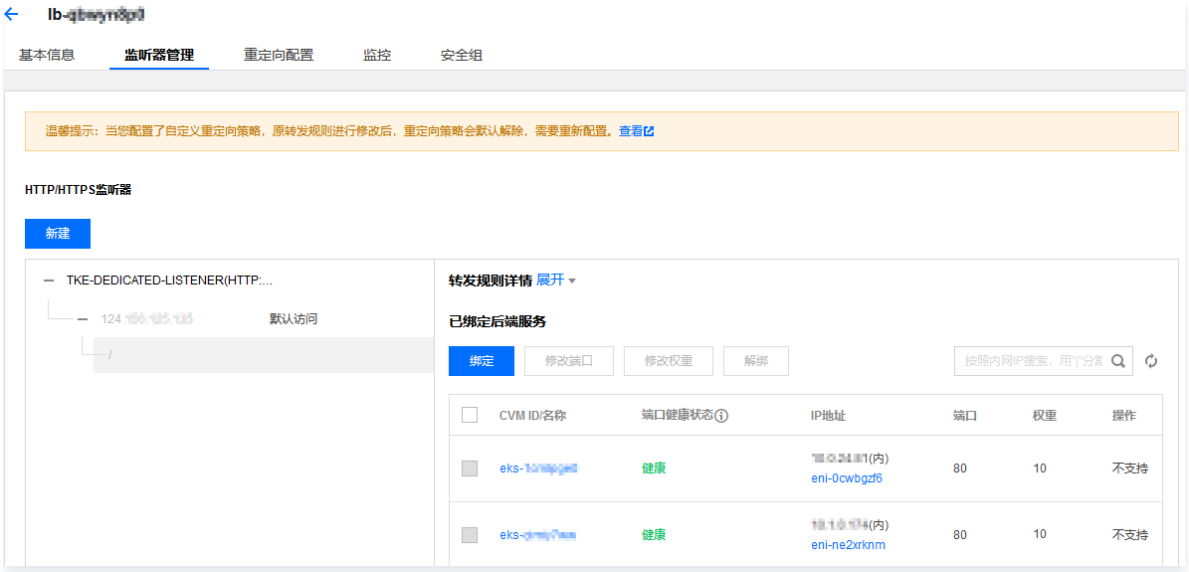
- 若 CLB 实例“实例类型”为内网，则其 VIP 只能在所属的 VPC 内访问。
  由于 TKE Serverless 集群中 Pods 的 IP 是 VPC 内的弹性网卡的 IP，所以可以在 Pods 中访问集群内任何 Service 或 Ingress 的 CLB 实例的 VIP。

> **注意：**
> 通常 LoadBalancer 系统都存在回环问题（例如 AzureLoad Balancer 问题排查指南），请勿在工作负载所属的 Pods 中通过该工作负载（经 Service 或 Ingress）对外暴露的 VIP 访问该工作负载提供的服务。即 Pods 不要通过 VIP（包含“内网类型”及“外网类型”）访问 Pods 自己提供的服务。否则可能导致延迟增加，或者（在 VIP 对应的规则下只有一个 RS/Pod 时）访问不通。

- 若 CLB 实例“实例类型”为公网，则其 VIP 可以在有公网访问能力的环境中访问。
  若要在集群内访问公网 VIP，请确保已经通过配置 NAT 网关或其他方式为集群开启了公网访问能力。

查看 CLB 下的 RS 包括（且仅包括）预期的 Pods 的 IP + 端口

您可在 CLB 管理页面中，选择**监听器管理**页面，查看（7层协议）转发规则、（4层协议）绑定的后端服务。其中的 IP 地址预期即为各个 Pod 的 IP。TKE Serverless 为某个 Ingress 创建的 CLB 示例图如下：



确认对应的 Endpoints 是否正常

如果已正确地为工作负载（Workload）设置了标签（Labels），并且正确地为 Service 资源设置了选择算符（Selectors），则在工作负载的 Pods 成功运行之后，即可通过 `kubectl get endpoints` 命令查看 Pods 被 K8S 列入到 Service 对应的 Endpoints 的就绪地址列表中。示例图如下：

```
[flyma@VM_227_154_centos ~]$
[flyma@VM_227_154_centos ~]$ kubectl get pod -n demo -o wide
NAME                    READY   STATUS    RESTARTS   AGE   IP          NODE                                                NOMINATED NODE   READINESS GATES
nginx-7c7c647ff7-4b8n5  1/1     Running   0          18h   10.0.24.70  cls-4plj5ho0-virtual-kubelet-subnet-e714bobz-0   <none>           <none>
nginx-7c7c647ff7-b95zc  1/1     Running   0          18h   10.0.24.81  cls-4plj5ho0-virtual-kubelet-subnet-e714bobz-0   <none>           <none>
nginx-7c7c647ff7-ckhb9  1/1     Running   0          18h   10.1.0.174  cls-4plj5ho0-virtual-kubelet-subnet-l9dkwcyx-0   <none>           <none>
[flyma@VM_227_154_centos ~]$
[flyma@VM_227_154_centos ~]$ kubectl get deployment -n demo nginx
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
nginx   3/3     3            3           18h
[flyma@VM_227_154_centos ~]$
[flyma@VM_227_154_centos ~]$ kubectl get svc -n demo internal-clb-nginx
NAME                TYPE           CLUSTER-IP   EXTERNAL-IP   PORT(S)        AGE
internal-clb-nginx  LoadBalancer   10.0.15.28   10.0.15.28    80:30082/TCP   18h
[flyma@VM_227_154_centos ~]$
[flyma@VM_227_154_centos ~]$ kubectl get ep -n demo internal-clb-nginx
NAME                ENDPOINTS                                                  AGE
internal-clb-nginx  10.0.24.70:80,10.0.24.81:80,10.1.0.174:80                  18h
[flyma@VM_227_154_centos ~]$
[flyma@VM_227_154_centos ~]$
```

而已创建、但状态异常的 Pods 会被 K8S 列入到 Service 对应的 Endpoints 的未就绪地址列表中。示例图如下：

```
[flyma@VM_227_154_centos ~]$ kubectl get pod -n demo -o wide
NAME                    READY   STATUS             RESTARTS   AGE   IP          NODE                                                NOMINATED NODE   READINESS GATES
nginx-7c7c647ff7-4b8n5  0/1     ImagePullBackOff   0          18h   10.0.24.70  cls-4plj5ho0-virtual-kubelet-subnet-e714bobz-0   <none>           <none>
nginx-7c7c647ff7-b95zc  1/1     Running            0          18h   10.0.24.81  cls-4plj5ho0-virtual-kubelet-subnet-e714bobz-0   <none>           <none>
nginx-7c7c647ff7-ckhb9  1/1     Running            0          18h   10.1.0.174  cls-4plj5ho0-virtual-kubelet-subnet-l9dkwcyx-0   <none>           <none>
[flyma@VM_227_154_centos ~]$
[flyma@VM_227_154_centos ~]$
[flyma@VM_227_154_centos ~]$ kubectl get ep -n demo internal-clb-nginx -o yaml
apiVersion: v1
kind: Endpoints
metadata:
  annotations:
    endpoints.kubernetes.io/last-change-trigger-time: "2020-12-22T11:41:25+08:00"
  creationTimestamp: "2020-12-21T08:58:22Z"
  name: internal-clb-nginx
  namespace: demo
  resourceVersion: "2482053203"
  selfLink: /api/v1/namespaces/demo/endpoints/internal-clb-nginx
  uid: 7893ab4c-f8b4-45d9-a4df-21daabb1c42e
subsets:
- addresses:
  - ip: 10.0.24.81
    nodeName: cls-4plj5ho0-virtual-kubelet-subnet-e714bobz-0
    targetRef:
      kind: Pod
      name: nginx-7c7c647ff7-b95zc
      namespace: demo
      resourceVersion: "2475611832"
      uid: 38c40d5f-9ab6-4647-9e75-642b8203268e
  - ip: 10.1.0.174
    nodeName: cls-4plj5ho0-virtual-kubelet-subnet-l9dkwcyx-0
    targetRef:
      kind: Pod
      name: nginx-7c7c647ff7-ckhb9
      namespace: demo
      resourceVersion: "2475610295"
      uid: c839e449-aa5b-4cf7-a5ff-a35df7262811
  notReadyAddresses:
  - ip: 10.0.24.70
    nodeName: cls-4plj5ho0-virtual-kubelet-subnet-e714bobz-0
    targetRef:
      kind: Pod
      name: nginx-7c7c647ff7-4b8n5
      namespace: demo
      resourceVersion: "2482053202"
      uid: f5d2a2d3-676d-4323-98dd-4aaf168e6e22
  ports:
  - name: tcp-80-80
    port: 80
    protocol: TCP
[flyma@VM_227_154_centos ~]$
```

注意：

对于异常的 Pods，可通过 kubectl describe 命令查看异常的原因。示例命令如下：

```
kubectl describe pod nginx-7c7c647ff7-4b8n5 -n demo
```

确认 Pods 是否能够正常提供服务

即使 Running 状态的 Pods 也可能无法正常对外提供服务。例如，未监听指定的协议+端口、Pods 内部逻辑错误、处理过程阻塞等。您可通过 `kubectl exec` 命令登录至 Pod 内，并使用 `telnet/wget/curl` 命令或自定义的客户端工具直接访问 Pod IP+端口。若 Pod 内直接访问失败，则需要进一步分析导致 Pod 无法正常提供服务的原因。

确认 Pod 绑定的安全组是否放通 Pods 提供服务的协议和端口

安全组控制 Pods 的网络访问策略，如同 Linux 服务器中的 IPTables 规则。请结合实际情况进行查看：

通过容器服务控制台创建工作负载

交互流程会强制要求指定一个安全组，TKE Serverless 将使用该安全组控制 Pods 的网络访问策略。用户选择的安全组会被存储在工作负载的 `spec.template.metadata.annotations` ，最终添加到 Pods 的 annotations 中。示例如下：

```
[flyma@VM_227_154_centos ~]$ kubectl get pod -n demo nginx-7c7c647ff7-ckhb9 -o yaml
apiVersion: v1
kind: Pod
metadata:
  annotations:
    eks.tke.cloud.tencent.com/cpu-type: intel
    eks.tke.cloud.tencent.com/security-group-id: sg-7wsmhh5d
  creationTimestamp: "2020-12-21T08:55:17Z"
  generateName: nginx-7c7c647ff7-
  labels:
    k8s-app: nginx
    pod-template-hash: 7c7c647ff7
    qcloud-app: nginx
  name: nginx-7c7c647ff7-ckhb9
  namespace: demo
  ownerReferences:
  - apiVersion: apps/v1
    blockOwnerDeletion: true
    controller: true
    kind: ReplicaSet
    name: nginx-7c7c647ff7
    uid: dc06824c-425c-4229-929d-46d9afe393b6
  resourceVersion: "2475610295"
  selfLink: /api/v1/namespaces/demo/pods/nginx-7c7c647ff7-ckhb9
  uid: c839e449-aa5b-4cf7-a5ff-a35df7262811
spec:
  containers:
  - image: nginx
    imagePullPolicy: Always
    name: nginx
    resources:
```

使用 kubectl 命令创建工作负载

若您通过 kubectl 命令创建工作负载，且没有（通过 annotations）为 Pods 指定安全组，则 TKE Serverless 会使用账号下的同地域的默认项目的 default 安全组。查看步骤如下：

1. 登录私有网络控制台，选择左侧导航栏中的 安全组 。
2. 在“安全组”页面上方，选择同地域的默认项目。
3. 可在列表中查看 default 安全组，可单击**修改规则**查看详情。如下图所示：

联系我们

若至此仍未找到问题原因，您可通过 在线客服 或 提交工单 联系 TKE 团队解决问题。

超级节点相关

最近更新时间：2025-01-08 17:10:12

如何禁止 Pod 调度到某个按量计费超级节点?

默认情况下，TKE 普通集群添加了按量计费超级节点节点池后，会在 Node 资源不足时，自动向按量计费超级节点调度 Pod。Serverless 集群则会自动在多个按量计费超级节点随机调度 Pod。
此时如果您不希望向某按量计费超级节点（代表某子网/可用区）调度，可通过以下两种方式封锁按量计费超级节点实现禁止调度：

- 通过 容器服务控制台 对节点进行封锁操作，详情可参见 封锁节点 。
- 通过命令行执行如下命令实现禁止调度：

```
$kubectl cordon $按量计费超级节点名称
```

如何禁止 TKE 普通集群在资源不足时自动调度到按量计费超级节点?

需要在 kube-system 命名空间下，新建名为 eks-config 的 configmap。执行命令如下：

```
$kubectl create configmap eks-config --from-literal=AUTO_SCALE_EKS=false -n kube-system
```

将 AUTO_SCALE_EKS 的 value 设置为 false，即可关闭 TKE 普通集群向按量计费超级节点的自动调度机制。

如何手动调度 Pod 到按量计费超级节点?

默认按量计费超级节点会自动添加 Taints 以降低调度优先级，如需手动调度 Pod 到按量计费超级节点或指定按量计费超级节点，通常需要为 Pod 添加对应的 Tolerations。但并非所有的 Pod 均可以调度到按量计费超级节点上，详情请参见 超级节点调度说明 。为方便使用，您可以在 Pod Spec 中指定 nodeselector 。示例如下：

```
spec:
    nodeSelector:
      node.kubernetes.io/instance-type: eklet
```

TKE 的管控组件会判断该 Pod 是否可以调度到按量计费超级节点，若不支持则不会调度到按量计费超级节点。

如何强制调度 Pod 到按量计费超级节点，无论按量计费超级节点是否支持该 Pod?

> **注意：**
> 强制调度到按量计费超级节点，Pod 可能会创建失败，失败原因可参见 如何手动调度 Pod 到按量计费超级节点 。

如需强制调度 Pod 到按量计费超级节点，除了为 Pod 指定 nodeselector 或 nodename，也必须添加对应的 Tolerations。

1. 为 Pod Spec 中指定 nodeselector。示例如下：

```
spec:
  nodeSelector:
    node.kubernetes.io/instance-type: eklet
```

或在 Pod Spec 指定 nodename。示例如下：

```
spec:
   nodeName: $按量计费超级节点名称
```

2. 为 Pod 添加 Tolerations。示例如下：

```
spec:
   tolerations:
```

```
  - effect: NoSchedule
    key: eks.tke.cloud.tencent.com/eklet
    operator: Exists
```

如何自定义按量计费超级节点 DNS?

TKE Serverless 集群支持按量计费超级节点特性，您可通过在 yaml 中定义 annotation 的方式，实现自定义 DNS 等能力。详情可参见 按量计费超级节点 annotation 说明 。

镜像仓库相关

最近更新时间：2025-02-11 15:42:52

Serverless 集群如何使用容器镜像服务 TCR?

Serverless 集群如需使用容器镜像服务 TCR，需要确保 已选择对应的镜像访问凭证 和 Serverless 集群到镜像服务网络打通 。

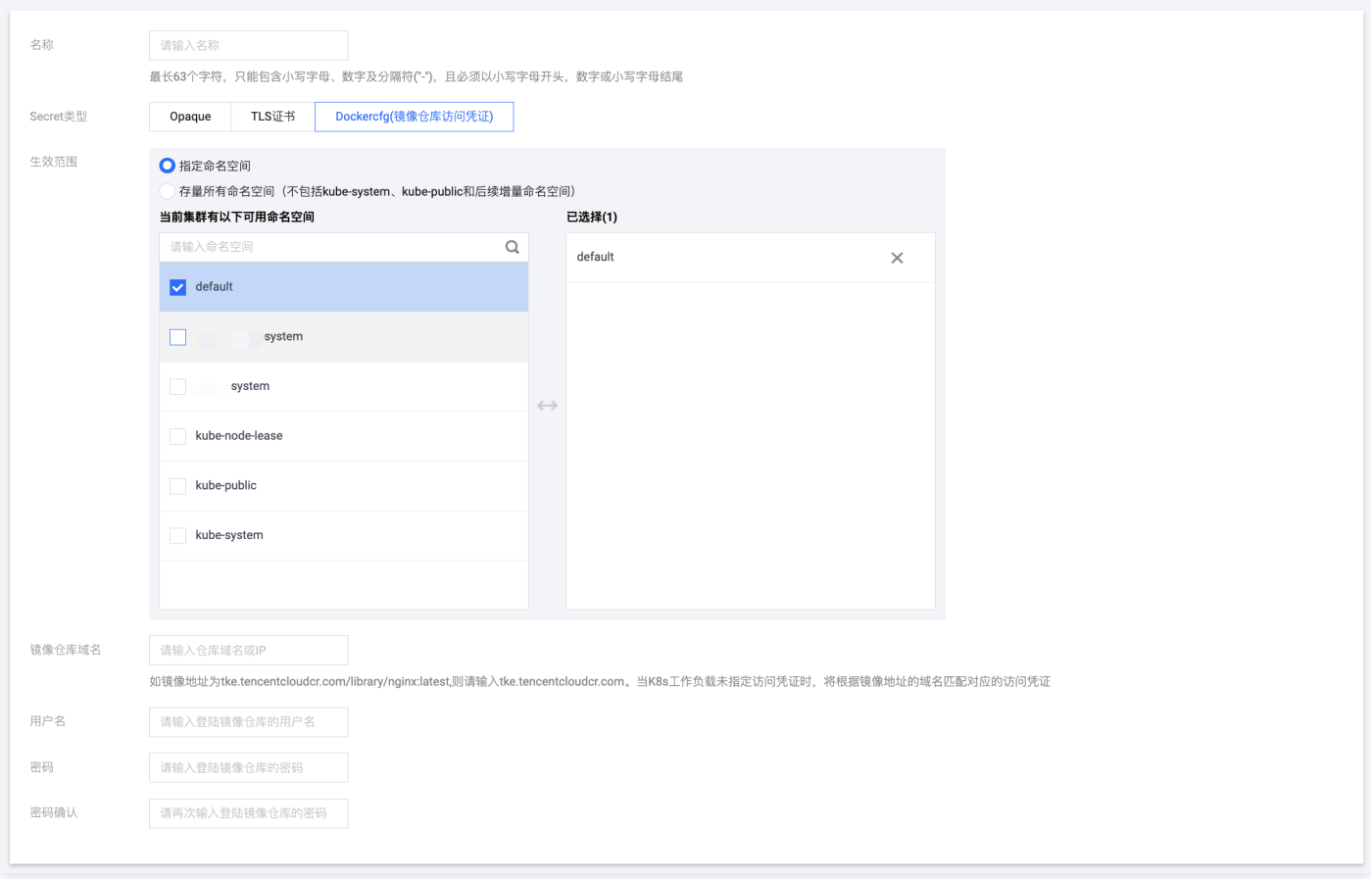
确保已选择对应的镜像访问凭证

容器镜像默认私有，因此在创建工作负载时：

1. 不指定访问凭证，自动匹配命名空间下已有访问凭证，需要确保当前命名空间下，已有镜像仓库所属的 TCR 实例的访问凭证。
2. 手动指定访问凭证，支持同时指定多个访问凭证。

您可参考以下步骤创建镜像访问凭证：

1. 登录 容器服务控制台 。
2. 在集群列表页面，单击您需要创建访问凭证的 Serverless 集群名称，进入 Serverless 集群详情页。
3. 在左侧导航中选择**配置管理 > Secret**，单击**新建**。
4. 选择 **Dockercfg(镜像仓库访问凭证)**，选择需要下发的命名空间，并配置仓库域名、用户名、密码。



确保 Serverless 集群到镜像服务网络打通

Serverless 集群与容器镜像服务之间的网络默认不通，因此在拉取镜像时会报网络不通的错误：

```
dial tcp x.x.x.x:443: i/o timeout
```

解决步骤

处理方式有以下两种：

处理方式	说明
方式1：内网访问方式（推荐）	容器镜像服务新建内网访问链路，并配置内网域名解析，Serverless 集群通过新建的内网链路访问容器镜像服务。
方式2：外网访问方式	Serverless 集群开启外网访问，通过公网访问容器镜像服务，同时容器镜像服务也需允许公网访问。

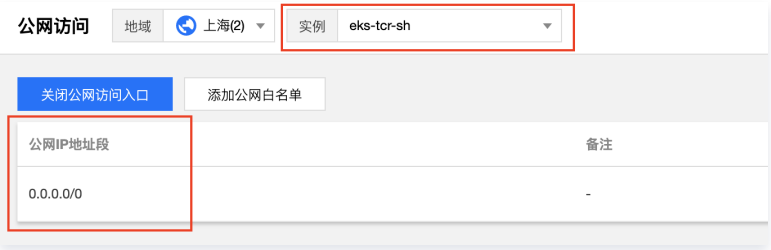
方式1：内网访问方式（推荐）

1. 新建内网访问链路
   1.1 登录 容器镜像服务控制台，选择左侧导航栏中的**访问控制** > **内网访问**。
   1.2 在“内网访问”页面中选择对应的地域和实例，单击**接入私有网络**。
   1.3 在弹出的“新建内网访问链路”窗口中，配置私有网络及子网信息。如下图所示：

2. 配置完成并单击**确定**，该内网访问链路将启动新建，并自动配置内网解析。

方式2：外网访问方式

1. 确保容器镜像服务开启公网访问
   1.1 容器镜像服务控制台，选择左侧导航栏中的**访问控制** > **公网访问**。
   1.2 在**公网访问**页面中选择对应的地域和实例。
   1.3 为对应的容器镜像服务实例开启公网访问。



   测试体验阶段，可将公网 IP 地址段设置为：0.0.0.0/0。后续运行阶段，可将下文步骤2中涉及的 NAT 网关出口的弹性 IP 添加到公网白名单。
2. 为 Serverless 集群开启公网访问。
   Serverless 集群默认不开通外网访问，需通过 NAT 网关进行外网访问，详情可参见 通过 NAT 网关访问外网 。

完成 NAT 网关配置后，将 Serverless 集群所在子网关联至 NAT 网关的路由表，并确保 NAT 网关出口的弹性 IP 在访问容器镜像服务的白名单里（请参见上文 步骤1），Serverless 集群即可正常访问容器镜像服务，从公网拉取镜像。

常见报错指引

- 报错：`Error: failed to do request: Head "xxxx/manifests/late-st": dial tcp xxx:443: i/o timeout`

  如果报错信息中包含 “443: i/o timeout”，大部分情况是因为 TKE Serverless 到 TCR 网络不通。请参考上文 确保 Serverless 集群到镜像服务网络打通，选择其中一种访问方式，打通 TKE Serverless 到 TCR 的网络。注意，<tcr-name>.tencentcloudcr.com 的域名默认会解析成公网 IP，报错时请注意识别 “dial tcp xxx” 里的 IP 地址是公网还是内网，按实际情况处理。
- 报错：

  ```
  Error: code = Unknown, pull access denied, repository does not exist or may require authorization: server message: insufficient_scope: authorization failed
  ```

  如果报错信息中包含 “insufficient_scope: authorization failed”，说明 TKE Serverless 到 TCR 网络已经开通，但权限不足。可能原因为命名空间不存在、密钥不正确、密钥不适用正在拉取的镜像等。
- 报错：`Error: code = NotFound, failed to resolve reference "xxx:xxx": not found`

  如果报错信息中包含 “not found”，说明镜像不存在。

如果以上报错指引无法解决问题，请参见 容器镜像服务常见问题指引 。

Serverless 集群如何使用自建的自签名镜像仓库或 HTTP 协议镜像仓库?

问题描述

在 Serverless 集群使用自建镜像仓库的镜像创建工作负载，可能会遇到 “ErrImagePull” 报错，拉取镜像失败。如下图所示：

问题原因

通常在保证网络连通性的前提下，可能有以下两个原因导致该问题：

- 自建镜像仓库采用 HTTPS 协议，但是 HTTPS 协议证书是自签名。
- 自建镜像仓库采用 HTTP 协议。

上述两个问题都可以通过在工作负载 Yaml 配置里的 PodTemplate 中添加 Annotation 来解决。

解决步骤

HTTPS 自签名镜像仓库

如果自建的镜像仓库是 HTTPS 协议且证书是自签名。则需要在 PodTemplate 中，添加如下 Annotation，使其跳过证书检验：

eks.tke.cloud.tencent.com/registry-insecure-skip-verify: 镜像仓库地址（多个用 “,” 隔开，或者填写 all）

如下图所示：

```
spec:
  replicas: 1
  selector:
    matchLabels:
      k8s-app: sampleapp
      qcloud-app: sampleapp
  template:
    metadata:
      annotations:
        eks.tke.cloud.tencent.com/registry-insecure-skip-verify: your-registry-url
      creationTimestamp: null
      labels:
        k8s-app: sampleapp
        qcloud-app: sampleapp
```

说明：
如果 Pod 内多个容器的镜像从不同仓库拉取，可填写多个镜像仓库地址，中间用 “,” 隔开。或填写 “all”，表示 Pod 内所有容器镜像的相关仓库均跳过证书检验。

HTTP 协议镜像仓库

说明
默认 Serverless 集群运行时会使用 HTTPS 协议拉取镜像，因此如果镜像仓库支持的协议为 HTTP，则也需要通过 Annotation 进行说明。

通过命令 “$kubectl describe pod $podname”，发现输出 “**http: server gave HTTP response to HTTPS client**” 报错，则说明所访问的镜像仓库使用 HTTP 协议。如下图所示：

```
Events:
  Type     Reason     Age                  From                        Message
  ----     ------     ----                 ----                        -------
  Normal   Scheduled  11m                  default-scheduler           Successfully assigned defau
lt/busybox-764db787c8-4w894 to eklet-subnet-es398vv5
  Normal   Starting   11m                  eklet, eklet                Starting pod sandbox eks-71
cpsr8b
  Normal   Starting   10m                  eklet, eklet-subnet-es398vv5  Sync endpoints
  Normal   Pulling    9m12s (x4 over 10m)  eklet, eklet-subnet-es398vv5  Pulling image "10.16.100.17
4:5000/busybox"
  Warning  Failed     9m12s (x4 over 10m)  eklet, eklet-subnet-es398vv5  Failed to pull image "10.16
.100.174:5000/busybox": rpc error: code = Unknown desc = failed to pull and unpack image "10.16.100.
174:5000/busybox:latest": failed to resolve reference "10.16.100.174:5000/busybox:latest": failed to
 do request: Head "https://10.16.100.174:5000/v2/busybox/manifests/latest": http: server gave HTTP r
esponse to HTTPS client
  Warning  Failed     9m12s (x4 over 10m)  eklet, eklet-subnet-es398vv5  Error: ErrImagePull
  Normal   BackOff    8m58s (x6 over 10m)  eklet, eklet-subnet-es398vv5  Back-off pulling image "10.
16.100.174:5000/busybox"
  Warning  Failed     38s (x42 over 10m)   eklet, eklet-subnet-es398vv5  Error: ImagePullBackOff
```

解决此问题需要在 PodTemplate 中，添加如下 Annotation，使其使用 HTTP 协议访问镜像仓库：
eks.tke.cloud.tencent.com/registry-http-endpoint: 镜像仓库地址(多个用 “,” 隔开，或者填写all)

如下图所示：

```
spec:
  replicas: 1
  selector:
    matchLabels:
      k8s-app: sampleapp
      qcloud-app: sampleapp
  template:
    metadata:
      annotations:
        eks.tke.cloud.tencent.com/registry-http-endpoint: your-registry-url
      creationTimestamp: null
      labels:
        k8s-app: sampleapp
        qcloud-app: sampleapp
```

> **说明**
> 如果 Pod 内多个容器的镜像从不同仓库拉取，可填写多个镜像仓库地址，中间用 “,” 隔开。或填写 “all”，表示 Pod 内所有容器镜像均采用 HTTP 协议拉取。

镜像仓库地址填写说明

以上两个 Annotation 均涉及镜像仓库地址的填写，多个仓库地址可用 “,” 隔开。

> **注意：**
> 如果镜像仓库有端口号，则需要带上端口号。

例如，镜像地址为 10.16.100.174:5000/busybox:latest，则 Annotation 的 value 应填为 “10.16.100.174:5000”，即 “eks.tke.cloud.tencent.com/registry-insecure-skip-verify: 10.16.100.174:5000”。

运维类
基础监控相关

最近更新时间：2024-11-13 11:32:22

基础监控常见问题

节点 cpu/memory 分配量为什么会超出节点资源规格?

原因：node 层级的 cpu/memory 分配量指标依赖节点上各个 pod 的 cpu/memory request 来计算，在计算时没有将 failed 的 pod 排除。
示例：节点规格是 4c8g，节点上目前运行3个 pod（资源 request 用量如下）：

- pod1 正常运行其 request 为 2c4g；
- pod2 正常运行其 request 为 1c2g；
- pod3 状态为 failed 其 request 为 0.5c1g；

此时节点剩余可调度资源为 4-2-1=1c、8-4-2=2g，pod4 request 为 0.8c1.5g，满足调度器筛选，正常被调度到该节点上。此时节点上共4个 pod，3个正常1个异常，此时 node 层级的分配量为 4.3c8.5g（因计算时没有将 failed 的 pod 排除，因此超过了节点规格）。
该问题已在5月新版本中修复，即计算 node 资源分配量已把异常 pod 排除。

为什么 pod 状态显示正常，但监控指标 k8s_workload_abnormal 展示异常?

原因：该指标根据 workload 下 pod 是否异常来判断，pod 是否异常取决于 `pod.status.condition` 下的四个 Type。当这四个指标同时为 `True` 时 `k8s_workload_abnormal` 才会认为是正常，否则认为是异常。

- PodScheduled：Pod 已经被调度到某节点。
- ContainersReady：Pod 中所有容器都已就绪。
- Initialized：所有的 Init 容器都已成功完成。
- Ready：Pod 可以为请求提供服务，并且应该被添加到对应服务的负载均衡池中。

daemonSet tke-monitor-agent 报错原因总结

现象：无法解析域名 `receiver.barad.tencentyun.com`，导致指标上报失败，用户集群没有监控数据。
原因：节点 DNS 被修改。
解决措施：给 DaemonSet tke-monitor-agent 添加 hostAlias。参考代码如下：

```
hostAliases:
- hostnames:
  - receiver.barad.tencentyun.com
  ip: 169.254.0.4
```

日志采集相关

最近更新时间：2023-10-24 10:31:22

日志采集通用问题

集群配置日志采集后，为什么在日志服务控制台查看不到日志？

发生日志查看不到或者缺失的情况，请检查是否存在以下问题：

- **检查所选的日志 topic 是否开启了索引。**索引配置是使用日志服务进行检索分析的必要条件。若未开启，则无法查看日志。配置索引的详细操作，请参见 日志服务配置索引 。

- **日志、审计、事件是否选用同一个 topic。**若选用同一个 topic 则会发生日志覆盖，造成缺失。
- **检查所选的 topic 是否选用了两种提取模式。**新的提取模式会覆盖旧的提取模式。
- **检查是否存在软链接。**如果选择采集类型为“容器文件路径”时，对应的“容器文件路径”不能为软链接，否则会导致软链接的实际路径在采集器的容器内不存在，导致采集日志失败。

如果您的问题仍未解决，请 在线咨询 联系我们。

配置好日志规则后，日志在哪查看？

1. 登录 日志服务控制台 ，选择左侧导航中的**检索分析**。
2. 在**检索分析**页面，选择地域、需要查看日志的日志集和日志主题，开启全文索引，即可检索分析日志。

Prometheus 监控相关

最近更新时间：2024-08-28 18:00:11

TKE Serverless 如何对接腾讯云原生 Prometheus 监控?

1. 登录 容器服务控制台 ，选择左侧导航中的 **Prometheus 监控**。
2. 创建监控实例，操作详情请参见 创建监控实例 。
3. 完成创建后，在 **Prometheus 监控**列表页中单击**监控实例名称**进入监控实例详情页。
4. 在监控实例详情页，选择**数据采集 > 集成容器服务**。
5. 单击**关联集群**。如下图所示：
   - **集群类型**：选择 “Serverless 集群” 。
   - **集群**：勾选当前 VPC 下需要关联的集群。
6. 单击**确定**完成关联集群。
7. 在**关联集群**页签，单击集群 ID 右侧的**数据采集配置**，完成数据采集规则配置。操作详情请参见 数据采集配置 。
8. 在**基本信息**页签，查看 Grafana 信息。登录指定的 Grafana 地址并输入账号密码即可查看监控数据。

TKE Serverless 如何对接自建 Prometheus?

前提条件

- 已创建 Prometheus。
- 已安装 Prometheus Operator。
- 已配置 Grafana。

在 TKE Serverless 集群中，需要获取以下监控指标：

指标类型	采集源	发现类型
k8s资源指标	kube-state-metrics	通过 coredns 访问域名
容器运行时指标	pod 的 metrics 接口	k8s_sd pod 级别

监控 k8s 资源指标

若您希望监控 k8s 的资源指标，可以通过在 TKE Serverless 集群内部署 kube-state-metrics 组件及编写 ServiceMonitor 实现。

1. **在 TKE Serverless 集群内部署 kube-state-metrics 组件。**

   如果您在 TKE Serverless 的集群内已经部署了 Prometheus Operator 会发现对应的 kube-state-metrics 组件和 node exporter 的 Pod 是 pending 状态，这是因为它们并不适用于 TKE Serverless 集群的场景，node exporter 在 TKE Serverless 集群的监控中不需要使用，可以直接删除该 pod，同时我们需要重新部署 kube-state-metrics 组件，具体的部署内容如下所示：
   - kube-state-metrics-ClusterRole

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  labels:
    app.kubernetes.io/name: kube-state-metrics
    app.kubernetes.io/version: 1.9.7
  name: tke-kube-state-metrics
rules:
  - apiGroups:
      - ""
    resources:
      - configmaps
      - secrets
      - nodes
      - pods
```

```
      - services
      - resourcequotas
      - replicationcontrollers
      - limitranges
      - persistentvolumeclaims
      - persistentvolumes
      - namespaces
      - endpoints
    verbs:
      - list
      - watch
  - apiGroups:
      - extensions
    resources:
      - daemonsets
      - deployments
      - replicasets
      - ingresses
    verbs:
      - list
      - watch
  - apiGroups:
      - apps
    resources:
      - statefulsets
      - daemonsets
      - deployments
      - replicasets
    verbs:
      - list
      - watch
  - apiGroups:
      - batch
    resources:
      - cronjobs
      - jobs
    verbs:
      - list
      - watch
  - apiGroups:
      - autoscaling
    resources:
      - horizontalpodautoscalers
    verbs:
      - list
      - watch
  - apiGroups:
      - authentication.k8s.io
    resources:
      - tokenreviews
    verbs:
      - create
  - apiGroups:
      - authorization.k8s.io
    resources:
      - subjectaccessreviews
    verbs:
      - create
  - apiGroups:
```

```
      - policy
    resources:
      - poddisruptionbudgets
    verbs:
      - list
      - watch
  - apiGroups:
      - certificates.k8s.io
    resources:
      - certificatesigningrequests
    verbs:
      - list
      - watch
  - apiGroups:
      - storage.k8s.io
    resources:
      - storageclasses
      - volumeattachments
    verbs:
      - list
      - watch
  - apiGroups:
      - admissionregistration.k8s.io
    resources:
      - mutatingwebhookconfigurations
      - validatingwebhookconfigurations
    verbs:
      - list
      - watch
  - apiGroups:
      - networking.k8s.io
    resources:
      - networkpolicies
    verbs:
      - list
      - watch
  - apiGroups:
      - coordination.k8s.io
    resources:
      - leases
    verbs:
      - list
      - watch
```

- kube-state-metrics-service-ClusterRoleBinding

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  labels:
    app.kubernetes.io/name: kube-state-metrics
    app.kubernetes.io/version: 1.9.7
  name: tke-kube-state-metrics
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: tke-kube-state-metrics
subjects:
  - kind: ServiceAccount
```

```
    name: tke-kube-state-metrics
    namespace: kube-system
```

- kube-state-metrics-deployment

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app.kubernetes.io/name: kube-state-metrics
    app.kubernetes.io/version: 1.9.7
  name: tke-kube-state-metrics
  namespace: kube-system
spec:
  replicas: 1
  selector:
    matchLabels:
      app.kubernetes.io/name: kube-state-metrics
  template:
    metadata:
      labels:
        app.kubernetes.io/name: kube-state-metrics
        app.kubernetes.io/version: 1.9.7
    spec:
      containers:
        - image: ccr.ccs.tencentyun.com/tkeimages/kube-state-metrics:v1.9.7
          livenessProbe:
            httpGet:
              path: /healthz
              port: 8080
            initialDelaySeconds: 5
            timeoutSeconds: 5
          name: kube-state-metrics
          ports:
            - containerPort: 8080
              name: http-metrics
            - containerPort: 8081
              name: telemetry
          readinessProbe:
            httpGet:
              path: /
              port: 8081
            initialDelaySeconds: 5
            timeoutSeconds: 5
          securityContext:
            runAsUser: 65534
      serviceAccountName: tke-kube-state-metrics
```

- kube-state-metrics-service

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app.kubernetes.io/name: kube-state-metrics
    app.kubernetes.io/version: 1.9.7
  name: tke-kube-state-metrics
  namespace: kube-system
```

```
spec:
  clusterIP: None
  ports:
    - name: http-metrics
      port: 8180
      targetPort: http-metrics
    - name: telemetry
      port: 8181
      targetPort: telemetry
  selector:
    app.kubernetes.io/name: kube-state-metrics
```

- kube-state-metrics-serviceaccount

```
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    app.kubernetes.io/name: kube-state-metrics
    app.kubernetes.io/version: 1.9.7
  name: tke-kube-state-metrics
  namespace: kube-system
```

2. **在 TKE Serverless 集群内部署 ServiceMonitor。**
   ServiceMonitor 可以定义如何监控一组动态服务，部署 kube-state-metrics-servicemonitor 后，Prometheus 可以通过 kube-state-metrics 来收集 k8s 的资源指标。具体的部署内容如下所示：
   - kube-state-metrics-servicemonitor

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  labels:
    app.kubernetes.io/name: kube-state-metrics
    app.kubernetes.io/version: 1.9.7
  name: kube-state-metrics
  namespace: kube-system
spec:
  endpoints:
    - interval: 15s
      port: http-metrics
      scrapeTimeout: 15s
      honorLabels: true
  jobLabel: app.kubernetes.io/name
  selector:
    matchLabels:
      app.kubernetes.io/name: kube-state-metrics
```

监控容器运行时指标

TKE Serverless 中的 Pod 通过暴露9100端口向外提供监控数据，您可以通过访问 `podip：9100/metrics` 获取监控数据指标。相较于容器服务 TKE 标准的监控配置，监控 TKE Serverless 需要修改相应的配置文件，建议使用 Operator 的 additional scrape config 配置。此外，您也可以通过在 Pod 中添加 annotation 的方式对指定的 Pod 进行监控。

1. **通过配置 Operator 的 additional\sscrape\sconfig 获取监控数据指标。**
   若您希望通过访问 `podip：9100/metrics` 获取监控数据指标，可执行以下步骤：
   1.1 新建 prometheus-additional.yaml 文件。
   1.2 在文件中添加 scrape_configs。scrape_configs 内容如下所示：

```
- job_name: eks-info
  honor_timestamps: true
  metrics_path: /metrics
  scheme: http
  kubernetes_sd_configs:
  - role: pod
  bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
  tls_config:
    insecure_skip_verify: true
  relabel_configs:
  - source_labels: [__meta_kubernetes_pod_ip]
    separator: ;
    regex: (.*)
    target_label: __address__
    replacement: ${1}:9100
    action: replace
  - source_labels: [__meta_kubernetes_pod_name]
    separator: ;
    regex: (.*)
    target_label: pod_name
    replacement: ${1}
    action: replace
  metric_relabel_configs:
  - source_labels: [__name__]
    separator: ;
    regex: node_network_receive_packets_total
    target_label: __name__
    replacement: container_network_receive_packets_total
    action: replace
  - source_labels: [__name__]
    separator: ;
    regex: node_network_receive_bytes_total
    target_label: __name__
    replacement: container_network_receive_bytes_total
    action: replace
  - source_labels: [__name__]
    separator: ;
    regex: node_network_transmit_bytes_total
    target_label: __name__
    replacement: container_network_transmit_bytes_total
    action: replace
  - source_labels: [__name__]
    separator: ;
    regex: node_network_transmit_packets_total
    target_label: __name__
    replacement: container_network_transmit_packets_total
    action: replace
  - source_labels: [pod_name]
    separator: ;
    regex: (.*)
    target_label: pod
    replacement: $1
    action: replace
  - source_labels: [__name__]
    separator: ;
    regex: (container_network.*|pod_.*)
    replacement: $1
    action: keep
  - separator: ;
```

```
    regex: pod_name|node|unInstanceId|workload_kind|workload_name
    replacement: $1
    action: labeldrop
```

1.3 完成部署后，连接 Grafana 获取相应数据。

2. **通过在 Pod 中添加 annotation 对指定 Pod 进行监控。**

若您希望通过在 Pod 中添加 annotation 的方式对指定的 Pod 进行监控，可执行以下步骤：

2.1 修改需要进行采集的 Pod 的 yaml 文件，在 spec.template.metadata.annotations 中配置以下内容：

```
prometheus.io/scrape: 'true'
prometheus.io/port: '9100'
prometheus.io/path: 'metrics'
```

2.2 配置 scrape_configs。配置 scrape_configs 后，prometheus 会对所有配置过采集信息为 true 的 Pod 进行监控。scrape_configs 请参考以下配置：

```
- job_name: kubernetes-pods
  honor_timestamps: true
  metrics_path: /metrics
  scheme: http
  kubernetes_sd_configs:
  - role: pod
  relabel_configs:
  - source_labels: [__meta_kubernetes_pod_annotation_prometheus_io_scrape]
    separator: ;
    regex: "true"
    replacement: $1
    action: keep
  - source_labels: [__meta_kubernetes_pod_annotation_prometheus_io_path]
    separator: ;
    regex: (.+)
    target_label: __metrics_path__
    replacement: $1
    action: replace
  - source_labels: [__address__, __meta_kubernetes_pod_annotation_prometheus_io_port]
    separator: ;
    regex: ([^:]+)(?::\d+)?;(\d+)
    target_label: __address__
    replacement: $1:$2
    action: replace
  - separator: ;
    regex: __meta_kubernetes_pod_label_(.+)
    replacement: $1
    action: labelmap
  - source_labels: [__meta_kubernetes_namespace]
    separator: ;
    regex: (.*)
    target_label: kubernetes_namespace
    replacement: $1
    action: replace
  - source_labels: [__meta_kubernetes_pod_name]
    separator: ;
    regex: (.*)
    target_label: kubernetes_pod_name
    replacement: $1
    action: replace
```

服务类

最近更新时间：2024-05-22 10:24:51

创建服务常见问题

服务的名称为什么不能重复?

服务名称是当前集群下的服务的唯一标识，服务之间可以通过服务名称+访问端口的形式互相访问。

创建服务能否使用非腾讯云或 dockerhub 镜像的第三方镜像?

您可以通过登录到主机执行 docker login 命令登录到第三方镜像仓库拉取。

使用外网服务的有什么前置条件?

确保集群内的云服务器拥有外网带宽，否则外网服务将创建失败。

内存限制，CPU 限制如何填写?

详情请参见 容器服务资源限制说明 。

创建服务时的特权级是什么意思?

开启该选项会使得容器内程序具有真正的 root 权限。在容器内程序需要进行高级系统操作时建议开启，如搭建 nfs 服务器。

服务能创建包年包月计费模式的负载均衡吗?

包年包月计费模式的负载均衡不能在服务删除时同步进行删除，会使得资源无法回收，导致用户负载均衡资源的浪费。
服务自动创建的负载均衡计费模式均为按量计费，用户可以通过使用已有负载均衡的方式使用包年包月的已有负载均衡。详情请参见 Service 使用已有 CLB 及 Service 工作原理 。

负载均衡可以在创建时就指定安全组吗?

可以，目前支持以下两个方案，实现服务使用负载均衡时指定安全组：

- 使用已有负载均衡。先创建负载均衡并配置安全组，再挂载给服务。详情请参见 Service 使用已有 CLB 。
- 可在服务中通过 `TkeServiceConfig` 配置安全组，负载均衡创建时会根据配置使用对应安全组。如需使用此功能，请 在线咨询 进行申请。

> **说明：**
> 集群内进行服务访问时，建议不要通过负载均衡 IP 进行访问，以避免出现访问不通的情况。
> 一般情况下，4层负载均衡会绑定多台 Node 作为 real server（rs），使用时需要限制 client 和 rs 不能存在于同一台云服务器上，否则会有一定概率导致报文回环失败。
> 当 Pod 去访问负载均衡时，Pod 为源 IP，当其传输到内网时负载均衡也不会做 snat 处理将源 IP 转化为 Node IP，则负载均衡收到报文时无法判断是哪个 Node 发送的，避免回环策略就无法生效，所有的 rs 都可能被转发。当转发到 client 所在的 Node 上时，负载均衡就无法收到回包，从而导致访问不通。

更新服务容器数量常见问题

更新容器数量需注意哪些问题?

需确认 CPU、内存资源充足，否则容器将创建失败。

能否将容器数量设为0?

可以，通过将容器数量设置为0，保存服务配置并且释放资源占用。

更新服务配置常见问题

更新服务是否支持滚动更新?

支持滚动更新和快速更新两种方式。

公网负载均衡可以切换为内网负载均衡吗?

可以，目前支持公网切换至 VPC 内网、VPC 内网切换至公网及 VPC 不同子网间切换。详情请参见 Service 生命周期管理 。

> **注意：**

- 若为服务负责负载均衡资源的生命周期管理，则负载均衡及其公网 IP 将会被释放。
- 公网切换内网的过程并不是瞬间的，公网负载均衡服务下线到内网负载均衡并提供服务，此过程需要一定的时间。建议您先在集群中配置一个内网服务资源，并进行测试。等到流量切换完成之后，再删除原有公网服务资源。

删除服务常见问题

删除服务后服务创建的负载均衡会自动销毁么？

删除服务时，将会同时删除创建该服务时自动创建的负载均衡。若在创建服务时选用的是已有负载均衡，则该负载均衡将不会受到任何影响。

删除服务是否会影响业务数据？

删除服务不会删除业务容器，数据不会受到影响，该操作无需提前备份数据。

服务运行常见问题

如何设置容器系统时间为北京时间？

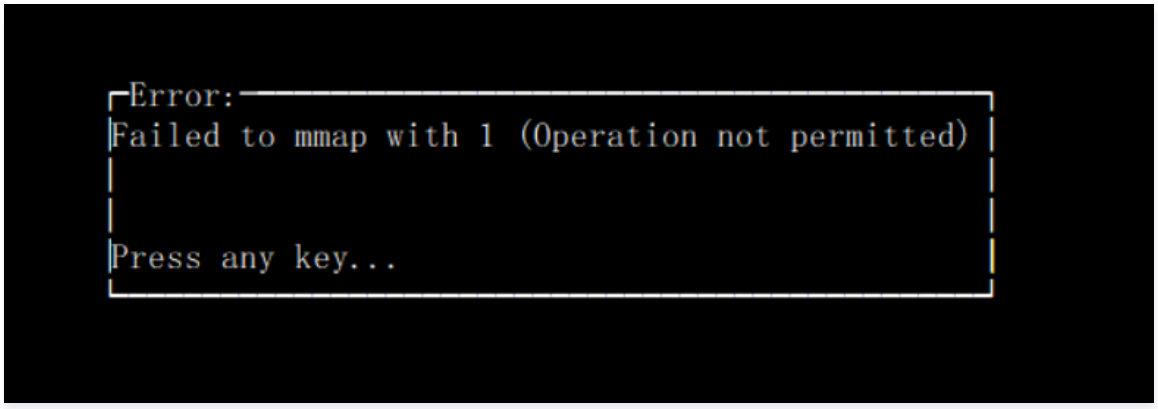
容器默认使用 UTC 时间，使用容器时经常碰到容器系统时间和北京时间差8小时的问题，解决方法是在 dockerfile 中创建时区文件。详情请参见 解决容器内时区不一致问题 。

Dockerhub 部分镜像如 ubuntu、php 和 busybox 等在容器服务里运行异常怎么办？

运行异常是因为没有设置启动命令或者默认的启动命令为 bash，导致容器执行完启动程序就退出了。要使容器一直运行，容器里 PID 为1的进程必须是常驻进程，否则 PID 为1的进程一结束容器就退出了。对于部分镜像如 centos 等，可以使用 /bin/bash 作为运行命令， -c sleep 800000 作为运行参数来创建服务，在控制台填运行参数时 -c 和 sleep 800000 必须放在两行。
目前已知的使用默认参数启动不了服务的镜像包括这些：clearlinux、ros、mageia、amazonlinux、ubuntu、clojure、crux、gcc、photon、java、debian、oraclelinux、mono、bash、buildpack-deps、golang、sourcemage、swift、openjdk、centos、busybox、docker、alpine、ibmjava、php和python。

容器执行 perf top -p 查看进程 CPU 情况提示 “Operation not permitted”

容器执行 `perf top -p` 查看进程 CPU 情况提示 “Operation not permitted”，如下图所示：



Docker 默认配置文件阻止了重要的系统调用，perf_event_open 可能会泄漏主机上的大量信息所以被禁止使用。如果您需要调用请配置特权容器或者直接修改 Pod yaml 字段 privileged 为 true，您需自行评估安全风险。

Service/Ingress FAQ

最近更新时间：2025-02-08 10:29:52

如何判断更新配置后是否成功下发？

您可以先查看资源的 Event 信息，然后根据资源同步的返回码或错误码来判断资源同步是否遇到问题。如果返回码或错误码表明同步失败，您可以参考 Service&Ingress 常见报错和处理 进一步查看错误信息并采取相应的措施来解决问题。

负载均衡监听器上没有绑定任何后端，应该怎么排查问题？

您可以根据接入方式分情况进行排查：

- 通过 NodePort 接入：
  - 检查 Service 是否使用了后端选择能力，导致没有节点匹配上。
  - 检查 Node 是否配置了 node.kubernetes.io/exclude-from-external-load-balancers label，声明节点不被用作外部流量接入。
  - 检查 Service 是否开启了 Local 绑定的能力，或是 Ingress 引用了 Local 模式的 Service，同时没有健康的工作负载。
  - 检查 Ingress，注意检查 Service 类型是否为 ClusterIP，ClusterIP 类型的 Service 不能作为 Ingress 后端使用。
  - 用户是一个独立集群，只包含 Master 节点，默认 Master 节点不会作为后端承接入口流量。如果用户不添加普通节点，将会导致没有可用的后端节点，从而无法访问集群中的应用程序。
- 通过直连方式接入：
  - 检查工作负载是否采用了 HostNetwork 网络。
  - Ingress 注意检查 Service 是否存在，指定的 Service 端口是否存在。
  - Service 注意检查 Endpoint 是否为空，以此来判断 Service 的 LabelSelector 是否出现错误。

Pod 滚动更新卡住，Pod Status 中显示 cloud.tencent.com/load-balancer-backendgroup-ready 不能更新到正常状态，如何排查问题？

您可以按照以下流程进行排查：

1. 根据失败的工作负载 Pod，查询 Status 中的 Condition Message。目前 Pod 的 Condition 信息中会包含健康检查失败的负载均衡实例以及监听器等信息。

```
status:
  conditions:
  - lastProbeTime: null
    lastTransitionTime: "2022-12-14T07:54:33Z"
    message: readinessGate cloud.tencent.com/load-balancer-backendgroup-ready is False, Readiness
check failed according to l4 listener: lbl-ro8ixxxx UDP:21000) of CLB lb-qvnrxxxx. Service: ns-
production/gz3-service
    reason: LoadBalancerNetworkGroupNotReady
    status: "False"
    type: cloud.tencent.com/load-balancer-backendgroup-ready
```

   例如上面这个 Case，我们就可以关注到 Pod 关联的 Service(ns-production/gz3-service)，流量接入不正确，Pod 没有绑定到 lb-qvnrxxxx 或是健康检查不通过。

2. 确认后端是否正确绑定了负载均衡上。

   2.1 用户负载均衡绑定数量达到上限，导致工作负载没有绑定到 CLB 上，健康检查不通过。（默认 CLB 后端上限为200）

   2.2 用户工作负载使用了 HostNetwork 的网络模式，或非弹性网卡网络模式。工作负载的网络模式不支持直连，导致没有绑定为后端。

   2.3 查询 Service(ns-production/gz3-service) 同步的其他错误信息，解决配置错误等原因导致后端未绑定问题。

```
# 以下命令都可以查询到Event信息
kubectl describe service -n ns-production gz3-service
kubectl get event -n ns-production | grep gz3-service
```

   2.4 找不到 Ingress 关联的 CLB ，说明 CLB 被删除了，可能是用户在 CLB 控制台上操作了删除 CLB，导致 Pod 无法绑定到 CLB（删除操作可在 操作审计 中确认）。如果是复用的 CLB，需修改 Ingress，切换到其它没被删除的 CLB 上；如果是自动创建的 CLB，需删除重建一下

Ingress，具体请参见 重建 Ingress 。

3. 当后端已经绑定在负载均衡时：

   3.1 在负载均衡控制台，确认监听器下的 Pod 的健康状态是否正常。当负载均衡健康检查失败时，可以通过请求后端服务，判断后端服务是否可用。（根据 CLB 运营平台上查询到的健康检查配置进行请求）

```
curl -H "Host: gz3.service.com" -X HEAD "9.148.100.100:80/checkServiceStatus"
curl -H "Host: gz3.service.com" -X GET "9.148.100.100:80/checkServiceStatus"
```

   3.2 确认 Pod 所在的子机的安全组是否有拦截，Pod 所在的子网的 ACL 是否有拦截，在 CLB 控制台可以针对后端进行连通性探测。

开启直连之后，Pod 的请求来源被改成了节点 IP，应该怎么解决？

您可以注意检查 IpMasq 配置，避免来自负载均衡的直连请求被进行 IP 伪装。这个问题会影响 GlobalRoute 直连和 RouteENI 弹性网络直连。

当前通过 NodePort 转发的工作负载，可以直接更新成直连 Pod 吗？

可以，您可以通过控制台表单或者修改资源注解的方式，直接将流量转发的方式切换成直连。
需要注意以下几点：

1. 直连转发相较于 NodePort 转发，会避免流量进行 FullNAT，而直接从负载均衡转发到 Pod 的网卡。这个区别会导致来源 IP、会话保持等特性出现一些区别。
2. 当用户 NodePort 切换成直连时，组件会优先绑定新的直连后端，然后删除原有的 NodePort 后端，不会造成断流。但是要注意集群的规模，如果用户 NodePort 的数量已经达到或超过 CLB 的后端限制（默认为200），那么就只能先删除原有的绑定后端再绑定，可能会出现短时间的断流。

相关文档：

1. 直连功能介绍与使用文档：使用 LoadBalancer 直连 Pod 模式 Service
2. NodePort 转发示意图：Service 后端选择

集群节点超200了，导致负载均衡上有的节点没有绑定，会有问题吗？

如果是部分节点没有绑定，不会存在问题。负载均衡监听器后端数量默认限制200个。区分以下两种接入方式来理解这个问题。

- NodePort 接入。
  - 默认的接入方式。当用户发现负载均衡上绑定的是节点，并且端口是 Service 上的 NodePort 端口时，基本可以确认您使用的是该接入方式。
  - 每一个节点上的 NodePort 都会收到来自负载均衡的流量，然后随机转发给工作负载。所以并不一定需要绑定集群所有节点。
- 直连接入。
  - 弹性网卡网络常见的接入方式。当用户发现负载均衡上绑定的是 Pod 的 IP 和服务端口时，基本可以确认您使用的是该接入方式。
  - 绕过了 NodePort 转发，负载均衡会直接将流量转发到工作负载上，这就需要 Pod 全部绑定在负载均衡后端。
  - 再考虑到滚动更新会导致副本数量有额外增加，用户需要保证工作负载最大的副本数小于200（例如工作负载副本数150个，每批滚动最多新增工作负载50个）。如果业务需要工作负载的规模需要超过这个值，需要及时通过 工单 方式联系客服为您开通白名单。

相关文档：

1. Kubernetes Service NodePort：https://kubernetes.io/docs/concepts/services-networking/service/#type-nodeport
2. 直连功能介绍与使用文档：使用 LoadBalancer 直连 Pod 模式 Service

Ingress 的证书过期了怎么办

直接在我的证书中更新 Ingress 所关联的证书即可，Ingress 中引用的证书 ID 会自动更新。

镜像仓库类

最近更新时间：2025-01-08 17:10:12

开通镜像仓库常见问题

命名空间有什么作用？

命名空间是标识用户私人镜像的地址前缀。

镜像仓库的账户是什么？

默认是用户的腾讯云账号（QQ 号）。

开通时创建的密码忘记了怎么办？

可以通过控制台重置密码。

创建镜像常见问题

创建镜像有配额限制么？

默认配额是单个地域可创建500个镜像仓库，单个镜像仓库可创建100个镜像版本。
如果您需要更多的配额项数量，可通过 在线咨询 提出配额申请。

创建的镜像可以分享给其他用户么？

暂不提供该功能。

创建的镜像如何使用？

需要先上传可用的镜像版本，通过具体的镜像版本来创建服务。

删除镜像常见问题

如何删除镜像某一个版本？

直接在控制台指定删除某一个具体的版本。

在镜像列表删除镜像会删除该镜像的所有版本么？

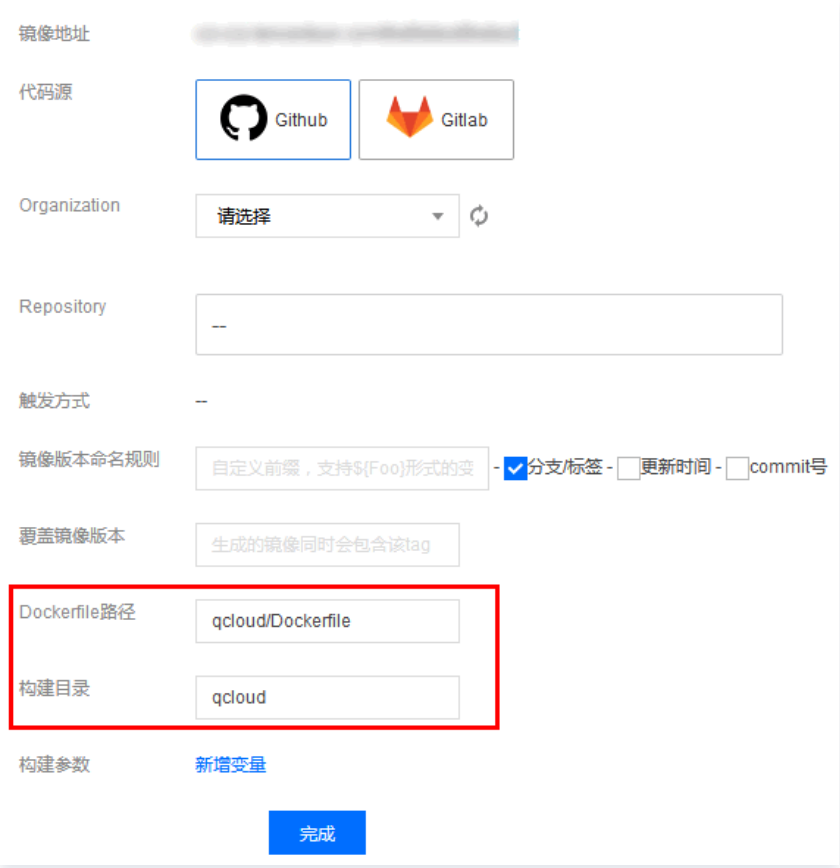
是的，删除镜像时会删除该镜像的所有镜像版本，请提前备份好数据。

镜像构建常见问题

使用源码构建镜像时 Dockerfile 路径与构建目录该怎么填？

- 若不填写路径，系统使用以下默认值：
  - Dockerfile 路径默认值：代码仓库根目录下的 Dockerfile（ `Dockerfile` ）。
  - 构建目录默认值：代码仓库根目录（ `./` ）。

- 若需填写路径，则填写以项目为根路径的相对路径。如下图所示：



源码构建功能使用 Dockerfile 路径及构建目录细节是什么？

源码构建功能首先 clone 用户指定的仓库，切换到相应的分支（branch）或标签（tag），然后在代码仓库根目录执行 `docker build -f $DOCKERFILE_PATH $WORKDIR` 命令构建出容器镜像。

Dockerfile 文件中的源路径应该怎么写？

对于 `COPY`， `ADD` 等涉及源路径的命令，源路径应该写成相对于构建目录的相对路径。

远程终端类

最近更新时间：2024-11-07 16:54:32

容器里面没有 bash，怎么办?

如果容器内没有 bash，您可以在命令行中输入您想执行的命令，屏幕会显示该命令的返回结果。您可以将命令行看做一个缺少自动补全等其他功能的精简版 bash。建议您先执行安装 bash 的命令，再执行后续操作。

为什么运行 apt-get 安装软件如此之慢?

如果您在使用 `apt-get` 安装软件速度过慢，有可能是因为机器访问国外软件源速度过慢引起。我们根据不同操作系统的容器为您提供对应的提速方案，您可根据实际情况进行操作：

注意事项

选择提速解决方案之前，请您准确识别容器操作系统，避免操作无法正常进行。您可参考以下方式进行识别：

- Ubuntu 系统：执行命令 `cat /etc/lsb-release` ，验证是否存在 `/etc/lsb-release` 文件。
- CentOS 系统：执行命令 `cat /etc/redhat-release` ，验证是否存在 `/etc/redhat-release` 文件。
- Debian 系统：执行命令 `cat /etc/debian_version` ，验证是否存在 `/etc/debian_version` 文件。

解决方案

Ubuntu 16.04系统

对于系统为 Ubuntu 16.04的容器，您可以在终端运行以下命令，将 apt 的源设置为腾讯云的源服务器：

```
cat << ENDOF > /etc/apt/sources.list
deb http://mirrors.tencentyun.com/ubuntu/ xenial main restricted universe multiverse
deb http://mirrors.tencentyun.com/ubuntu/ xenial-security main restricted universe multiverse
deb http://mirrors.tencentyun.com/ubuntu/ xenial-updates main restricted universe multiverse
deb-src http://mirrors.tencentyun.com/ubuntu/ xenial main restricted universe multiverse
deb-src http://mirrors.tencentyun.com/ubuntu/ xenial-security main restricted universe multiverse
deb-src http://mirrors.tencentyun.com/ubuntu/ xenial-updates main restricted universe multiverse
ENDOF
```

CentOS 7系统

对于系统为 CentOS 7的容器，您可以在终端执行以下操作，直接修改源地址提高安装速度：

1. 将以下代码复制并粘贴至容器内运行：

```
cat << ENDOF > /etc/yum.repos.d/CentOS-Base.repo
[os]
name=Qcloud centos os - \$basearch
baseurl=http://mirrors.tencentyun.com/centos1/\$releasever/os/\$basearch/
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
[updates]
name=Qcloud centos updates - \$basearch
baseurl=http://mirrors.tencentyun.com/centos1/\$releasever/updates/\$basearch/
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
#[centosplus]
#name=Qcloud centosplus - \$basearch
#baseurl=http://mirrors.tencentyun.com/centos1/\$releasever/centosplus/\$basearch/
#enabled=1
#gpgcheck=1
#gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
```

```
#[cloud]
#name=Qcloud centos contrib - \$basearch
#baseurl=http://mirrors.tencentyun.com/centos1/\$releasever/cloud/$basearch/openstack-kilo/
#enabled=1
#gpgcheck=1
#gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
#[cr]
#name=Qcloud centos cr - \$basearch
#baseurl=http://mirrors.tencentyun.com/centos1/\$releasever/cr/\$basearch/
#enabled=1
#gpgcheck=1
#gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
[extras]
name=Qcloud centos extras - \$basearch
baseurl=http://mirrors.tencentyun.com/centos1/\$releasever/extras/\$basearch/
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
#[fasttrack]
#name=Qcloud centos fasttrack - \basearch
#baseurl=http://mirrors.tencentyun.com/centos1/\$releasever/fasttrack/\$basearch/
#enabled=1
#gpgcheck=1
#gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
ENDOF
```

2. 执行以下命令，清空并重建 YUM 缓存。

```
yum clean all && yum clean metadata && yum clean dbcache && yum makecache
```

Debian 系统

对于系统为 Debian 的容器，您可以在终端运行以下命令，将 apt 的源设置为腾讯云的源服务器：

```
cat << ENDOF > /etc/apt/sources.list
deb http://mirrors.tencentyun.com/debian stretch main contrib non-free
deb http://mirrors.tencentyun.com/debian stretch-updates main contrib non-free
deb http://mirrors.tencentyun.com/debian-security stretch/updates main

deb-src http://mirrors.tencentyun.com/debian stretch main contrib non-free
deb-src http://mirrors.tencentyun.com/debian stretch-updates main contrib non-free
deb-src http://mirrors.tencentyun.com/debian-security stretch/updates main
ENDOF
```

问题总结

在容器中直接修改源地址为临时解决方案，当容器被重新调度后，您所做的修改将会失效，建议您在创建镜像时解决该问题。具体的操作方法如下：
修改创建容器镜像的 Dockerfile，在 Dockerfile 的 RUN 字段中，添加对应操作系统 解决方案 中提供的信息修改源地址。例如，在一个基于 Ubuntu 系统的镜像的 Dockerfile 中，加入以下内容：

```
RUN cat << ENDOF > /etc/apt/sources.list
deb http://mirrors.tencentyun.com/ubuntu/ xenial main restricted universe multiverse
deb http://mirrors.tencentyun.com/ubuntu/ xenial-security main restricted universe multiverse
deb http://mirrors.tencentyun.com/ubuntu/ xenial-updates main restricted universe multiverse
#deb http://mirrors.tencentyun.com/ubuntu/ xenial-proposed main restricted universe multiverse
#deb http://mirrors.tencentyun.com/ubuntu/ xenial-backports main restricted universe multiverse
deb-src http://mirrors.tencentyun.com/ubuntu/ xenial main restricted universe multiverse
deb-src http://mirrors.tencentyun.com/ubuntu/ xenial-security main restricted universe multiverse
```

```
deb-src http://mirrors.tencentyun.com/ubuntu/ xenial-updates main restricted universe multiverse
#deb-src http://mirrors.tencentyun.com/ubuntu/ xenial-proposed main restricted universe multiverse
#deb-src http://mirrors.tencentyun.com/ubuntu/ xenial-backports main restricted universe multiverse
ENDOF
```

当登录容器后，发现没有 vim，netstat 等工具，怎么办？

您可以通过 apt-get install vim， net-tools 等命令下载您所需要的工具（CentOS 下执行 yum install vim）。

为什么运行 apt-get install 命令，提示找不到工具？

您可以通过以下操作，安装软件。

1. 执行以下命令，升级软件列表。

```
apt-get update
```

2. 执行以下命令， 安装软件（CentOS 下执行 yum updateinfo）。

```
apt-get install
```

如果想在容器中使用自制的工具，怎么办？

您可以在进入远程终端页面后，单击右下方的文件助手，即可进行上传和下载操作。

如何拷贝现场文件，例如 dump 或者日志到本地分析？

您可以在进入远程终端页面后，单击右下方的文件助手，即可进行上传和下载操作。

为什么用不了文件上传到容器或者下载到本地功能？

可能因为您的容器镜像里没有安装 tar 程序，您可以通过 apt-get install vim，net-tools 等命令（CentOS 下执行 yum install vim）先安装 tar 程序再重试。

为什么之前安装的工具不见了？

可能因为 kubernetes 重新调度您的容器，调度过程中会拉取镜像生成新的容器，如果镜像里面没有您之前安装的工具，新的容器是不会包含这些工具的。建议您在制作镜像时，安装一些常用的排错工具。

怎么复制控制台里的文字？

只要选中您想复制的内容，即可复制被选中的文字。

怎么粘贴复制好的文字？

同时按下 Shift + Insert 即可。

为什么会断开链接？

可能因为您在腾讯云其他页面对容器、云服务器进行操作更改了容器的状况，也有可能是长时间（3分钟）不进行任何操作，服务器断开了这个链接。

运行 top 命令时出现 "TERM environment variable not set" 错误，怎么办？

执行命令 `export TERM=linux` 即可。

为何进入绝对路径较长的目录后，bash 提示符只显示 “<” 和部分路径？

因为默认的 bash 提示符被设置为显示 “用户名@主机名 当前目录”。如果当前路径长于一定长度，bash 默认会显示 “<” 与路径的最后一部分。

事件类

最近更新时间：2023-07-26 17:18:22

报错：Back-off restarting failed docker container

说明：如果报错信息中包含 “Back-off restarting failed Docker container”，说明正在重启异常的 Docker 容器。
解决方法：检查镜像中执行的 Docker 进程是否异常退出，若镜像内并无持续运行的进程，可在创建服务的页面中添加执行脚本。

报错：fit failure on node: Insufficient cpu

说明：如果报错信息中包含 “Insufficient CPU”，说明集群 CPU 不足。
解决方法：原因是节点无法提供足够的计算核心，请在服务页面修改 CPU 限制或者对集群进行扩容。

报错：no nodes available to schedule pods

说明：如果报错信息中包含 “No nodes available to schedule pods”，说明集群资源不足。
解决方法：原因是没有足够的节点用于承载实例，请在服务页面修改服务的实例数量或者 CPU 限制。

报错：pod failed to fit in any node

说明：如果报错信息中包含 “Pod failed to fit in any node”，说明没有合适的节点可供实例使用。
解决方法：原因是服务配置了不合适的资源限制，导致没有合适的节点用于承载实例，请在服务页面修改服务的实例数量或者 CPU 限制。

报错：Liveness probe failed

说明：如果报错信息中包含 “Liveness probe failed”，说明容器健康检查失败。
解决方法：请检查镜像内容器进程是否正常，检查检测端口是否配置正确。

报错：Error syncing pod, skipping

Error syncing pod, skipping failed to "StartContainer" for with CrashLoopBackOff: "Back-off 5m0s restarting failed container
说明：如果报错信息中包含 “Error syncing pod, skipping”，说明容器进程崩溃或退出。
解决方法：请检查容器内是否有持续运行的前台进程，若有检查其是否有异常行为。

如果以上解决方法未能解决您的问题，您可以咨询 在线客服 或 提交工单 来寻求帮助。

资源管理类

最近更新时间：2024-09-29 16:49:42

为什么容器要设置 Request/Limit?

容器在运行时通常需要消耗 CPU/内存资源，假设没有任何配置的情况下，容器能使用的资源上限为当前所在的节点可被分配的资源量。

- 一台节点通常会运行很多容器。
  假设一个节点上只有一个容器，那么这个容器在没有完全消耗节点资源的时间里，节点上空闲的资源被浪费了。现代一台普通的个人电脑通常可以运行数百个进程，同样的，一台节点通常会运行很多容器，但这引发了另外一个问题：容器之间可能抢夺资源，而节点本身的资源却是固定的。
- 需要使用 Limit 控制容器资源使用上限。
  Kubernetes 的世界里，通过 Limit 来限定一个容器能使用的资源量上限，如果容器申请的资源量大于 Limit，它就会被压制用量甚至被驱逐到其他节点上。
- 需要使用 Request 保证容器资源使用的下限。
  只设置 Limit 来控制容器资源使用上限就足够了吗？设想：如果一个10核的节点上运行了100个容器，但每个容器至少需要1核才能正常启动和运行，这时节点上的所有容器都无法正常运行。因此，Kubernetes 使用 Request 来保证容器最少的资源供给量。

总的来说，Kubernetes 使用 Request 和 Limit 来保证和限制一个容器使用的 CPU/内存的资源量。

CPU 和内存的单位是什么?

CPU

CPU 默认单位：核。您也可以使用带小数的 CPU 资源量。当您定义一个容器 CPU Request 为0.5时，所请求的 CPU 是请求1.0核 CPU 时的一半。同时，0.5等价于500m，可以看作 “500 millicpu”，读作“五百毫核”。

内存

内存默认单位：字节。您也可以使用普通的整数，或者带有 数量单位 后缀来表示内存。例如，以下表达式所代表的是大致相同的值：

```
128974848、129e6、129M、128974848000m、123Mi
```

> **注意：**
> 请注意后缀的大小写。例如，如果请求400m临时存储，实际上所请求的值是0.4字节。同理，如果申请400Mi字节（400Mi）或者400M字节，实际上所请求的值是0.4字节。

如何理解 Request/Limit?

Kubernetes 里的 Request/Limit 是通过 CPU Share 和 CPU Quota 技术实现的。

CPU Share

假设一个机器上运行多个容器，多个容器是如何分配资源的？您需要了解 CPU Shares 的概念。
CPU Shares 是属于一项 Linux Control Groups（cgroup）的特性，CPU Shares 控制容器中的进程可以使用的 CPU time。
CPU time 是 CPU 用于处理计算机程序或操作系统的指令的时间量，不是实际日常生活中的时间概念。例如进程进入中断、挂起、休眠等行为时，CPU time 不会增加，但进程恢复运行后 CPU time 会从中断前的时间点继续增加。

CPU Shares 特点

1. CPU Shares 是相对的概念，不是绝对的含义。
   容器的 CPU Shares 是用于在不同容器之间调度 CPU time 的相对值。单独看 CPU Shares 的数字本身没有任何意义。例如，将一个容器 A 的 CPU Shares 设置为512，并不会提供有关容器将获得多少 CPU time 的信息。如果此时将另一个容器 B 的 CPU Shares 设置为1024，意味着 B 容器将获得两倍于 A 容器的 CPU time。也就是说，它仍然没有提供有关每个容器将获得的实际 CPU time 的信息，只有它们之间的相对量。如果 A、B 同时运行在一个3核的设备上，理论上每个时刻将分别获得1核和2核的 CPU。如果运行在6核的设备上，则分别获得2核和4核。如果运行在一个0.3核的设备上，则变成了0.1核和0.2核。
2. CPU Shares 只有在资源竞争时才会发生。
   - 只有 A、B 容器期望在相同时间运行时，才会使用您设置的 CPU Shares 值去分配 CPU 核数。
   - 如果只有一个容器在运行，该容器可以用所有的 CPU。

- 如果同时有多个容器在运行，则通过设置的 CPU Shares 值去分配节点上可分配的 CPU 核数。
- 非运行中的容器即使配置 CPU Shares，也不会影响到运行中的容器关于 CPU time 的划分。

3. CPU Shares 设计的目的在于最大化 CPU 资源的利用率。
   - 任何容器，不管 CPU Shares 值是多少，都有可能用到一台节点上的所有 CPU 资源。
   - 在 CPU 发生竞争时，可以通过 CPU Shares 值来判断每个容器到底该获得多少 CPU time。

CPU Quota

CPU Quota 是用来限制容器的资源使用上限，即使一个节点资源还有剩余量，但容器依旧无法使用超过 CPU Quota 的数值。

Kubernetes 中的 CPU Share

Kubernetes 的 Request/Limit 是由 CPU Share 和 CPU Quota 来实现的。但 Request 还有更多的含义：

1. Request 是一个绝对值，不是相对值，保证一个容器的最小可用的资源量。
2. Request 用来给调度器做判断，寻找集群可剩余调度资源量大于当前 Pod Request 的节点中，最优的节点用来调度当前 Pod。
3. 在发生资源竞争时，会利用 CPU Share 的相对值概念，分配 CPU。

没有 Request 的 Pod

没有 Request 的 Pod 可以调度到任意节点，因为任意节点的剩余可调度资源都满足该 Pod 要求。但是在发生竞争时，它将不会获得任何的资源，此时 Pod 可能会无限期地缺乏资源。

实际应用

在终端里创建一个可以交互的 busybox Pod：

```
kubectl run -i --tty --rm busybox \
  --image=busybox \
  --restart=Never \
  --requests='cpu=50m,memory=50Mi' -- sh
```

在交互栏里使用如下命令让该 Pod 可以用满当前节点上空闲的 CPU 和内存：

```
while true; do true; done
dd if=/dev/zero of=/dev/shm/fill bs=1k count=1024k
```

在另一终端里查看当前 Pod 的资源用量：

```
kubectl top pods
NAME       CPU(cores)    MEMORY(bytes)
busybox    460m          65Mi
```

可以看出，当前 busybox Pod 的 CPU 和内存的资源用量都大于 Request。但理论上因为 busybox 运行的是无限循环的程序，应该会把所有的 CPU 都消耗完才对，为什么只消耗了460m？因为此时集群中还有其他的 Pod 和进程，它们都会通过 CPU Share 彼此竞争集群中的 CPU 资源。

总结

- Kubernetes 用 Request 保证容器对资源的最小用量。
- Kubernetes 用 Limit 限制容器对资源的最大用量。
- 输入数值的时候，要注意默认单位：CPU 默认单位：核；内存默认单位：字节。
- 当发生资源竞争时，Kubernetes 根据不同容器对 Request 的申请量的比例分配资源。

Nginx Ingress FAQ

最近更新时间：2024-07-10 10:35:01

手动替换镜像升级版本后导致原有证书失效

背景

在手动替换 ingress-nginx-controller 镜像进行版本升级后，可能会导致原有的证书失效，从而引起 controller 自带的 webhook 工作异常。

处理流程

1. 删除保存在集群中的证书 secret：
   - 对于所有升级之前版本在 v0.49.3 以下，或者未配置监听命名空间的 ingress-nginx-controller，secret 位于 kube-system 下：

     ```
     kubectl -n kube-system delete secret <nginx-ingress名称>-ingress-nginx-admission
     ```

   - 对于其他配置了 watch-namespace 的 ingress-nginx，secret 位于 watch-namespace 下：

     ```
     kubectl -n <watch-namespace> delete secret <nginx-ingress名称>-ingress-nginx-admission
     ```

2. 删除 nginxingress 对应的 webhook：

   ```
   kubectl delete validatingwebhookconfigurations <nginx-ingress名称>-ingress-nginx-admission
   ```

3. 为 NginxIngress CR 添加任意 label 重新触发证书生成：

   ```
   kubectl label nginxingresses <nginx-ingress名称> resync=true
   ```