Tencent Kubernetes Engine

TKE標準クラスターガイド

製品ドキュメント

カタログ：

TKE標準クラスターガイド
テンセントクバネティスエンジン

最終更新日：：2023-05-06 19:41:07

製品紹介

テンセントクバネティスエンジン（Tencent Kubernetes Engine、TKE）は、高い拡張性と性能を有するコンテナ管理サービスです。ホストされているCVMインスタンスクラスターで、アプリケーションを手軽に実行することができます。このサービスを使用すると、クラスター管理インフラストラクチャをインストール・運用・保守・拡張する必要がありません。簡単なAPI呼び出しだけで、Dockerアプリケーションの開始と停止、クラスターの完全な状態の照会を行い、さまざまなクラウドサービスを使用することができます。お客様はリソース要件および可用性要件に基づいてクラスター内のコンテナの配置を調整し、業務やアプリケーションの特定の要件を満たすことができます。
Tencent Kubernetes Engine（TKE） はネイティブKubernetesに基づいて コンテナを中心とするソリューションを提供し、ユーザーの開発、テストおよび運用保守プロセスの環境問題を解決し、ユーザーがコストを削減し、効率を向上させることを可能にします。TKEはネイティブKubernetes APIに完全に互換性があり、Tencent CloudのCBS、CLBなどのKubernetesプラグインを拡張し、同時にTencent Cloud Virtual Private Cloudを基礎として、信頼性が高く、高性能なネットワークソリューションを実現します。

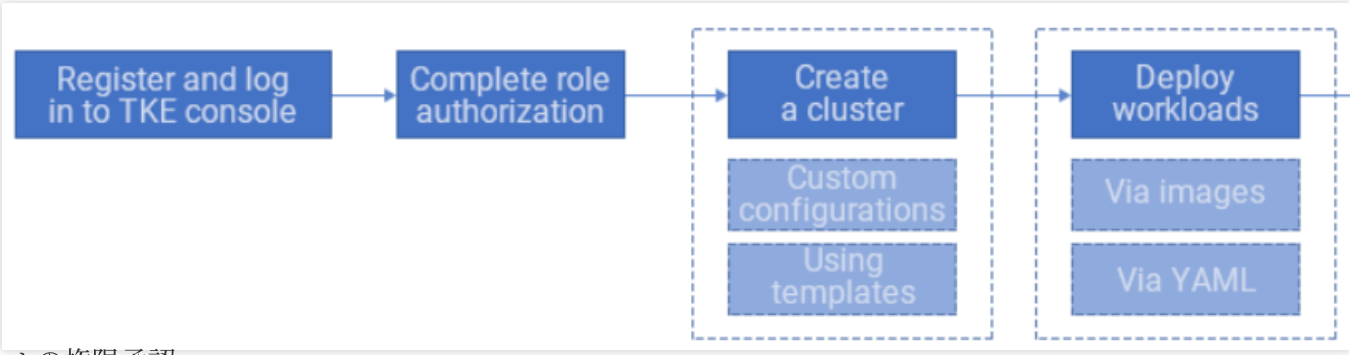
用語の説明

TKEの使用は、以下の基本概念に関連しています。
クラスター：コンテナの実行に必要なクラウドリソースの集合を指します。複数のCVM、CLBなどのクラウドリソースを含みます。
インスタンス：インスタンス（Pod）は、1つまたは複数の関連付けられたコンテナで構成され、これらのコンテナは同一のストレージとネットワークスペースを共有します。
ワークロード：Kubernetesのリソースオブジェクトで、インスタンス（Pod）レプリカの作成の管理、スケジューリングおよびライフサイクル全体の自動制御に使用されます。
Service：複数の同じ設定のインスタンス（Pod）およびこれらのインスタンス（Pod）にアクセスするルールで構成されたマイクロサービスです。
Ingress：Ingressは外部のHTTP（S）トラフィックをサービス（Service）にルーティングするために使用されるルール集合です。
アプリケーション：TKEに統合されたHelm 3.0関連機能です。helm chart、コンテナイメージ、ソフトウェアサービスなど各種製品およびサービスを提供する能力です。

イメージウェアハウス：Dockerイメージを保存するために使用され、DockerイメージはTKEにデプロイするために使用されます。

使用手順

TKEの使用手順は下図に示すとおりです。



1. ロールの権限承認
TKEコンソールに登録してログインし、サービス権限承認を完了して関連リソースの操作権限を取得すると、TKE製品の使用を開始することができます。
2. クラスターの作成
新規クラスターをカスタマイズすることも、テンプレートを使用して新規クラスターを作成することもできます。
3. ワークロードのデプロイ
イメージのデプロイ、YAMLファイルのオーケストレーションの2種類の方法を使用してワークロードをデプロイすることをサポートしています。詳細については、ワークロード管理をご参照ください。
4. ワークロードの作成が完了したら、監視、アップグレード、スケーリングなどの操作によってPodのライフサイクルを管理します。

製品料金

TKEは仕様が異なるホスティングクラスターに対し、対応するクラスター管理料金、およびユーザーが実際に使用するクラウドリソース料金を課金します。課金モードおよび具体的な価格については、TKE課金概要をご参照ください。

関連サービス

複数のCVMを購入することによってTKEクラスターを構成し、コンテナはCVM内で実行します。その他の情報については、CVM製品ドキュメントをご参照ください。

クラスターはプライベートネットワーク下で確立することができ、クラスター内のホストは異なるアベイラビリティーゾーンのサブネット下に割り当てることができます。その他の情報については、プライベートネットワーク製品ドキュメントをご参照ください。
CLBを使用して、複数のクラウドサービスインスタンスにまたがるクライアントのリクエストトラフィックを自動的に割り当て、ホスト内のコンテナに転送に転送することができます。その他の情報については、CLB製品ドキュメントをご参照ください。
TKEクラスターおよびコンテナインスタンスの実行統計データの監視には、Tencent Cloud監視可能プラットフォームを使用することができます。その他の情報については、Tencent Cloud監視可能プラットフォーム製品ドキュメントをご参照ください。

購入ガイド
TKE課金概要

最終更新日：：2024-02-27 11:08:36

課金項目

コンテナーサービスTKEを使用する時、製品の使用料金はクラスター管理料金とクラウド製品リソース料金から構成されます。

クラスター管理料金

クラスター管理料金**ホスティングクラスターのみに対して請求**。TKEによるホスティングクラスターは、高可用性・高性能・高拡張性・高安定性のフルホスティングパネルを提供し、 クラスターの構築や拡張などの操作を簡素化します。これにより、クラスターの管理やメンテナンスに手をかけず、コンテナー化したアプリケーションの開発に集中できるようになります。そのため、異なる構成のホスティングクラスターに対して、TKEはそれなりのクラスター管理料金を請求します。料金の詳細については、クラスター管理料金をご参照ください。

クラウド製品リソース料金

TKEの使用中に作成したほかのクラウド製品リソース（CVM、CBS、CLBなど）に対して、対応するクラウド製品の課金方式に従って料金を請求します。料金の詳細については、クラウド製品リソース料金をご参照ください。

クラスター管理料金

課金モデル

TKEの課金方式は主に従量制課金（後払い）を採用します。

課金項目	課金モデル	支払方式	課金単位
クラスター （個）	従量制課金	購入時 料金凍結、時間単位で請求	ドル/時間

製品価格

説明：

ノードはKubernetes Nodeを指し、CVMノード、BMノード、サードパーティノード、仮想ノードなどが含まれます。

アイドル状態のクラスターに対して、クラスター管理料金を請求しません。

最大管理ノード数	価格 （ドル/時間）

5	0.02040816
20	0.06279435
50	0.11459969
100	0.19152276
200	0.40031397
500	0.8021978
1000	1.47252747
3000	2.44897959
5000	4.40188383

クラウド製品リソース料金

TKEの使用中に作成したほかのクラウド製品リソース（CVM、CBS、CLBなど）に対して、対応するクラウド製品の課金方式に従って料金を請求します。料金の詳細については、各製品の課金説明をご参照ください。

クラウド製品	課金説明
クラウドサーバー CVM	CVM課金モデル
クラウドディスク CBS	CBS価格一覧
ロードバランサー CLB	CLB課金説明

注意：

TKEはKubernetesベースの宣言式サービスです。TKEが作成したロードバランサーCLBやクラウドディスクCBSなどのIaaSサービスリソースを必要としなくなる場合、具体的なサービスリソース画面から削除するのではなく、TKEコンソールから該当するサービスリソースを削除してください。そうしない場合、TKEは削除されたサービスリソースを再作成し、関連の料金が請求されます。例えば、TKEにロードバランサーCLBサービスリソースがすでに作成されているとします。ロードバランサーコンソールからこのCLBインスタンスを削除すると、TKEは宣言式APIにより新しいCLBインスタンスを作成します。

購入説明

最終更新日：：2022-03-31 15:04:01

ご購入時の注意事項

説明：
ご購入時の注意事項に従わないためにサービスが利用できない場合、対応するサービス利用不可時間は、サービス利用不可時間として計上されません。詳しくは、TKEサービスレベル利用規約。

クラスターをご購入時に、クラスターパネルコンポネントの高負荷によるクラスター利用不可を回避するために、以下の推奨スペックを参考にして、業務に応じて適切なクラスター構成を選択してください。
例えば、1つのクラスターに50ノードをデプロイするが、Podを2000もデプロイする場合、最大管理ノードが50ではなく、100のクラスター構成を選択すべきです。

注意：PodにはNamespace配下のすべての状態のPodが含まれ、システムコンポーネント関連のPod（cni-agentなど）が含まれません。

最大管理ノード数	最大Pod数（推奨）	最大ConfigMap数（推奨）	最大CRD 数（推奨）
5	150	32	150
20	600	64	600
50	1500	128	1250
100	3000	256	2500
200	6000	512	5000
500	15000	1024	10000
1000	30000	2048	20000
3000	90000	4096	50000
5000	150000	6144	100000

リージョンとアベイラビリティーゾーン

最終更新日：：2023-05-06 19:41:07

リージョン

概要

リージョン（Region）とは物理的なデータセンターがある地理的区域のことです。Tencent Cloudは異なるリージョン間を完全に隔離し、リージョン間での最大限の安定性とフォールトトレランスを保証します。アクセスの遅延を減らし、ダウンロードスピードを向上させるために、ご自身のお客様に最も近いリージョンを選択することをお勧めします。
下表を確認するか、またはAPIインターフェースによってリージョンリストの照会を行うことで、完全なリージョンリストを確認できます。

関連の特性

異なるリージョン間のネットワークは完全に隔離され、異なるリージョン間のクラウドサービスは、**デフォルトではプライベートネットワークを介して通信できません。**。
異なるリージョン間のクラウドサービスは、パブリックIP を介してInternetにアクセスする方式で通信を行うことができます。また異なるプライベートネットワークに属するクラウドサービスは、 CCNを介して通信を行うことができ、この通信方式の方がより高速で、安定しています。
CLBは、現在、デフォルトで同一リージョンのトラフィック転送をサポートし、ローカルドメインのCVMをバインドします。 クロスリージョンバインディング機能を有効にすることによって、CLBをサポートし、リージョン間でCVMをバインドすることができます。

アベイラビリティーゾーン

概要

アベイラビリティーゾーン（Zone）とは、Tencent Cloudの同一リージョン内で電源とネットワークが互いに独立している物理的なデータセンターのことです。お客様のビジネスのオンラインサービスの持続性を確保するため、アベイラビリティーゾーン間の障害を互いに隔離し（大規模な災害または大規模な電力設備の障害を除く）、障害の影響が拡散しないようにすることを目的としています。独立したアベイラビリティーゾーンでインスタンスを起動することにより、単一障害点の影響からアプリケーションを保護することができます。
APIインターフェースによってアベイラビリティーゾーンリストの照会を行うことで、完全なアベイラビリティーゾーンリストを確認できます。

関連の特性

同一リージョンで異なるアベイラビリティーゾーンに位置していても、同一のプライベートネットワーク下のクラウド製品はプライベートネットワークを介して相互接続されており、プライベートIPを使用して直接アクセスすることができます。

説明

プライベートネットワークの相互接続は同じアカウント下のリソースの相互接続を指し、異なるアカウントのリソースのプライベートネットワークは完全に隔離されます。

中国

リージョン	アベイラビリティーゾーン
華南地域（広州） ap-guangzhou	広州1区（販売終了） ap-guangzhou-1
	広州2区（販売終了） ap-guangzhou-2
	広州3区 ap-guangzhou-3
	広州4区 ap-guangzhou-4
	広州6区 ap-guangzhou-6
	広州7区 ap-guangzhou-7
華東地域（上海） ap-shanghai	上海1区（販売終了） ap-shanghai-1
	上海2区 ap-shanghai-2
	上海3区 ap-shanghai-3
	上海4区 ap-shanghai-4
	上海5区 ap-shanghai-5

	上海8区 ap-shanghai-8
華東地域（南京） ap-nanjing	南京1区 ap-nanjing-1
	南京2区 ap-nanjing-2
	南京3区 ap-nanjing-3
華北地域（北京） ap-beijing	北京1区（販売終了） ap-beijing-1
	北京2区 ap-beijing-2
	北京3区 ap-beijing-3
	北京4区 ap-beijing-4
	北京5区 ap-beijing-5
	北京6区 ap-beijing-6
	北京7区 ap-beijing-7
西南地域（成都） ap-chengdu	成都1区 ap-chengdu-1
	成都2区 ap-chengdu-2
西南地域（重慶） ap-chongqing	重慶1区 ap-chongqing-1
中国香港・マカオ・台湾地区（中国香港） ap-hongkong	中国香港1区（中国香港ノードは中国香港・マカオ・台湾地区をカバーするために使用されます）（販売終了） ap-hongkong-1
	中国香港2区（中国香港ノードは中国香港・マカオ・台湾地区をカバーするために使用されます）

	ap-hongkong-2
	中国香港3区（中国香港ノードは中国香港・マカオ・台湾地区をカバーするために使用されます） ap-hongkong-3

説明：
済南、杭州、福州、武漢、長沙、石家庄リージョンは現在ベータ版テスト中です。ご利用をご希望の場合はビジネスマネージャーまでご連絡ください。

その他の国と地域

リージョン	アベイラビリティーゾーン
アジア太平洋東南（シンガポール） ap-singapore	シンガポール1区（シンガポールノードはアジア太平洋東南地区をカバーするために使用されます） ap-singapore-1
	シンガポール2区（シンガポールノードはアジア太平洋東南地区をカバーするために使用されます） ap-singapore-2
	シンガポール3区（シンガポールノードはアジア太平洋東南地区をカバーするために使用されます） ap-singapore-3
	シンガポール4区（シンガポールノードはアジア太平洋東南地区をカバーするために使用されます） ap-singapore-4
アジア太平洋東南（ジャカルタ） ap-jakarta	ジャカルタ1区（ジャカルタノードはアジア太平洋東南地区をカバーするために使用されます） ap-jakarta-1
	ジャカルタ2区（ジャカルタノードはアジア太平洋東南地区をカバーするために使用されます） ap-jakarta-2
アジア太平洋東北（ソウル） ap-seoul	ソウル1区（ソウルノードはアジア太平洋東北地区をカバーするために使用されます） ap-seoul-1
	ソウル2区（ソウルノードはアジア太平洋東北地区をカバーするために使用されます）

	ap-seoul-2
アジア太平洋東北（東京） ap-tokyo	東京1区（東京ノードのアベイラビリティーゾーンはアジア太平洋東北地区をカバーします） ap-tokyo-1
	東京2区（東京ノードのアベイラビリティーゾーンはアジア太平洋東北地区をカバーします） ap-tokyo-2
アジア太平洋南部（ムンバイ） ap-mumbai	ムンバイ1区（ムンバイノードはアジア太平洋南部地区をカバーするために使用されます） ap-mumbai-1
	ムンバイ2区（ムンバイノードはアジア太平洋南部地区をカバーするために使用されます） ap-mumbai-2
アジア太平洋東南（バンコク） ap-bangkok	バンコク1区（バンコクノードはアジア太平洋東南地区をカバーするために使用されます） ap-bangkok-1
	バンコク2区（バンコクノードはアジア太平洋東南地区をカバーするために使用されます） ap-bangkok-2
北アメリカ地域（トロント） na-toronto	トロント1区（トロントノードは北米地区をカバーするために使用されます） na-toronto-1
南アメリカ地区（サンパウロ） sa-saopaulo	サンパウロ1区（サンパウロノードは南米地区をカバーするために使用されます） sa-saopaulo-1
米国西部（シリコンバレー） na-siliconvalley	シリコンバレー1区（シリコンバレーノードは米国西部をカバーするために使用されます） na-siliconvalley-1
	シリコンバレー2区（シリコンバレーノードは米国西部をカバーするために使用されます） na-siliconvalley-2
米国東部（バージニア） na-ashburn	バージニア1区（バージニアノードは米国東部地区をカバーするために使用されます） na-ashburn-1
	バージニア2区（バージニアノードは米国東部地区をカバーするために使用されます）

	na-ashburn-2
欧州地区（フランクフルト） eu-frankfurt	フランクフルト1区（フランクフルトノードは欧州地区をカバーするために使用されます） eu-frankfurt-1
	フランクフルト2区（フランクフルトノードは欧州地区をカバーするために使用されます） eu-frankfurt-2

リージョンとアベイラビリティーゾーンを選択する方法

リージョンとアベイラビリティーゾーンの選択に関しては、次のいくつかの要素を考慮に入れる必要があります。
CVMの所在リージョン、ご自身およびご自身のターゲットユーザーが存在する地理的位置です。
CVMを購入する際には、ご自身のお客様に最も近いリージョンを選択することで、アクセスの遅延を減らし、アクセススピードを向上させることをお勧めします。
CVMと他のクラウド製品との関係です。
他のクラウド製品を選択する際には、可能な限りすべて同一リージョン、同一アベイラビリティーゾーンとすることで、各クラウド製品間でプライベートネットワークによる通信を行えるようにし、アクセスの遅延を減らし、アクセススピードを向上させることをお勧めします。
業務の高可用性および障害復旧の観点。
VPCが1つしかない場合であっても、業務を少なくとも異なるアベイラビリティーゾーンにデプロイすることで、アベイラビリティーゾーン間の障害隔離を保証し、異なるアベイラビリティーゾーン間の障害復旧を実現できるようにすることをお勧めします。
異なるアベイラビリティーゾーン間ではネットワークの通信遅延が発生する可能性がありますので、業務上の実際のニーズを考慮して評価を行い、高可用性と低遅延との間で最適なバランスをとる必要があります。
他の国および地域のホストにアクセスする必要がある場合は、他の国および地域のCVMを選択してアクセスすることをお勧めします。中国でCVMを作成し、他の国および地域のホストにアクセスした場合、アクセス遅延が比較的大きくなりますので、推奨しません。

リソースの位置の説明

ここではTencent Cloudのどのリソースがグローバルなものか、どのリソースがリージョンは区別するがアベイラビリティーゾーンは区別しないのか、どのリソースがアベイラビリティーゾーンベースであるのかについてご説明します。

リソース	リソースID形式	タイプ	説明

	<リソースの略称>-8桁の数字および文字		
ユーザーアカウント	制限なし	グローバルで一意	ユーザーは同一のアカウントを使用してTencent Cloudの世界各地のリソースにアクセスできます。
SSHキー	skey-xxxxxxxx	全リージョンで利用可能	ユーザーはSSHキーを使用して、アカウント下の任意のリージョンのCVMをバインドできます。
CVMインスタンス	ins-xxxxxxxx	単一リージョンの単一アベイラビリティーゾーンでのみ利用可能	ユーザーは特定のアベイラビリティーゾーンでのみCVMインスタンスを作成できます。
カスタムイメージ	img-xxxxxxxx	単一リージョンの複数のアベイラビリティーゾーンで利用可能	ユーザーはインスタンスのカスタムイメージを作成し、それを同一のリージョンの異なるアベイラビリティーゾーンで使用できます。他のリージョンで使用する際は、イメージコピー機能を使用してカスタムイメージを他のリージョンにコピーする必要があります。
Elastic IP	eip-xxxxxxxx	単一リージョンの複数のアベイラビリティーゾーンで利用可能	Elastic IPアドレスはあるリージョンで作成し、なおかつ同一のリージョンのインスタンスとのみバインドできます。
セキュリティグループ	sg-xxxxxxxx	単一リージョンの複数のアベイラビリティーゾーンで利用可能	セキュリティグループはあるリージョンで作成し、なおかつ同一のリージョンのインスタンスとのみバインドできます。Tencent Cloudはユーザー向けに3つのデフォルトセキュリティグループを自動作成します。
CBS	disk-xxxxxxxx	単一リージョンの単一アベイラビリティーゾーンでのみ利用可能	ユーザーは特定のアベイラビリティーゾーンでのみCBSを作成でき、なおかつ同一のアベイラビリティーゾーンのインスタンス上にのみマウントできます。
スナップショット	snap-xxxxxxxx	単一リージョンの複数のアベイラビリ	あるCBSにスナップショットを作成すると、ユーザーはこのリージョン下でこのスナップショットを使用して他の操作（CBSの作成など）を行うことができます。

		ティーゾーンで利用可能	
CLB	clb-xxxxxxxx	単一リージョンの複数のアベイラビリティーゾーンで利用可能	CLBは単一リージョンの異なるアベイラビリティーゾーンのCVMにバインドしてトラフィックの転送を行うことができます。
VPC	vpc-xxxxxxxx	単一リージョンの複数のアベイラビリティーゾーンで利用可能	VPCはあるリージョンで作成し、異なるアベイラビリティーゾーンで同じVPCに属するリソースを作成することができます。
サブネット	subnet-xxxxxxxx	単一リージョンの単一アベイラビリティーゾーンでのみ利用可能	ユーザーは異なるアベイラビリティーゾーン間にサブネットを作成することはできません。
ルートテーブル	rtb-xxxxxxxx	単一リージョンの複数のアベイラビリティーゾーンで利用可能	ルートテーブルはVPCの位置属性に従うため、ユーザーが作成する際は特定のVPCを指定する必要があります。

関連操作

インスタンスを他のアベイラビリティーゾーンに移行する

起動済みのインスタンスのアベイラビリティーゾーンの変更を行うことはできませんが、ユーザーは他の方法でインスタンスを他のアベイラビリティーゾーンに移行することができます。移行には、元のインスタンスからカスタムイメージを作成し、カスタムイメージを使用して新しいアベイラビリティーゾーンでインスタンスを起動し、新しいインスタンスの設定を更新するというプロセスが含まれます。

1. 現在のインスタンスのカスタムイメージを作成します。詳細な情報については、カスタムイメージの作成をご参照ください。

2. 現在のインスタンスのネットワーク環境がVPCであり、かつ移行後も現在のプライベートIPアドレスを保持する必要がある場合、ユーザーはまず現在のアベイラビリティーゾーンのサブネットを削除し、その後、新しいアベイラビリティーゾーンに元のサブネットと同一のIPアドレスの範囲でサブネットを作成することができます。削

除できるのは、使用可能なインスタンスを含まないサブネットのみであることに注意する必要があります。このため、現在のサブネット内のすべてのインスタンスを新しいサブネットに移動する必要があります。
3. 作成したカスタムイメージを使用して、新しいアベイラビリティーゾーンに新しいインスタンスを作成します。ユーザーは、元のインスタンスと同一のインスタンスタイプと設定を選択することも、新しいインスタンスタイプと設定を選択することもできます。詳細な情報については、インスタンスの作成をご参照ください。
4. 元のインスタンスをすでにElastic IPアドレスにバインド済みの場合は、古いインスタンスのバインドを解除してから新しいインスタンスとのバインドを行います。詳細な情報については、Elastic IPをご参照ください。
5. （オプション）元のインスタンスが従量課金タイプの場合は、元のインスタンスの破棄を選択することもできます。詳細な情報については、インスタンスの破棄をご参照ください。

イメージを他のリージョンにコピーする

ユーザーのインスタンス起動、インスタンス確認などの動作はすべてリージョン属性を区別するものです。ユーザーが起動したいインスタンスのイメージがそのリージョンに存在しない場合は、イメージをそのリージョンにコピーする必要があります。詳細な情報については、イメージをコピーするをご参照ください。

クラスター割り当て量購入制限

最終更新日：：2023-05-09 09:31:13

各ユーザーに対して、Tencent CloudのTKEクラスターは地域ごとに固定したクォーターを割り当てています。

TKE クォーター制限

ユーザーごとに購入できるTKEクォーターはデフォルトでは以下となります。より多くのクォーター項目が必要である場合、クォーター申請チケット でクォーターを申請することができます。

> 注意：
> 2019年10月21日より、ユーザークラスターがサポートする5000未満の最大ノードクォーターは、すべて5000に調整されました。

クォーター項目	デフォルト値	参照可能なエントリ	クォーター拡張可否
単一地域配下のクラスター	20	TKE 概要画面の右下側	はい
単一クラスター配下のノード	5000		
単一地域配下のイメージネームスペース	10		
単一地域配下のイメージリポジトリ	500		
単一イメージ配下のイメージバージョン	100		

CVMクォーター制限

Tencent CloudのTKEによって生成されたCVMは、CVMの購入制限にかけられます。詳しくは、CVM購入制約をご参照ください。ユーザーごとに購入できるCVMクォーターはデフォルトでは以下のとおりとなります。より多くのクォーター項目が必要である場合、クォーター申請チケット でクォーターを申請することができます。

クォーター項目	デフォルト値	参照可能なエントリ	クォーター拡張可否
単一利用可能リージョン配下の従量制課金CVM	30台または60台	CVM 概要画面-各地域のリソース	はい

クラスター設定制限

説明：
クラスター設定制限は、クラスターの規模を制限しており、現在変更不可です。

設定項目	アドレス範囲	影響範囲	参照可能なエリート	変更可否
VPC ネットワーク-サブネットワーク	カスタム設定	当該サブネットワークに追加できるノード数	クラスターに対応するVPCサブネットワーク一覧-利用可能なIP数	• 変更不可 • 新規サブネットワーク利用可能
コンテナーのネットワークセグメントCIDR	カスタム設定	• クラスター内部の最大ノード数 • クラスター内部の最大service数 • ノードごとの最大Pod数	クラスター基本情報画面-コンテナーのネットワークセグメント	変更不可

リソース制限の説明

2021年1月13日より、Tencent CloudのTKEシステムは、ノード数（nodeNum）が5以下（0 < nodeNum ≤ 5）、また、ノード数が5より大きいが20未満（5 < nodeNum < 20）のクラスターにおけるネームスペースに自動的に一連のリソースクォーターを適用します。これらのクォーターはリムーブ不可であり、クラスターにデプロイされたアプリケーションの潜在バグからクラスターパネルを保護し、その安定性を確保します。

これらのクォーターを確認する場合、以下のコマンドを実行します。

```
kubectl get resourcequota tke-default-quota -o yaml
```

指定されたネームスペースの `tke-default-quota` オブジェクトを確認する場合、 `--namespace` オプションを使用し、ネームスペースを指定してください。

具体的なクォーター制限は以下の通りです：

クラスター規模	クォーター制限
0 < nodeNum ≤ 5	総数の制限 Pod：4000，configMap：3000，CustomResourceDefinition(CRD)：4000
5 < nodeNum < 20	総数の制限 Pod：8000，configMap：6000，CustomResourceDefinition(CRD)：8000
20 ≤ nodeNum	制限なし

特殊な運用シーンでクォーターを調整する必要がある場合、チケットの提出で申請してください。

コンテナノードハードディスクの設定

最終更新日：：2022-03-31 15:04:01

説明

TKEがクラスターを作成または拡張する時、業務需要に合わせて、コンテナーノードのシステムディスクのタイプとサイズ、データディスクのタイプとサイズを設定し、異なるタイプのディスクを選択することができます。

アドバイス

1.コンテナーのディレクトリがシステムディスクに格納されるため、50Gのシステムディスクを作成することを推奨します。
2.システムディスクに何か要求がある場合、クラスターを初期化する時、dockerのディレクトリをデータディスクに移行することを推奨します。

TKEノードパブリックネットワークIPの説明

最終更新日：：2023-05-23 11:40:10

パブリックネットワークから直接業務セキュリティにアクセスできないが、パブリックネットワークへのアクセスができるようにするという要望がある場合、Tencent CloudのNATゲートウェイをご利用ください。以下でNATゲートウェイを使用してパブリックネットワークにアクセスする方法をご紹介します。

パブリックネットワークIP

デフォルトでは、クラスターを作成する時、クラスターのノードにパブリックネットワークIP
を割り当てます。割り当てられたパブリックネットワークIPは以下の役割を果たします：

- パブリックネットワークIPでクラスターのノードマシンにログインします。
- パブリックネットワークIPでパブリックネットワークサービスにアクセスします。

パブリックネットワーク帯域幅

パブリックネットワークサービスを作成する時、パブリックネットワークロードバランサーが使用するのはノードの帯域幅とトラフィックです。パブリックネットワークサービスを提供する場合、ノードにパブリックネットワーク帯域幅があることを確保する必要があります。業務がパブリックネットワークを必要としなければ、パブリックネットワーク帯域幅を購入しなくても構いません。

NATゲートウェイ

CVMはElastic IPアドレスをバインドしません。Internetにアクセスするすべてのトラフィックは NATゲートウェイ経由で転送されます。このソリューションでは、CVMがInternetにアクセスするトラフィックは、内部ネットワーク経由でNATゲートウェイに転送されるため、CVM購入時のパブリックネットワークの帯域幅に制限されません。また、NATゲートウェイが生じたネットワークトラフィックはCVMのパブリックネットワークのアウトバンドを占有しません。NATゲートウェイ経由でInternetにアクセスするには、以下を実施してください：

ステップ1：NATゲートウェイを作成します

1. VPCコンソールにログインし、左側のナビゲーションバーで【 NAT Gateway】をクリックします。
2. 「NATゲートウェイ」管理画面で、**Create**をクリックします。
3. 表示された「Create an NAT Gateway 」ウィンドウに、以下のパラメータを記入します。

- ゲートウェイ名：カスタム。
- 所属ネットワーク：NATゲートウェイサービスのプライベートネットワークを選択します。
- ゲートウェイタイプ：必要に応じて選択してください。ゲートウェイタイプは作成後にも変更可能です。
- アウトバンド帯域幅上限：必要に応じて選択します。
- Elastic IP：NATゲートウェイにElastic IPを割り当てます。既存Elastic IPを選択するか新規購入してElastic IPを割り当ててください。

4. **Create**をクリックして、NATゲートウェイの作成を完了します。

> 注意：
> NATゲートウェイ作成時にリース料金は1時間凍結されます。

ステップ2：関連サブネットワークが関連付けられたルーティングテーブルを設定します

> 説明：
> NATゲートウェイを作成した後、サブネットのトラフィックをNATゲートウェイに転送するように、VPCコンソールのルーティングテーブル画面でルーティングルールを設定する必要があります。

1. 左側のナビゲーションバーで【Route Table】をクリックします。
2. ルーティングテーブル一覧で、Internetにアクセスするサブネットワークが関連付けられたルーティングテーブルのID/名前をクリックして、ルーティングテーブルの詳細画面に入ります。
3. 「Routing Policy」欄で、「+ New routing policies」をクリックします。
4. 表示された「Add routing」ウィンドウに、**Destination**を記入し、**Next Hop Type**に**NAT gateway**、** Next Hop**に作成したNATゲートウェイのIDを指定します。
5. **OK**をクリックします。
   上記のように設定した後、このルーティングテーブルに関連付けられたサブネットワークにおけるCVMがIntenetにアクセスするトラフィックはNATゲートウェイに転送されます。

その他のソリューション

案1：Elastic IPアドレスを使用します

CVMはElastic IPアドレスだけをバインドし、NATゲートウェイを使用しません。この案では、CVMがInternetにアクセスするすべてのトラフィックは、Elastic IPアドレス経由で転送され、CVM購入時のパブリックネットワークの帯域幅に制限されます。パブリックネットワークへのアクセスによって生じた料金は、CVMネットワーク課金

モデルによって計算されます。
使用方法：Elastic IPアドレス使用手引書をご参照ください。

案2：NATゲートウェイとastic IPアドレス両方を使用します

NATゲートウェイとastic IPアドレス両方を使用する案では、CVMからInternetへのアクセスによって生じたすべてのトラフィックは、内部ネットワーク経由でNATゲートウェイに転送されます。返されたパケットもNATゲートウェイ経由でCVMに転送されます。この部分のトラフィックは、CVM購入時のパブリックネットワークの帯域幅に制限されません。NATゲートウェイが生じたネットワークトラフィックはCVMのパブリックネットワークのアウトバンドを占有しません。InternetからのトラフィックがCVMのElastic IPアドレスにアクセスする場合、CVMが返したパケットは全部Elastic IPアドレス経由で転送され、生じたパブリックネットワークのアウトバンドトラフィックは、CVM購入時のパブリックネットワークの帯域幅に制限されます。パブリックネットワークへのアクセスによって生じた料金は、CVMネットワーク課金モデルによって計算されます。

> 注意：
> ご利用中のアカウントで帯域幅パックによる帯域共有機能をアクティブ化した場合、NATゲートウェイが生じたアウトバンドトラフィックは帯域幅パック全体に基づき計算されます(ネットワークトラフィック料金を重複して請求しません）。NATゲートウェイの高いアウトバンドによって多額な帯域幅パック料金が生じることを防ぐために、NATゲートウェイのアウトバンドに制限をかけることを推奨します。

TKEセキュリティグループの設定

最終更新日：：2023-10-24 14:31:46

セキュリティはみんなが関心を持っている問題です。Tencent Cloudは、セキュリティを製品設計上の最も重要な要因とし、製品にセキュリティ隔離を厳しく要求します。TKEもこれを重要視しています。Tencent Cloudの基幹ネットワークは十分なセキュリティメカニズムを提供しています。TKEの基幹ネットワークは、より豊富なネットワーク機能を提供するTencent Cloudバーチャルプライベートクラウド VPC を採用します。本書では、TKEでセキュリティグループを使用するベストプラクティスを紹介することで、セキュリティポリシーの選択に協力します。

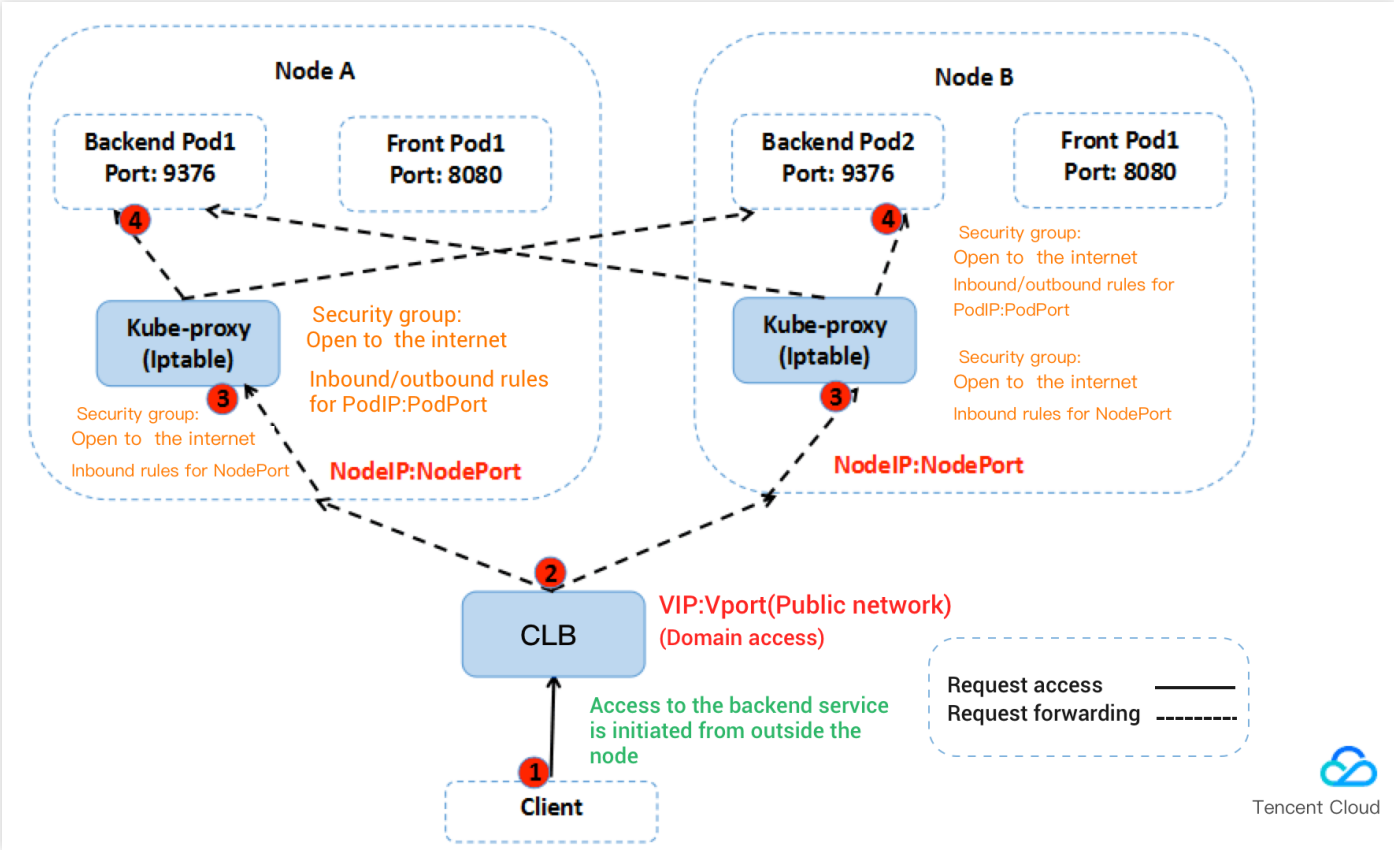
セキュリティグループ

セキュリティグループは状態付きのパケットフィルタリング型仮想ファイアウォールであり、1台または複数台のCVMのネットワークアクセス制御を設定するために使用され、Tencent Cloudによって提供される重要なネットワークセキュリティ隔離の手段です。セキュリティグループの詳細は、セキュリティグループをご参照ください。

TKEを使用しセキュリティグループを選択する時検討すべきのポイント

- コンテナークラスターにおいて、サービスインスタンスが分散式にデプロイされるため、異なるサービスインスタンスはクラスターのノードに分散されます。同一クラスター配下のホストを同じセキュリティにバインドし、クラスターのセキュリティグループにほかのCVMを追加しないことを推奨します。
- 外部に対して、セキュリティは必要最小限の権限を付与します。
- 以下のTKE使用規則を適用する必要があります。
- コンテナーインスタンスネットワークとクラスターノードネットワークへのアクセスを許可します
  サービスがホストノードにアクセスする時、Kube-proxyモジュールが設定したiptables規則に従って、リクエストをサービスの任意1つのインスタンスに転送します。サービスのインスタンスがほかのノードに存在する可能性があるため、ノードをまたがったアクセスが行われます。例えば、アクセスするデスティネーションIPに、サービスインスタンスIP、クラスター上のほかのノードIP、ノード上のクラスターのcbr0ブリッジIPがある場合、- 対向側のノードでコンテナーインスタンスネットワークとクラスターノードネットワークへのアクセスを許可する必要があります。
- 同一VPCにおける異なるクラスターがお互いアクセスする場合、対応するクラスターのコンテナーネットワークとノードネットワークへのアクセスを許可する必要があります。
- SSHによるノードへのログインが必要な場合、22ポートを開放する必要があります。

- ノードの30000 - 32768ポートを開放します。
  アクセル経路で、ロードバランサーを介してパケットをコンテナークラスターのNodeIP：NodePortに転送してください。そのうち、NodeIPはクラスター上の任意のノードのホストIPであり、NodePortはサービス作成時にコンテナークラスターがデフォルトでサービスに割り当てたポートです。NodePortの範囲は30000 - 32768です。
  下図ではパブリックネットワークからサービスにアクセスすることを例とします：



TKEのデフォルトセキュリティグループ規則

ノードのデフォルトセキュリティグループ規則

クラスターノード間の通信には一部のポートを開放する必要があります。無効なセキュリティグループをバインドすることによってクラスターの作成に失敗することを防ぐために、TKEはデフォルトのセキュリティグループ設定規則を用意しています。具体的には、下表をご参照ください：

> 注意：
> 現在のデフォルトセキュリティグループが業務需要を満足できず、このセキュリティグループにバインドされるクラスターをすでに作成している場合、セキュリティグループ規則管理を参照し、このセキュリティ

グループに対して、確認や変更などを行ってください。

インバンド規則

プロトコル規則	ポート	ソース	ポリシー	備考
ALL	ALL	コンテナーネットワーク CIDR	許可	コンテナーネットワークにおけるPod間通信の許可
ALL	ALL	クラスターネットワーク CIDR	許可	クラスターネットワークにおけるノード間通信の許可
tcp	30000 - 32768	0.0.0.0/0	許可	MasterとWorkerのノード間通信の許可
udp	30000 - 32768	0.0.0.0/0	許可	MasterとWorkerのノード間通信の許可
icmp	-	0.0.0.0/0	許可	ICMPプロトコルの許可、Ping対応

アウトバウンド規則

プロトコル規則	ポート	ソース	ポリシー
ALL	ALL	0.0.0.0/0	許可

説明：

- アウトバンド規則をカスタマイズする場合、ノードネットワークセグメントとコンテナーネットワークセグメントへのアクセスを許可する必要があります。
- コンテナーノードでこの規則を設定すれば、異なる方法でクラスターにおけるサービスにアクセスできます。
- クラスターにおけるサービスへのアクセス方法については、Service管理サービスへのアクセス方法をご参照ください。

独立クラスターMasterのデフォルトセキュリティグループ規則

独立クラスターを作成する時、デフォルトではMasterにTKEデフォルトセキュリティグループをバインドします。これは、クラスター作成後に、MasterとNodeが通信できないリスクやServiceにアクセスできないリスクを減らすためです。デフォルトのセキュリティグループ設定規則については、下表をご参照ください：

説明：
作成したセキュリティグループの権限はTKEサービスロールが継承します。詳しくは、サービスがロールに権限を付与する説明をご参照ください。

インバンド規則

プロトコル	ポート	ネットワークセグメント	ポリシー	備考
ICMP	ALL	0.0.0.0/0	許可	Ping対応
TCP	30000 - 32768	クラスターネットワークCIDR	許可	MasterとWorkerのノード間通信の許可
UDP	30000 - 32768	クラスターネットワークCIDR	許可	MasterとWorkerのノード間通信の許可
TCP	60001,60002,10250,2380,2379,53,17443,50055,443,61678	クラスターネットワークCIDR	許可	API Server通信の許可
TCP	60001,60002,10250,2380,2379,53,17443	コンテナーネットワークCIDR	許可	API Server通信の許可
TCP	30000 - 32768	コンテナーネットワークCIDR	許可	Service通信の許可
UDP	30000 - 32768	コンテナーネットワークCIDR	許可	Service通信の許可
UDP	53	コンテナーネットワークCIDR	許可	CoreDN通信の許可

プロトコル	ポート	ネットワークセグメント	ポリシー	備考
UDP	53	クラスターネットワークCIDR	許可	CoreDN通信の許可

アウトバウンド規則

プロトコル規則	ポート	ソース	ポリシー
ALL	ALL	0.0.0.0/0	許可

クラスターの追加リソースが所属するプロジェクトの説明

最終更新日：：2022-03-31 15:04:02

クラスタ追加リソースが所属するプロジェクトの説明

概要

プロジェクトごとに財務清算などを行う場合、以下をお読みください：

1. クラスターにはプロジェクト属性がありません。クラスター内部のCVMやロードバランサーなどのリソースにはプロジェクト属性があります。
2. クラスターの追加リソースが所属するプロジェクト：当該クラスターに追加したリソースだけを当該プロジェクトの配下とします。

アドバイス

1. クラスターにおけるすべてのリソースを同一プロジェクトの配下とすることを推奨します
2. クラスターにおけるCVMを異なるプロジェクトに分散させる場合、CVMコンソールでプロジェクトを移行する必要があります。
3. CVMの所属するプロジェクトが異なる場合、CVMが所属する「セキュリティグループインスタンス」も異なるため、同じクラスターにおけるCVMの「セキュリティグループルール」を統一することを推奨します。

コンテナアプリケーションのクラウドへのデプロイ Check List

最終更新日：：2023-05-06 19:41:07

概要

セキュアで高効率な、安定的で可用性の高い業務のクラウド化は、クラウド関係者にとっての共通のニーズです。このニーズを実現するには、システムの可用性、データの信頼性、運用保守の安定性の3つが完璧に結びつく必要があります。ここでは評価項目、影響の説明、評価の参照という3つの角度からコンテナアプリケーションのクラウドへのデプロイの各チェック項目について述べることで、クラウド化の障害を取り除き、スムーズかつ効率的に業務をTKEに移行できるよう支援します。

チェック項目

システム可用性

カテゴリー	評価項目	タイプ	影響の説明	評価の参照
クラスター	クラスター作成前に、業務シナリオを踏まえて事前にノードネットワークとコンテナネットワークを計画することで、その後の業務拡張が制限されないようにします。	ネットワーク計画	クラスターの存在するサブネットまたはコンテナネットワークセグメントが小さいと、クラスターが実際にサポートする有効ノード数が業務に必要な容量より少なくなる可能性があります。	ネットワーク計画 コンテナおよびノードネットワークの設定
	クラスター作成前に、Direct Connect、Peering Connection、コンテナネットワークセグメントとサブネットネットワークセグメントなどの関連ネットワークセグメントの計画を事前に整理することで、ネットワークセグメントの競合が発生して業務に影響することがないようにします。	ネットワーク計画	単純なネットワーキングシナリオの場合はページの表示に従ってクラスターの関連ネットワークセグメントを設定し、競合を避けます。Peering Connection、Direct Connect、VPNなどの、業務上の複雑なネットワーキングシナリオの場合は、不適切なネットワーク計画によって業務全体の正常な相互アクセスに影響が出ます。	-
	クラスター作成の際、自動的に	デプ	セキュリティグループは重要なセ	TKEセキュリ

	デフォルトのセキュリティグループを新規作成してバインドし、業務ニーズに応じてカスタムセキュリティグループルールを設定することもできます。	ロイ	キュリティ隔離手段であり、セキュリティポリシーの設定が不適切な場合はセキュリティ関連リスクおよびサービスの接続性などの問題が生じる可能性があります。	ティグループの設定
	ContainerdとDockerはTKEが現在サポートしているランタイムコンポーネントであり、適用ケースがそれぞれ異なります。クラスター作成の際は、業務シナリオに応じて適切なコンテナランタイム（Container Runtime）コンポーネントを選択してください。	デプロイ	クラスターの作成後にランタイムコンポーネントおよびバージョンを変更する場合は、クラスター内のノードプールに所属していない増分ノードにのみ有効であり、既存ノードには影響しません。	ContainerdとDockerの選択方法
	デフォルトでは、Kube-proxyはiptablesを使用してServiceとPod間のロードバランシングを実現します。クラスター作成の際は、IPVSを迅速に有効化してトラフィックを受け入れ、ロードバランシングを実現することができます。	デプロイ	現在はクラスター作成時のIPVS有効化をサポートしており、それ以降は全クラスターに対して有効になり、無効化することはできません。	クラスターでのIPVSの有効化
	クラスター作成の際は、業務シナリオに応じて、独立クラスターとホスティングクラスターのうちどちらか適切なクラスターモードを選択します。	デプロイ	ホスティングクラスターのMasterとEtcdはユーザーリソースではなく、Tencent Cloudのテクニカルチームが一元的に管理および保守を行っているため、ユーザーはMasterとEtcdのデプロイ規模およびサービスパラメータを変更することはできません。変更したい場合は独立デプロイモードのクラスターを選択してください。	クラスターの概要 クラスターのホスティングモードの説明
ワークロード情報	ワークロードの作成時にCPUとメモリの制限範囲を設定することで、業務の堅牢性を向上させる必要があります。	デプロイ	同一のノード上に複数のアプリケーションをデプロイする場合、リソースの上限と下限を設定していないアプリケーションに、アプリケーションの異常によるリソース漏洩の問題が発生した場合、その他のアプリケーションにリソースが割り当てられずにエラーが起き、かつその監視情報にも誤差が生じる可能性があります。	ワークロードのリソース制限の設定

	ワークロードの作成時にコンテナヘルスチェック（「コンテナLivenessチェック」および「コンテナReadinessチェック」）を設定できます。	信頼性	コンテナヘルスチェックが設定されていないと、ユーザーの業務に異常が発生した場合にPodが認識できず、自動的に再起動してサービスを再開することができなくなり、最終的にPodのステータスは正常にもかかわらず、Pod内の業務は異常という現象が発生することがあります。	サービスヘルスチェックの設定
	サービスを作成する際は、実際のアクセスの必要性に応じて適切なアクセス方式を選択する必要があります。現在は、パブリックネットワークアクセス、クラスター内のみのアクセス、VPCプライベートネットワークアクセス、ホストポートアクセスの4種類をサポートしています。	デプロイ	適切でないアクセス方式を選択すると、サービス内外のアクセスロジックが混乱し、リソースの浪費が発生する可能性があります。	Serviceの管理
	ワークロードの作成の際は、Pod単体のレプリカ数を設定することは避け、ご自身の業務に合わせてノードスケジューリングポリシーを合理的に設定してください。	信頼性	Pod単体のレプリカ数を設定した場合、ノードの異常またはインスタンスの異常が発生した場合、サービスにも異常が生じます。Podのスケジューリングを確実に成功させるため、スケジューリングルールを設定後に、ノードにコンテナのスケジューリング用のアイドルリソースがあることを確実にしてください。	Pod数の調整 ワークロードのスケジューリングルールの設定

データ信頼性

カテゴリー	評価項目	タイプ	影響の説明	評価の参照
コンテナデータの永続化	アプリケーションPodのデータストレージについては、実際のニーズに応じて適切なデータボリュームタイプを選択します。	信頼性	ノードの異常が復旧できない場合、ローカルディスク内に存在するデータは復元できませんが、この場合クラウドストレージは極めて高いデータ信頼性を提供することができます。	Volumeの管理

運用保守安定性

カテゴリー	評価項目	タイプ	影響の説明	評価の参照
エンジニアリング	CVM、VPC、サブネット、CBSなどのリソースのクォータがお客様のニーズを満たしているかどうか。	デプロイ	クォータが不足するとリソースの作成に失敗することがあります。自動スケーリングを設定しているユーザーは特に、使用しているクラウドサービスのクォータが十分であることを保障する必要があります。	クラスター購入クォータ制限 クォータ制限
	クラスターのノード上で、カーネルパラメータ、システム設定、クラスターのコアコンポーネントのバージョン、セキュリティグループ、LB関連パラメータなどをユーザーが任意に変更することは推奨しません。	デプロイ	TKEクラスター機能の異常またはノード上にインストールしたKubernetesコンポーネントの異常をもたらし、ノードのステータスが利用不可に変わり、アプリケーションをこのノードにデプロイできなくなる可能性があります。	TKEのハイリスク操作
能動的な運用保守	TKEはさまざまな次元での監視およびアラート機能を提供すると同時に、TCOPが提供する基本的なリソース監視と合わせて、より詳細な指標によるカバーを保証することができます。監視アラートを設定することで、異常の際に速やかにアラートを受信して故障位置の特定を行うことができます。	監視	監視アラートを設定しないと、コンテナクラスターのパフォーマンスの正常な基準を確立できず、異常が発生した場合に適時にアラートを受信できないため、環境の巡回検査を手動で行う必要があります。	アラートの設定 監視データの確認 監視およびアラート指標リスト

クラスター管理
群集をつくる

最終更新日：：2023-05-09 14:41:26

ここでは、TKEコンソールを使用して標準クラスターを作成する方法、ならびにクラスターに必要なVirtual Private Cloud（VPC）、サブネット、セキュリティグループなどのリソースを作成する方法についてご説明します。

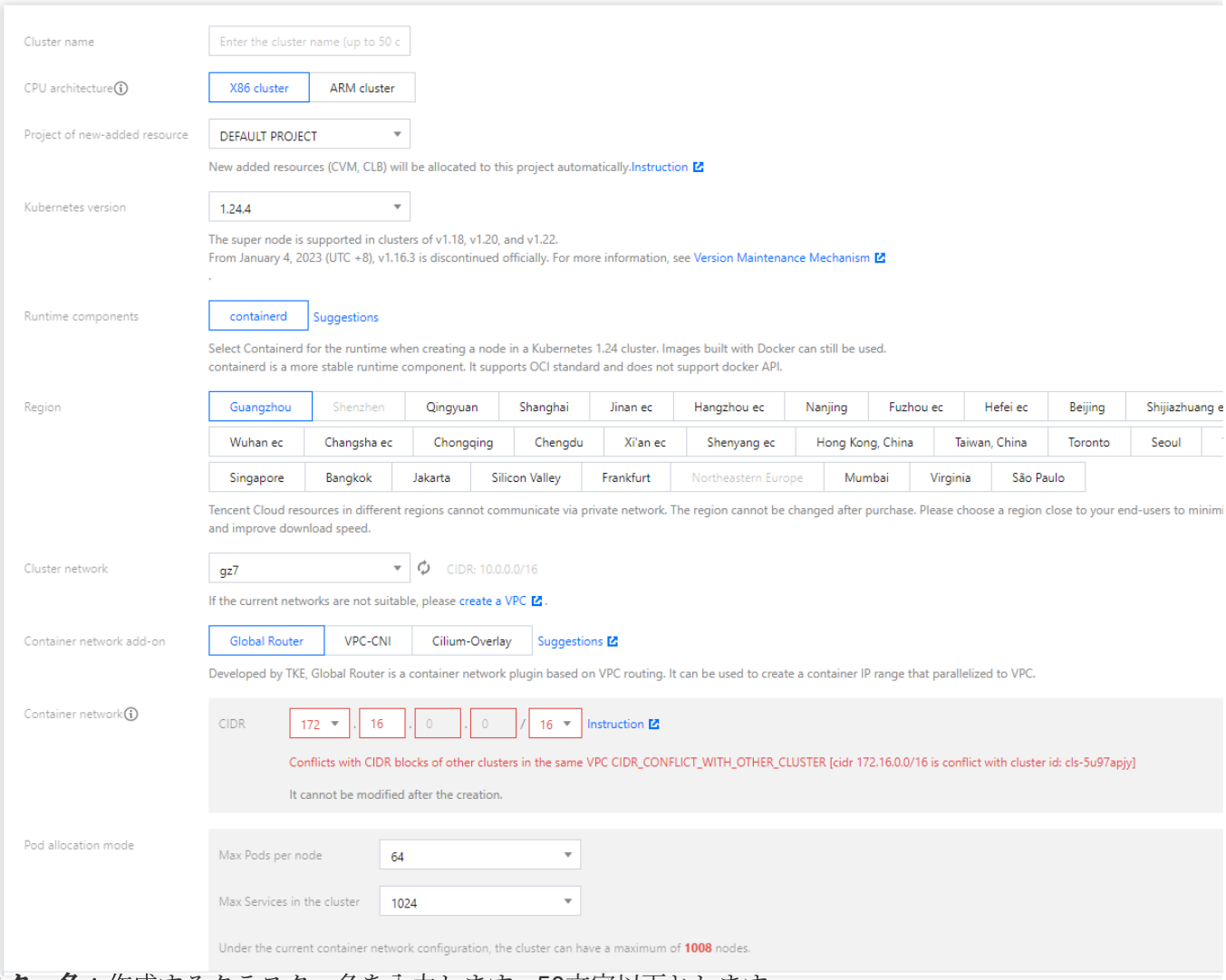
前提条件

クラスターを作成する前に、次の作業を完了する必要があります。
Tencent Cloudアカウントの登録を行います。
TKEコンソールへの初回ログイン時に、現在のアカウントでTencent Kubernetes Engine（TKE）がCloud Virtual Machine（CVM）、Cloud Load Balancer（CLB）、Cloud Block Storage（CBS）などのクラウドリソースを操作する権限を承認する必要があります。詳細については、サービス権限の承認をご参照ください。
- ネットワークタイプがVPCのコンテナクラスターを作成するには、ターゲットリージョンでVPCの作成を行い、VPCのターゲットアベイラビリティーゾーンでサブネットの作成を行う必要があります。
システムによって自動的に作成されたデフォルトのセキュリティグループを使用しない場合は、ターゲットリージョンでセキュリティグループの作成を行い、業務上のニーズを満たすセキュリティグループルールを追加する必要があります。
Linuxインスタンスの作成時にSSHキーペアをバインドしたい場合は、ターゲットプロジェクト下でSSHキーの作成を行う必要があります。
クラスター作成の過程ではVPC、サブネット、セキュリティグループなどの複数のリソースを使用します。リソースの所在リージョンには一定のクォータの制限があります。詳細については、クラスター購入クォータの制限をご参照ください。

コンソールからクラスターを作成する

1. クラスター情報の入力

1. TKEコンソールにログインし、左側ナビゲーションバーの**クラスター**をクリックします。
2. 「クラスター管理」ページで、クラスターリストの上にある**新規作成**をクリックします。
3. **標準クラスター**を選択し、**作成**をクリックします。
4. 「クラスターの作成」ページで、クラスターの基本情報を設定します。下図に示すとおりです。



クラスター名：作成するクラスター名を入力します。50文字以下とします。

追加リソースの所属プロジェクト：実際のニーズに応じて選択します。追加されるリソースはこのプロジェクト下に自動的に割り当てられます。

Kubernetesのバージョン：複数のKubernetesのバージョンから選択できます。Supported Versions of the Kubernetes Documentationで各バージョンの特性を確認して比較できます。

ランタイムコンポーネント：**docker**と**containerd**の2種類から選択できます。詳細については、Containerdと Dockerの選択方法をご参照ください。

所在リージョン：所在地の地理的位置に基づいて最寄りのリージョンを選択することをお勧めします。アクセスの遅延を減らし、ダウンロードスピードを向上させることができます。詳細については、リージョンとアベイラビリティーゾーンをご参照ください。

クラスターネットワーク：クラスター内のホストにノードネットワークアドレスの範囲内のIPアドレスを割り当てます。詳細については、コンテナとノードネットワークの設定をご参照ください。

コンテナネットワークプラグイン：GlobalRouterモード、VPC-CNIモード、Cilium-Overlayモードをご提供しています。詳細については、コンテナネットワークモードの選択方法をご参照ください。

コンテナネットワーク：クラスター内のコンテナにコンテナネットワークアドレスの範囲内のIPアドレスを割り当てます。詳細については、コンテナとノードネットワークの設定をご参照ください。

- **イメージプロバイダ**：パブリックイメージとカスタムイメージの2つのタイプのイメージをサポートしています。詳細については、イメージの概要をご参照ください。
OS：実際のニーズに応じて選択します。
クラスターの記述：クラスターの関連情報を入力します。この情報は**クラスター情報**のページに表示されます。
高度な設定（オプション）：
Tencent Cloudタグ：クラスターにタグ付けを行うと、リソースのカテゴリー管理を行うことができます。詳細については、タグによるリソースの照会をご参照ください。
削除保護：有効化すると、このクラスターがコンソールまたはTencent Cloud APIによって誤って削除されることがなくなります。
Kube-proxyプロキシモデル：iptablesまたはipvsを選択できます。ipvsはクラスター内で大規模なサービスを実行するケースに適しており、有効化後に無効化することはできません。詳細については、クラスターでのIPVSの有効化をご参照ください。
カスタムパラメータ：カスタムパラメータを指定してクラスターを設定できます。詳細については、カスタムKubernetesコンポーネント起動パラメータをご参照ください。
ランタイムのバージョン：コンテナランタイムコンポーネントのバージョンを選択します。
5. **次へ**をクリックします。

2. モデルの選択

「モデルの選択」の手順で、課金モデルを確認し、アベイラビリティーゾーンおよび対応するサブネットを選択し、ノードのモデルを確認します。
1. **ノードソース**を選択します。**新規追加ノード**および**既存ノード**の2つのオプションを提供しています。
新規追加ノード
既存ノード
新規追加ノード、すなわち新たに追加するCVMによってクラスターを作成する場合の詳細は次のとおりです。
クラスタータイプ：**ホスティングクラスター**および**独立クラスター**の2つのオプションを提供しています。
ホスティングクラスター：クラスターのMasterとEtcdはTencent Cloudが管理および保守を行います。
独立クラスター：クラスターのMasterとEtcdはお客様が購入されたCVM上にデプロイされます。
クラスターの仕様：業務の実際の状況に応じて適切なクラスター仕様を選択します。詳細については、クラスター仕様の選択方法をご参照ください。クラスターの仕様は手動で調整するか、または自動アップグレード機能によって自動調整することができます。
課金モデル：**従量課金**の課金モデルを提供しています。詳細については、課金モデルをご参照ください。
Worker設定：**ノードソース**で**新規追加ノード**を選択した場合、**クラスタータイプ**で**ホスティングクラスター**を選択した場合は、このモジュール下のすべての設定項目がデフォルトになっています。実際のニーズに応じて変更できます。
アベイラビリティーゾーン：同時に複数のアベイラビリティーゾーンを選択してMasterまたはEtcdをデプロイすることで、クラスターのより高い可用性を保証できます。
ノードネットワーク：同時に複数のサブネットのリソースを選択してMasterまたはEtcdをデプロイすることで、クラスターのより高い可用性を保証できます。

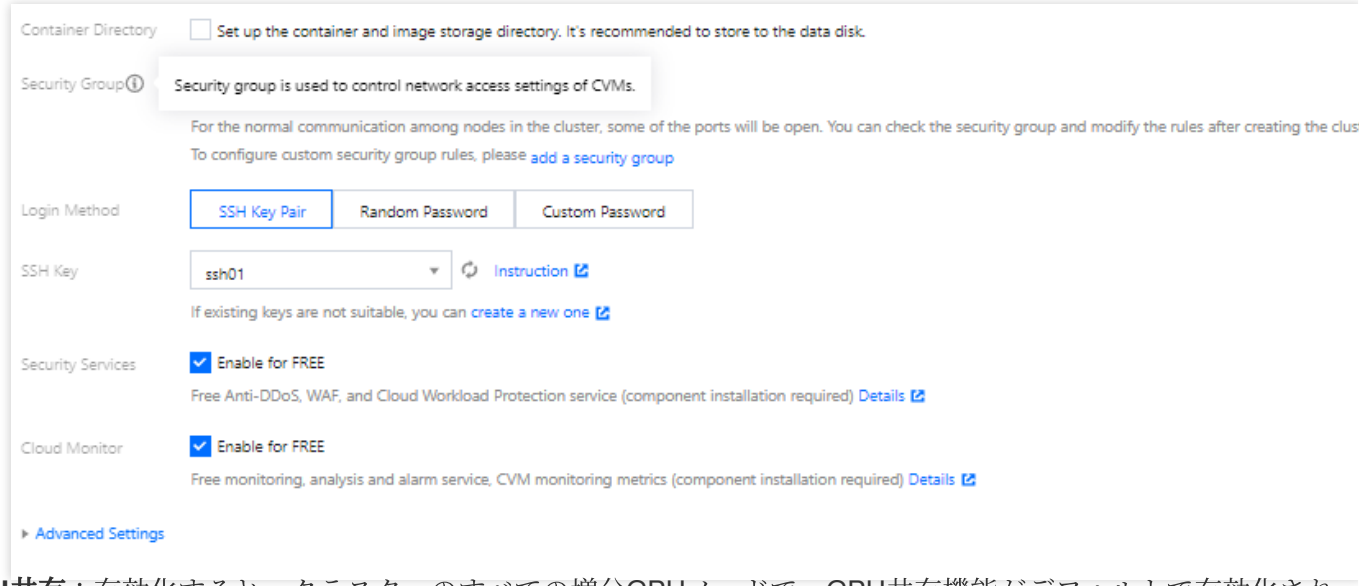
モデル：CPU 4コアより大きいモデルを選択します。具体的な選択プランについては、インスタンス仕様をご参照ください。
システムディスク：デフォルトでは「通常のCBS 50G」となります。モデルに応じてローカルディスク、CBS、SSD CBS、高性能CBSを選択できます。詳細については、ストレージの概要をご参照ください。
データディスク：MasterとEtcdに他のアプリケーションをデプロイすることは推奨しません。デフォルトではデータディスクは設定されておらず、購入後に追加することができます。
パブリックネットワークブロードバンド：**無料のパブリックIPを割り当てる**にチェックを入れると、システムがパブリックIPを無料で割り当てます。課金モデルは2種類あり、その詳細についてはパブリックネットワーク課金モデルをご参照ください。
ホスト名：OS内部のコンピュータ名（`kubectl get nodes`コマンドによって表示されるnode name）であり、この属性がクラスターの属性となります。ホスト名には次の2種類の命名方式があります。
自動命名：ノードのhostnameがデフォルトでノードプライベートIPアドレスとなります。
手動命名：一括連続命名または指定パターン文字列命名をサポートしています。小文字アルファベット、数字、ハイフン「-」、ドット「.」のみをサポートします。記号は先頭または末尾には使用できず、連続して使用することもできません。その他の命名ルールのガイドについては、一括連続命名または指定パターン文字列命名をご参照ください。
注意
kubernetes nodeの命名の制限により、ホスト名を手動で命名する場合は小文字アルファベットのみ使用可能です（例：`cvm{R:13}-big{R:2}-test`）。
インスタンス名：コンソールに表示されるCVMインスタンス名です。この属性はホスト名の命名方式の制限を受けます。
ホスト名が自動命名方式の場合：一括連続命名または指定パターン文字列命名をサポートします。インスタンス名はデフォルトで自動生成され、形式は`tke_クラスターid_worker`となります。
ホスト名が手動命名方式の場合：インスタンス名はホスト名と同じになり、再設定の必要はありません。
CVM数：インスタンス数です。実際のニーズに応じて選択します。
説明
クラスタータイプで**独立クラスター**を選択した場合、Master&Etcdノードの設定項目はWorkerの設定を参照することも可能です。この数を少なくとも3台にデプロイします。異なるアベイラビリティーゾーン間のデプロイも可能です。

既存ノード、すなわち既存のCVMを使用してクラスターを作成する場合の詳細は次のとおりです。
注意
選択するCVMはシステムの再インストールが必要です。再インストールすると、CVMのシステムディスクのすべてのデータがクリアされます。
選択したCVMはクラスターの所属プロジェクトに移行します。またCVMのプロジェクト移行によってセキュリティグループのバインドが解除されるため、セキュリティグループを再バインドする必要があります。
CVMの設定の際にデータディスクのマウントパラメータを入力した場合、これらのパラメータは**Masterノードと Wokerノードのすべてに対して有効**となります。その他の関連の注意事項については、既存ノードの追加の、

データディスクのマウントパラメータについての説明をご参照ください。
クラスタータイプ：**ホスティングクラスター**および**独立クラスター**の2つのオプションを提供しています。
ホスティングクラスター：クラスターのMasterとEtcdはTencent Cloudが管理および保守を行います。
独立クラスター：クラスターのMasterとEtcdはお客様が購入されたCVM上にデプロイされます。
クラスターの仕様：業務の実際の状況に応じて適切なクラスター仕様を選択します。詳細については、クラスター仕様の選択方法をご参照ください。クラスターの仕様は手動で調整するか、または自動アップグレード機能によって自動調整することができます。
Worker設定：実際のニーズに応じて既存のCVMにチェックを入れます。
2. **次へ**をクリックし、CVMの設定を開始します。

3. CVMの設定

1. 「CVMの設定」の手順で、次の情報を参照してCVMの設定を行います。下図に示すとおりです。



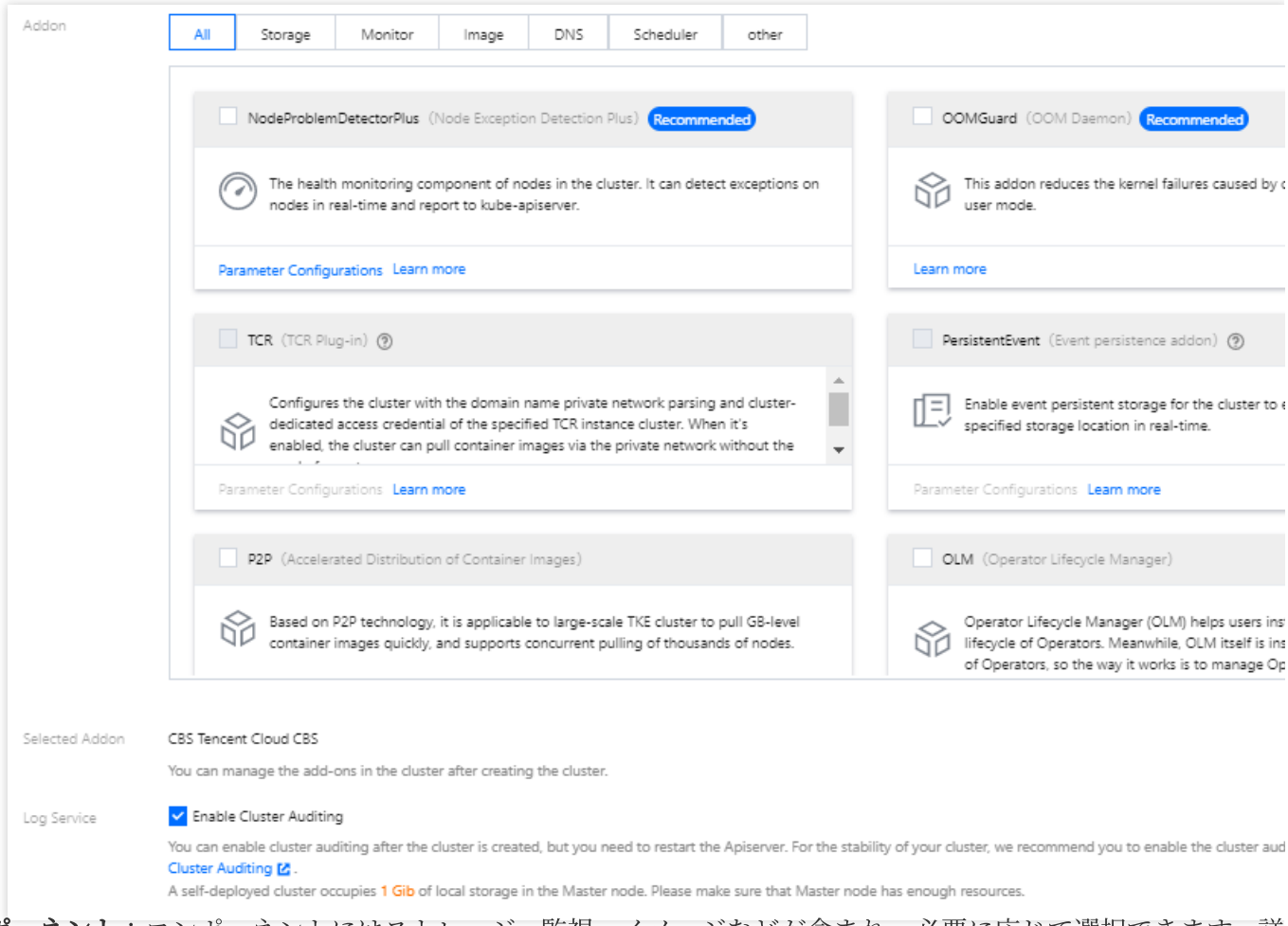
qGPU共有：有効化すると、クラスターのすべての増分GPUノードで、GPU共有機能がデフォルトで有効化されます。隔離機能を有効化するかどうかはLabelで制御できます。GPU共有スケジューリング機能を正常に使用するにはコンポーネントのインストールが必要な点にご注意ください。
コンテナディレクトリ：チェックを入れると、コンテナとイメージストレージディレクトリを設定することができます。データディスクへの保存を推奨します（例：`/var/lib/docker`）。
セキュリティグループ：セキュリティグループにはファイアウォールの機能があり、CVMのネットワークアクセス制御の設定に用いられます。次の設定をサポートしています。
デフォルトのセキュリティグループを新規作成してバインドする場合、デフォルトのセキュリティグループルールをプレビューできます。
セキュリティグループを追加し、業務ニーズに応じてセキュリティグループルールをカスタム設定できます。詳しい情報についてはTKEセキュリティグループの設定をご参照ください。
ログイン方式：3種類のログイン方式があります。

今すぐキーをバインドする：キーペアはアルゴリズムによって生成された一対のパラメータであり、一般的なパスワードと比べてより安全なCVMへのログイン方式です。詳細については、SSHキーをご参照ください。
パスワードの自動生成：自動生成されたパスワードはサイト内メッセージで送信されます。
パスワードの設定：対応するパスワードを表示に従って設定してください。
セキュリティ強化：Anti-DDoS、WAFおよびクラウドイメージホスト保護がデフォルトで無料でアクティブ化されます。詳細については、 T-Sec Cloud Workload Protection公式サイトページをご参照ください。
Tencent Cloud Observability Platform：クラウド製品の監視、分析、アラート実施がデフォルトで無料でアクティブ化され、コンポーネントをインストールしてホスト監視指標を取得できます。詳細については、Tencent Cloud Observability Platform（TCOP）をご参照ください。
2.（オプション）**高度な設定**をクリックすると、下図のように、その他の情報を確認または設定することができます。

CAMロール：今回のバッチで作成したすべてのノードに同一のCAMロールをバインドすることで、このロールにバインドされた権限承認ポリシーをノードに付与することができます。詳細については、インスタンスロールの管理をご参照ください。
ノード起動設定：カスタムデータを指定してノードを設定します。これはすなわち、ノードの起動後に設定を実行するスクリプトです。スクリプトが再入可能であり、ロジックが再試行可能であることを確実にする必要があります。スクリプトとそれが生成するログファイルは、ノードの `/usr/local/qcloud/tke/userscript` パスで確認することができます。
コルドン：「コルドンを有効化する」にチェックを入れると、新しいPodをこのノードにスケジューリングできなくなり、ノードのコルドンを手動でキャンセルするか、またはカスタムデータ内でコルドンキャンセルコマンドを実行する必要があります。必要に応じて設定してください。
Label：**追加**をクリックすると、Labelをカスタム設定できます。クラスターの初期化によって作成されたノードにはここで設定したLabelが自動的に追加され、その後、Labelによるノードのフィルタリング、管理を行うことができるようになります。
3. **次へ**をクリックし、コンポーネントの設定を開始します。

4. コンポーネントの設定

1.「コンポーネントの設定」の手順で、次の情報を参照してコンポーネントの設定を行います。下図に示すとおりです。



コンポーネント：コンポーネントにはストレージ、監視、イメージなどが含まれ、必要に応じて選択できます。詳細については、拡張コンポーネントの概要をご参照ください。

Prometheus監視サービス：有効化すると、データ収集ルールを実際のニーズに応じて柔軟に設定し、なおかつ必要に応じてアラートルールを設定することができます。設定が完了すると、Grafanaを開いて監視データを確認できるようになります。詳細については、Prometheus監視サービスをご参照ください。

Cloud Log Service：クラスター審査サービスがデフォルトでアクティブ化されます。詳細については、クラスター審査をご参照ください。

2. **次へ**をクリックし、設定情報をチェックして確認します。

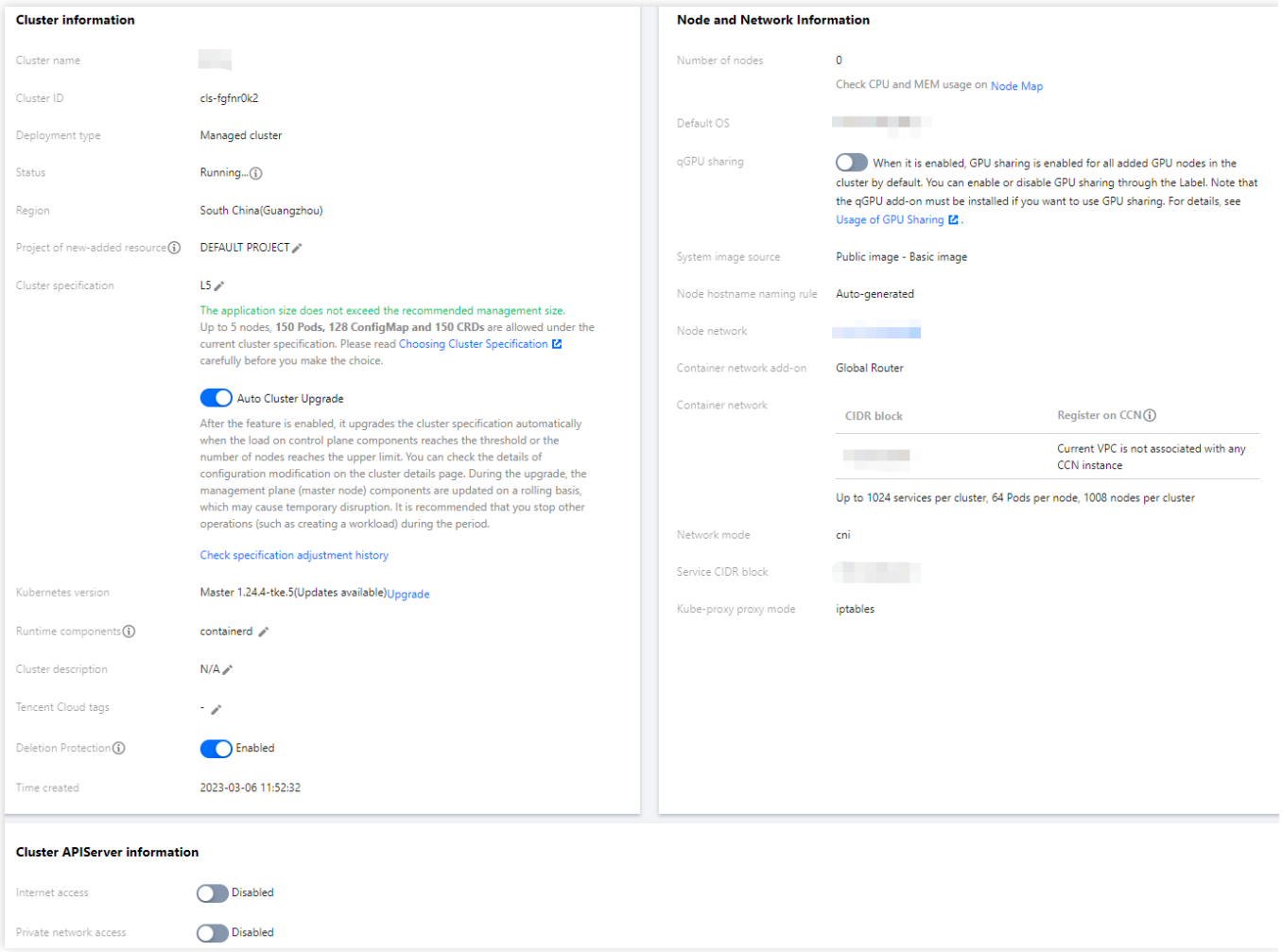
5. 情報の確認

「情報の確認」ページで、クラスターの選択済みの設定情報および料金を確認し、**私はTKEサービスレベル契約を読み、同意しました。にチェックを入れ、完了をクリック**すると、クラスターの作成が完了します。

6. クラスターの確認

作成が完了したクラスターはクラスターリストに表示されます。クラスターIDをクリックしてクラスター詳細ページに進むことができます。クラスターの「基本情報」ページで、クラスター情報、ノードおよびネットワーク情報

などを確認することができます。下図に示すとおりです。



APIによるクラスター作成

CreateClusterインターフェースを使用してクラスターを作成することもできます。詳細な情報については、クラスター作成APIドキュメントをご参照ください。

クラスター接続

最終更新日：：2023-05-19 11:31:50

ユースケース

KubernetesコマンドラインツールKubectlを使用して、ローカルクライアントマシンからTKEクラスターに接続できます。このドキュメントでは、クラスターに接続する方法について説明します。

注意

プラットフォームをセキュリティコンプライアンスの要件に合致させ、Tencent Kubernetes Engineのクラスターアクセスの安全性および安定性を向上させるために、cls-xxx.ccs.tencent-cloud.comのドメイン名は北京時間2022年8月10日に正式にオフラインになります。

使用体験を保証し、スムーズな移行をしやすくするために、プラットフォームは北京時間7月20日にクラスターアクセス機能（リリース後、クラスター内パブリックネットワークアクセスを有効化して新バージョンを使用します。既存のクラスターは切り替えが必要です）をアップグレードします。北京時間7月20日から8月10日までにクラスターアクセスを再度有効化することによって新しいバージョンのクラスターアクセスに切り替え、それによってドメイン名がオフラインになってから正常にクラスターにアクセスできなくなることを回避します。このドキュメント内のクラスターアクセスの有効化およびKubeconfigの取得を参照して操作を行ってください。

新しいアーキテクチャと古いアーキテクチャには以下のような違いがあります。クラスターアクセスに関するTencent Cloud APIは新しいアーキテクチャに互換性があり、スムーズな移行をサポートします。関連するAPIを呼び出す場合は、**7月20日**までに関連する設定をしてください。

1. 新しいアーキテクチャはパブリックネットワークドメイン名の解析機能を提供しません。プラットフォームは入力したドメイン名にセキュリティ署名を行います。アクセスの安全を保証するために、ご自身でパブリックネットワークドメイン名解決を設定する必要があります。
2. 新しいアーキテクチャでパブリックネットワークアクセスを有効化する場合、セキュリティグループを入力することによってソースの権限承認を設定する必要があります。
3. 新しいアーキテクチャでパブリックネットワークアクセスを有効化する場合、アカウントでCLBを作成します。パブリックネットワークCLB課金項目の詳細については標準アカウントタイプ課金説明をご参照ください。

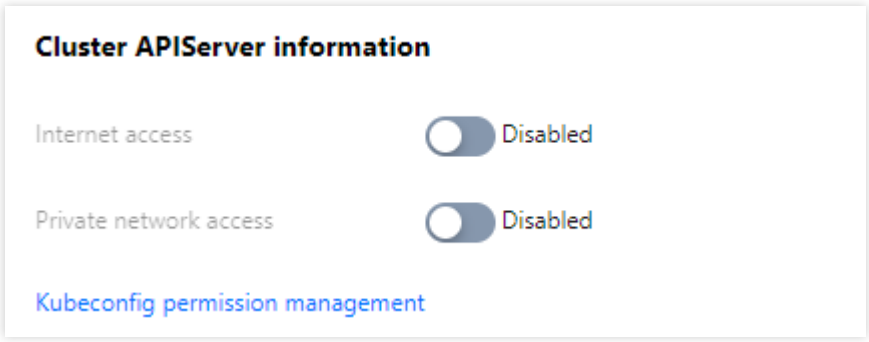
方法1：Cloud Shellによってクラスターを接続する

TKEはTencent Cloud Shellを統合し、Tencent Cloudコンソールにおいてワンクリックでクラスターを接続する機能を実現できます。kubectlによってクラスターに対する柔軟な操作を実現します。

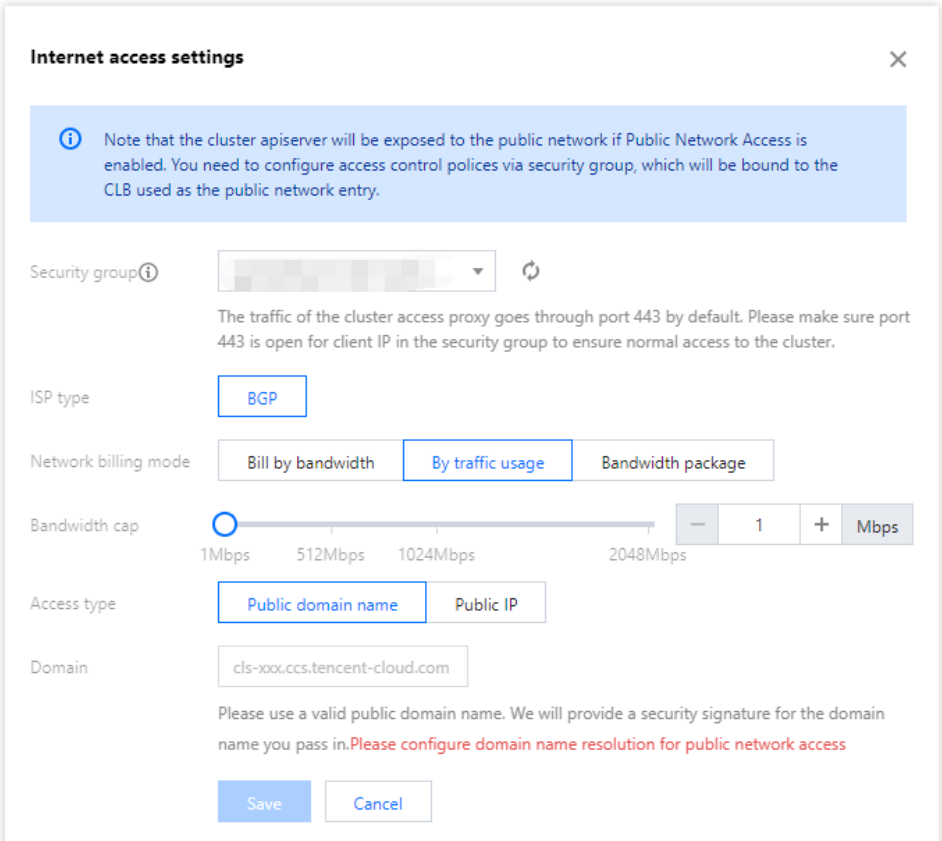
操作手順

ステップ1：クラスターのパブリックネットワークアクセスを有効化する

1. TKEコンソールにログインし、左側ナビゲーションバーのクラスターを選択します。
2. **クラスター管理**ページで、クラスター所在リージョンを選択し、目標のクラスターIDをクリックし、クラスター詳細ページに進みます。
3. **クラスター基本情報**ページで、下図に示すようにクラスターアクセスの有効化状態を確認します。



4.  をクリックしてパブリックネットワークアクセスを有効化します。パブリックネットワークアクセスを有効化する場合、下図に示すように、関連パラメータを設定する必要があります。



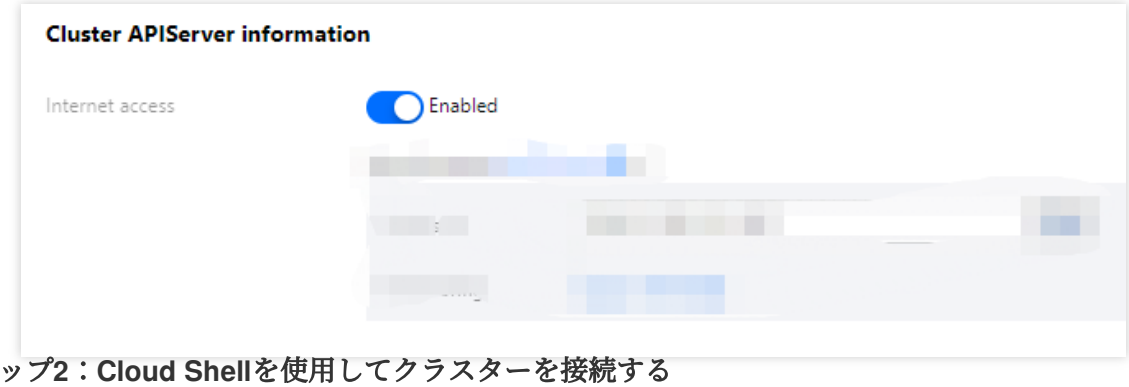
セキュリティグループ：パブリックネットワークアクセスを有効化すると、1つのパブリックネットワークCLBをアクセスポートとして自動的に割り当てます。セキュリティグループによってソース権限承認を設定すると、セ

キュリティグループをパブリックネットワークCLBにバインドし、それによってアクセス制御の効果を達成します。

キャリアタイプ、ネットワーク課金モデル、帯域幅の上限：CLB関連パラメータについては、CLB作成ガイドをご参照ください。実際のニーズに応じて設定する必要があります。

アクセス方式：パブリックネットワークのドメイン名を選択した後、カスタムドメイン名を入力する必要があります。入力したドメイン名にセキュリティ署名が実行され、ご自身でパブリックネットワーク解析を設定する必要があります。CLBのデフォルトドメインを選択する場合、手動でドメイン名解析などの操作を設定する必要はありません。

5. 下図に示すとおり、パブリックネットワークアクセスが有効化されたことを確認します。



ステップ2：Cloud Shellを使用してクラスターを接続する

1. TKEコンソールにログインし、左側ナビゲーションバーのクラスターを選択します。

2. **クラスター管理**ページで、クラスター所在リージョンを選択し、目標クラスター右側の**その他 > クラスターの接続を**クリックすると、下図のように表示されます。

3. コンソールの下方にCloud Shell入口が表示されます。直接コマンドボックス内にkubectlコマンドを入力します。

方法2：ローカルコンピュータでクラスターを接続する

前提条件

curlをインストールしてください。

操作手順

ステップ1：Kubectlツールをインストールする

1. Installing and Setting up kubectlを参照して、Kubectlツールをインストールします。OSのタイプに応じて、Kubectlを入手する方法を選択できます。

注意

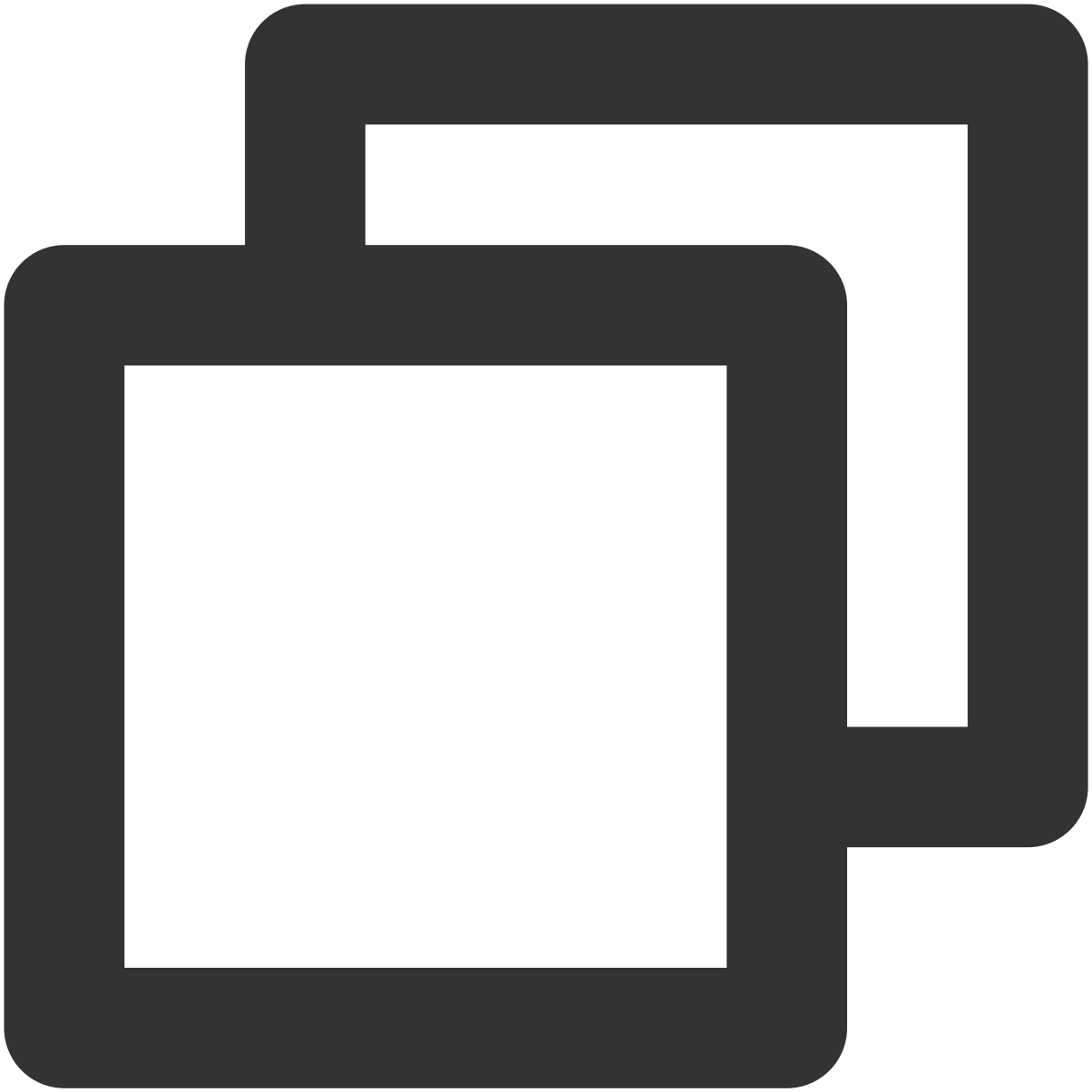
Kubectlツールをインストールしている場合、このステップを無視してください。
実際のニーズに応じて、コマンドライン内の「v1.18.4」を業務に必要なKubectlバージョンに置換してください。
クライアントのKubectlとサーバーのKubernetesの最新バージョンが一致している必要があります。**基本情報**の「クラスター情報」モジュールでKubernetesのバージョンを確認することができます。

Mac OS Xシステム

Linuxシステム

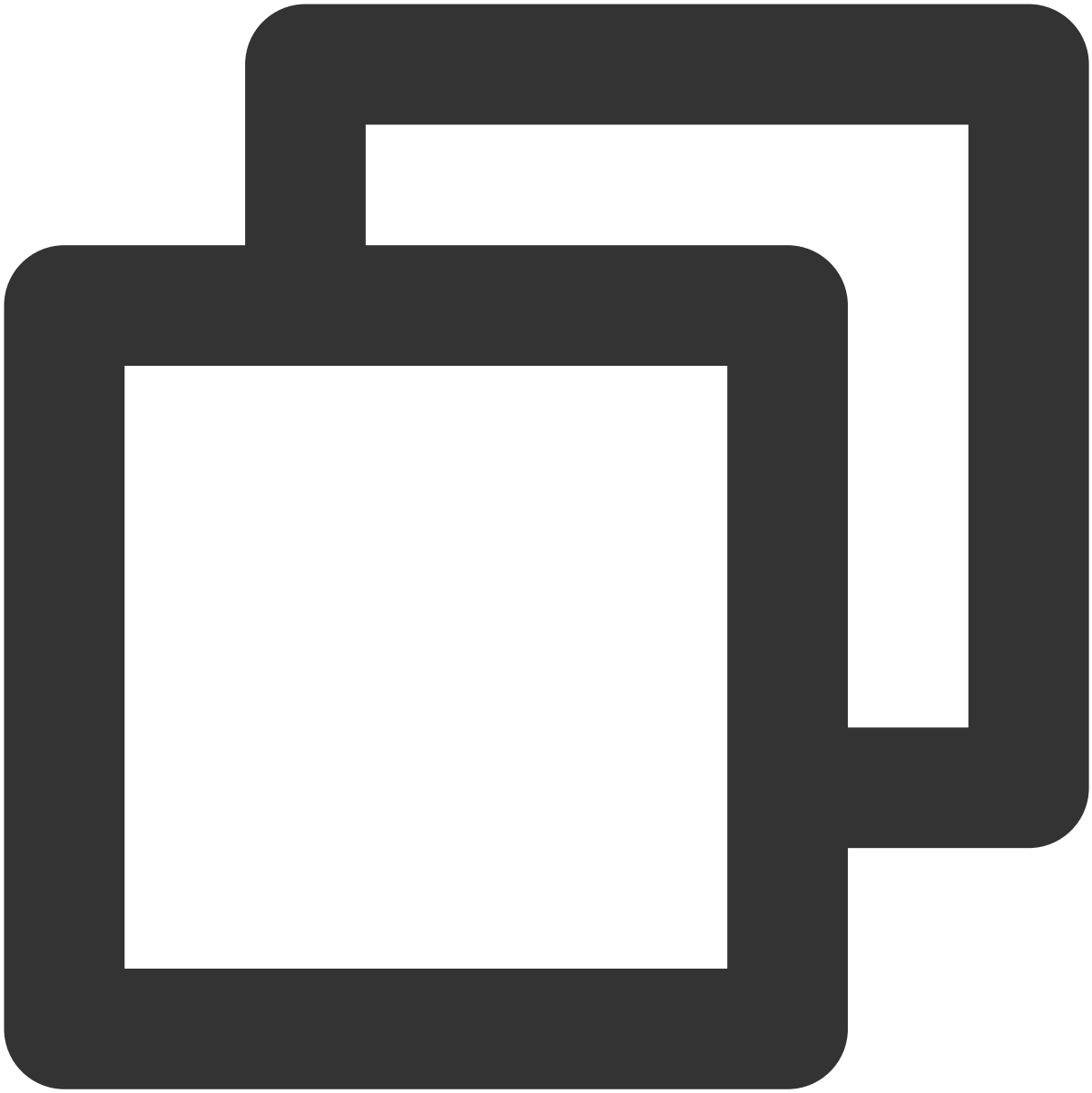
Windowsシステム

次のコマンドを実行して、Kubectlツールを取得します：



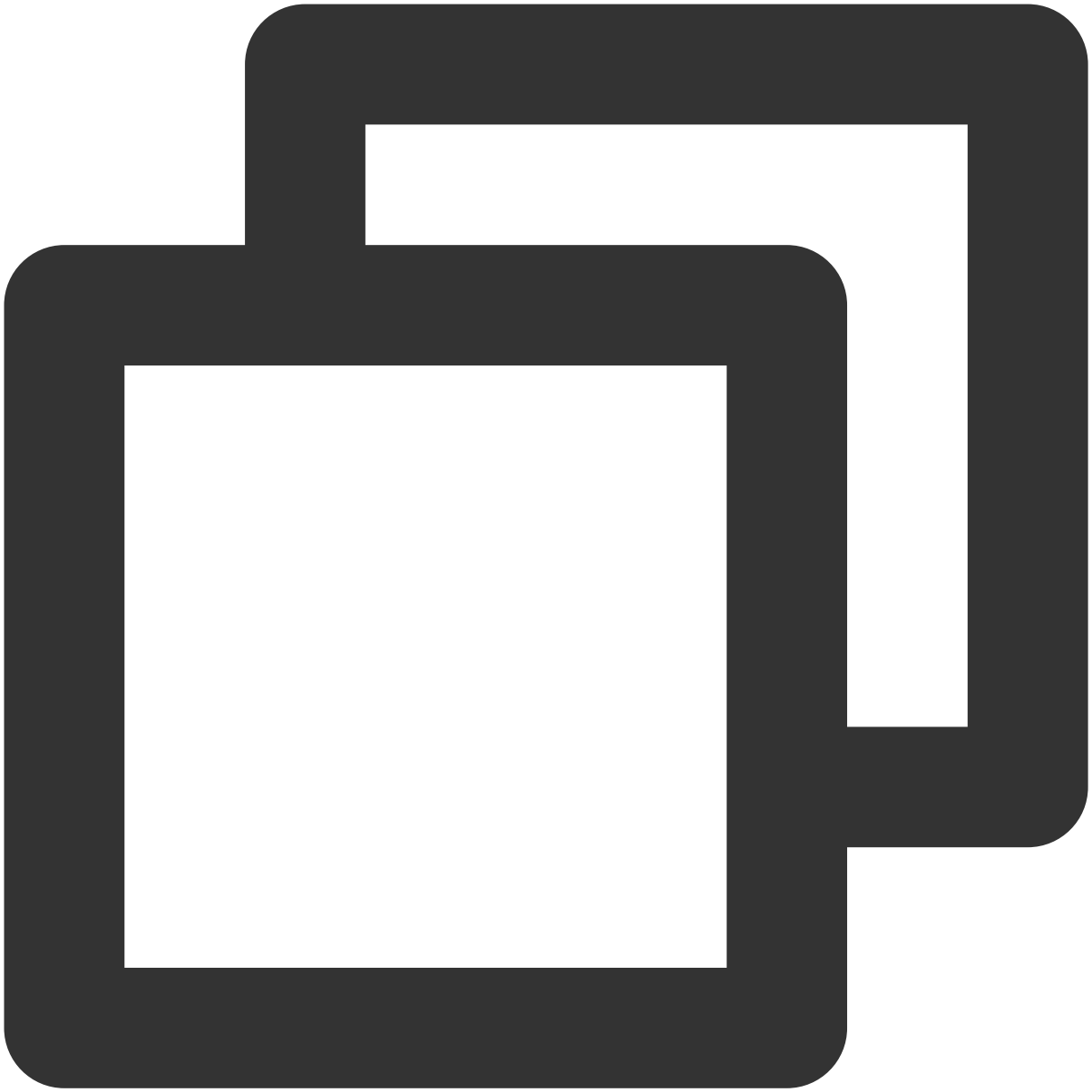
```
curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.18.4/bin/darw
```

次のコマンドを実行して、Kubectlツールを取得します：



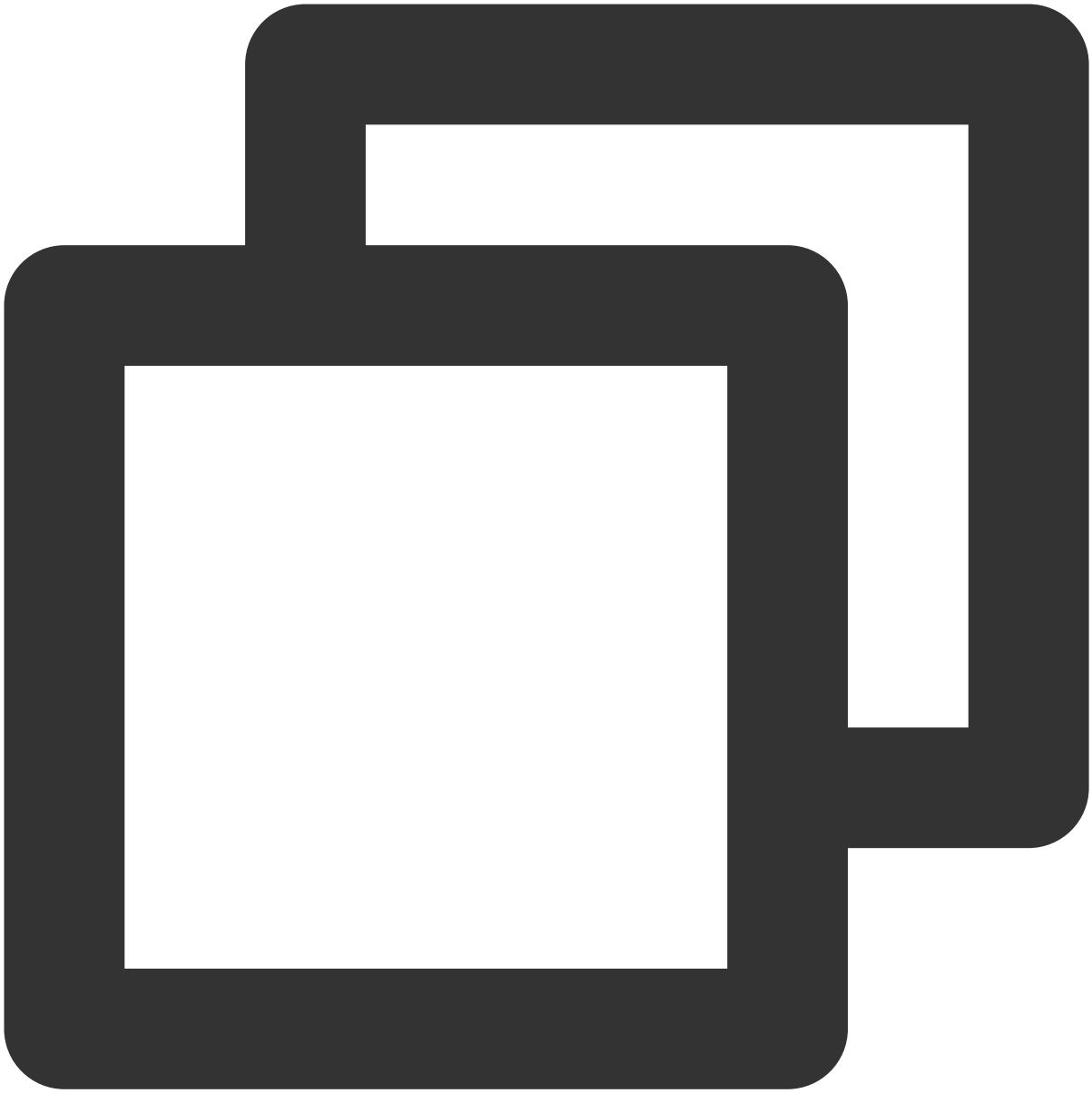
```
curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.18.4/bin/linu
```

次のコマンドを実行して、Kubectlツールを取得します：



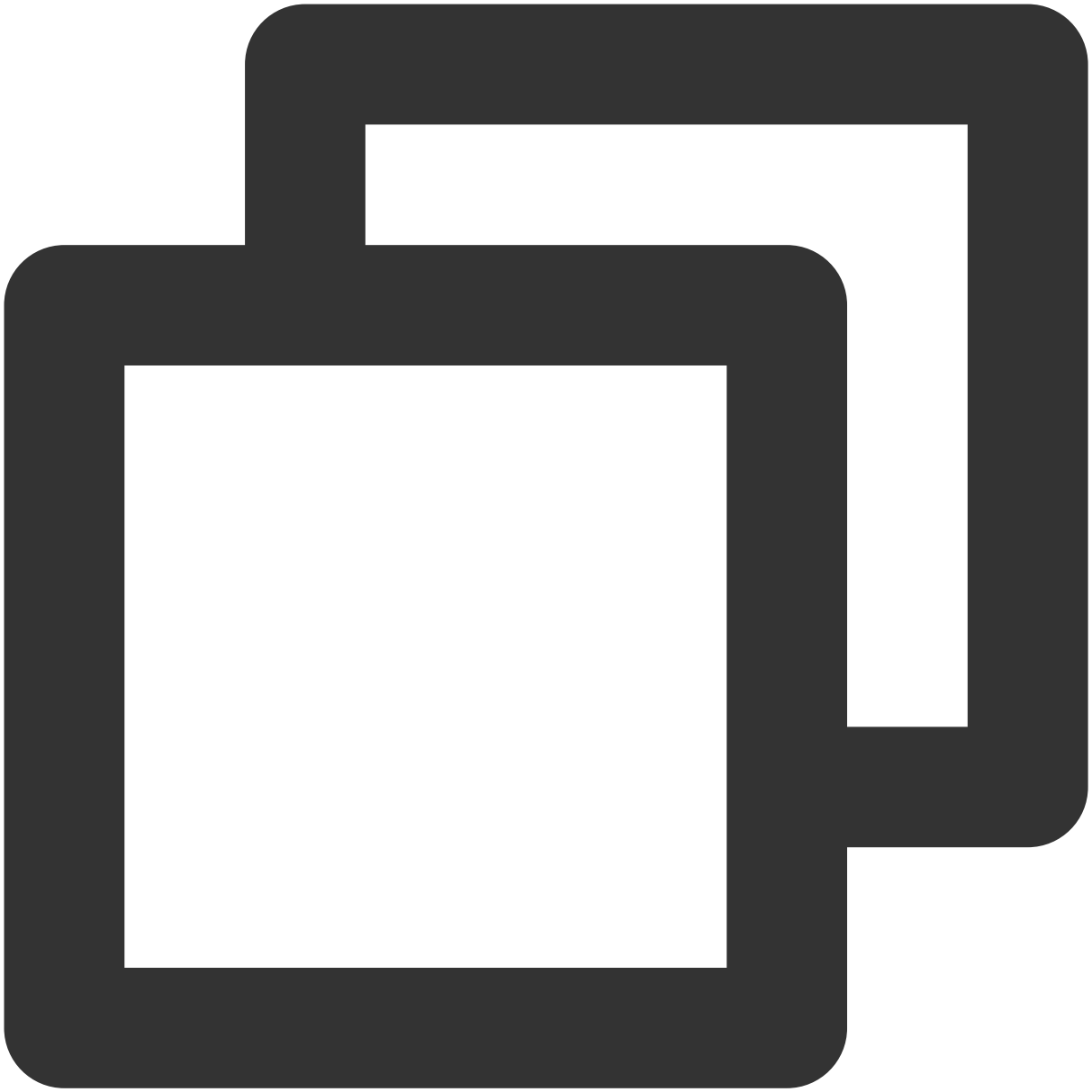
```
curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.18.4/bin/wind
```

2. このステップではLinuxシステムを例にとります。以下のコマンドを実行し、実行権限を追加します。



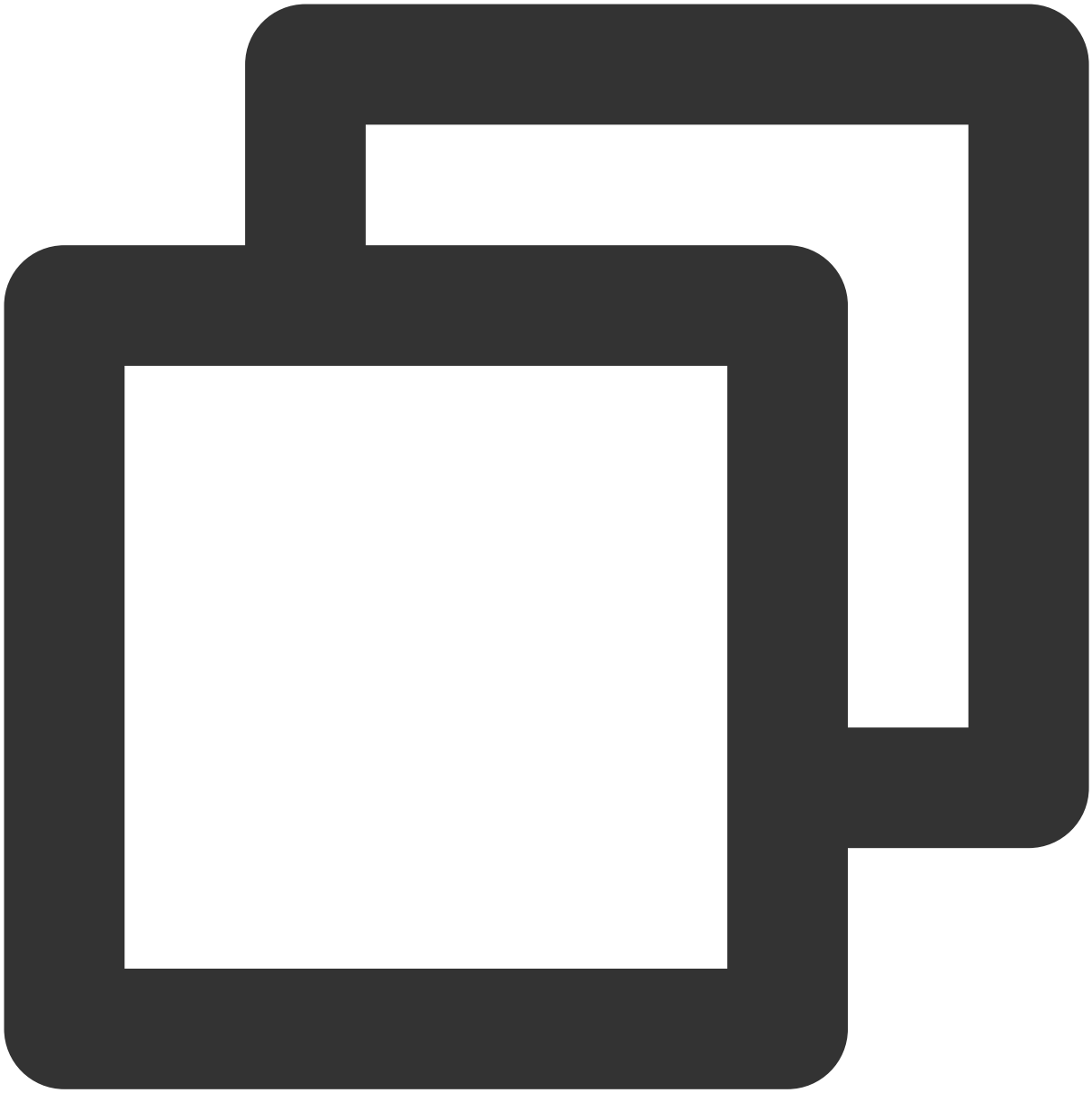
```
chmod +x ./kubectl
sudo mv ./kubectl /usr/local/bin/kubectl
```

3. 以下のコマンドを実行し、インストール結果をチェックします。



```
kubectl version
```

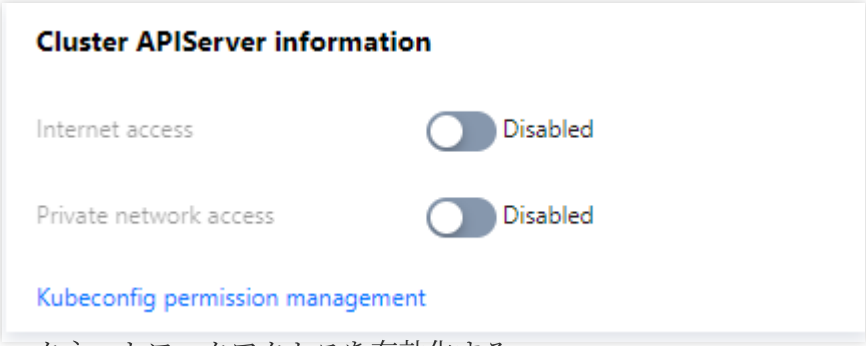
次のようなバージョン情報が出力されている場合は、インストールが成功したことを意味します。



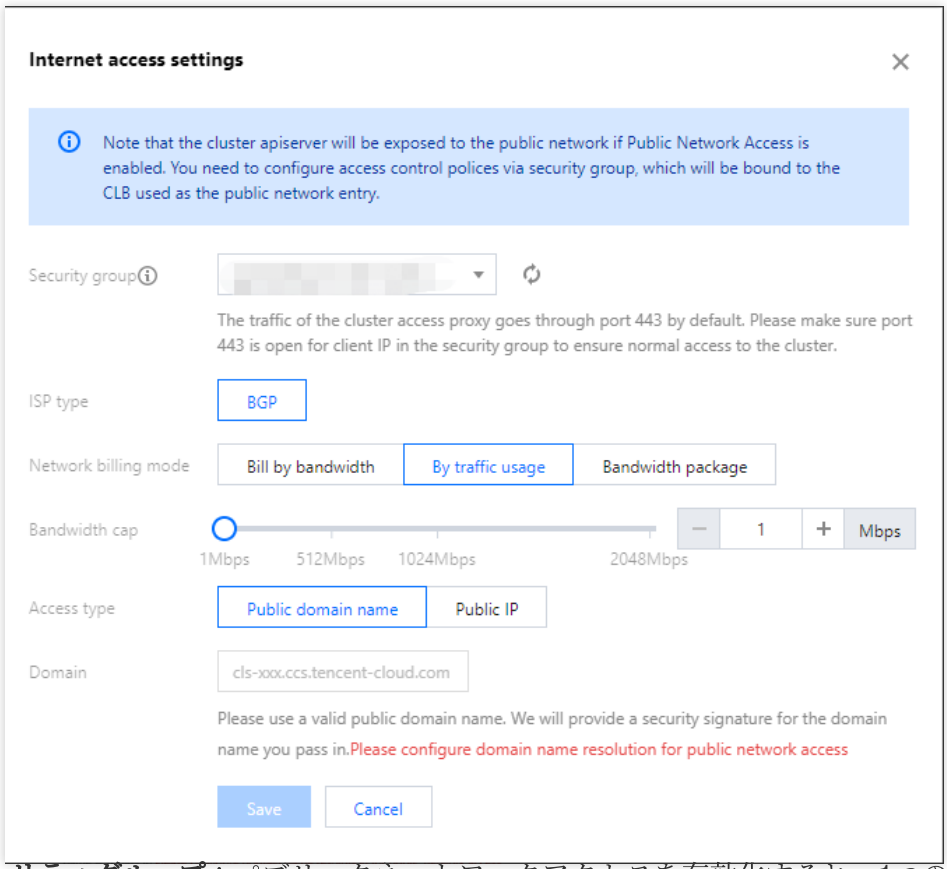
```
Client Version: version.Info{Major:"1", Minor:"5", GitVersion:"v1.5.2", GitCommit:"
```

ステップ2：クラスターのアクセスを有効化する

1. TKEコンソールにログインし、左側ナビゲーションバーのクラスターを選択します。
2. 「クラスター管理」ページで、クラスター所在リージョンを選択し、目標**クラスターID/名**をクリックし、クラスター詳細ページに進みます。
3. クラスター基本情報ページで、下図に示すようにクラスターアクセスの有効化状態を確認します。

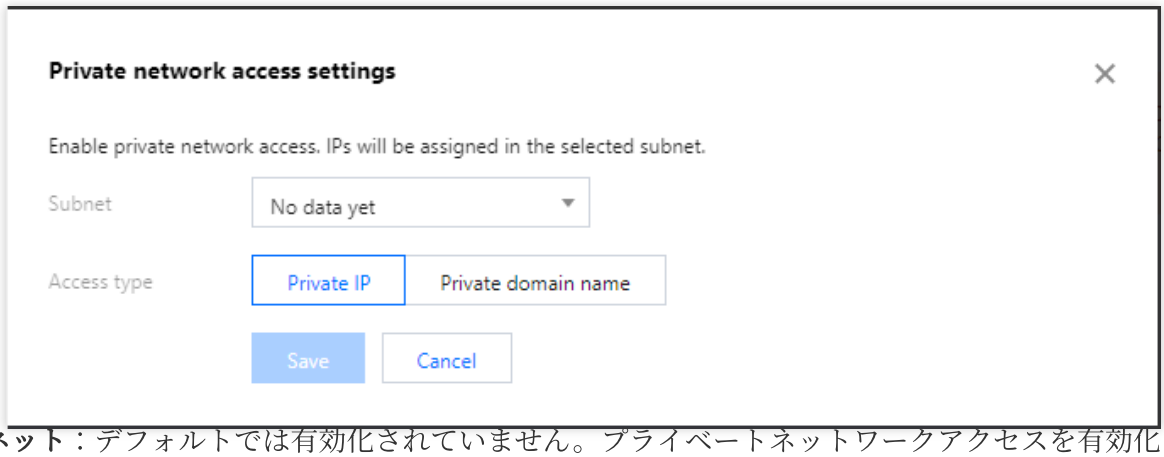


パブリックネットワークアクセスを有効化する
プライベートネットワークアクセスを有効化する
パブリックネットワークアクセスを有効化する場合、下図に示すように関連パラメータを設定する必要があります。



セキュリティグループ：パブリックネットワークアクセスを有効化すると、1つのパブリックネットワークCLBをアクセスポートとして自動的に割り当てます。セキュリティグループによってソース権限承認を設定すると、セキュリティグループをパブリックネットワークCLBにバインドし、それによってアクセス制御の効果を達成します。
キャリアタイプ、ネットワーク課金モデル、帯域幅の上限：CLB関連パラメータについては、CLB作成ガイドをご参照ください。実際のニーズに応じて設定する必要があります。
アクセス方式：パブリックネットワークのドメイン名を選択した後、カスタムドメイン名を入力する必要があります。入力したドメイン名にセキュリティ署名が実行され、ご自身でパブリックネットワーク解析を設定する必

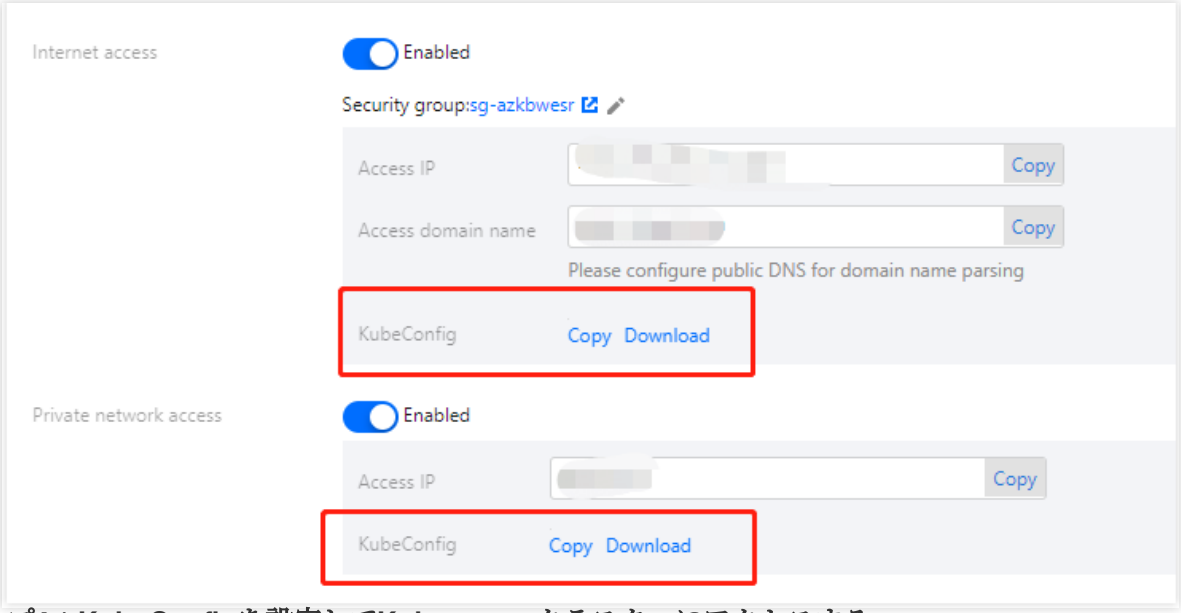
要があります。CLBのデフォルトドメインを選択する場合、手動でドメイン名解析などの操作を設定する必要はありません。
プライベートネットワークアクセスを有効化する場合、下図に示すように関連パラメータを設定する必要があります。



サブネット：デフォルトでは有効化されていません。プライベートネットワークアクセスを有効化した場合、サブネットを設定する必要があり、有効化に成功すると設定したサブネット内でIPアドレスを割り当てます。
アクセス方式：プライベートネットワークドメインを選択した後、カスタムドメイン名を入力する必要があります。入力したドメイン名にセキュリティ署名が実行され、ご自身でプライベートネットワーク解析を設定する必要があります。プライベートネットワークIPを選択すると、プライベートネットワークIPを割り当ててセキュリティ署名します。
KubernetesのService IPを使用する：クラスター詳細ページで、左側の**サービスとルーティング > Service**を選択してdefaultネームスペースにあるKubernetesのservice IPを取得します。 Kubeconfigファイル内のclusters.cluster.serverフィールドをhttps://< `IP` >:443に置換すれば完了です。**注意：**Kubernetes serviceはClusterIP モードです。クラスター内アクセスにのみ適用されます。

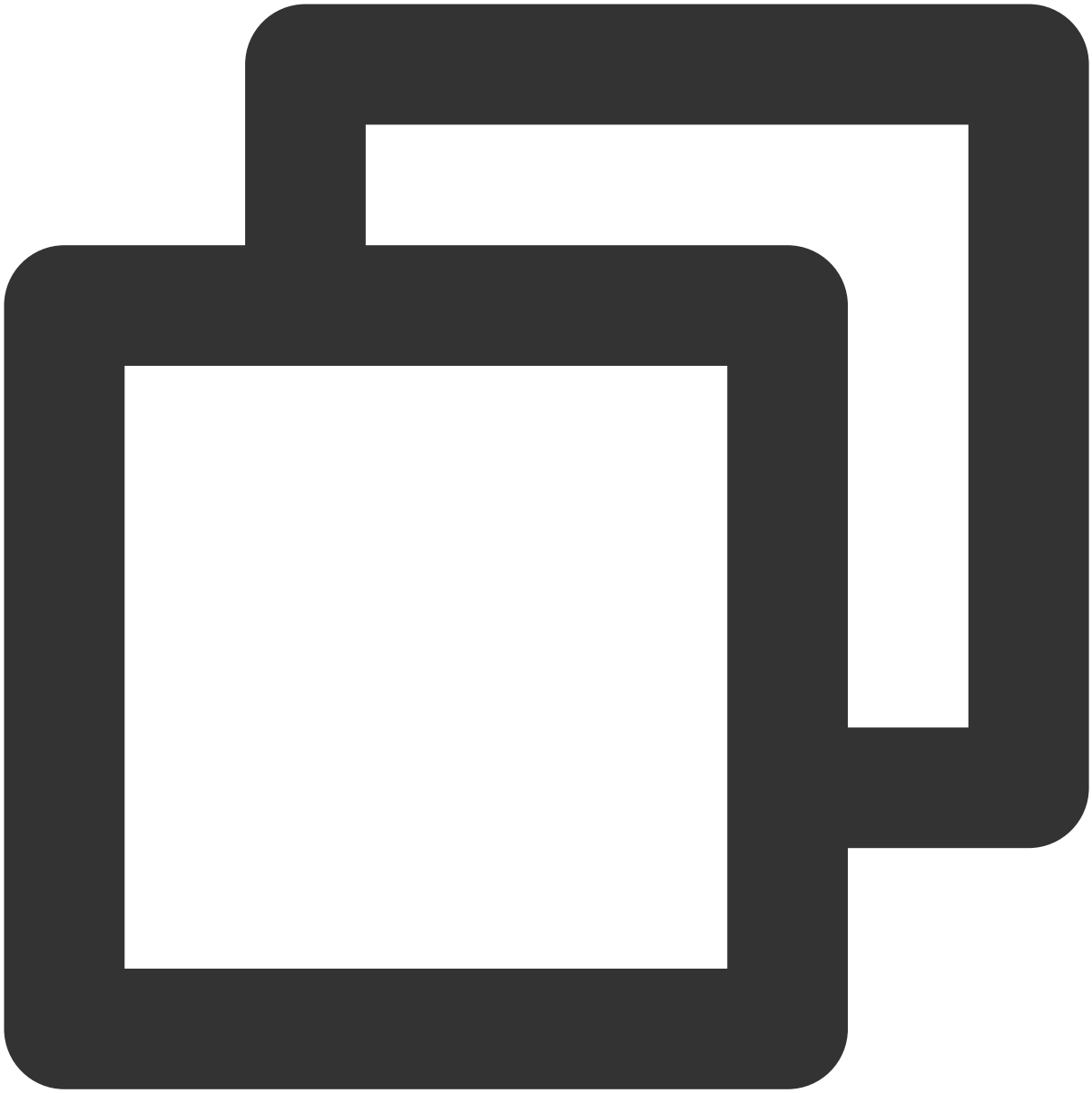
ステップ3：KubeConfigの取得

TKEは2種類のKubeConfigを提供し、それぞれパブリックネットワークアクセスおよびプライベートネットワークアクセスに使用されます。クラスターアクセスを有効化すると、以下のステップに従って対応するKubeconfigを取得することができます。
1. **クラスター詳細 > 基本情報**で、「クラスターのAPIServer情報」を確認します。
2. クラスターアクセスに対応するスイッチの下方で、**Kubeconfig**をコピーまたはダウンロードするか、パブリックネットワークアクセスセキュリティグループ、アクセスドメイン名（アクセス有効化時に設定）、アクセスIPを確認します。下図のように表示されます。

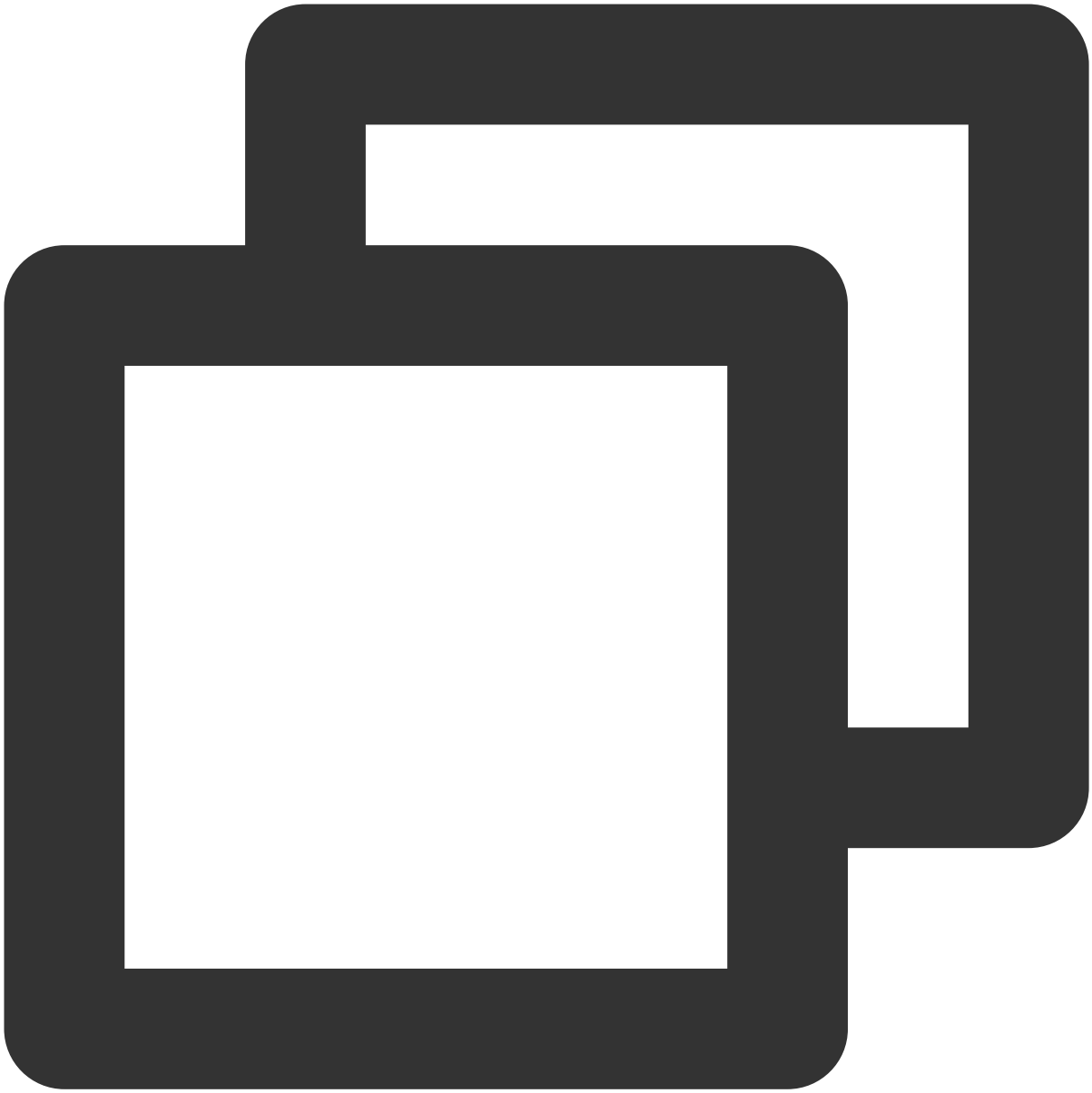


ステップ4：KubeConfigを設定してKubernetesクラスターにアクセスする

1. 実際の状況に応じてクラスター認証情報を設定します。
構成前に、現在のアクセスクライアントはクラスターのアクセス証明が構成されているかどうかを確認してください：
いいえの場合は、 `~/.kube/config` ファイル内容が空白であり、直接取得したKubeconfigアクセス証明書の内容をコピーして `~/.kube/config` に貼り付けることができます。クライアントに `~/.kube/config` ファイルがない場合、直接作成することができます。
はいの場合、取得したKubeconfigを指定の位置にダウンロードし、順次以下のコマンドを実行して複数のクラスターのconfigを結合します。



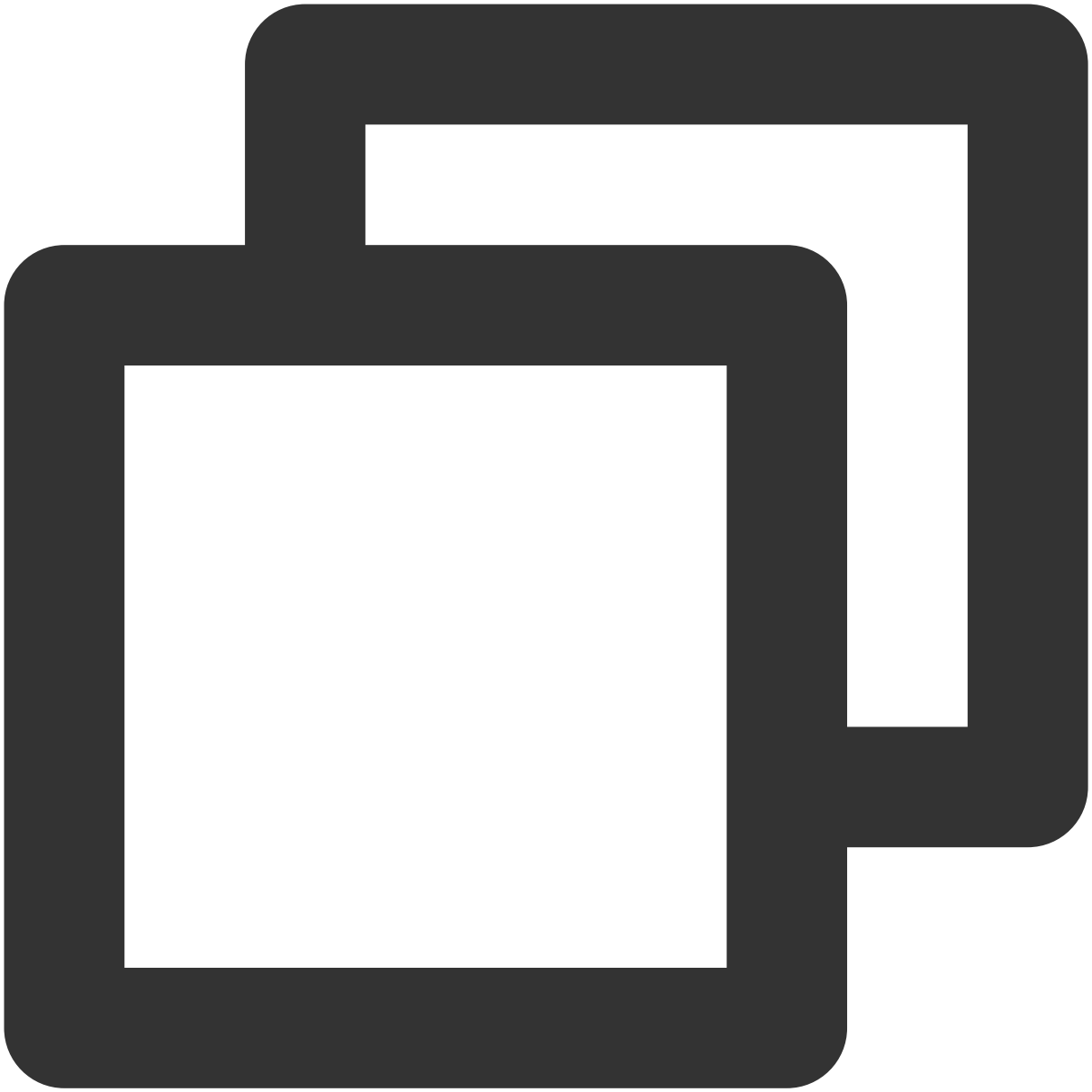
```
KUBECONFIG=~/.kube/config:~/Downloads/cls-3jju4zdc-config kubectl config view --mer
```



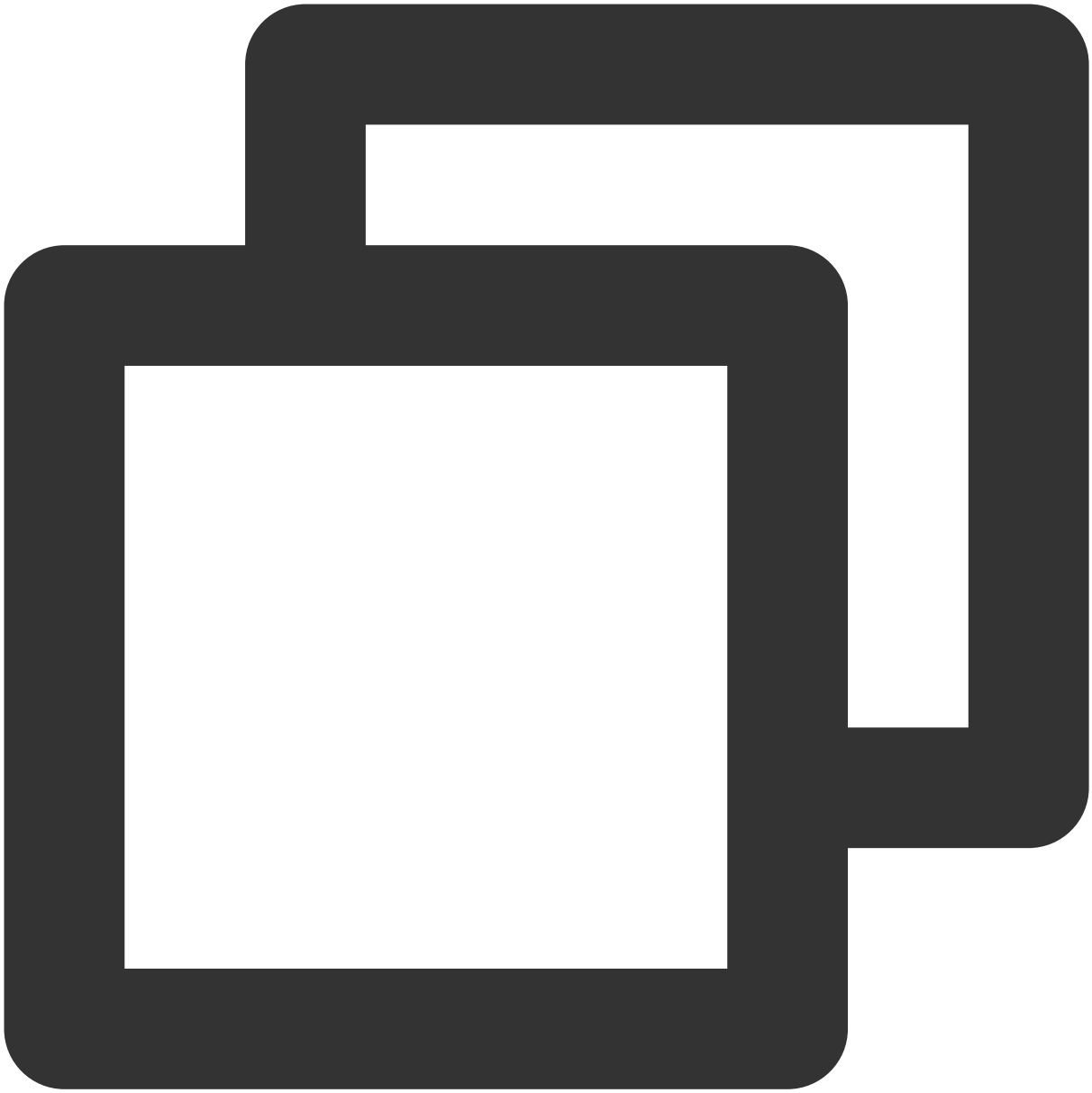
```
export KUBECONFIG=~/.kube/config
```

その中で、 `~/Downloads/cls-3jju4zdc-config` はクラスターのKubeconfigのファイルパスです。ローカルにダウンロードした後の実際のパスに置き換えてください。

2. Kubeconfigの設定が完了したら、順次以下のコマンドを実行します。contextを確認して切り替えることで、このクラスターにアクセスします。

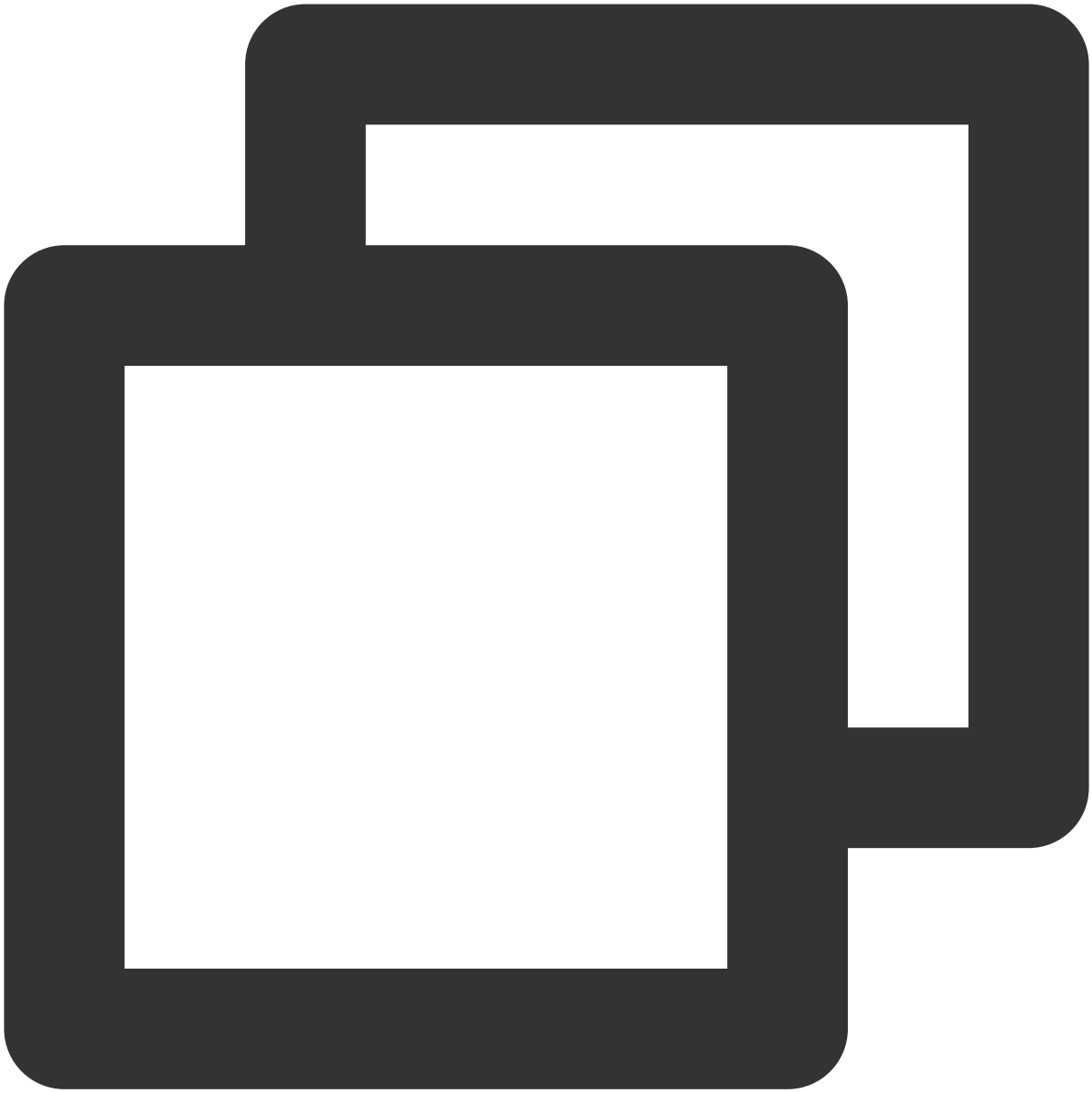


```
kubectl config get-contexts
```



```
kubectl config use-context cls-3jju4zdc-context-default
```

3. 以下のコマンドを実行し、 クラスターに正常にアクセスできるかどうかをチェックします。



```
kubectl get node
```

接続できない場合は、パブリックネットワークアクセスまたはプライベートネットワークアクセスのエントリーがオンにされているかどうかを確認し、アクセスクライアントが指定されたネットワーク環境にあることを確保してください。

関連説明

Kubectlコマンドラインの説明

KubectlはKubernetesクラスターの操作に使用されるコマンドラインツールです。ここではkubectl構文、一般的なコマンド操作をカバーし、よくある例を提供します。各コマンド（すべてのメインコマンドおよびサブコマンドを含む）に関する詳細な情報については、 kubectl参考ドキュメント を参照するか `kubectl help` コマンドを使用して詳細なヘルプを確認することができます。

イメージ

TKE-Optimizedシリーズイメージの説明

最終更新日：：2023-04-28 15:31:35

TencentOS-kernelはTencent Cloudチームによってメンテナンスされているカスタマイズカーネルです。Tencent LinuxはTencent Cloudのこのカーネルバージョンを含むパブリックイメージであり、TKEは現在このイメージをサポートしてデフォルトのオプションとしています。

Tencent Linuxパブリックイメージがリリースされる前に、イメージの安定性を向上させ、より多くの機能を提供するため、TKEチームはTKE-Optimizedシリーズのイメージを制作してメンテナンスします。現在コンソールはクラスターの新規作成によるTKE-Optimizedイメージの選択をサポートしていません。

注意

TKE-Optimizedイメージのクラスターを使用しても影響を受けず、継続して使用することができます。Tencent Linux 2.4に切り替えた場合、追加ノードにTencent Linux2.4を使用することを推奨します。既存のノードは影響を受けず、継続して使用することができます。

Centos7.6 TKE OptimizedイメージとTencent Linux 2.4イメージの使用は完全な互換性があります。

Ubuntu 18.04 TKE Optimizedイメージのユーザー空間ツールと Tencent Linuxに完全な互換性はなく、ノードの設定を変更するスクリプトは自分で新しいバージョンを適応させる必要があります。

Worker ノードの概要
ノードの追加

最終更新日：：2023-05-19 11:27:06

操作シナリオ

次の方法でクラスターにノードを追加することができます。
ノードの新規作成
既存ノードの追加

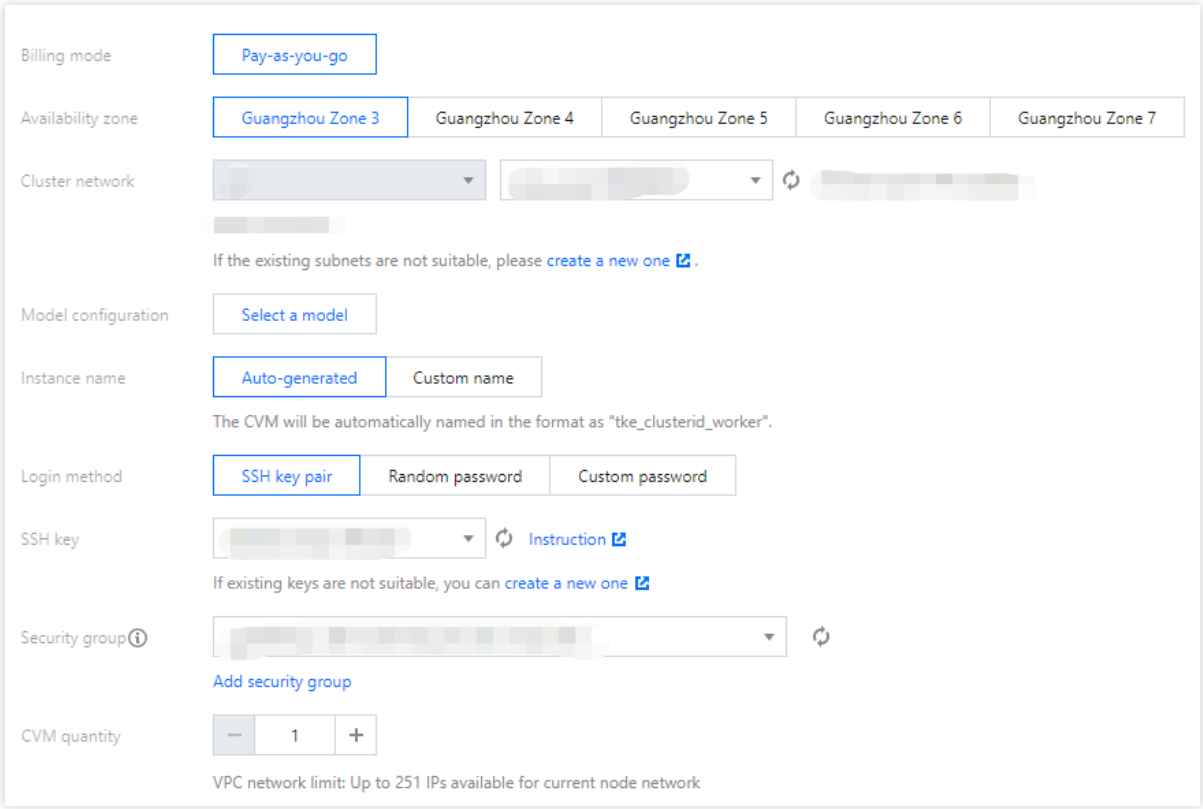
前提条件

TKEコンソールにログインしていることが必要です。

操作手順

ノードの新規作成

1. 左側ナビゲーションバーで、クラスターをクリックし、「クラスター管理」ページに進みます。
2. CVMを作成したいクラスターIDをクリックし、このクラスターの詳細ページに進みます。
3. ページ左側の**ノード管理** > **ノード**を選択し、ノードリストページに進み、**ノードの新規作成**をクリックします。
4. 「ノードの新規作成」ページで、実際のニーズに応じて関連パラメータを設定します。下図に示すとおりです。



主なパラメータの情報は次のとおりです。

課金モデル：**従量課金**の課金モデルを提供しています。詳細については、課金モデルをご参照ください。

アベイラビリティーゾーン：このパラメータはアベイラビリティーゾーン下の利用可能なサブネットのリストをフィルタリングするためだけに用いられます。

クラスターネットワーク：今回の新規作成ノードを割り当てるIPのサブネットを選択します。1回のノード作成操作では単一のサブネットのみ選択可能です。

モデルコンフィグレーション：**モデルを選択してください**をクリックし、ポップアップした「モデルコンフィグレーション」ウィンドウで次の情報を参照し、必要に応じて選択してください。

モデル：CPUコア数、メモリサイズ、インスタンスタイプによるフィルタリングをサポートしています。詳細については、インスタンス仕様をご参照ください。

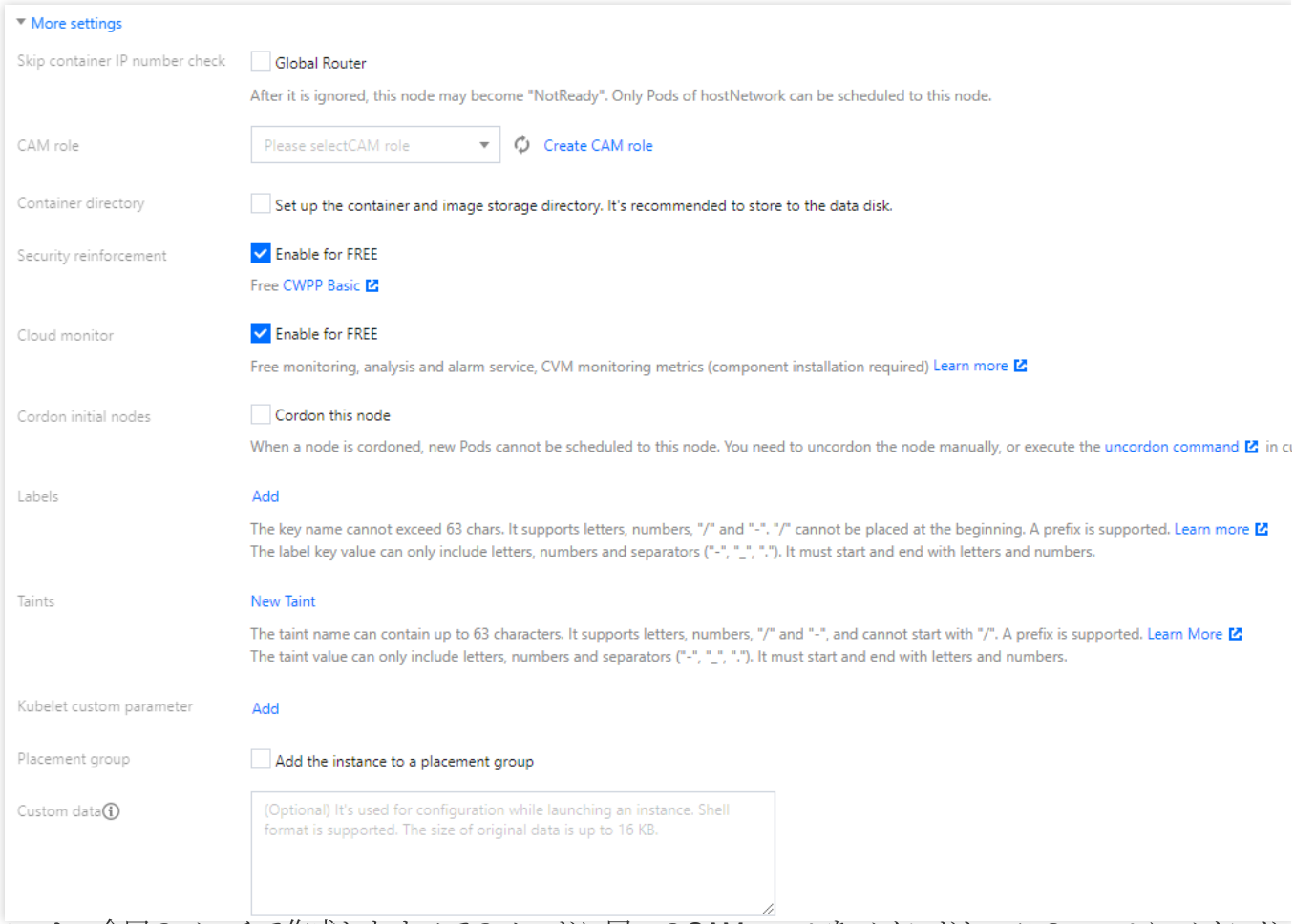
システムディスク：ストレージの制御、CVMの実行スケジューリングを行うシステムの集合です。選択したモデルで選択可能なシステムディスクのタイプを確認できます。CBSタイプをご参照の上、実際のニーズに応じて選択してください。

データディスク：すべてのユーザーデータの保存に用いられます。

インスタンス名：コンソールに表示されるCVMインスタンス名です。この属性はホスト名の命名方式の制限を受けます。次の2種類の命名方式があります。

自動命名：ホスト名が自動命名方式の場合、一括連続命名または指定パターン文字列命名をサポートします。最大で60文字を入力できます。インスタンス名はデフォルトで自動生成され、形式は `tke_クラスター id_worker` となります。

手動命名：ホスト名が手動命名方式の場合、インスタンス名はホスト名と同じになり、再設定の必要はありません。
ログイン方式：Tencent Cloudは、以下の3種類のログイン方式をご提供しています。実際の状況に応じて選択してください。
今すぐキーをバインドする：キーペアはアルゴリズムによって生成された一対のパラメータであり、一般的なパスワードと比べてより安全なCVMへのログイン方式です。詳細については、SSHキーをご参照ください。
SSHキー：この設定項目は**今すぐキーをバインドする**ログイン方式の場合にのみ表示され、ドロップダウンリストで既存のキーを選択すれば完了です。新規作成が必要な場合は、SSHキーの作成をご参照ください。
パスワードの自動生成：自動生成されたパスワードはサイト内メッセージで送信されます。
パスワードの設定：対応するパスワードを表示に従って設定してください。
セキュリティグループ：デフォルトでクラスター作成時に設定したセキュリティグループです。実際のニーズに応じて変更または追加できます。
数量：作成するインスタンス数です。実際のニーズに応じて選択してください。
5.（オプション）「ノードの新規作成」ページの**その他の設定**をクリックすると、下図のように、その他の情報を確認または設定することができます。



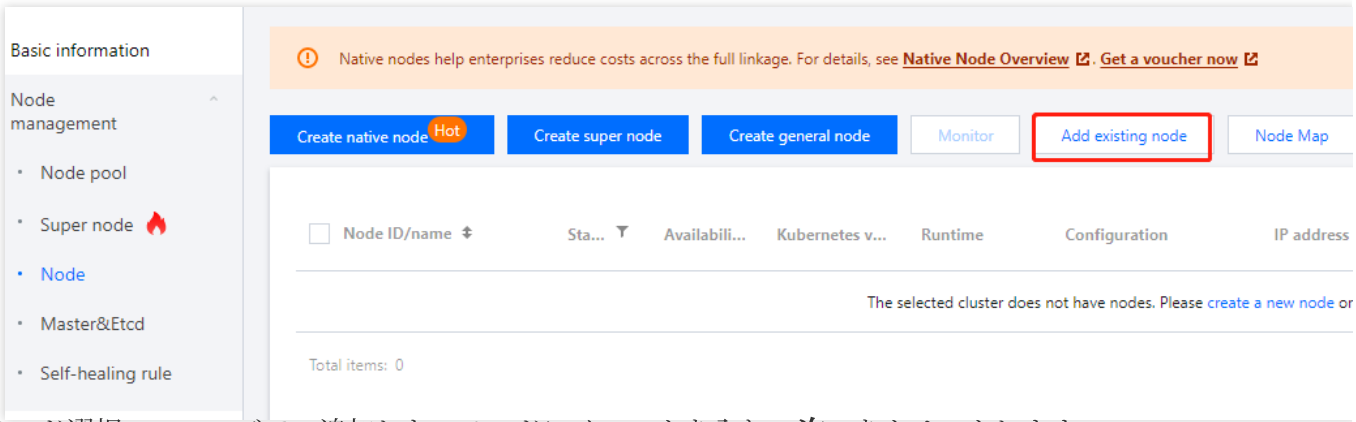
CAMロール：今回のバッチで作成したすべてのノードに同一のCAMロールをバインドし、このロールにバインドされた権限承認ポリシーをノードに付与することができます。詳細については、インスタンスロールの管理をご参照ください。

コンテナディレクトリ：チェックを入れると、コンテナとイメージストレージディレクトリを設定することができます。データディスクへの保存を推奨します（例： `/var/lib/docker` ）。
セキュリティ強化：Anti-DDoS、WAFおよびクラウドイメージホスト保護がデフォルトで無料でアクティブ化されます。詳細については、 T-Sec Cloud Workload Protection公式サイトページをご参照ください。
Tencent Cloud Observability Platform：クラウド製品の監視、分析、アラート実施がデフォルトで無料でアクティブ化され、コンポーネントをインストールしてホスト監視指標を取得できます。詳細については、Tencent Cloud Observability Platformをご参照ください。
初期ノードのコルドン：「コルドンを有効化する」にチェックを入れると、新しいPodをこのノードにスケジューリングできなくなり、ノードのコルドンを手動でキャンセルするか、またはカスタムデータ内でコルドンキャンセルコマンドを実行する必要があります。必要に応じて設定してください。
Label：**Labelの追加**をクリックすると、Labelをカスタム設定できます。その後、Labelによるノードのフィルタリング、管理を行うことができるようになります。
カスタムデータ：カスタムデータを指定してノードを設定します。これはすなわち、ノードの起動後に設定を実行するスクリプトです。スクリプトが再入可能であり、ロジックが再試行可能であることを確実にする必要があります。スクリプトとそれが生成するログファイルは、ノードの `/usr/local/qcloud/tke/userscript` パスで確認することができます。

既存ノードの追加

注意

現在は同じVPCにあるCVMの追加のみサポートしています。
パブリックゲートウェイCVMをクラスターに追加しないでください。このタイプのCVMを再インストールしてクラスターに追加した後にDNS異常が発生した場合、このノードが利用できなくなります。
1. 左側ナビゲーションバーで、クラスターをクリックし、「クラスター管理」ページに進みます。
2. 追加したい既存ノードのクラスターIDをクリックし、このクラスターの詳細ページに進みます。
3. **ノード管理** > **ノード**を選択し、**既存ノードの追加**をクリックします。下図に示すとおりです。



4. 「ノード選択」のページで、追加したいノードにチェックを入れ、**次へ**をクリックします。
5. 「CVMの設定」ページで、クラスターに追加したいCVMを設定します。主要パラメータの情報は次のとおりです。

データディスクマウント：フォーマットとマウントの関連設定：デバイス名を入力し、システムおよびマウントポイントをフォーマットする必要があります。

注意

High IO型、ハイパフォーマンスHCCタイプのモデルにNVMeタイプのデータディスクをマウントしたい場合は、他のモデルと一緒に操作せず、データディスクにファイルシステムボリュームラベルを設定する方法で、単独でのクラスター追加を行うことをお勧めします。

重要データは事前にバックアップしておきます。すでにご自身でディスクをフォーマットしている場合はシステムのフォーマットを選択する必要はなく、マウントポイントのみを入力します。

入力したフォーマットとマウントの設定は今回のバッチで追加したすべてのノードに対して有効になります。入力したデバイス名（/dev/vdbなど）が想定と同じかどうかを確認してください（CBSにホットスワップなどの操作を行った場合、デバイス名が変わる可能性があります）。

ディスクに対してパーティション/LVMを行っている場合は、デバイス名にはパーティション名/LVM名を記入し、対応するフォーマット・マウントパラメータを設定します。

誤ったデバイス名を入力した場合、システムがエラーを表示し、ノードの初期化フローを終了します。

入力したマウントポイントが存在しない場合は、システムが対応するするディレクトリを作成し、エラーは表示されません。

チェックを入れない：データディスクマウントオプションを設定しない場合は、手動またはスクリプトを使用してマウントすることができます。

チェックを入れる：デバイス名を入力し、システム（フォーマットを選択しなくても可）、マウントポイントをフォーマットする必要があります。ブロックデバイス/dev/vdbをext4にフォーマットし、ディレクトリ/var/lib/dockerにマウントしたい場合は、デバイス名を `/dev/vdb` 、フォーマットシステムを `ext4` 、マウントポイントを `/var/lib/docker` にそれぞれ設定することができます。

コンテナディレクトリ：コンテナとイメージを保存するディレクトリを設定します。データディスクへの保存を推奨します。

OS：OSはクラスターレベルであり、クラスター詳細ページで変更できます。変更後にノードを新規追加または再インストールする際は新しいOSを使用します。

ログイン方式：

パスワードの設定：対応するパスワードを表示に従って設定してください。

今すぐキーをバインドする：キーペアはアルゴリズムによって生成された一対のパラメータであり、一般的なパスワードと比べてより安全なCVMへのログイン方式です。具体的な詳細については、SSHキーをご参照ください。

パスワードの自動生成：自動生成されるパスワードです。このパスワードはサイト内メッセージで送信されます。

セキュリティグループ：CVMのネットワークアクセス制御の設定に用いられます。実際のニーズに応じて選択してください。また、**セキュリティグループの新規作成**をクリックし、他のポートを許可することもできます。

6. **完了**をクリックします。

通常ノード管理
ノードプールの概要

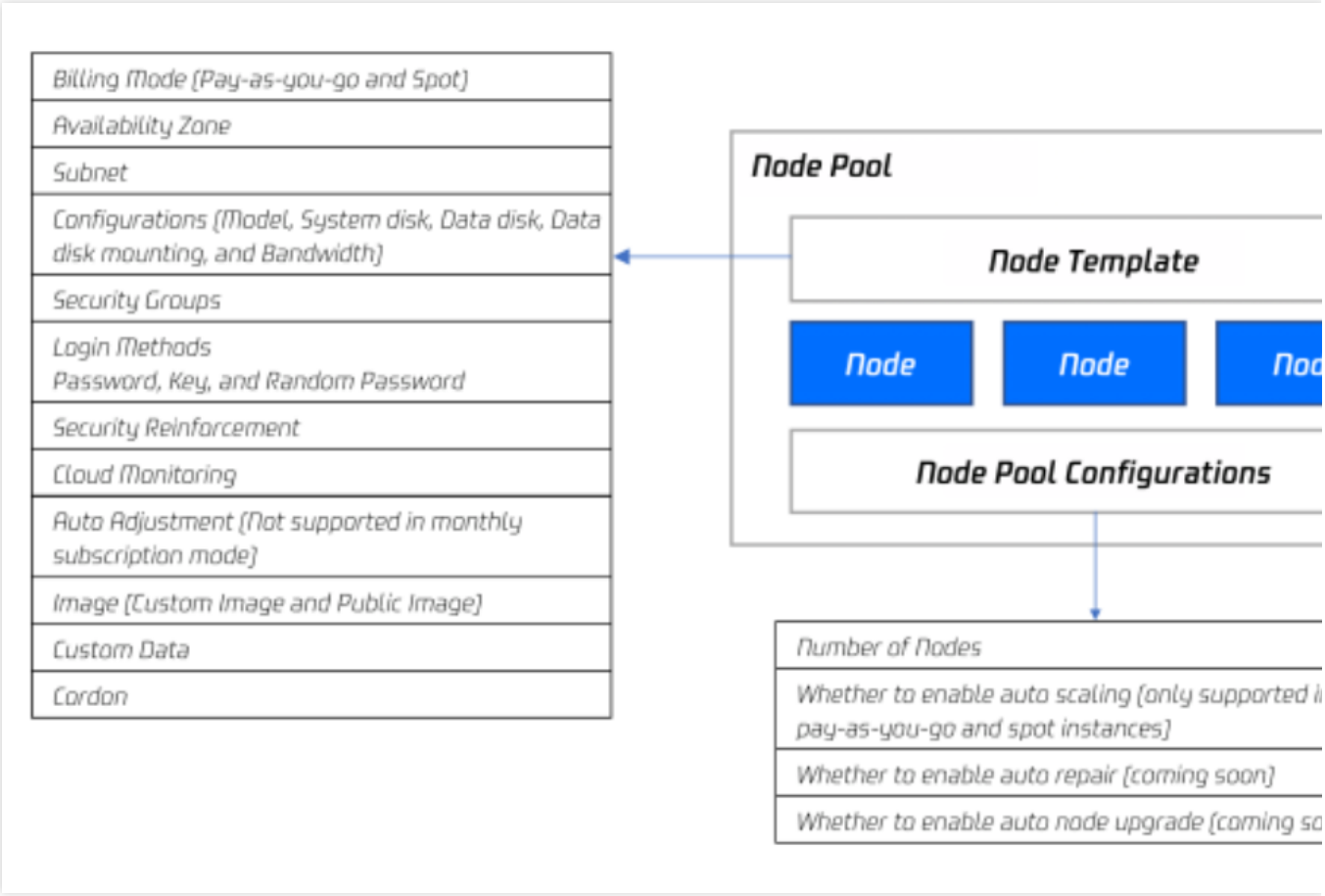
最終更新日：：2023-05-06 19:41:07

概要

Kubernetesクラスター内のノードの管理効率を上げるため、Tencent Kubernetes Engine（TKE）はノードプールのコンセプトを導入しました。ノードプールの基本機能を利用することで、ノードの作成、管理、破棄が、より簡単に素早くなり、ノードのダイナミックスケーリングを実現しました。
人的コスト削減のため、クラスターでリソース不足によりインスタンス（Pod）のスケジューリングができない場合は、自動でスケーリングをトリガーします。
リソースコスト削減のため、ノードアイドルなどのスケーリング条件が揃った場合には、スケーリングを自動でトリガーします。

製品アーキテクチャ

ノードプール全体のアーキテクチャ図は次のとおりです。



通常の状況では、ノードプール内のノードはすべて次のような同一属性を有しています。
ノードオペレーティングシステム。
課金タイプ（現在は従量課金とスポットインスタンスをサポートしています）。
CPU/メモリ/GPU。
ノードKubernetesインスタンスの起動パラメータ。
ノードのカスタマイズ起動スクリプト。
ノードのKubernetes LabelおよびTaintの設定。
この他、TKEはノードプールについて、同時に次の機能を展開します。
CRDを用いたノードプールの管理をサポートします。
ノードプールレベルの各ノードのPod数の上限。
ノードプールレベルの自動修復および自動アップグレード。

ユースケース

業務で大規模なクラスターの使用が必要な場合は、大規模なクラスターの使いやすさを高めるため、ノードプールを利用したノード管理をお勧めします。次の表では、複数の大規模クラスターの管理ケースについて説明します。ノードプールが各ケースで発揮するそれぞれの機能を表示します。

シーン	機能
クラスターには多くの異種ノード（モデルコンフィグレーションが異なります）が存在しますが	ノードプールを介してノードのグループ管理をルール化できます。
クラスターは頻繁にノードのスケーリングを行う必要がありますが	ノードプールによって運用効率を高め、人的コストを削減することができます。
クラスター内のアプリケーションプログラムのスケジューリングルールは複雑ですが	ノードプールのタグにより指定業務のスケジューリングルールを素早く指定できます。
クラスター内ノードの日常のメンテナンス	ノードプールによりKubernetesバージョンアップ、Dockerバージョンアップの操作が簡単にできます。

関連概念

TKEの自動スケーリングは、Tencent Cloud自動スケーリング（AutoScaling）および Kubernetesコミュニティの cluster-autoscaler によって実現します。関連コンセプトの説明です。
CA：cluster-autoscalerは、コミュニティオープンソースコンポーネントです。主にクラスターの自動スケーリングを行います。
AS：AutoScalingは、Tencent Cloudの自動スケーリングサービスです。
ASG：AutoScaling Groupは、いずれかの具体的なノードプール（ノードプールは自動スケーリングサービスが提供するスケーリンググループに依存します。ノードプール1つにつき1つのスケーリンググループに対応しますので、ノードプールのみに注目してください）です。
ASA：AS activityは、いずれかの回のスケーリングイベントです。
ASC：AS configは、ASの起動設定、つまりノードテンプレートです。

ノードプール内のノードの種類

異なるシーンでのニーズに応えるため、ノードプール内のノードは2つのタイプに分けられます。
説明：
特殊なシーンがない場合は、既存のノード機能の追加使用はお勧めしません。例えば、ノードの権限を新しく作成せず、既存のノードを介してのみクラスターをスケールアウトする場合は、既存のノードの一部パラメータを追加することで、あなたが定義したノードのテンプレートとの不一致がおこり、自動スケーリングに参加できなくなる可能性があります。

ノードのタイプ	ノードのソース	自動スケーリングをサポートしているかどうか	ノードプールからの削除方式	ノード数が**数量の調整**の影響を受けるかどうか

スケーリンググループ内のノード	自動スケールアウトあるいは手動での数量調整	はい	自動スケールインあるいは手動での数量調整	はい
スケーリンググループ外のノード	ユーザーが手動でノードプールに参加	いいえ	ユーザーが手動で削除	いいえ

ノードプールの自動スケーリングの原理

ノードプールの自動スケーリング機能をご利用になる前に、原理についての次の説明にしっかり目を通してください。

ノードプールの自動スケールアウトの原理

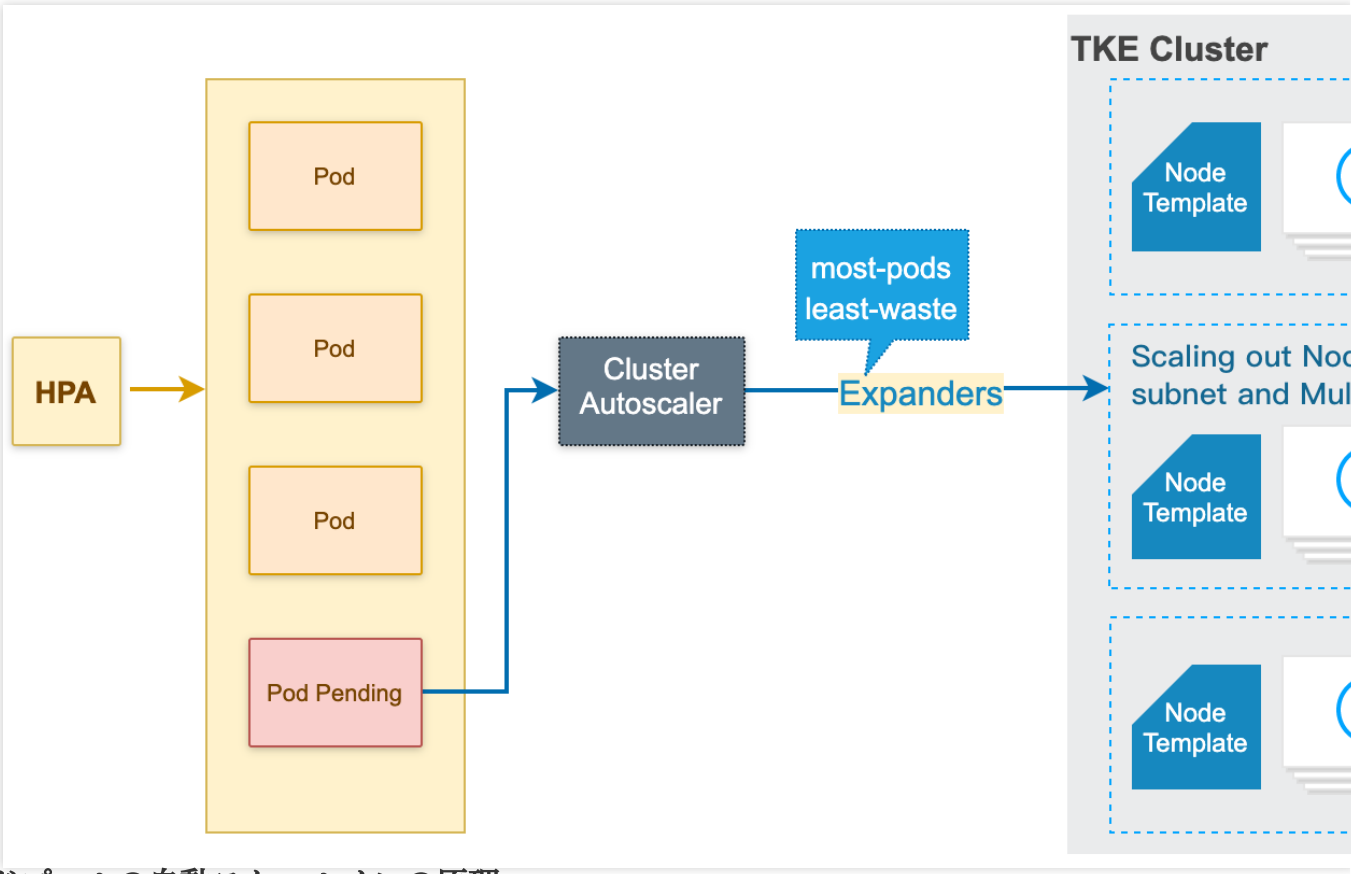
1. クラスターがリソース不足（クラスターの計算/ストレージ/ネットワークなどリソースがPodのrequest/親和性のルールを満たしていない）の場合は、CA（Cluster Autoscaler）が、スケジューリングができずにPendingしているPodをモニタリングします。
2. CAは、各ノードプールのノードテンプレートに基づいてスケジューリング判断を行い、適切なノードテンプレートを選択します。
3. 複数のテンプレートが適していた場合、つまり複数の拡張可能なノードプールが選択肢となった場合、CAは **expanders** を呼び出してその中から最適なテンプレートを選択し、対応するノードプールに対してスケールアウトを行います。
4. 指定のノードプールに対してスケールアウト（複数のサブネット/モデルポリシーに基づく）を行い、2種類のリトライポリシー（ノードプールの作成で設定できます）を提供します。スケールアウトに失敗した場合は、ご自身で設定されたリトライポリシーに基づいてリトライします。

説明

特定のノードプールにスケールアウトを行う場合は、ご自身で作成されたノードプールのサブネットおよびその後で設定した複数のモデルコンフィグレーションでスケールアウトを行うことをお勧めします。通常の状況では、**先に複数モデルのポリシーを保証し、次にマルチアベイラビリティーゾーン/サブネットのポリシーを保証します。**

例えば複数のモデルA、B、複数のサブネット1、2、3を設定すると、A1、A2、A3、B1、B2、B3に基づいてトライします。A1が完売している場合は、B1ではなくA2でトライします。

ノードプールの自動スケールアウトの原理は、次の図のとおりです。



ノードプールの自動スケールインの原理

1. CA（Cluster Autoscaler）がモニタリングした分配率（つまりRequest値は、CPU分配率とMEM分配率の最大値を取ります）は、設定のノードを下回ります。分配率を計算する際に、Pod占有リソースにDaemonsetタイプが計算されないよう設定できます。

2. CAは、クラスターの状態がスケールインにトリガーできるかどうかを判断します。それには次の要件を満たす必要があります。

ノードのアイドル時間についての要求（デフォルトでは10分）。

クラスターのスケールアウトのバッファ時間についての要求（デフォルトでは10分）。

3. CAはこのノードがスケールインの条件に合致しているかどうかを判断します。必要に応じて次の**スケールインしない条件**（条件を満たすノードはCAによってスケールインされません）に従って設定できます。

ローカルストレージのノードを含みます。

Kube-system namespaceを含み、DaemonSet管理でないPodのノード。

説明

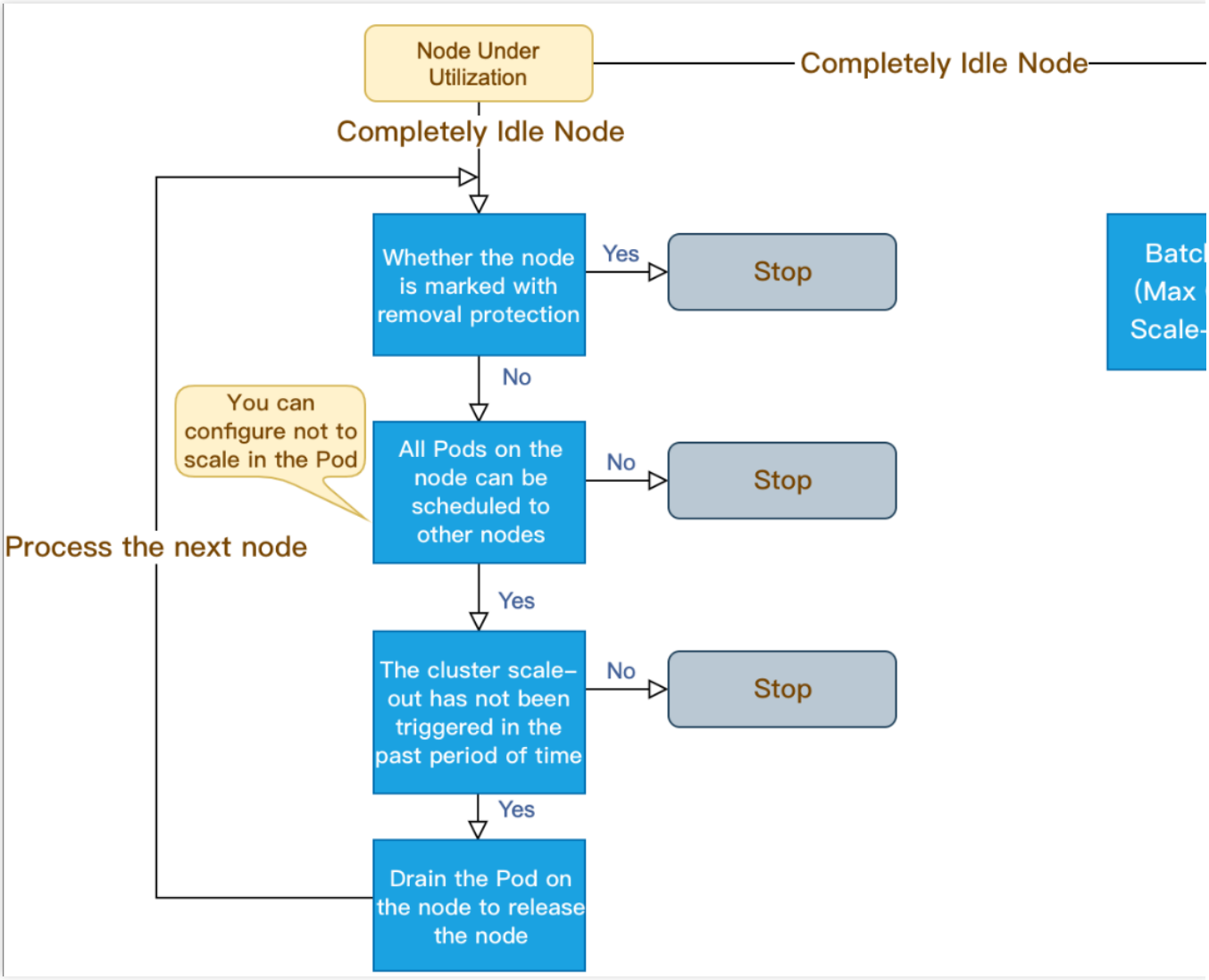
上記のスケールイン条件はクラスター次元で有効化します。さらに細かい粒度でノードを守りスケールインを避ける場合は、**スケールインの保護**機能が利用できます。

4. CAはノード上のPodをドレイン後に、ノードをリリース/シャットダウンします。

ノードが完全にアイドル状態になると、スケールインの同時実行が可能になります（スケールインの最大同時実行数が設定できます）。

完全なアイドル状態でないノードは個別にスケールインします。

ノードプールの自動スケールインの原理は、次の図のとおりです。



機能点および注意事項

機能点	機能説明	注意事項
ノードプールの作成	ノードプールの追加	単一クラスターではノードプールが20を超えることをお勧めしません。
ノードプールの削除	ノードプールを削除する際、ノードプール内のノードを破棄するかどうかの選択ができます。 ノードを削除した場合もしない場合も、ノードはクラスター内では保持されません。	ノードプールを削除する際、破棄するノードを選択するとノードは保持されなくなります。後から新しいノードの使用が必要になった場合には、再度作成ができます。
ノードプールの自動スケーリングの開始	自動スケーリングをオンにすると、ノードプール内のノード数はクラスターの負荷状況に従って自動調整されます。	スケーリンググループコンソールでは、自動スケーリングのオン/オフを行わないでください。

ノードプールの自動スケーリングの終了	自動スケーリングをオフにすると、ノードプール内のノード数はクラスターの負荷状況に従わずに自動調整されます。	
ノードプールのノード数の調整	ノードプール内のノード数の直接調整をサポートします。 ノード数を減らす際は、ノード退出ポリシー（デフォルトでは最も古いノードを退出）に従い、スケーリンググループ内からノードをスケールインします。このスケールイン動作はスケーリンググループにより行われるため、TKEは具体的なスケールインノードを関知できず、事前に動作のドレイン/ブロックができませんのでご注意ください。	自動スケーリング開始後は、ノードプールサイズの手動調整をお勧めしません。スケーリンググループコンソールでは、スケーリンググループの希望インスタンス数の直接調整をしないでください。特殊な状況でないかぎり、ノードプールでは自動スケールインを使用し手動でのスケールインはしないでください。自動スケールインでは、まずノードマーカーをスケジューリングできなくしてから、ノード上のすべてのPodをドレインあるいは削除し、その後ノードの再リリースを行います。
ノードプール設定の調整	ノードプールの名称、オペレーティングシステム（OS）、スケーリンググループのノード数範囲、Kubernetes labelおよびTaintの変更が可能です。	LabelおよびTaintを変更すると、ノードプール内のノードすべてが有効化され、Podが再度スケジューリングされる可能性がありますので、慎重に変更してください。
既存ノードの追加	クラスターに属さないインスタンスをノードプールに追加することができます。その要件は次のとおりです。 インスタンスとクラスターが同一のVirtual Private Cloudに属していること。 インスタンスがその他クラスターで使用されておらず、かつインスタンスとノードプールの設定が同一モデル、同一課金モデルであること。 クラスター内のどのノードプールにも属していないノードを追加できます。ノードインスタンスとノードプールの設定は同一モデル、同一課金モデルであること。	特殊な状況でないかぎり、既存ノードの追加はお勧めしません。ノードプールを直接新規作成されることを推奨します。
ノードプール内ノードの削除	ノードプール内の任意のノードの削除をサポートします。削除時、ノードをクラスター内に保留するかどうかの選択ができます。	スケーリンググループコンソールでは、スケーリンググループ内にノードを追加しないでください。データの不一致により重大な影響が生じる恐れがあります。
オリジナルスケーリンググ	既存のスケーリンググループのノードプールへの切り替えをサポートします。変換後は、ノードプールはオリジナルスケーリン	不可逆的な操作ですので、ノードプールの機能を理解してから切り替えを行ってください。

ループのノードプールへの変換	グループの機能を完全に引き継ぎ、そのスケジューリンググループの情報は表示されなくなります。 クラスターメモリ量のすべてのスケーリンググループの切り替えが完了した後は、スケーリンググループにポータルを提供しません。	

関連する操作

TKEコンソール にログインし、次のドキュメントを参考に、ノードプールについての操作を行ってください。

ノードプールの作成

ノードプールの確認

ノードプールの調整

ノードプールの削除

ノードプールの作成

最終更新日：：2023-06-01 15:27:44

操作シナリオ

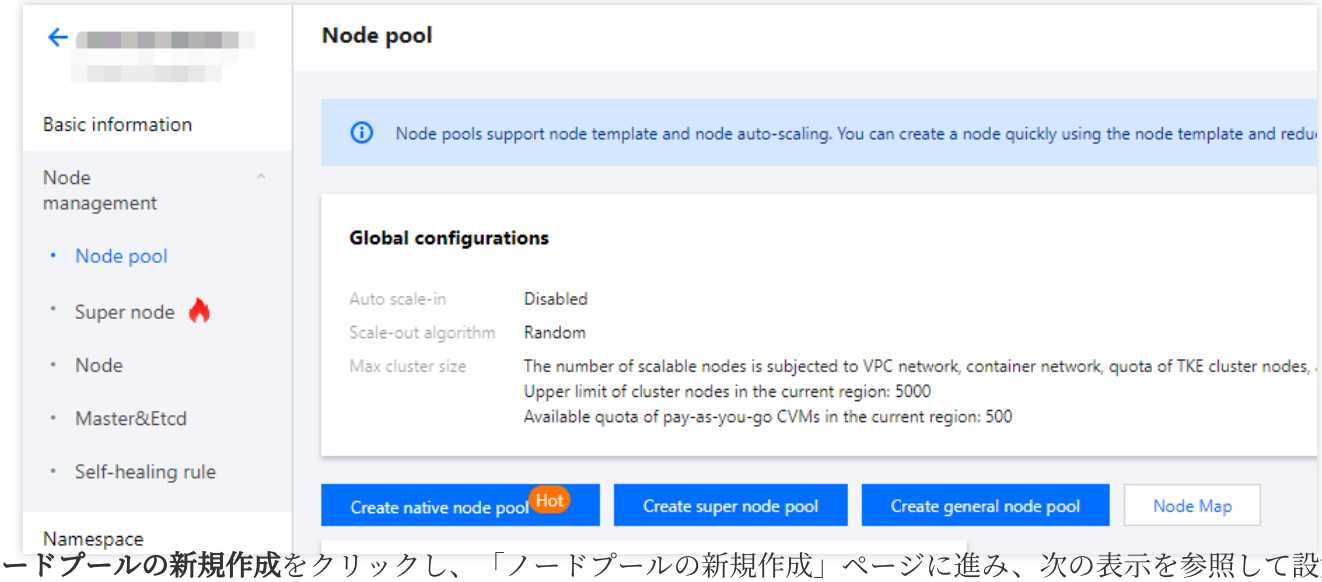
ここでは、TKEコンソールによってクラスター内にノードプールを作成する方法についてご説明し、ノードプールの確認、管理、削除などの関連操作についても記述します。

前提条件

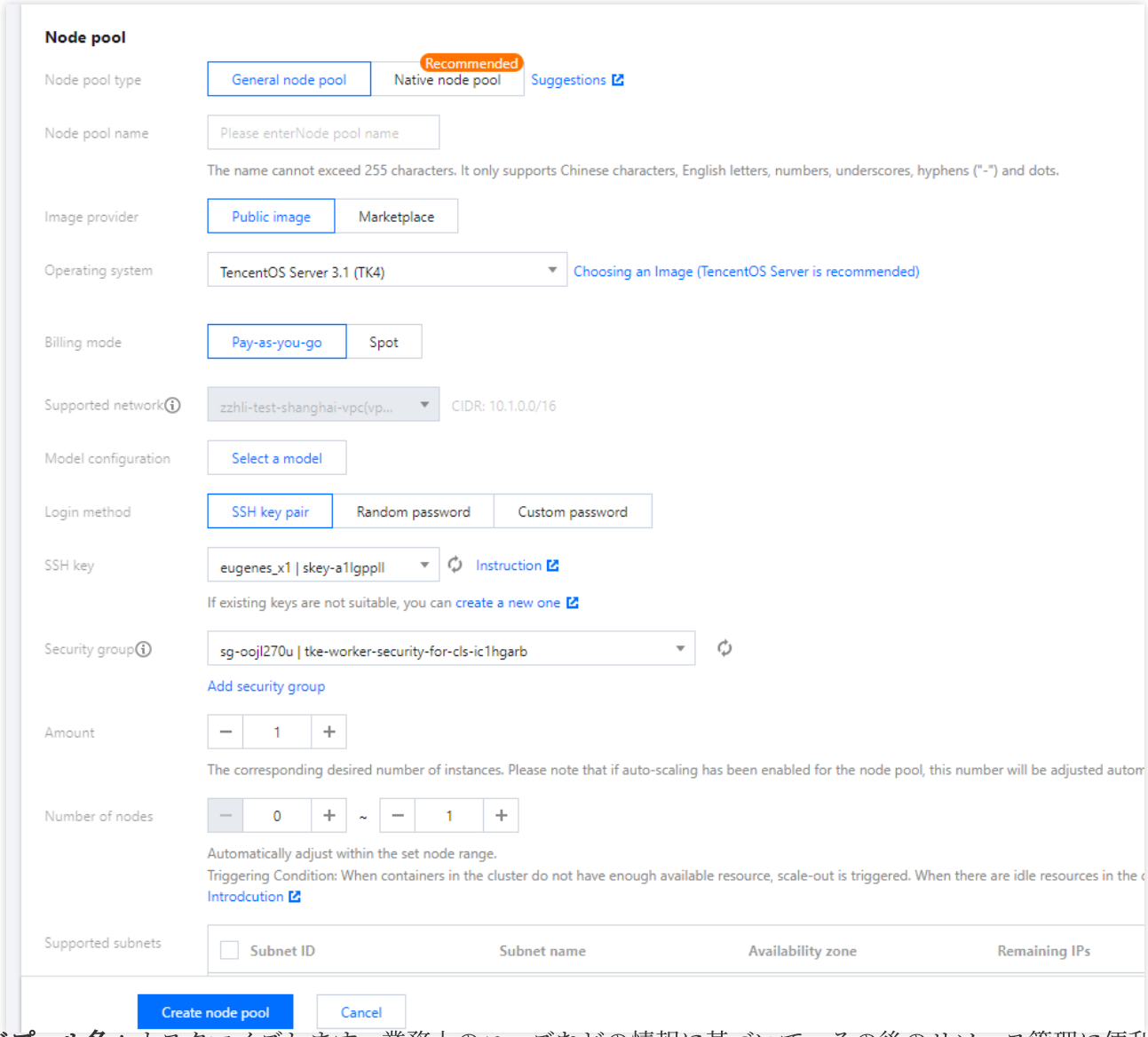
ノードプールの基本概念を理解している必要があります。
クラスターの作成が完了している必要があります。

操作手順

1. TKEコンソールにログインし、左側ナビゲーションバーの**クラスター**をクリックします。
2. 「クラスター管理」リストページでターゲットのクラスターIDを選択し、このクラスターの詳細ページに進みます。
3. 左側メニューバーの**ノード管理** > **ノードプール**を選択し、「ノードプールリスト」ページに進みます。下図に示すとおりです。



4. **ノードプールの新規作成**をクリックし、「ノードプールの新規作成」ページに進み、次の表示を参照して設定します。下図に示すとおりです。



ノードプール名：カスタマイズします。業務上のニーズなどの情報に基づいて、その後のリソース管理に便利なように命名することができます。

OS：実際のニーズに応じて選択します。このOSはノードプールの次元で有効になり、変更が可能です。変更後の新しいOSはノードプール内の増分ノードにのみ有効であり、既存ノードには影響しません。

課金モデル：**従量課金**、**スポットインスタンス課金**の2種類の課金モデルを提供しています。実際のニーズに応じて選択してください。詳細については、課金モデルの比較をご参照ください。

サポートネットワーク：システムがノードネットワークアドレスの範囲内のIPアドレスをクラスター内のホストに割り当てます。

注意

このオプションはクラスターの次元の設定項目のため、変更はできません。

モデルコンフィグレーション：**モデルを選択してください**をクリックし、ポップアップした「モデルコンフィグレーション」ウィンドウで次の情報を参照し、必要に応じて選択してください。

アベイラビリティーゾーン：起動設定にはアベイラビリティーゾーンの情報は含まれません。このオプションは選択したアベイラビリティーゾーン下で利用可能なインスタンスタイプをフィルタリングするためだけに用いら

れます。
モデル：CPUコア数、メモリサイズ、インスタンスタイプによるフィルタリングをサポートしています。詳細については、インスタンス仕様をご参照ください。
システムディスク：ストレージの制御、CVMの実行スケジューリングを行うシステムの集合です。選択したモデルで選択可能なシステムディスクのタイプを確認できます。CBSタイプをご参照の上、実際のニーズに応じて選択してください。
データディスク：すべてのユーザーデータの保存に用いられます。次のガイドに従って設定してください。モデルごとに対応するデータディスクの設定が異なるため、次の表を参照して設定を行ってください。

モデル	データディスクの設定
標準型、メモリ型、コンピューティング型、GPUモデル	デフォルトではチェックが入っていません。チェックを入れる場合は、実際の状況に応じてCBSの設定およびフォーマット設定を行ってください。
High IO型、ビッグデータ型	デフォルトでチェックが入っており、なおかつ変更できません。デフォルトで購入したローカルディスクに対し、カスタマイズしたフォーマット設定を行うことができます。
バッチ型	デフォルトでチェックが入っており、なおかつチェックを外すことができます。チェックを入れる場合は購入したデフォルトのローカルディスクのみサポートし、デフォルトのローカルディスクに対してカスタマイズしたフォーマット設定を行うことができます。

データディスクの追加（オプション）：**データディスクの追加**をクリックし、上の表を参照して設定します。
パブリックネットワークブロードバンド：デフォルトでは**無料のパブリックIPを割り当てる**にチェックが入っており、システムがパブリックIPを無料で割り当てます。トラフィック課金、帯域幅課金という2種類のモデルをサポートしています。パブリックネットワーク課金モデルをご参照の上、実際の状況に応じて選択し、ネットワーク速度のカスタム設定を行ってください。
ログイン方式：Tencent Cloudは、以下の3種類のログイン方式をご提供しています。実際の状況に応じて選択してください。
今すぐキーをバインドする：キーペアはアルゴリズムによって生成された一対のパラメータであり、一般的なパスワードと比べてより安全なCVMへのログイン方式です。詳細については、SSHキーをご参照ください。
SSHキー：この設定項目は**今すぐキーをバインドする**ログイン方式の場合にのみ表示され、ドロップダウンリストで既存のキーを選択すれば完了です。新規作成が必要な場合は、SSHキーの作成をご参照ください。
パスワードの自動生成：自動生成されたパスワードはサイト内メッセージで送信されます。
パスワードの設定：対応するパスワードを表示に従って設定してください。
セキュリティグループ：デフォルトでクラスター作成時に設定したセキュリティグループです。実際のニーズに応じて変更または追加できます。
数量：希望するインスタンス数に対応します。実際のニーズに応じて選択してください。
注意

ノードプールで自動スケーリングを有効にしている場合、この数量はクラスターの負荷に伴って自動調整されます。

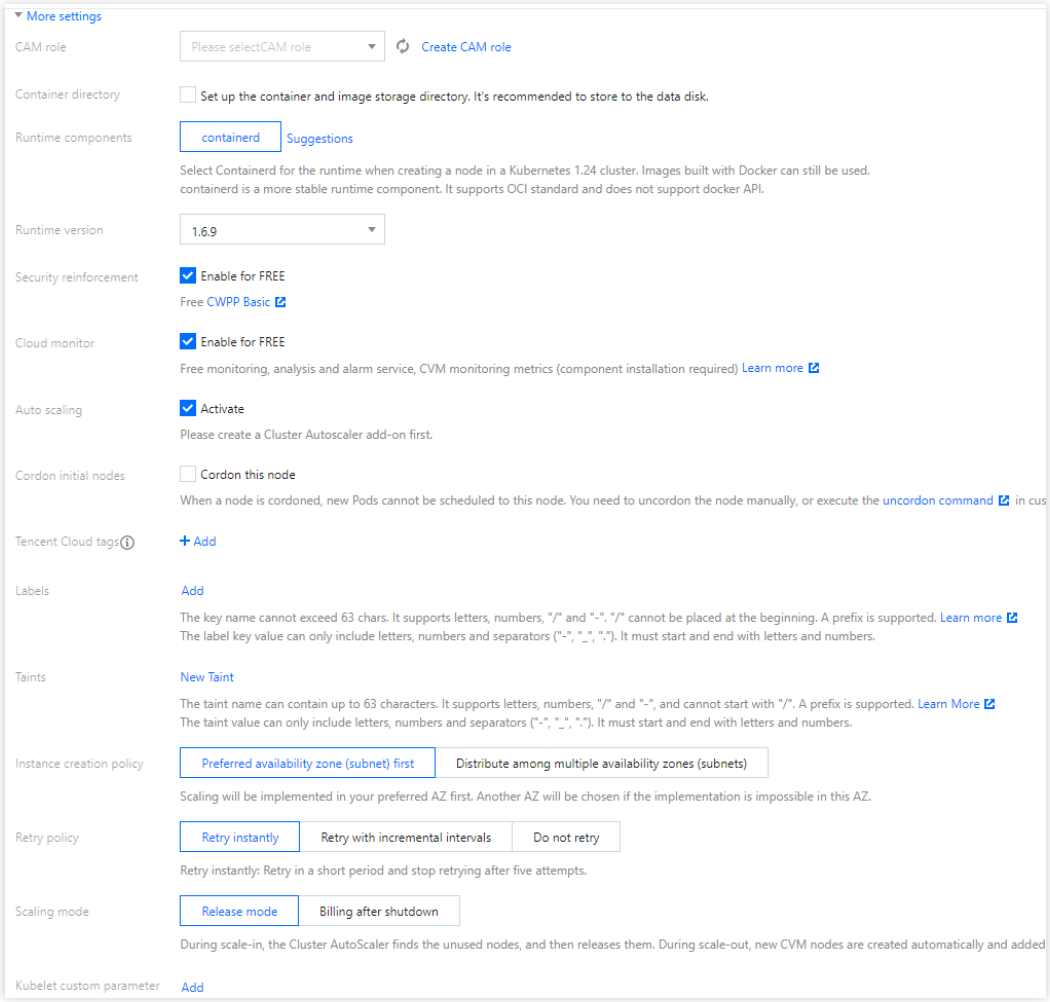
ノード数の範囲：ノード数は設定されたノードの範囲内で自動的に調整され、設定された範囲を超えることはありません。

サポートサブネット：実際のニーズに応じて、適切で利用可能なサブネットを選択してください。

説明

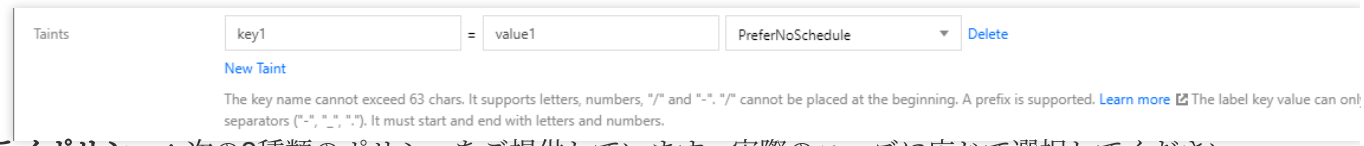
ノードプールのデフォルトのマルチサブネット拡張ポリシーは次のとおりです。マルチサブネットを設定し、ノードプールを拡張する場合（手動スケーリングまたは自動スケーリング）、サブネットリストの順序を優先順位としてノードの作成を試みます。優先順位が最も高いサブネットが作成に成功できた場合、常にそのサブネットで作成されます。

5.（オプション）**その他の設定**をクリックすると、下図のように、その他の情報を確認または設定することができます。



CAMロール：ノードプールのすべてのノードに同一のCAMロールをバインドすることで、このロールにバインドされた権限承認ポリシーをノードに付与することができます。詳細については、インスタンスロールの管理をご参照ください。

コンテナディレクトリ：チェックを入れると、コンテナとイメージストレージディレクトリを設定することができます。データディスクへの保存を推奨します（例： `/var/lib/docker` ）。
セキュリティ強化：Anti-DDoS、WAFおよびクラウドイメージホスト保護がデフォルトで無料でアクティブ化されます。詳細については、 T-Sec Cloud Workload Protection公式サイトページをご参照ください。
Tencent Cloud Observability Platform：クラウド製品の監視、分析、アラート実施がデフォルトで無料でアクティブ化され、コンポーネントをインストールしてホスト監視指標を取得できます。詳細については、Tencent Cloud Observability Platform（TCOP）をご参照ください。
自動スケーリング：デフォルトでは**有効**にチェックが入っています。
初期ノードのコルドン：**コルドンを有効化する**にチェックを入れると、新しいPodをこのノードにスケジューリングできなくなり、ノードのコルドンを手動でキャンセルするか、またはカスタムデータ内でコルドンキャンセルコマンドを実行する必要があります。必要に応じて設定してください。
Label：**Labelの追加**をクリックすると、Labelをカスタム設定できます。このノードプール下に作成されたノードにはここで設定したLabelが自動的に追加され、その後、Labelによるノードのフィルタリング、管理を行うことができるようになります。
Taints：ノードの属性であり、一般的に `Tolerations` と組み合わせて使用します。ここではノードプール下のすべてのノードにTaintsを設定することができ、条件に合致しないPodをこれらのノード上にスケジューリングしないようにし、かつこれらのノード上にすでに存在する、条件に合致しないPodがドレインされるようにすることができます。
説明
Taintsの内容は一般的に、 `key` 、 `value` 、 `effect` の3つの要素で構成されます。このうち `effect` が取れる値には通常、次の3種類が含まれます。
PreferNoSchedule： 非強制的条件であり、容認できないtaintが設定されているノード上へのPodのスケジューリングをできる限り避けます。
NoSchedule： ノード上にtaintが存在する場合、対応する容認がないPodは決してスケジューリングされません。
NoExecute：ノード上にtaintが存在する場合、対応する容認がないPodがそのノード上にスケジューリングされないだけでなく、そのノード上にすでに存在するPodもドレインされます。
Taints `key1=value1:PreferNoSchedule` を設定した場合の例では、コンソールの設定は下図のようになります。



リトライポリシー：次の2種類のポリシーをご提供しています。実際のニーズに応じて選択してください。
クイックリトライ：すぐにリトライを行い、比較的短時間内で迅速にリトライします。一定の回数（5回）を超えて連続で失敗すると、それ以上リトライを行いません。
間隔漸増リトライ：間隔漸増リトライでは、連続失敗回数が増えるにつれてリトライ間隔が徐々に長くなります。リトライ間隔は秒レベルから1日までと、それぞれ異なります。

スケーリングモード：次の2種類のスケーリングモードをご提供しています。実際のニーズに応じて選択してください。
リリースモード：スケールインの際はCluster AutoScalerが判断したアイドルノードを自動的に解放し、スケールアウトの際は新しいノードを自動的に作成してスケーリンググループに追加します。
シャットダウンモード： スケールアウトの際はシャットダウン済みのノードに対して優先的に起動操作を行い、それでもノード数が要件に満たない場合は新しいノードを作成します。スケールインの際はアイドルノードをシャットダウンし、ノードが停止済みインスタンスの非課金化をサポートしている場合はモデルの料金がかからなくなります。詳細については、停止済み従量課金インスタンスの非課金化の説明をご参照ください。それ以外のノードはシャットダウン後も引き続き料金がかかります。
カスタムデータ：カスタムデータを指定してノードを設定します。これはすなわち、ノードの起動後に設定を実行するスクリプトです。スクリプトが再入可能であり、ロジックが再試行可能であることを確実にする必要があります。スクリプトとそれが生成するログファイルは、ノードの `/usr/local/qcloud/tke/userscript` パスで確認することができます。
5. **ノードプールの作成**をクリックすればノードプールが作成されます。

関連操作

ノードプールの作成が完了した後、次の操作ガイドを参照して、その後のノードプールの管理を行うことができます。
ノードプールの確認
ノードプールの調整
ノードプールの削除

ネイティブノード管理
ネイティブノードの概要

最終更新日：：2023-05-08 18:28:50

ネイティブノードとは何ですか。

ネイティブノードは、Tencent Cloud のTKE（Tencent Kubernetes Engine）チームがKubernetes環境に向けて打ち出した、まったく新しいノードのタイプです。Tencent Cloudの膨大なコアとなるコンテナ運用保守の技術的蓄積を拠り所として、ネイティブ化した安定性かつ迅速なレスポンスのK8sノードの管理機能をユーザーに提供します。

製品の優位性

FinOpsの理念を搭載し、クラウドにおけるリソースのコストの最適化をサポートします。

HouseKeeperを搭載し、リソーステーブルを可視化することで、ノードリソースの利用率を向上させ、クラウドのコスト削減と効率アップの実現をサポートします。
ロードにおいてインテリジェントにRequestを推奨し、リソースのアイドルタイムを減少させます。
専有の動的スケジューリング機能を提供しており、以下の特徴があります。
Cloud Load Balancer：ノードの実態とロードの履歴状況に基づき、ノードのリソースのロードをよりバランシングします。
ボックスの拡張：ノードCPU/メモリリソースを拡大し、量をスケジューリングすることが可能で、ノードのボックス率を100%以上に向上させます。
業務の整理：目標利用率を設定することで、ノードの継続的なスケジューリングが保証されるため、業務のリソースのデプロイを、より「集中」させることができます。

多次元ノード管理機能は、オールラウンドに運用保守の負担を軽減させます。

K8s運用保守の新パラダイム：インフラステートメント式APIを提供し、workloadの管理と同じようにノードを管理するため、kubernetes api、Tencent Cloud API、コンソールの3種類の方法でノードを管理することができます。
インテリジェント運用保守システムの自社開発：オペレーティングシステム/実行時/k8sにおいて、故障の検出および自動アップグレードをサポートしており、ユーザーの運用保守の負担軽減のために多次元でサポートします。
Tencent Cloud内のクラウドネイティブテクノロジーと組み合わせて実践します。オペレーティングシステム、実行時、kubernetesに対して全面的にパラメータのチューニングおよび適用を行い、ノードが初期化されるため安定性が大きく向上します。

ネイティブノード VS 通常ノード

ネイティブノードには、通常ノードのすべての機能が含まれ、かつ全面的に強化されています。

モジュール	ネイティブノード	通常ノード
管理モード	ノードの管理モード：プラットフォームは、リソース管理および安定的な運用保守の面において、クライアントによる意思決定と分析をサポートするための機能を提供します。	Serverfulモード：ユーザー分析、ユーザー決定、ユーザー実行
インフラステートメント式管理	サポート	サポートしません。
Podインプレースのアップグレード・ダウングレード	サポート	サポートしません。
カーネルパラメータチューニングなどのカスタム設定のエントリー	サポート	サポートしません。
ノードの故障時の自己回復	自社開発したオペレーティングシステム、K8s環境、実行時レベルの故障検出および自己回復機能	コミュニティNPD（メンテナンス停止済み）
スケジューラ	ネイティブノード専用スケジューラは、バーチャル拡大をサポートしており、リソースのスケジューリングができます。	コミュニティDynamicScheduler、DeScheduler
Requestインテリジェント推奨	推奨とワンクリック更新のサポート	サポートしません。
Jobを占有できます	サポート	サポートしません。
ノード管理	カーネル / Nameserver / Hostsパラメータ設定、前提スクリプト / 結論スクリプト機能をサポートします	サポートしません。

課金モデル

ネイティブノードは、複数のCVMインスタンスタイプをサポートしており、アプリケーションの規模および業務特性に基づき、適切なインスタンスを選択してデプロイすることができます。TKEは、ノードインスタンスが消耗

したリソース（CPU、メモリ、GPUおよびシステムディスクを含む）に対し、インスタンスタイプとリソースの仕様に基づいて課金します。
ネイティブノードは、**従量課金**の課金モデルをサポートしています。

課金モデル	**従量課金**
支払方法	後払い（購入時に料金を凍結し、1時間ごとに決済します）
課金単位	米ドル/秒
単価	比較的高単価
最少使用時間	1秒ごとに課金され、1時間ごとに決済されます。いつでも購入とリリースができます
ユースケース	トランスコード、ビッグデータ、eコマースにおける買い占めなどの定期的な計算シーンまたは自動スケーリング（HPA）を有効化した周期的なオンラインサービスシーンに適しています。

リージョンおよびアベイラビリティーゾーン

以下のリージョンでネイティブノードを使用することができます。

中国

リージョン			リージョンの略称
中国	パブリッククラウドリージョン	北京	ap-beijing
		南京	ap-nanjing
		上海	ap-shanghai
		広州	ap-guangzhou
		成都	ap-chengdu
		重慶	ap-chongqing
		中国香港	ap-hongkong
		台北	ap-taipei

その他の国と地域

リージョン			リージョンの略称

アジア太平洋	パブリッククラウドリージョン	シンガポール	ap-singapore
		ムンバイ	ap-mumbai
		ジャカルタ	ap-jakarta
		ソウル	ap-seoul
		バンコク	ap-bangkok
		東京	ap-tokyo
北アメリカ		シリコンバレー	na-siliconvalley
		バージニア	na-ashburn
		トロント	na-toronto
南アメリカ		サンパウロ	sa-saopaulo
欧州		フランクフルト	eu-frankfurt

ネイティブノードの関連操作

説明

ネイティブノードに対するグループ管理をより便利にするために、**ノードプール**からパラメータが同じネイティブノードを作成し、管理することをお勧めします。

ネイティブノードの新規作成は、コンソール、Kubernetes API 、Tencent Cloud APIの3種類の方法によって、クラスター内のネイティブノードの作成を完了させることをお勧めします。

ネイティブノードの削除

故障時の自己回復ルール

ステートメント式操作の実践

ネイティブノードのスケーリング

Podインプレースのアップロード・ダウングレード

Managementパラメータの紹介

ネイティブノードのライフサイクル

最終更新日：：2023-05-08 18:28:50

ライフサイクルのステータスの説明

ステータス	説明
ヘルス	ノードが正常に実行され、クラスターに接続されています。
異常	ノードの実行に異常があり、クラスターに接続されていません。
作成中	ノードは作成中のため、クラスターに接続されていません。作成中のノードは、**機器の購入、コンポーネントのインストール、ノードの登録**操作を完了後、正常にクラスターに接続されます。
ドレイン中	ノードは、Podによるその他のノードへの接続をドレインしています。
再起動中	ノードは再起動中で、クラスターに接続することができないため、新しいPodが、このノードへスケジューリングすることを許可していません。
ブロック済み	ノードはブロック済みのため、新しいPodが、このノードへスケジューリングすることを許可していません。

ネイティブノードパラメータの説明

最終更新日：：2023-04-28 11:08:19

ネイティブノード基本パラメータの説明

パラメータ	サポート詳細	説明
モデル	標準型：S2、S5、SA2、SA3 コンピューティング型：C3、C4 メモリ型：MA3、M5 GPU型：GN7、GNV4、PNV4 High IO型：IT5	コンソールのモデル表示は、選択したアベイラビリティーゾーンに残っているリソースの量と相関します。モデルに関するその他のご要望については、チケットを提出からサポートを求めることができます。
システムディスク	高性能クラウドディスク、SSDクラウドディスク	システムディスクは100G以上に増設することをお勧めします。
データディスク	高性能クラウドディスク、SSDクラウドディスク	デフォルトでは、データディスクはバインドされていません。
パブリックネットワーク帯域幅	EIPのバインドをサポート	デフォルトでは、ノードのパブリックネットワーク帯域幅は有効になっていません。
OS	TencentOS Server 3.1	Tencent Cloud仮想化プラットフォームにより、カーネル最適化などの技術によってクラウドネイティブなリソース分離技術に対応し、コンテナアプリケーションのシナリオに応じた包括的なパラメータチューニングを行います。
ノードログイン	SSHログイン	ノードログインはデフォルトで有効になっていますので、コンソールからログインするノードのSSHキーの有効化と発行が行えます。
GPUドライバー	450/470ドライバー	ノードプール作成時にターゲットドライバーを選択できます。
実行時	Containerd 1.4.3	コンテナチームはContainerdログのパフォーマンスを最適化し、NginxコンテナのCPU使用率を50%から90%に引き上げます（この特性はコミュニティにマージされています）。
Kubernetes	K8sメジャーバージョン >= 1.16	対応するK8sマイナーバージョンの要件：v1.16.3-tke.28以上、v1.18.4-tke.26以上、v1.20.6-tke.21以上。

ネイティブノードの新規作成

最終更新日：：2023-05-08 18:28:50

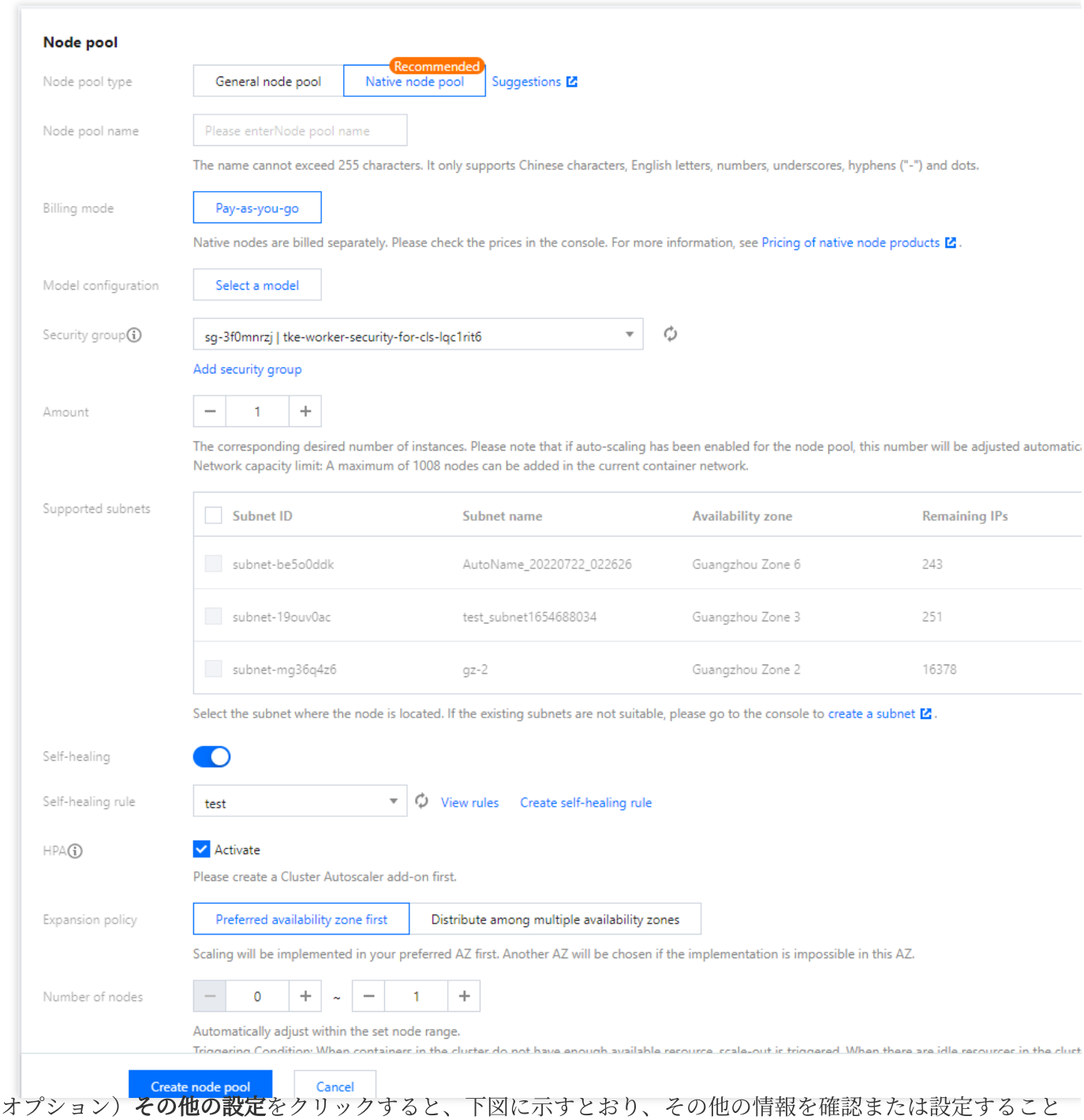
ここでは、コンソールとYAMLからネイティブノードを作成する方法をご紹介します。

前提条件

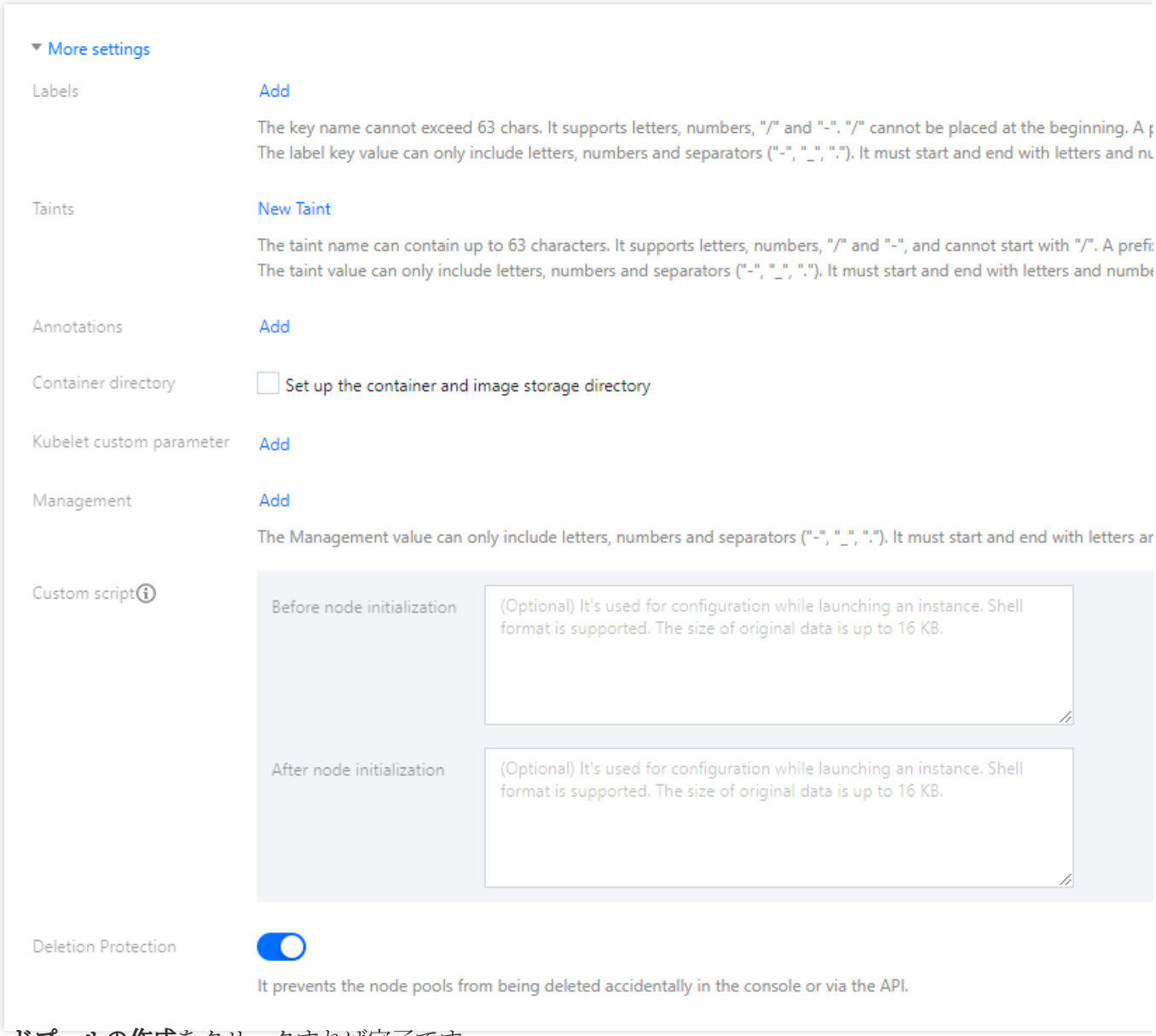
TKEコンソールにログインしていることが必要です。
TKE 標準クラスターを作成済みであることが必要です。まだ作成していない場合は、標準クラスターのクイック作成をご参照ください。
説明
ネイティブノードは、**ノードプール**による管理のみをサポートしています。

コンソールから作成

1. TKEコンソールにログインし、左側ナビゲーションバーから**クラスター**を選択します。
2. クラスターリストページで、クラスターIDをクリックし、このクラスターの詳細ページに進みます。
3. 左側メニューバーの**ノード管理 > ノードプール**を選択し、「ノードプールリスト」ページに進みます。
4. **ノードプールの新規作成**をクリックし、「ノードプールの新規作成」ページに進みます。パラメータの作成の説明の注意事項を参考に、下図に示すとおり、設定を行ってください。



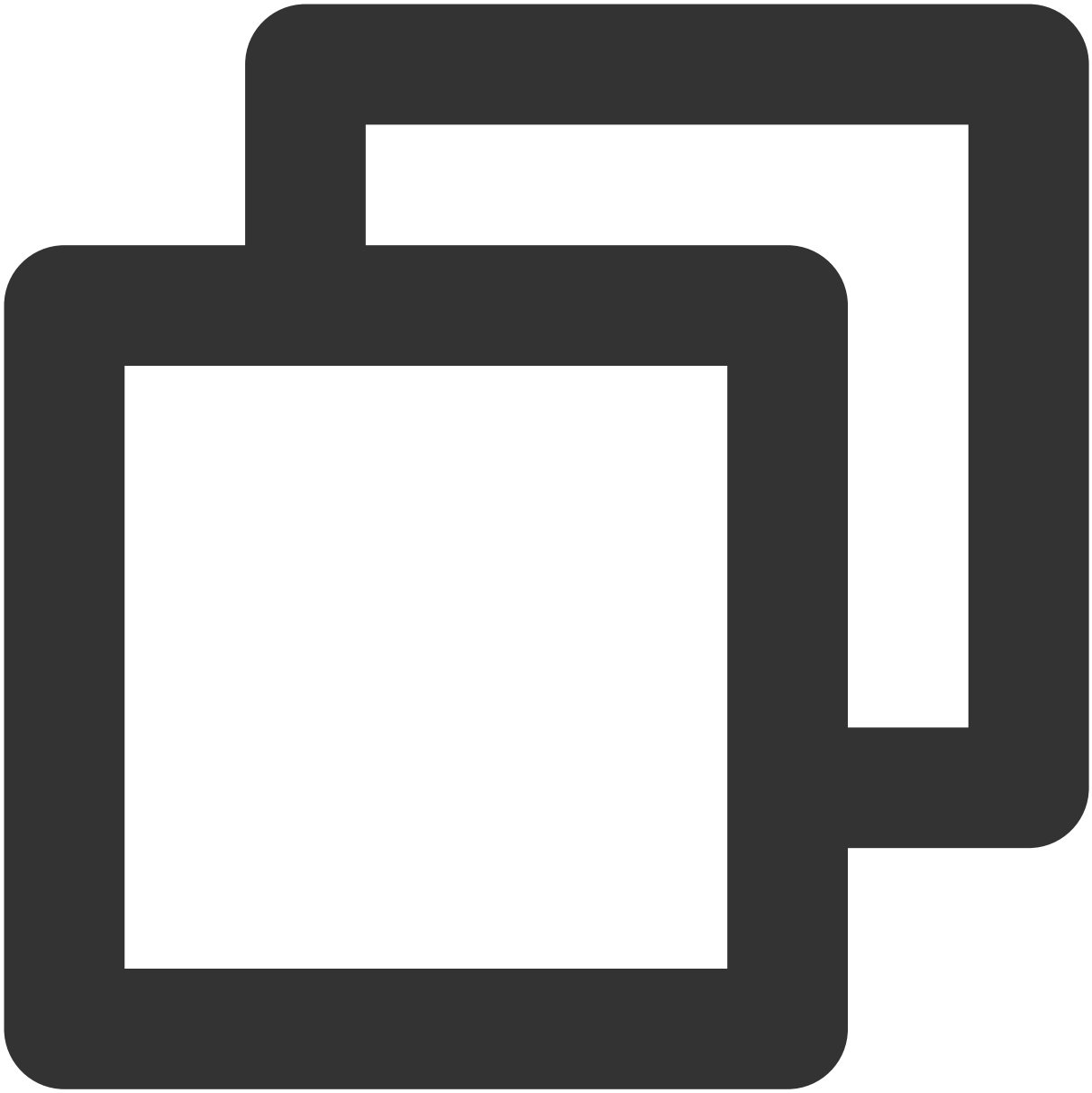
5. （オプション）**その他の設定**をクリックすると、下図に示すとおり、その他の情報を確認または設定することができます。



6. **ノードプールの作成**をクリックすれば完了です。

YAMLで作成します

ネイティブノードプールのkubernetesリソースは、下記のとおりです。YAMLフィールドについては、パラメータの作成の説明をご参照ください。



```
apiVersion: node.tke.cloud.tencent.com/v1beta1
kind: MachineSet
spec:
  type: Native
  displayName: mstest
  replicas: 2
  autoRepair: true
  deletePolicy: Random
  healthCheckPolicyName: test-all
  instanceTypes:
  - C3.LARGE8
```

```
  subnetIDs:
  - subnet-xxxxxxxx
  - subnet-yyyyyyyy
  scaling:
    createPolicy: ZonePriority
    maxReplicas: 100
  template:
    spec:
      displayName: mtest
      runtimeRootDir: /var/lib/containerd
      unschedulable: false
      metadata:
        labels:
          key1: "val1"
          key2: "val2"
      providerSpec:
        type: Native
        value:
          instanceChargeType: PostpaidByHour
          lifecycle:
            preInit: "echo hello"
            postInit: "echo world"
          management:
            hosts:
            - Hostnames:
              - lkongtest
              IP: 22.22.22.22
            nameservers:
            - 183.60.83.19
            - 183.60.82.98
            - 8.8.8.8
          metadata:
            creationTimestamp: null
          securityGroupIDs:
          - sg-xxxxxxxx
          systemDisk:
            diskSize: 50
            diskType: CloudPremium
```

パラメータの作成の説明

パラメータの所属	パラメータ項目	YAMLフィールド

モジュール		
スタートアップコンフィギュレーション	ノードプールのタイプ	フィールド名：spec.type フィールド値：Native
	ノードプールの名前	フィールド名：metadata.name フィールド値：demo-machineset（カスタマイズ）
	課金モデル	フィールド名：spec.template.spec.providerSpec.value.instanceChargeType フィールド値：PostpaidByHour（従量課金）
	モデルコンフィグレーション	モデル： フィールド名：spec.instanceTypes フィールド値：S2.MEDIUM4（コンソールを参考にしてその他のモデル仕様を取得することができます） システムディスク： フィールド名： spec.template.spec.providerSpec.value.systemDisk.diskSize/diskType フィールド値： diskSize: 50（システムディスクのサイズは、10の倍数でカスタマイズすることができ、最小で50Gです） diskType: CloudPremium / CloudSSD（システムディスクのタイプで、高性能/SSDをサポートしています）
	セキュリティグループ	フィールド名：spec.template.spec.providerSpec.value.securityGroupIDs フィールド値：sg-a7msxxx（セキュリティグループID）
	数量	フィールド名：spec.replicas

		フィールド値：7（カスタマイズ）
	コンテナネットワーク	フィールド名：spec.subnetIDs フィールド値：subnet-i2ghxxxx（コンテナサブネットID）
運用保守機能	故障時の自己回復	フィールド名：spec.autoRepair フィールド値：true（オン）/ false（オフ）
	チェックおよび自己回復ルール	フィールド名：spec.healthCheckPolicyName フィールド値：test-all（故障時の自己回復のCRの名前をバインドします）
	自動アップグレード(alphaバージョン)	-

アップグレード項目	-
運用保守ウィンドウ	-
最大アップグレードノード数	-
オートスケーリング	フィールド名：spec.scaling

	ノード数量の範囲	フィールド名：spec.scaling.maxReplicas / minReplicas フィールド値： maxReplicas: 7（カスタマイズ） minReplicas: 2（カスタマイズ）
	拡張ポリシー	フィールド名：spec.scaling.createPolicy フィールド値：ZonePriority（最初に選択したアベイラビリティーゾーンの優先）/ ZoneEquality（マルチアベイラビリティーゾーンの分散）
高度なパラメータ	Lables	フィールド名：spec.template.spec.metadata.labels フィールド値： key1: 「value1”」（lableのkey/valueはカスタマイズです）
	Taints	フィールド名：spec.template.spec.metadata.taints フィールド値：effect: NoSchedule/PreferNoSchedule/NoExecute（taintsタイプを入力します）

	コンテナディレクトリ	フィールド名：spec.template.spec.runtimeRootDir フィールド値：/data/containerd（カスタマイズ）
	kubelet カスタマイズパラメータ	このフィールドはホワイトリストとして制御され、チケット/販売後から申請によって開放されます。
	Management	フィールド名： spec.template.spec.providerSpec.value.management.hosts/nameservers/KernelArgs フィールド値： hosts: Hostnames: ['test'], IP: '22.22.22.22'nameservers: ['183.60.83.19', '183.60.82.98']KernelArgs:このフィールドはホワイトリストとして制御されています。チケットを提出して、サポートを受けることができます。
	カスタマイズスクリプト	フィールド名：spec.template.spec.providerSpec.value.lifecycle.preInit/postInit フィールド値： preInit: "echo hello"（ノードを初期化する前にスクリプトを実行し、カスタマイズします） postInit: "echo world"（ノードを初期化した後にスクリプトを実行し、カスタマイズします）

ネイティブノードの削除

最終更新日：：2023-05-08 18:28:50

ここでは、ノードプールからネイティブノードを削除し、ノードプールが管理するノード数を減少させる方法についてご紹介します。

注意事項

ステートメント式管理のためのネイティブノードにおいて、ノードプールの**ノード数**を変更していない状況では、ノードを直接削除すると新たなノードがすぐにノードプールに作成されます。
ネイティブノードは、ノードプールのディメンションに従ってグループを管理しており、ノードからクラスターへの移動はサポートしていません。
ネイティブノードの**削除すなわち破棄**、ノードリソースの完全なリリースについて、復元はサポートされていないため、慎重に操作することをお勧めします。

ネイティブノードの削除

従量課金のタイプ

1. TKEコンソールにログインし、左側ナビゲーションバーから**クラスター**を選択します。
2. クラスターリストページで、クラスターIDをクリックし、このクラスターの詳細ページに進みます。
3. 左側メニューバーの**ノード管理 > ノードプール**を選択し、「ノードプールリスト」ページに進みます。
4. 削除したいノードのノードプールIDをクリックし、「ノードリスト」ページに進みます。
5. ノードリストで**数量の調整**をクリックし、ポップアップウィンドウでノード数量を少なく設定します。例えば、現在ノードプールのネイティブノードの数量が5で、2つのノードを削除したい場合、数量を3に調整する必要があります。
6. **確定**をクリックし、ノードの削除を完了させます。

説明
従量課金のノードプールでノードの数量を少なく設定する場合、バックエンドは余分なノードのレプリカをランダムに削除することで、ノードプール内のノード数量を、ステートメント式が期待するノード数量に適合させます。

故障時の自己回復ルール

最終更新日：：2023-05-08 18:28:50

機能の説明

インフラに安定性がなく、環境に確定性がない場合は、さまざまな側面からシステムの故障が発生することになります。スタッフを頻発する運用保守の業務から解放するために、TKEチームは、故障時の自己回復機能を自社開発し、運用保守スタッフが迅速に問題を特定できるようにサポートします。プリセットしたプラットフォーム運用保守の経験から、さまざまな検査項目に対して最小の自己回復動作を提供します。この機能は、NPD Plusコンポーネントの基盤をさらに拡張させたもので、具体的には以下の特徴が含まれています。
システムは、人員が関与して解決しなければならない持続的な故障をリアルタイムで検出します。
故障範囲には、オペレーティングシステム、K8s環境、実行時などの数十種類の検査項目が含まれています。
プリセットした専門家の経験（スクリプトの修正の実行、コンポーネントの再起動）を基に、故障に対して迅速なレスポンスを行います。

検査項目の紹介

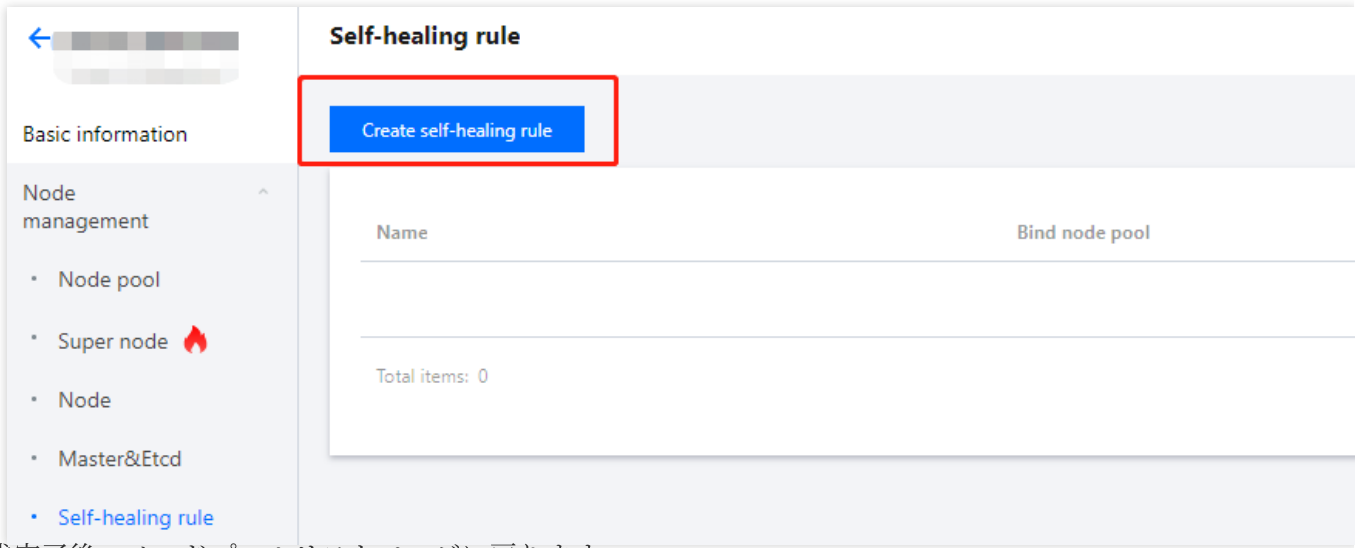
検査項目	説明	リスクレベル	自己回復動作
FDPressure	Too many files opened（ホストのファイルのディスクリプタの数量が、最大値の90%に達したかどうかを確認します)	low	-
RuntimeUnhealthy	List containerd task failed	low	RestartRuntime
KubeletUnhealthy	Call kubelet healthz failed	low	RestartKubelet
ReadonlyFilesystem	Filesystem is readonly	high	-
OOMKilling	Process has been oom-killed	high	-
TaskHung	Task blocked more then beyond the threshold	high	-
UnregisterNetDevice	Net device unregister	high	-
KernelOopsDivideError	Kernel oops with divide error	high	-
KernelOopsNULLPointer	Kernel oops with NULL pointer	high	-

Ext4Error	Ext4 filesystem error	high	-
Ext4Warning	Ext4 filesystem warning	high	-
IOError	IOError	high	-
MemoryError	MemoryError	high	-
DockerHung	Task blocked more then beyond the threshold	high	-
KubeletRestart	Kubelet restart	low	-

ノードの故障時の自己回復機能を有効化します

コンソールを介して

1. TKEコンソールにログインし、左側ナビゲーションバーから**クラスター**を選択します。
2. クラスターリストページで、クラスターIDをクリックし、このクラスターの詳細ページに進みます。
3. 左側メニューバーの**ノード管理 > 故障時の自己回復ルール**を選択し、「故障時の自己回復ルールリスト」ページに進みます。
4. **故障時の自己回復ルールの新規作成**をクリックし、下図に示すとおり、新しい故障時の自己回復ルールを作成します。

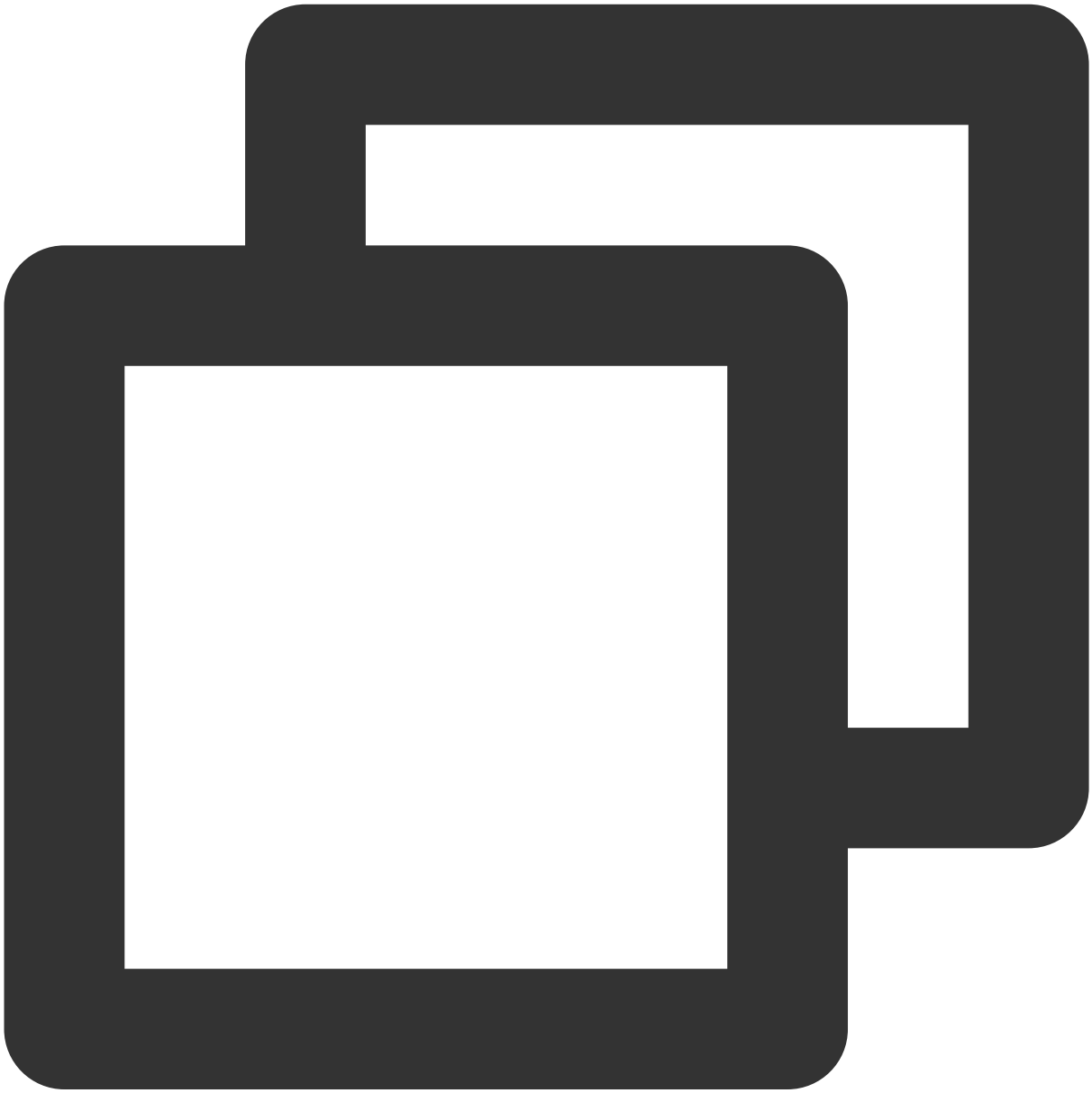


5. 作成完了後、ノードプールリストページに戻ります。
6. ノードプールIDをクリックし、ノードプール詳細ページに進みます。
7. ノードプール詳細ページの「運用保守情報」モジュールで、**編集**をクリックし、ノードプールの故障時の自己回復機能を有効化します。
8. 有効化すると、「運用保守記録」からリアルタイムで故障の検出の詳細を確認することができます。ステータスが「失敗」となっている場合は、この検査項目が有効化されていないことを示しています。

YAMLで操作します

1. 故障時の自己回復ルールの新規作成

コマンド `kubectl ceate -f demo-HealthCheckPolicy.yaml` に従って、クラスターに自己回復ルールを作成します。YAMLの設定は次のとおりです。



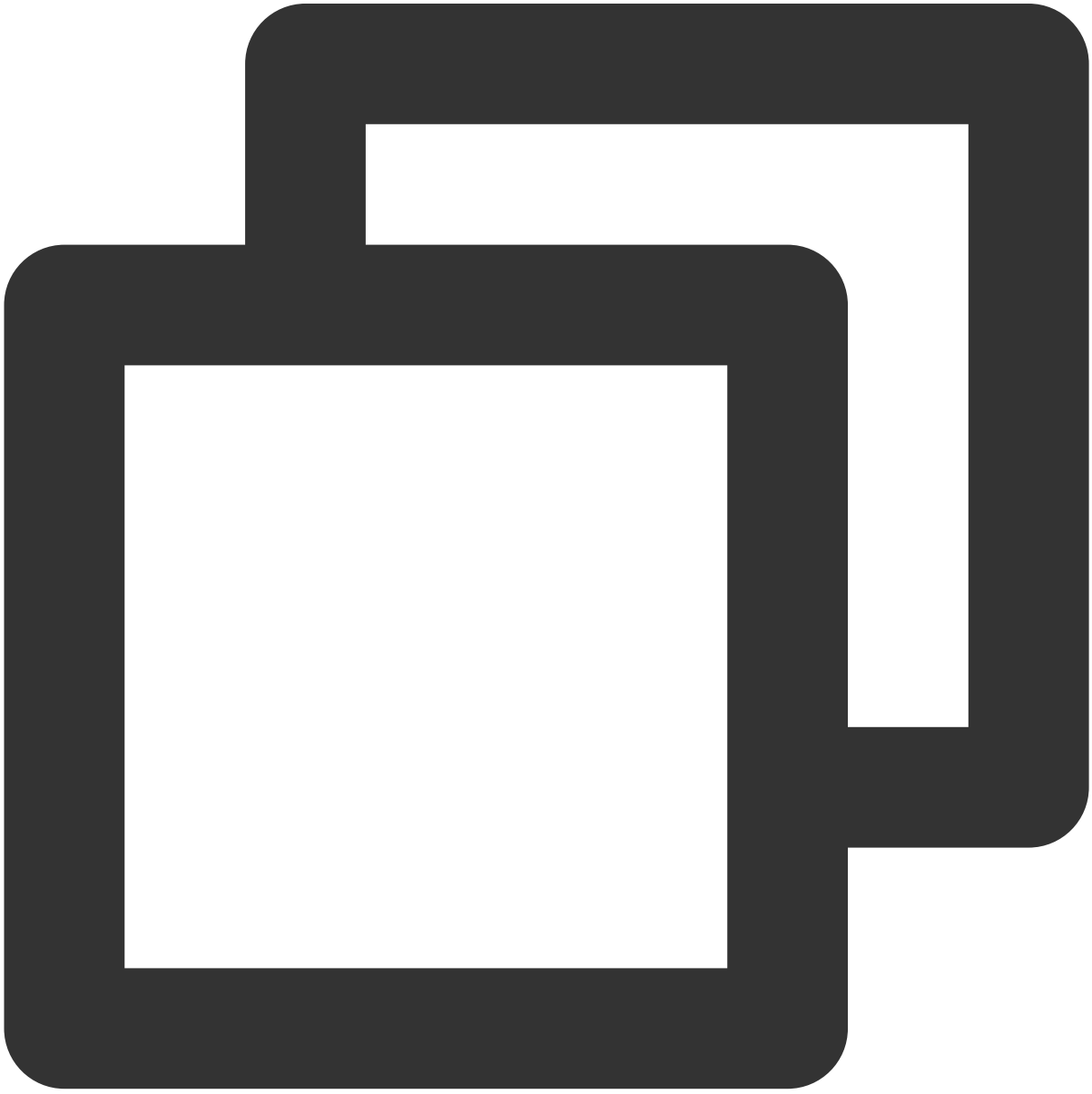
```
apiVersion: config.tke.cloud.tencent.com/v1
kind: HealthCheckPolicy
metadata:
  name: test-all
  namespace: cls-xxxxxxxx（クラスターID）
```

```
spec:
  machineSetSelector:
    matchLabels:
      key: fake-label
  rules:
  - action: RestartKubelet
    enabled: true
    name: FDPressure
  - action: RestartKubelet
    autoRepairEnabled: true
    enabled: true
    name: RuntimeUnhealthy
  - action: RestartKubelet
    autoRepairEnabled: true
    enabled: true
    name: KubeletUnhealthy
  - action: RestartKubelet
    enabled: true
    name: ReadonlyFilesystem
  - action: RestartKubelet
    enabled: true
    name: OOMKilling
  - action: RestartKubelet
    enabled: true
    name: TaskHung
  - action: RestartKubelet
    enabled: true
    name: UnregisterNetDevice
  - action: RestartKubelet
    enabled: true
    name: KernelOopsDivideError
  - action: RestartKubelet
    enabled: true
    name: KernelOopsNULLPointer
  - action: RestartKubelet
    enabled: true
    name: Ext4Error
  - action: RestartKubelet
    enabled: true
    name: Ext4Warning
  - action: RestartKubelet
    enabled: true
    name: IOError
  - action: RestartKubelet
    enabled: true
    name: MemoryError
  - action: RestartKubelet
```

```
    enabled: true
    name: DockerHung
  - action: RestartKubelet
    enabled: true
    name: KubeletRestart
```

2. 自己回復スイッチを有効化する

MachineSetで、フィールド `healthCheckPolicyName: test-all` を指定します。YAMLの設定は次のとおりです。



```
apiVersion: node.tke.cloud.tencent.com/v1beta1
kind: MachineSet
spec:
  type: Hosted
  displayName: demo-machineset
  replicas: 2
  autoRepair: true
  deletePolicy: Random
  healthCheckPolicyName: test-all
  instanceTypes:
  - C3.LARGE8
  subnetIDs:
  - subnet-xxxxxxxx
  - subnet-yyyyyyyy
......
```

ネイティブノードのスケーリング

最終更新日：：2023-05-08 18:28:50

利用説明

ネイティブノードの自動スケーリング機能は、コンテナプラットフォームの自社開発によって実現しました。通常ノードの自動スケーリング機能は、クラウド製品Auto Scaling（AS）に依存しています。
ネイティブノードプールは、自動スケーリングを有効化していません。
ノード数量の初期化は、設定した**ノード数**で決まります（フィールド名：replicas）。
ユーザーは、希望するノード数を手動で調整することができます。ノード数の上限は、バックエンドのデフォルト値の500とコンテナサブネットのIPの数によって制限されます。
ネイティブノードプールの自動スケーリングを有効化します。
ノード数量の初期化は、設定した**ノード数**で決まります（フィールド名：replicas）。
ノードの数量範囲（フィールド名：minReplicas/maxReplicas）を設定する必要があります。CAは、現在のノードプールのノード数量を制限されているこの範囲に調節します。
ユーザーによる必要なインスタンス数の手動調節は、サポートしていません。

ノードの自動スケーリング機能の有効化

パラメータの説明

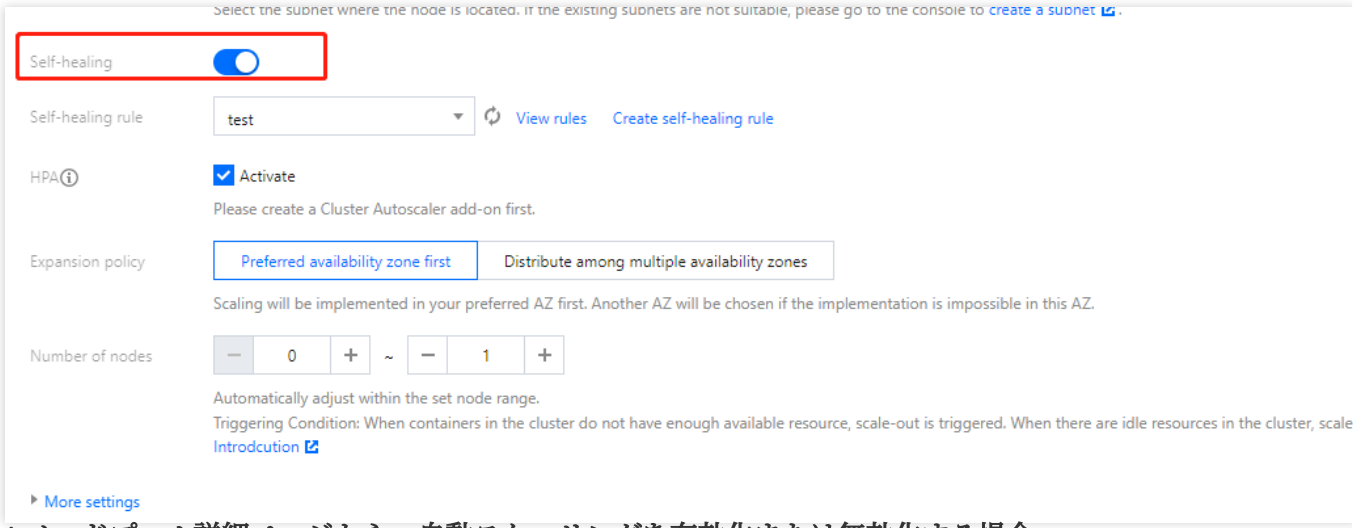
機能項目	フィールド名/値	説明
オートスケーリング	フィールド名：spec.scaling	デフォルトでは有効化にチェックが入れられており、有効化後、CAコンポーネントがこのタイプのノードプールに自動スケーリングを行います。
ノード数量の範囲	フィールド名： spec.scaling.maxReplicas/minReplicas フィールド値：カスタマイズ	ノードプール内のノード数量は、この範囲内の最小値/最大値に制限されます。ノードプールの自動スケーリングを有効化した場合、ネイティブノードの数量は設定された範囲内で自動的に調節されます。
拡張ポリシー	フィールド名： spec.scaling.createPolicy フィールド値：	最初に選択したアベイラビリティーゾーンの優先：自動スケーリングは、最初に選択したアベイラビリティーゾーンに、優先してスケーリングを実行します。最初に選択したアベイラビリティーゾーンにスケーリングを実行できない

	ZonePriority（最初に選択したアベイラビリティーゾーンの優先） ZoneEquality（マルチアベイラビリティーゾーンの分散）	場合に限って、その他のアベイラビリティーゾーンにスケーリングが行われます。 マルチアベイラビリティーゾーンの分散：スケーリンググループが指定したマルチアベイラビリティーゾーン（すなわち指定した複数のサブネット）に、ばらつきがないように最大限に努力してノードインスタンスを割り当てます。複数のサブネットを設定した場合に限って、このポリシーが有効化されます。

コンソールを介して

方法1：ノードプール作成ページから、自動スケーリングを有効化または無効化する場合

1. TKEコンソールにログインし、クラスターでノードプールを新規作成します。操作の詳細については、ネイティブノードの新規作成をご参照ください。
2. 「ノードプールの新規作成」ページで、下図に示すとおり、**自動スケーリングの有効化**にチェックを入れます。

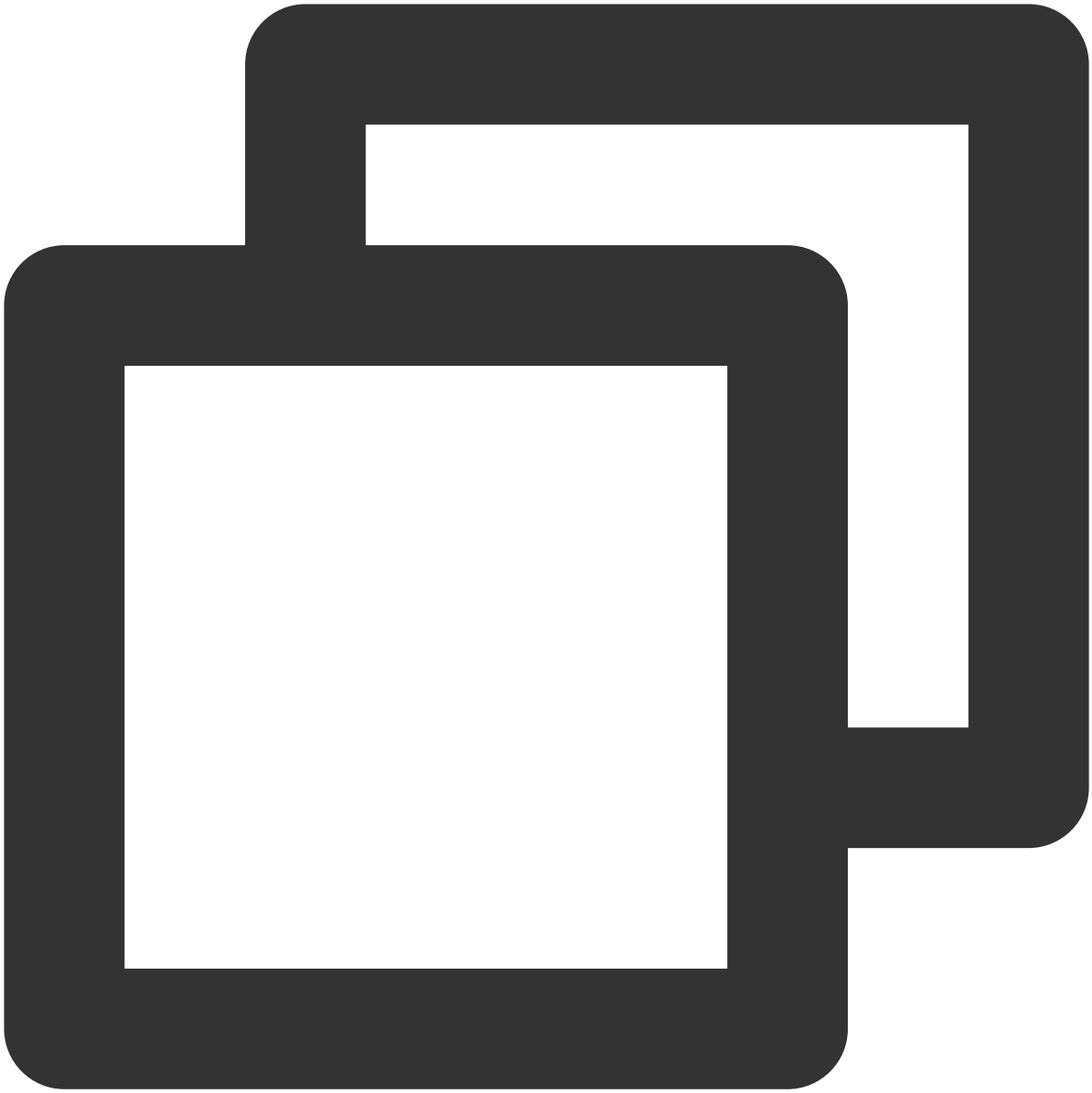


方法2：ノードプール詳細ページから、自動スケーリングを有効化または無効化する場合

1. TKEコンソールにログインし、左側ナビゲーションバーから**クラスター**を選択します。
2. クラスターリストページで、クラスターIDをクリックし、このクラスターの詳細ページに進みます。
3. 左側メニューバーの**ノード管理 > ノードプール**を選択し、「ノードプールリスト」ページに進みます。
4. ノードプールIDをクリックし、ノードプール詳細ページに進みます。
5. ノードプール詳細ページで、運用保守情報の右側の**編集**をクリックします。
6. **自動スケーリングの有効化**にチェックを入れ、**確定**すなわち自動スケーリングの有効化をクリックします。

YAMLで操作します

パラメータの紹介に基づき、ノードプールのYAMLで `scaling` フィールドを入力します。



```
apiVersion: node.tke.cloud.tencent.com/v1beta1
kind: MachineSet
spec:
  type: Native
  displayName: mstest
  replicas: 2
  autoRepair: true
  deletePolicy: Random
  healthCheckPolicyName: test-all
  instanceTypes:
  - C3.LARGE8
```

```
  subnetIDs:
  - subnet-xxxxxxxx
  - subnet-yyyyyyyy
  scaling:
    createPolicy: ZonePriority
    minReplicas: 10
    maxReplicas: 100
  template:
    spec:
      displayName: mtest
      runtimeRootDir: /var/lib/containerd
      unschedulable: false
......
```

スケーリング記録の確認

1. TKEコンソールにログインし、左側ナビゲーションバーから**クラスター**を選択します。
2. クラスターリストページで、クラスターIDをクリックし、このクラスターの詳細ページに進みます。
3. 左側メニューバーの**ノード管理 > ノードプール**を選択し、「ノードプールリスト」ページに進みます。
4. ノードプールIDをクリックし、ノードプール詳細ページに進みます。
5. 運用保守記録ページで、ノードのスケーリング記録を確認することができます。

ネイティブノードでSSHキーを有効化してログインします

最終更新日：：2023-05-08 18:28:50

概要

このドキュメントでは、ノードプールとノードのSSHキーの指定および変更を含む、ネイティブノードでSSHキーによるログインを有効化するための関連操作についてご紹介します。

操作手順

ノードプールのディメンション

ノードプールの作成時に指定する場合
既存のノードプールに指定する場合
作成時、ノードプールにグローバルディメンションのSSHキーを指定することができます。ノードプールから拡張したノードは、デフォルトではこのキーを発行します。
1. TKEコンソールにログインし、左側ナビゲーションバーから**クラスター**を選択します。
2. クラスターリストページで、クラスターIDをクリックし、このクラスターの詳細ページに進みます。
3. 左側メニューバーの**ノード管理 > ノードプール**を選択し、**ノードプール**ページで、**ネイティブノードプールの新規作成**をクリックします。
4. **ノードプールの新規作成**ページで、**SSHキー**を選択します。
5. **ノードプールの作成**をクリックすれば完了です。
ノードプールの詳細ページで起動設定モジュールを見つけ、関連するsshキーに変更を行います。関連キーの変更は、ノードプール内のノードの増分にのみ有効で、作成済みのネイティブノードに影響を与えることはありません。
1. TKEコンソールにログインし、左側ナビゲーションバーから**クラスター**を選択します。
2. クラスターリストページで、クラスターIDをクリックし、このクラスターの詳細ページに進みます。
3. 左側メニューバーの**ノード管理 > ノードプール**を選択し、**ノードプール**ページで、**ノードプール ID**をクリックし、ノードプール詳細ページに進みます。
4. ノードプール詳細ページで、ノード起動設定情報から**関連sshキー**の右側の

をクリックします。
5. **ノードプールのSSHキーの設定**ページで、SSHキーを選択します。
6. **確定**をクリックします。

ノードのディメンション

1つのノードを有効化/変更してログインする場合
複数のノードを一括して有効化/変更し、ログインする場合
ノードプールのディメンションのSSHキーを設定していない場合は、ノード操作から「ノードログイン」を選択し、ターゲットノードを有効化してログインします。この操作は、ノードに対して発行済みのSSHキーの変更もサポートしています。

1. TKEコンソールにログインし、左側ナビゲーションバーから**クラスター**を選択します。
2. クラスターリストページで、クラスターIDをクリックし、このクラスターの詳細ページに進みます。
3. 左側メニューバーの**ノード管理 > ノードプール**を選択し、**ノードプール**ページで、**ノードプール ID**をクリックし、ノードプール詳細ページに進みます。
4. ノードプールのノードリストページで、ノードがある行の右側の**その他 > ノードログイン**を選択します。

ノードリストのその他の操作では、ノード操作から「SSHキーの一括設定」を選択し、複数のターゲットノードにキーを同時発行することができます。

1. TKEコンソールにログインし、左側ナビゲーションバーから**クラスター**を選択します。
2. クラスターリストページで、クラスターIDをクリックし、このクラスターの詳細ページに進みます。
3. 左側メニューバーの**ノード管理 > ノードプール**を選択し、**ノードプール**ページで、**ノードプール ID**をクリックし、ノードプール詳細ページに進みます。
4. ノードプールのノードリストページで、複数のノードにチェックを入れ、**その他 > SSHキーの一括設定**を選択します。

説明：

1. ノードで発行されたSSHキーを変更しても、ノードが再起動することはありません。
2. ノードを有効化/変更後に初めてログインしたり、ノードが発行したSSHキーを変更したりした場合は、1〜2分後に有効化されます。
3. 複数のノードを一括して有効化/変更し、ログインする場合は、1回の操作におけるノード数を20個以内にすることをお勧めします。
4. 複数のノードを一括して有効化/変更し、ログインする場合に、ノードで発行済みのキーが上書きされることはありません。

Managementパラメータの紹介

最終更新日：：2023-05-08 18:28:50

機能概要

Managementパラメータは、ノードの共通カスタマイズ設定として統一されたポータルを提供します。このポータルからネイティブノードの基盤であるカーネルパラメータKernelArgsにチューニングをすることができます。また、Nameservers\\Hostsの設定をサポートすることで、業務をデプロイするための環境要件を満たすことができます。

Managementパラメータ項目

パラメータ項目	説明
Nameservers	業務のデプロイ環境の設定には、DNSサーバーアドレスが必要になります。
Hosts	業務のデプロイ環境の設定に必要なHostsです。
KernelArgs	カーネルパラメータを設定し、業務にパフォーマンスのチューニングを行います（現在、この機能はホワイトリストにリリースされており、チケットを提出して申請することができます）。

説明

システムコンポーネントの正常なインストールを確実にするため、ネイティブノードは、デフォルトではTencent Cloudの公式データライブラリのアドレス `nameserver = 183.60.83.19` 、 `nameserver = 183.60.82.98` に注入されます。

KernelArgsパラメータ

下記に調整がサポートされているOSパラメータおよび受け入れられている値を一覧で表示します。

ソケットとネットワークの最適化

大量かつ同時に処理することが予想されるセッションのプロキシノードに対しては、下記のTCPおよびネットワークオプションを使用して調整することができます。

番号	パラメータ	デフォルト値	許可される値 / 範囲	パラメー	範囲

				タタイプ	
1	"net.core.somaxconn"	32768	4096 - 3240000	int	The maximum length of the listening queue for each port in the system.
2	"net.ipv4.tcp_max_syn_backlog"	8096	1000 - 3240000	int	The maximum length of tcp SYN queue length.
3	"net.core.rps_sock_flow_entries"	8192	1024 - 536870912	int	The maximum size of hash table for RPS.
4	"net.core.rmem_max"	16777216	212992 - 134217728	int	The maximum size, in bytes, of the receive socket buffer.
5	"net.core.wmem_max"	16777216	212992 - 134217728	int	The maximum size, in bytes, of the send socket buffer.
6	"net.ipv4.tcp_rmem"	"4096 12582912 16777216"	1024 - 2147483647	string	The min/default/max size of tcp socket receive buffer.
7	"net.ipv4.tcp_wmem"	"4096 12582912 16777216"	1024 - 2147483647	string	The min/default/max size of tcp socket send buffer.
8	"net.ipv4.neigh.default.gc_thresh1"	2048	128 - 80000	int	The minimum number of entries that can be retained. If the number of entries is less than this value, the entries will not be recycled.
9	"net.ipv4.neigh.default.gc_thresh2"	4096	512 - 90000	int	When the number of entries exceeds this value, the GC will clear the entries longer than 5 seconds.
10	"net.ipv4.neigh.default.gc_thresh3"	8192	1024 -	int	Maximum allowable

			100000		number of non-permanent entries.
11	"net.ipv4.tcp_max_orphans"	32768	4096 - 2147483647	int	Maximal number of TCP sockets not attached to any user file handle, held by system. Increase this parameter properly to avoid the 'Out of socket memory' error when the load is high.
12	"net.ipv4.tcp_max_tw_buckets"	32768	4096 - 2147483647	int	Maximal number of timewait sockets held by system simultaneously. Increase this parameter properly to avoid "TCP: time wait bucket table overflow" error.

ファイルハンドラの制限：

大量のトラフィックによるサービスを提供する場合、サービスのトラフィックは、一般的に多数のローカルファイルからのものになります。占有部分のシステムメモリのみでより多くの量を処理させるために、以下のカーネル設定および内蔵の制限に若干の調整をすることができます。

番号	パラメータ	デフォルト値	許可される値 / 範囲	パラメータタイプ	範囲
1	"fs.file-max"	3237991	8192 - 12000500	int	Limit on the total number of fd, including socket, in the entire system.
2	"fs.inotify.max_user_instances"	8192	1024 - 2147483647	int	Limit on the total number of inotify instances.
3	"fs.inotify.max_user_watches"	524288	781250 - 2097152	int	The total number of inotify watches is limited. Increase this parameter to avoid "Too many open files" errors.

バーチャルメモリ：

以下の設定は、Linuxカーネルのバーチャルメモリ(VM)のサブシステムの操作およびディスクに対するダーティデータのwriteoutを行う際の調整に用いることができます。

番号	パラメータ	デフォルト値	許可される値 / 範囲	パラメータタイプ	範囲
1	"vm.max_map_count"	262144	65530 - 262144	int	The maximum number of memory map areas a process may have.

ワークスレッドの制限：

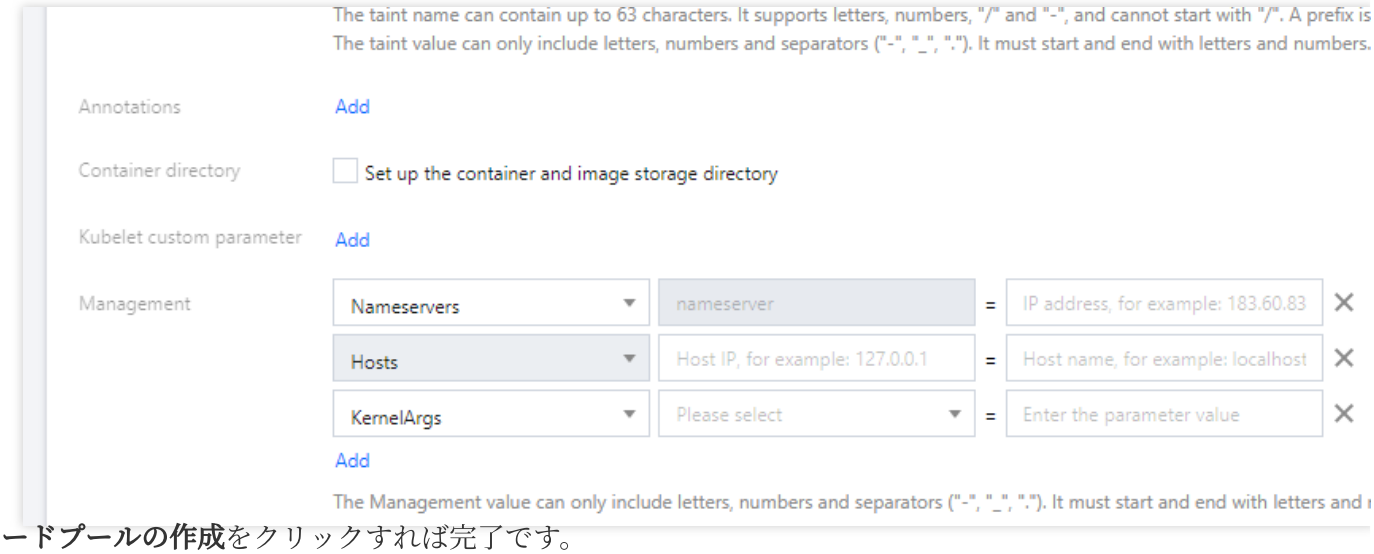
番号	パラメータ	デフォルト値	許可される値 / 範囲	パラメータタイプ	範囲
1	"kernel.threads-max"	4194304	4096 - 4194304	int	The system-wide limit on the number of threads (tasks) that can be created on the system.
2	"kernel.pid_max"	4194304	4096 - 4194304	int	PIDs greater than this value are not allocated; thus, the value in this file also acts as a system-wide limit on the total number of processes and threads.

ノードにManagementパラメータを設定

コンソールを介して

方法1：ノードを新規追加してManagementパラメータを設定する場合

1. TKEコンソールにログインし、ネイティブノードの新規作成ドキュメントを参考にして、ネイティブノードを作成します。
2. 「ノードプールの新規作成」ページで、**その他の設定**をクリックし、下図に示すとおり、ノードにManagementパラメータを設定します。

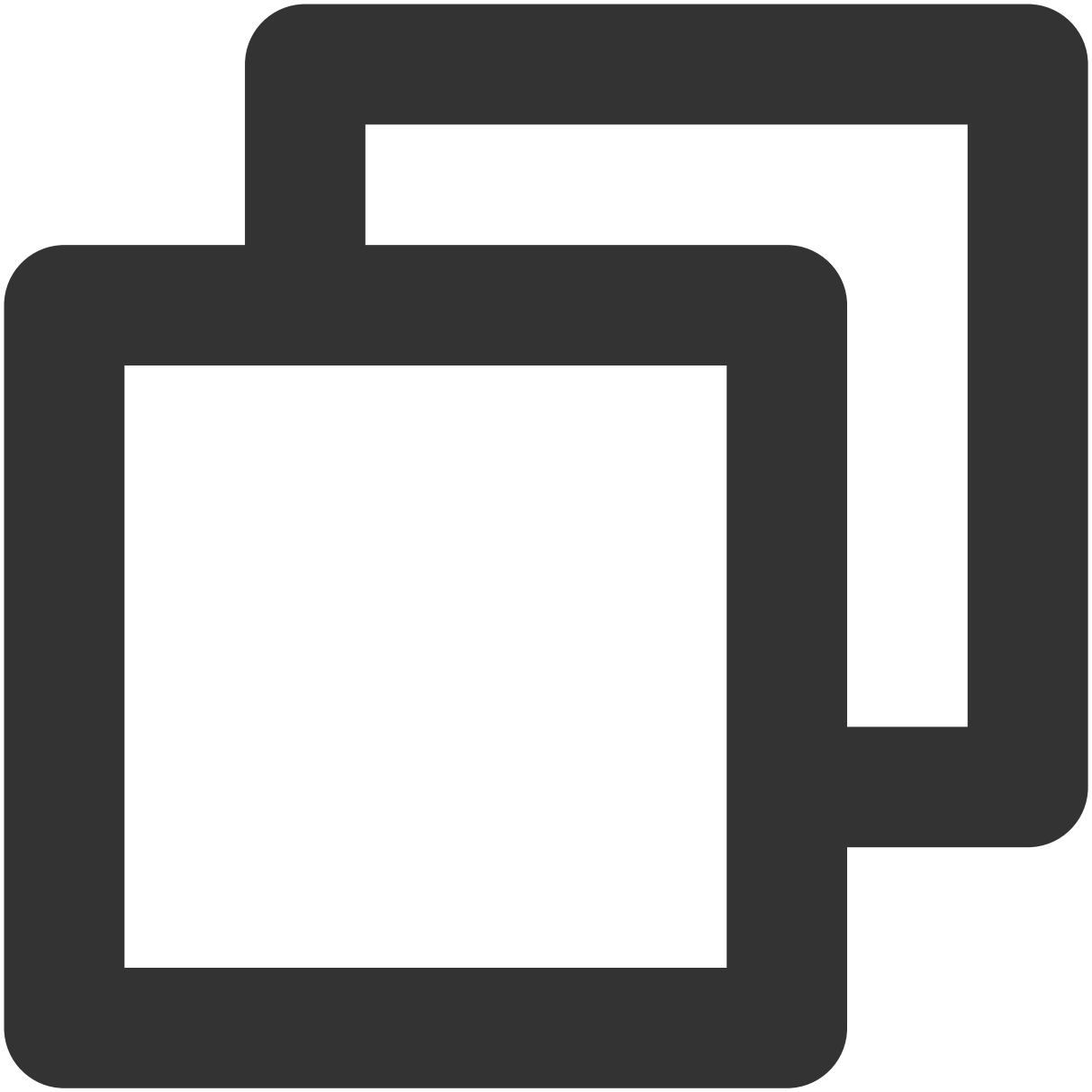


3. **ノードプールの作成**をクリックすれば完了です。

方法2：既存のノードにManagementパラメータを設定する場合

1. TKEコンソールにログインし、左側ナビゲーションバーから**クラスター**を選択します。
2. クラスターリストページで、クラスターIDをクリックし、このクラスターの詳細ページに進みます。
3. 左側メニューバーの**ノード管理 > ノードプール**を選択し、「ノードプールリスト」ページに進みます。
4. ノードプールIDをクリックし、「ノードリスト」ページに進みます。
5. 「詳細」タブで、**パラメータ設定 > 編集**をクリックし、Managementパラメータを変更します。

YAMLで操作します



```
apiVersion: node.tke.cloud.tencent.com/v1beta1
kind: MachineSet
spec:
  type: Native
  displayName: mstest
  replicas: 2
  autoRepair: true
  deletePolicy: Random
  instanceTypes:
  - C3.LARGE8
  subnetIDs:
```

```
  - subnet-xxxxxxxx
  template:
    spec:
      displayName: mtest
      providerSpec:
        type: Native
        value:
          instanceChargeType: PostpaidByHour
          # ここでmanagementパラメータ情報を入力します
          management:
            hosts:
            - Hostnames:
              - static.fake.com
              IP: 192.168.2.42
            - Hostnames:
              - common.fake.com
              IP: 192.168.2.45
            nameservers:
            - 183.60.83.19
            - 183.60.82.98
            - 8.8.8.8
            kernelArgs:
            - kernel.pid_max=65535
            - fs.file-max=400000
            - net.ipv4.tcp_rmem="4096 12582912 16777216"
            - vm.max_map_count="65535"
```

ネイティブノードのパブリックネットワークアクセスを有効化します

最終更新日：：2023-05-08 18:16:31

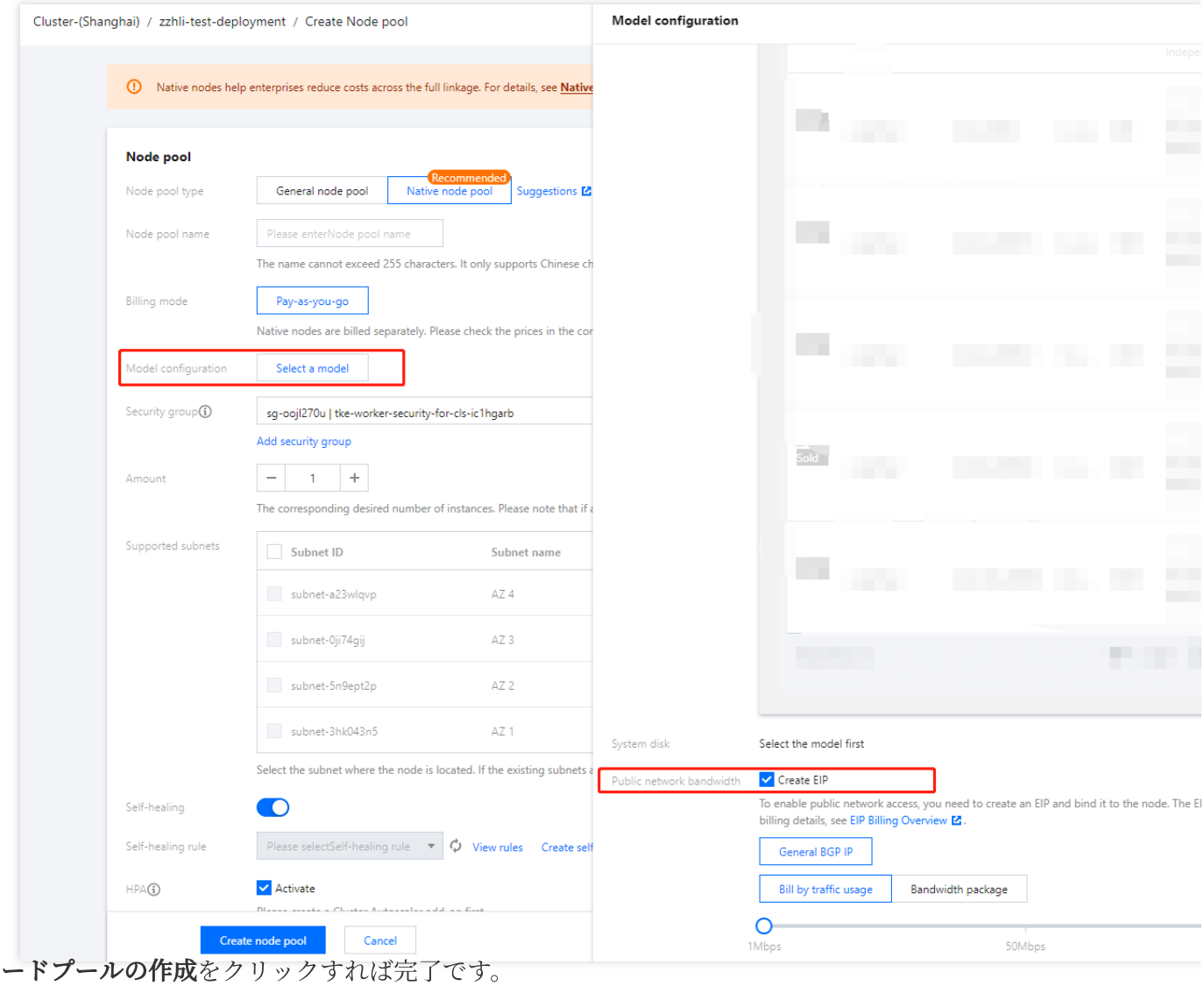
説明：
従来型アカウントタイプは、ネイティブノードによるパブリックネットワークアクセスの有効化をサポートしていません。詳細については、アカウントタイプの説明をご参照し、チケットを提出してアップグレードを行うことができます。
ここでは、コンソールとYAMLによって、ノードにElastic IP（EIP）をバインドし、パブリックネットワークを有効化する方法を主にご紹介します。

注意事項

パブリックネットワークアクセスのノードプールを有効化した場合は、新規追加された各ネイティブノードは、1つのEIPを作成してバインドします。
EIPとノードのライフサイクルを一致させると、ノードの破棄に伴って破棄されます。
ネイティブノードは、EIPに対して追加で課金されることはありません。

コンソールから、ネイティブノードのパブリックネットワークアクセスを有効化します

1. TKEコンソールにログインし、左側ナビゲーションバーから**クラスター**を選択します。
2. クラスターリストページで、クラスターIDをクリックし、このクラスターの詳細ページに進みます。
3. 左側メニューバーの**ノード管理 > ノードプール**を選択し、**ノードプール**ページで、**ネイティブノードプールの新規作成**をクリックします。
4. **ノードプールの新規作成**ページで、**モデルコンフィグレーション**をクリックし、**モデルコンフィグレーションページ**の下部で、下図に示すとおり、**EIPの作成**にチェックを入れます。



5. **ノードプールの作成**をクリックすれば完了です。

YAMLからネイティブノードのパブリックネットワークアクセスを有効化します

フィールドの紹介

フィールド名	フィールド値	意味
spec.template.spec.providerSpec.value.internetAccessible	addressType	`EIP` ：入力をしなかっ は、デフォルトで標準BG に、すなわち通常EIPにな す。 `HighQualityEIP` ：高 BGP IPで、すなわち高品 す。

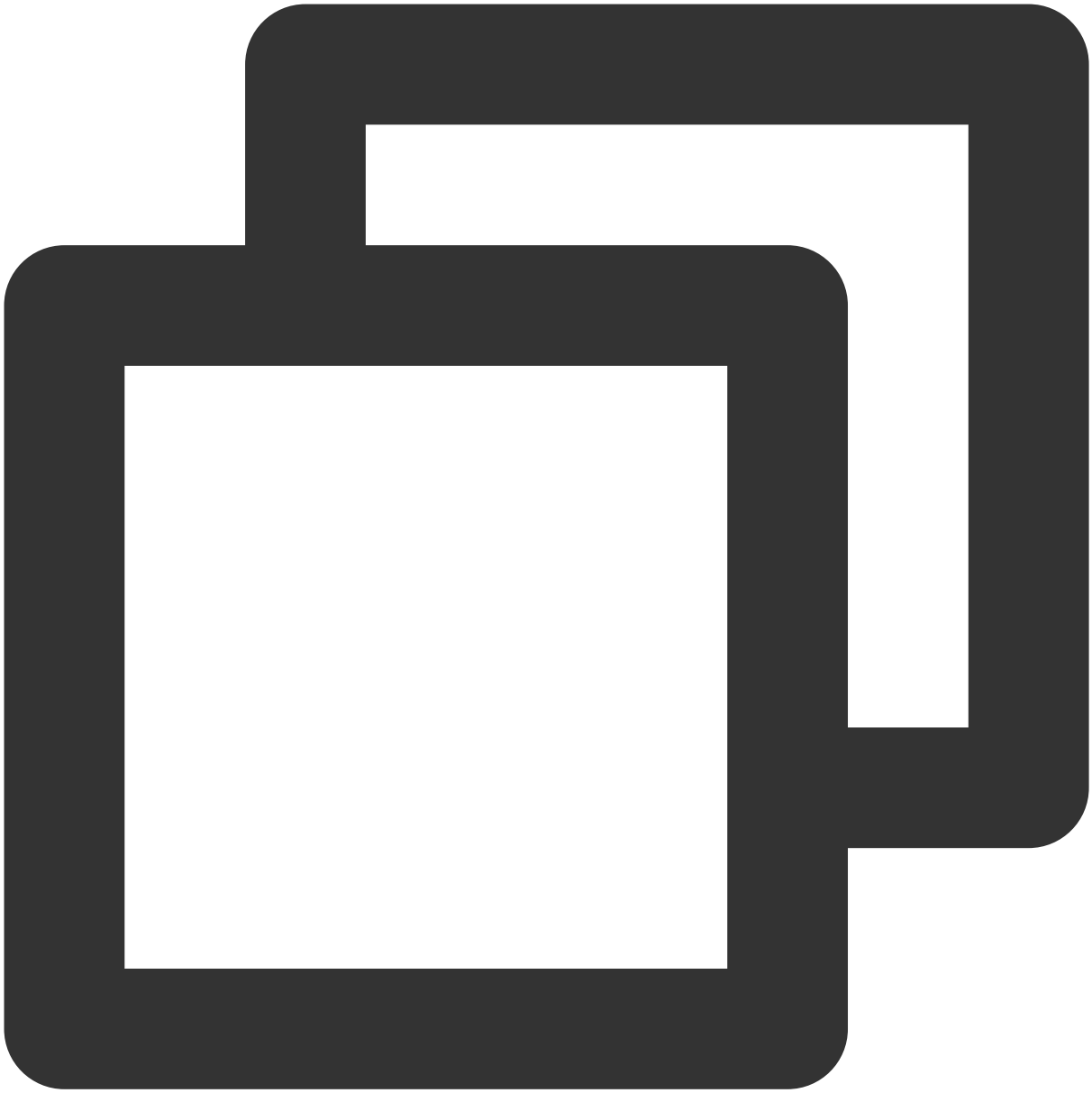
	chargeType	課金モデル：
		`TrafficPostpaidByH`
		トラフィックは1時間単位
		いです。
		`BandwidthPostpaidB`
		：帯域幅は1時間単位で
		です。
		`BandwidthPackage`：
		Bandwidth Packageの支払
		EIPでBandwidth Package
		イトリストがアクティブ
		ていることが必要です。

	maxBandwidthOut	帯域幅の上限で、単位はM す。
	bandwidthPackageID	Bandwidth Packageを指定 （帯域幅パッケージ IDを てください）。

説明

現在、高品質EIPのアカウントタイプは**標準アカウント**のみを、リージョンは**中国香港**のみを、課金モデルは**Bandwidth Package**のみをサポートしています。高品質BGP帯域幅パッケージがない場合は、VPCコンソール > **Bandwidth Package**に進み、作成してください。

YAMLの例



```
apiVersion: node.tke.cloud.tencent.com/v1beta1
kind: MachineSet
spec:
  deletePolicy: Random
  displayName: HighQualityEIP-test
  instanceTypes:
  - SA2.MEDIUM2
  replicas: 1
  scaling:
    createPolicy: ZonePriority
    maxReplicas: 4
```

```
  subnetIDs:
  - subnet-xxxxxxx
  template:
    metadata:
      labels:
        node.tke.cloud.tencent.com/machineset: np-ohh7gaek
    spec:
      providerSpec:
        type: Native
        value:
          instanceChargeType: PostpaidByHour
          lifecycle: {}
          management:
            nameservers:
            - 183.60.83.19
            - 183.60.82.98
          metadata:
            creationTimestamp: null
          securityGroupIDs:
          - sg-5lxe2r2p
          systemDisk:
            diskSize: 50
            diskType: CloudPremium
          internetAccessible:
            chargeType: BandwidthPackage
            bandwidthPackageID: bwp-95xr2686
            maxBandwidthOut: 100
            addressType: HighQualityEIP
      runtimeRootDir: /var/lib/containerd
  type: Native
```

Kubernetes Object Management
ワークロード
Deployment管理

最終更新日：：2022-04-18 18:45:25

概要

Deploymentは、Podのテンプレートを宣言し、Podの実行ポリシーを制御します。状態無しのアプリケーションのデプロイに適しています。サービスニーズに応じて、Deploymentで実行されているPodのレプリカの数、スケジューリングポリシー、更新ポリシーなどを宣言できます。

Deploymentコンソールのアクションガイド

Deploymentの作成

1. Tencent Kubernetes Engine（TKE）コンソールにログインして、左側ナビゲーションバーの【Clusters 】を選択してください。

2. Deploymentの作成を必要とするクラスターIDをクリックして、Deploymentが作成されるクラスター管理ページへ進みます。

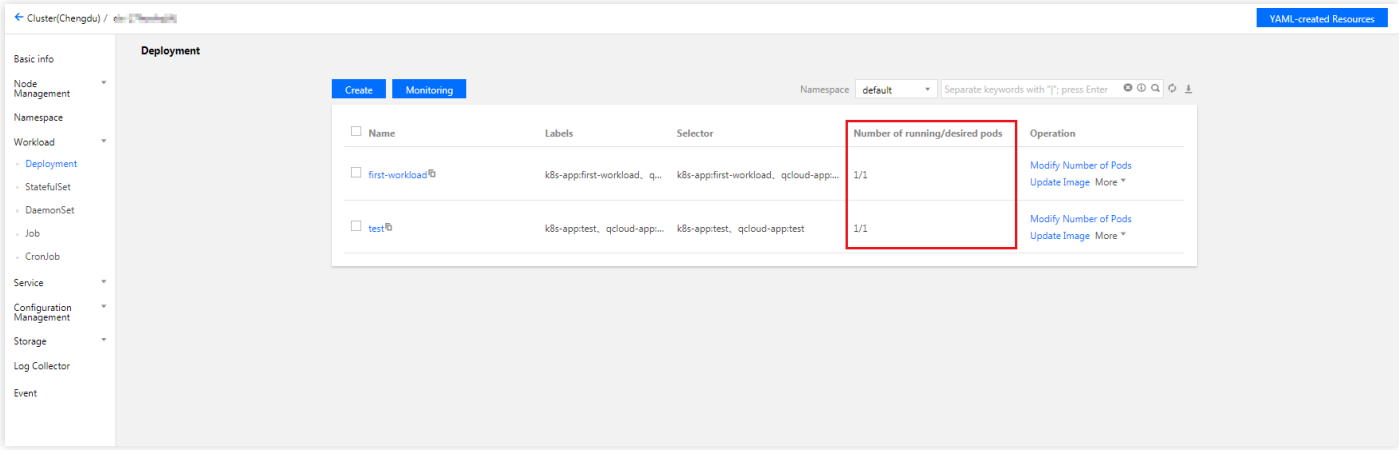
3. 【 Create 】をクリックして、「Create a workload」ページへ進みます。
   実際のニーズに応じて、Deploymentパラメータを設定します。キーパラメータの情報は以下の通りです：

- **ワークロード名**：カスタマイズされた名称を入力します。
- **ネームスペース**：実際のニーズに応じて選択します。
- **タイプ**：【Deployment（Podを拡張可能にデプロイする）】を選択します。
- **インスタンス内のコンテナ**：実際のニーズに応じて、DeploymentのPodに1つ又は複数の異なるコンテナを設定します。
  - **名称**：カスタマイズします。
  - **イメージ**：実際のニーズに応じて選択します。
  - **イメージバージョン（Tag）**：実際のニーズに応じて入力します。
  - **イメージプルポリシー**：次の3つのポリシーを提供し、ニーズに応じて選択してください。
    イメージプルポリシーが設定されていない場合は、イメージバージョンがnullまたは `latest` であるとき、

Alwaysポリシーを使用し、そうでないとき、IfNotPresentポリシーを使用します。
  - **Always**：常にリモートからこのイメージをプルします。
  - **IfNotPresent**：デフォルトではローカルイメージを使用しますが、ローカルでこのイメージがない場合は、リモートからこのイメージをプルします。
  - **Never**：ローカルイメージのみを使用します。ローカルでこのイメージがない場合は、異常を報告します。
- **CPU/メモリ制限**：Kubernetes リソース制限に基づいて、CPUおよびメモリの制限範囲を設定することができ、サービスのロバストを向上させます。
- **詳細設定**：「**作業ディレクトリ**」、「**実行コマンド**」、「**実行パラメータ**」、「**コンテナ健康診断**」および「**特権レベル**」などのパラメータを設定できます。

- **インスタンスの数**：実際のニーズに応じて調整方法を選択し、インスタンスの数を設定します。

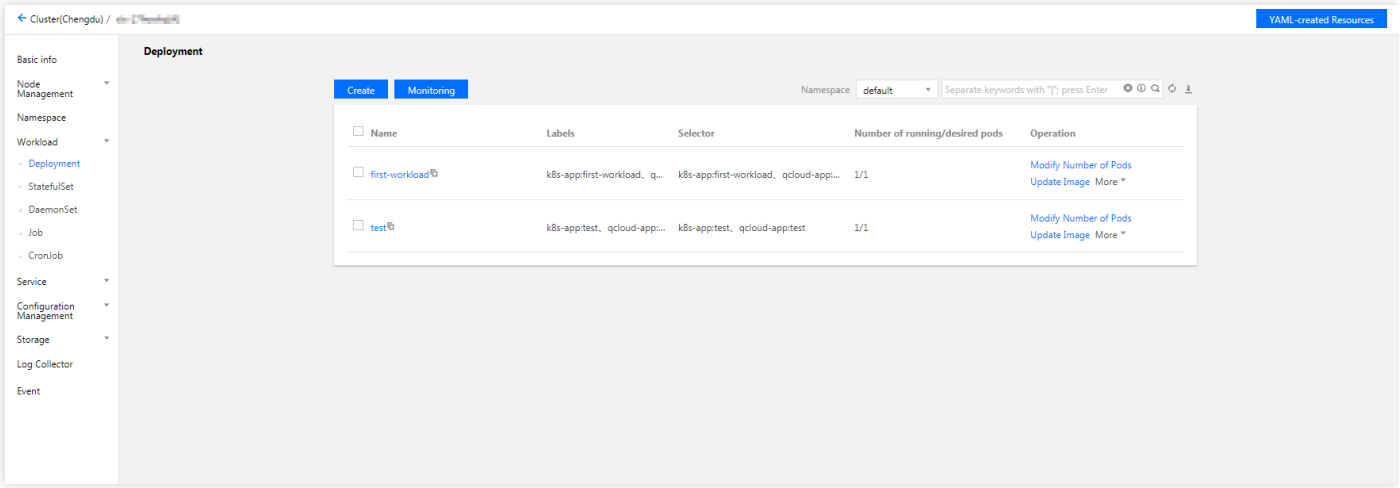
4. 【Create Workload】をクリックして、作成が完了します。下図の通りです：
実行数=目的数の場合、DeploymentでのすべてのPodが作成されたことを意味します。



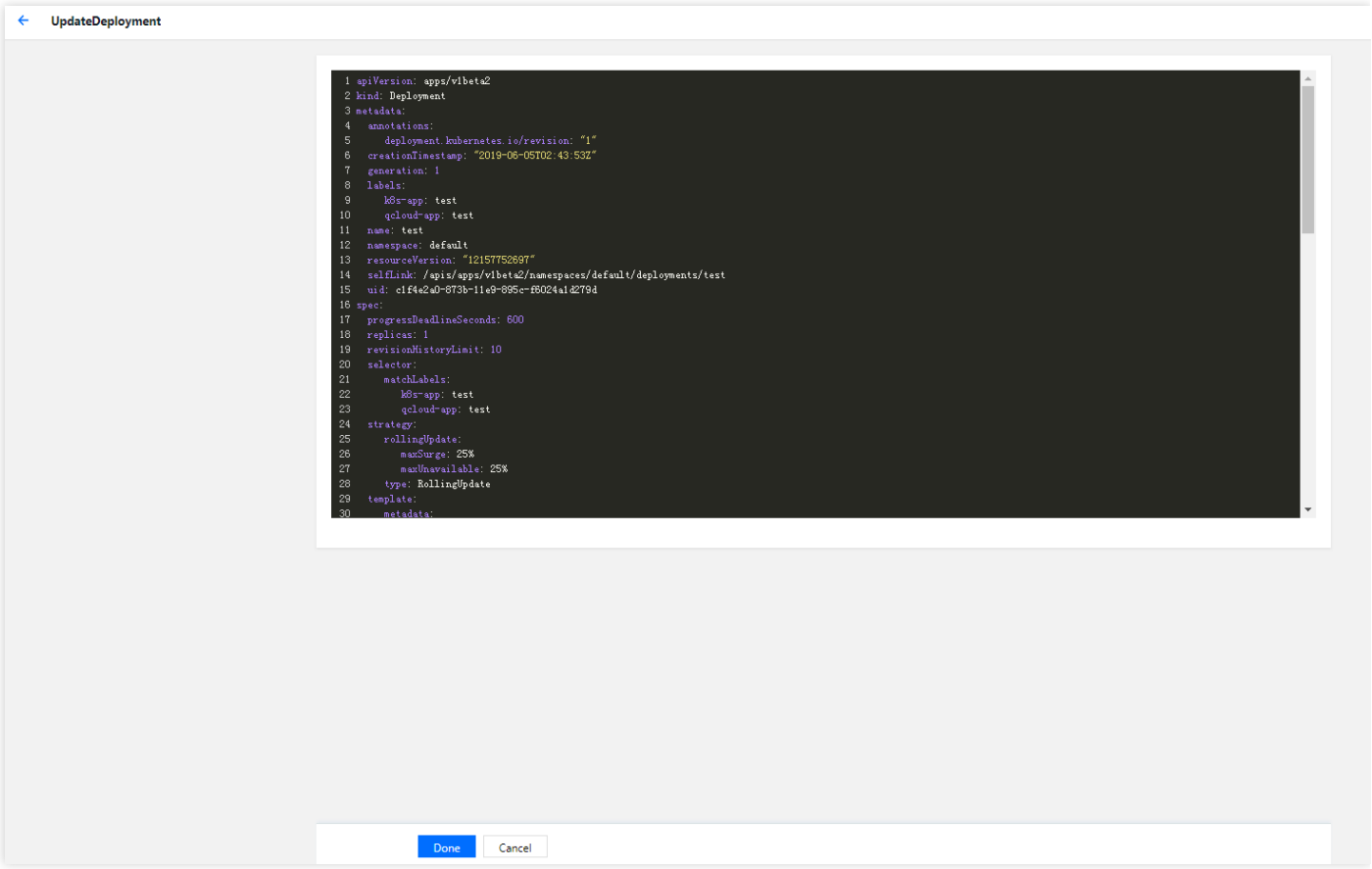
Deploymentの更新

YAMLの更新

1. TKEコンソールにログインして、左側ナビゲーションバーの【Clusters】を選択してください。
2. Deploymentの更新を必要とするクラスターIDをクリックして、Deploymentが更新されるクラスター管理ページへ進みます。下図の通りです：



3. YAMLの更新を必要とするDeployment行では、【More】>【Edit YAML】をクリックして、Deployment更新ページへ進みます。
4. 「Update Deployment」ページでYAMLを編集し、【Finish】をクリックしてYAMLが更新されます。下図の通りです：



Pod構成の更新

1. クラスター管理ページでは、Pod構成の更新を必要とするDeploymentのクラスターIDをクリックして、Pod構成が更新されるDeploymentのクラスター管理ページへ進みます。
2. Pod構成の更新を必要とするDeployment行では、【Updating Pod configurations】をクリックします。
3. 「Updating Pod configurations」ページでは、実際のニーズに応じて更新方法を変更し、パラメータを設定します。
4. 【Finish 】をクリックして、Pod構成が更新されます。

Deploymentのロールバック

1. TKEコンソールにログインして、左側ナビゲーションバーの【Clusters】を選択してください。
2. Deploymentのロールバックを必要とするクラスターIDをクリックして、Deploymentがロールバックされるクラスター管理ページへ進みます。
3. ロールバックを必要とするDeploymentの名称をクリックして、Deployment情報ページへ進みます。
4. 【 Modification History】タブを選択して、ロールバックを必要とするバージョンでは、【 Rollback 】をクリックします。
5. ポップアップ表示された「Roll back a resource」プロンプトでは、【Submit 】をクリックしてロールバックが完了します。

Pod数の調整

1. TKEコンソールにログインして、左側ナビゲーションバーの【Clusters 】を選択してください。

2. Pod数の調整を必要とするDeploymentのクラスターIDをクリックして、Pod数が調整されるDeploymentのクラスター管理ページへ進みます。

3. Pod数の調整を必要とするDeployment行では、【 Update Pod quantity 】をクリックして、Pod数の更新ページへ進みます。

4. 実際のニーズに応じてPod数を調整し、【Update Pod quantity 】をクリックして調整が完了します。

KubectlによるDeploymentのアクションガイド

YAMLの例

```
apiVersion: apps/v1beta2
kind: Deployment
metadata:
name: nginx-deployment
namespace: default
```

```
labels:
app: nginx-deployment
spec:
replicas: 3
selector:
matchLabels:
app: nginx-deployment
template:
metadata:
labels:
app: nginx-deployment
spec:
containers:
- name: nginx
image: nginx:latest
ports:
- containerPort: 80
```

- **kind**：Deploymentリソースタイプを標識します。
- **metadata**：Deploymentの名称、Namespace、Labelなどの基本情報です。
- **metadata.annotations**：Deploymentに関する追加説明であり、このパラメータを使用してTencent Cloud TKEの追加の拡張機能を設定できます。
- **spec.replicas**：Deploymentによって管理されるPod数です。
- **spec.selector**：Deployment管理Seletorによって選択されたPodのLabelです。
- **spec.template**：Deploymentによって管理されるPodの詳細なテンプレート構成です。

パラメータの詳細については、Kubernetes Deployment 公式ドキュメントで確認できます。

KubectlによるDeploymentの作成

1. YAMLの例を参照して、Deployment YAMLファイルを準備します。
2. Kubectlをインストールして、クラスターに接続します。アクションの詳細については、Kubectlによるクラスターへの接続をご参照ください。
3. 次のコマンドを実行して、Deployment YAMLファイルを作成します。

```
kubectl create -f Deployment YAMLファイル名称
```

例えば、ファイル名がnginx.YamlであるDeployment YAMLファイルを作成するには、次のコマンドを実行します：

```
kubectl create -f nginx.yaml
```

4. 次のコマンドを実行して、作成が成功したかどうかを確認します。

```
kubectl get deployments
```

下記のような情報が返された場合は、作成が成功したことを示しています。

```
NAME DESIRED CURRENT UP-TO-DATE AVAILABLE AGE
first-workload 1 1 1 0 6h
ng 1 1 1 1 42m
```

KubectlによるDeploymentの更新

KubectlによるDeploymentの更新には次の3つの方法が含まれています。その中で、方法1および方法2の両方とも、**Recreate**および**RollingUpdate**の2つの更新ポリシーをサポートします。

- Recreate更新ポリシーは、最初にすべてのPodを廃棄してから、Deploymentを再作成することです。
- RollingUpdate更新ポリシーは、DeploymentのPodを1つずつ更新するローリング更新ポリシーです。RollingUpdateは、一時停止、更新時間間隔の設定などもサポートします。

方法1

次のコマンドを実行して、Deploymentを更新します。

```
kubectl edit deployment/[name]
```

この方法は、単純なデバッグ確認に適しています。実稼働環境で直接使用することはお勧めしません。この方法を使用して、任意のDeploymentパラメータを更新できます。

方法2

次のコマンドを実行して、指定されたコンテナのイメージを更新します。

```
kubectl set image deployment/[name] [containerName]=[image:tag]
```

Deploymentの他のパラメータを変更せず、ビジネスが更新されたときに、コンテナイメージのみを更新することはお勧めします。

方法3

次のコマンドを実行して、指定されたリソースをローリング更新します。

```
kubectl rolling-update [NAME] -f FILE
```

KubectlによるDeploymentのロールバック

1. 次のコマンドを実行して、Deploymentの更新履歴を確認します。

```
kubectl rollout history deployment/[name]
```

2. 次のコマンドを実行して、指定されたバージョンの詳細を確認します。

```
kubectl rollout history deployment/[name] --revision=[REVISION]
```

3. 次のコマンドを実行して、前のバージョンへロールバックします。

```
kubectl rollout undo deployment/[name]
```

ロールバックバージョン番号を指定するには、次のコマンドを実行します。

```
kubectl rollout undo deployment/[name] --to-revision=[REVISION]
```

KubectlによるPod数の調整

Pod数の手動更新

次のコマンドを実行して、Pod数を手動で更新します。

```
kubectl scale deployment [NAME] --replicas=[NUMBER]
```

Pod数の自動更新

前提条件

クラスターのHPA機能をオンにします。TKEによって作成されたクラスターでは、デフォルトでHPA機能がオンになっています。

操作手順

次のコマンドを実行して、Deploymentの自動スケールアウトと自動スケールインを設定します。

```
kubectl autoscale deployment [NAME] --min=10 --max=15 --cpu-percent=80
```

KubectlによるDeploymentの削除

次のコマンドを実行して、Deploymentを削除します。

```
kubectl delete deployment [NAME]
```

StatefulSet管理

最終更新日：：2022-04-07 11:04:08

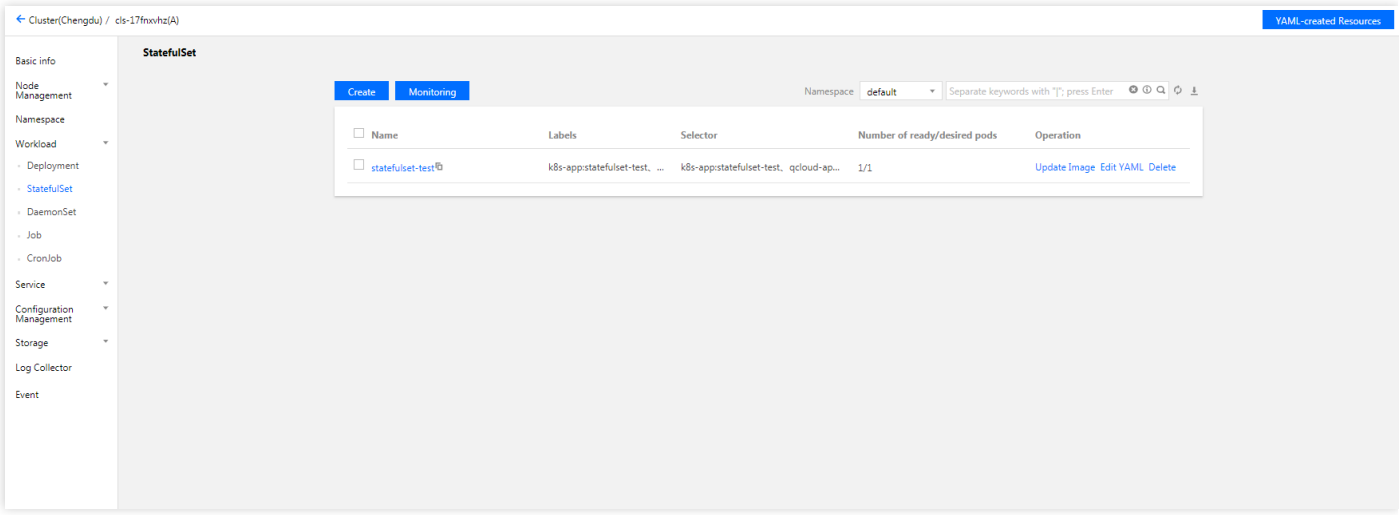
概要

StatefulSetは主に、状態ありのアプリケーションを管理するために使用されます。作成されたPodには、仕様に従って作成された持続的な識別子があります。識別子は、Podの移行または破棄のリスタート後も保持されます。持続的なストレージを必要とするとき、識別子を使用してストレージボリュームを1対1でマッピングできます。アプリケーションが持続的な識別子を必要としない場合は、Deploymentを使用してアプリケーションをデプロイすることはお勧めします。

StatefulSetコンソールのアクションガイド

StatefulSetの作成

1. Tencent Kubernetes Engine（TKE）コンソールにログインして、左側ナビゲーションバーの【 Clusters】を選択してください。
2. StatefulSetの作成を必要とするクラスターIDをクリックして、StatefulSetが作成されるクラスター管理ページへ進みます。
3. 【Workload】>【StatefulSet】を選択して、StatefulSet管理ページへ進みます。下図の通りです：



4. 【Create 】をクリックして、「 Create Workload」ページへ進みます。
   実際のニーズに応じて、StatefulSetパラメータを設定します。キーパラメータの情報は以下の通りです：

- **ワークロード名**：カスタマイズされた名称を入力します。

- **ネームスペース**：実際のニーズに応じて選択します。
- **タイプ**：【StatefulSet（状態セット有りの実行Pod）】を選択します。
- **インスタンス内のコンテナ**：実際のニーズに応じて、StatefulSetのPodに1つ又は複数の異なるコンテナを設定します。
  - **名称**：カスタマイズします。
  - **イメージ**：実際のニーズに応じて選択します。
  - **イメージバージョン（Tag）**：実際のニーズに応じて入力します。
  - **イメージプルポリシー**：次の3つのポリシーを提供し、ニーズに応じて選択してください。
    イメージプルポリシーが設定されていない場合は、イメージバージョンがnullまたは `latest` であるとき、Alwaysポリシーを使用し、そうでないとき、IfNotPresentポリシーを使用します。
    - **Always**：常にリモートからこのイメージをプルします。
    - **IfNotPresent**：デフォルトではローカルイメージを使用しますが、ローカルでこのイメージがない場合は、リモートからこのイメージをプルします。
    - **Never**：ローカルイメージのみを使用します。ローカルでこのイメージがない場合は、異常を報告します。
  - **CPU/メモリ制限**：Kubernetes リソース制限に基づいて、CPUおよびメモリの制限範囲を設定することができ、サービスのロバストを向上させます。
  - **詳細設定**：「**作業ディレクトリ**」、「**実行コマンド**」、「**実行パラメータ**」、「**コンテナ健康診断**」および「**特権レベル**」などのパラメータを設定できます。
- **インスタンスの数**：実際のニーズに応じて調整方法を選択し、インスタンスの数を設定します。

5. 【 Create a workload 】をクリックして、作成が完了します。

StatefulSetの更新

YAMLの更新

1. TKEコンソールにログインして、左側ナビゲーションバーの【Clusters】を選択してください。
2. YAMLの更新を必要とするクラスターIDをクリックして、YAMLが更新されるクラスター管理ページへ進みます。
3. 【Workload】>【StatefulSet】を選択して、StatefulSet情報ページへ進みます。
4. YAMLの更新を必要とするStatefulSet行から、【More】>【Edit YAML 】を選択して、StatefulSet更新ページへ進みます。
5. 「Update StatefulSet」ページでYAMLを編集し、【 Finish】をクリックしてYAMLが更新されます。

Pod構成の更新

1. クラスター管理ページでは、Pod構成の更新を必要とするStatefulSetのクラスターIDをクリックして、Pod構成が更新されるStatefulSetのクラスター管理ページへ進みます。
2. Pod構成の更新を必要とするStatefulSet行では、【Update Pod Configurations.】をクリックします。

3. 「Update Pod Configurations.」ページでは、実際のニーズに応じて更新方法を変更し、パラメータを設定します。
4. 【Finish】をクリックして、Pod構成が更新されます。

KubectlによるStatefulSetのアクションガイド

YAMLの例

```
apiVersion: v1
kind: Service ## ネットワークドメインをコントロールするためのHeadless Serviceを作成する
metadata:
name: nginx
namespace: default
labels:
app: nginx
spec:
ports:
- port: 80
name: web
clusterIP: None
selector:
app: nginx
---
apiVersion: apps/v1
kind: StatefulSet ### NginxのStatefulSetを作成する
metadata:
name: web
namespace: default
spec:
selector:
matchLabels:
app: nginx
serviceName: "nginx"
replicas: 3 # by default is 1
template:
metadata:
labels:
app: nginx
spec:
terminationGracePeriodSeconds: 10
containers:
- name: nginx
image: nginx:latest
ports:
```

```
- containerPort: 80
name: web
volumeMounts:
- name: www
mountPath: /usr/share/nginx/html
volumeClaimTemplates:
- metadata:
name: www
spec:
accessModes: [ "ReadWriteOnce" ]
storageClassName: "cbs"
resources:
requests:
storage: 10Gi
```

- **kind**：StatefulSetのリソースタイプを標識します。
- **metadata**：StatefulSetの名称、Labelなどの基本情報です。
- **metadata.annotations**：StatefulSetに関する追加説明であり、このパラメータを使用してTencent Cloud TKEの追加の拡張機能を設定できます。
- **spec.template**：StatefulSetによって管理されるPodの詳細なテンプレート構成です。
- **spec.volumeClaimTemplates**：PVC&PVを作成するテンプレートを提供します。

パラメータの詳細については、Kubernetes StatefulSet 公式ドキュメントで確認できます。

StatefulSetの作成

1. YAMLの例を参照して、StatefulSet YAMLファイルを準備します。
2. Kubectlをインストールして、クラスターに接続されます。アクションの詳細については、Kubectlによるクラスターへの接続をご参照ください。
3. 次のコマンドを実行して、StatefulSet YAMLファイルを作成します。

   ```
   kubectl create -f StatefulSet YAMLファイル名称
   ```

例えば、ファイル名がweb.yamlであるStatefulSet YAMLファイルを作成するには、次のコマンドを実行します：

```
kubectl create -f web.yaml
```

4. 次のコマンドを実行して、作成が成功したかどうかを確認します。

   ```
   kubectl get StatefulSet
   ```

下記のような情報が返された場合は、作成が成功したことを示しています。

```
NAME DESIRED CURRENT AGE
test 1 1 10s
```

StatefulSetの更新

次のコマンドを実行して、StatefulSetの更新ポリシータイプを確認します。

```
kubectl get ds/<daemonset-name> -o go-template='{{.spec.updateStrategy.type}}{{
"\n"}}'
```

StatefulSetには、次の2つの更新ポリシータイプが含まれています：

- OnDelete：デフォルトの更新ポリシー。この更新ポリシーでStatefulSetを更新した後、新しいStatefulSet Podを作成する前に、古いStatefulSet Podを手動で削除する必要があります。
- RollingUpdate：Kubernetes 1.7以降のバージョンをサポートします。この更新ポリシーでStatefulSetテンプレートを更新した後、古いStatefulSet Podが終了し、ローリング更新方式で新しいStatefulSet Pod（Kubernetes 1.7以降のバージョン）を作成します。

方法1

次のコマンドを実行して、StatefulSetを更新します。

```
kubectl edit StatefulSet/[name]
```

この方法は、単純なデバッグ確認に適しています。実稼働環境で直接使用することはお勧めしません。この方法を使用して、任意のStatefulSetパラメータを更新できます。

方法2

次のコマンドを実行して、指定されたコンテナのイメージを更新します。

```
kubectl patch statefulset <NAME> --type='json' -p='[{"op": "replace", "path": "/s
pec/template/spec/containers/0/image", "value":"<newImage>"}]'
```

StatefulSetの他のパラメータを変更せず、ビジネスが更新されたときに、コンテナイメージのみを更新することはお勧めします。

更新されたStatefulSetがローリング更新ポリシーである場合は、次のコマンドを実行して更新状態を表示できます。

```
kubectl rollout status sts/<StatefulSet-name>
```

StatefulSetの削除

次のコマンドを実行して、StatefulSetを削除します。

```
kubectl delete StatefulSet [NAME] --cascade=false
```

--cascade=falseパラメータは、KubernetesがStatefulSetのみを削除し、Podを削除しないことを意味します。Podを削除するには、次のコマンドを実行します：

```
kubectl delete StatefulSet [NAME]
```

StatefulSetに関するアクションの詳細については、Kubernetes公式ガイドで確認できます。

DaemonSet管理

最終更新日：：2022-04-07 11:04:08

概要

DaemonSetは主に、ノードログ収集などの常駐クラスターに存在するバックグラウンドプログラムをデプロイするために使用されます。DaemonSetは、指定されたPodがすべてまたは一部のノードで実行されていることを保証します。Podは、新しいノードがクラスターに追加されたときにも自動的にデプロイされます。Podは、ノードがクラスターから削除された後に自動的に回収されます。

スケジューリングの説明

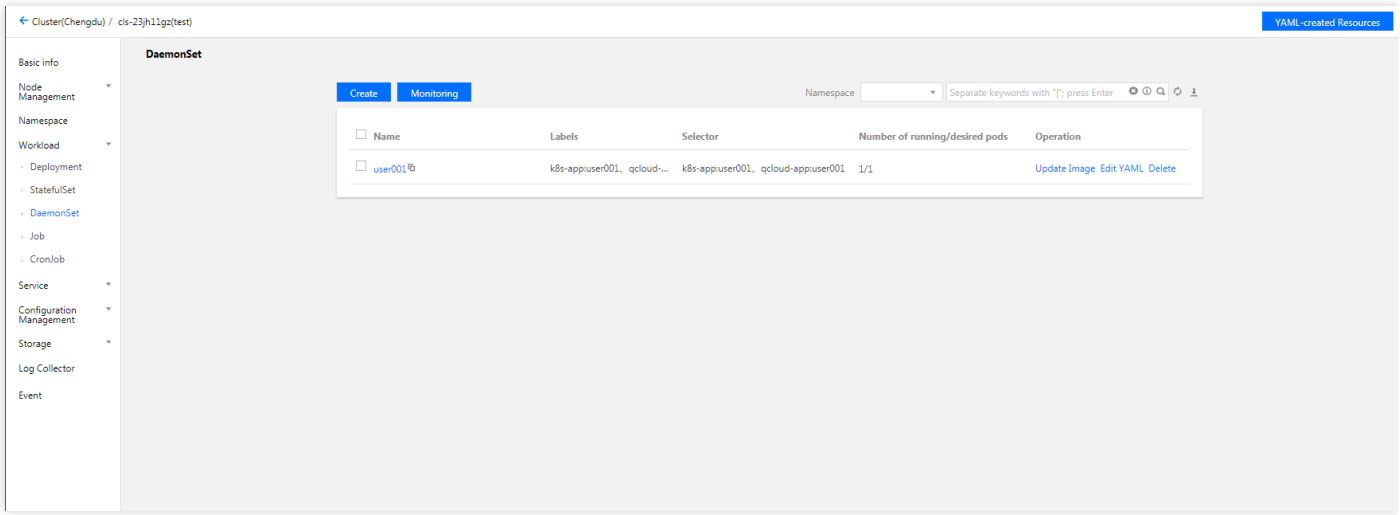
PodのnodeSelectorまたはaffinityパラメータが構成されている場合は、DaemonSetによって管理されるPodは、指定されたスケジューリングルールに従ってスケジューリングされます。PodのnodeSelectorまたはaffinityパラメータが構成されていない場合、Podはすべてのノードにデプロイされます。

DaemonSetコンソールのアクションガイド

DaemonSetの作成

1. Tencent Kubernetes Engine（TKE）コンソールにログインして、左側ナビゲーションバーの【Cluster】を選択してください。
2. DaemonSetの作成を必要とするクラスターIDをクリックして、DaemonSetが作成されるクラスター管理ページへ進みます。

3. 【Workload】>【DaemonSet】を選択して、DaemonSet情報ページへ進みます。以下の通りです：



4. 【 Create 】をクリックして、「 Create Workload」ページへ進みます。
実際のニーズに応じて、DaemonSetパラメータを設定します。キーパラメータの情報は以下の通りです。

- **ワークロード名**：カスタマイズされた名称を入力します。
- **ネームスペース**：実際のニーズに応じて選択します。
- **タイプ**：【DaemonSet（各ホストでPodを実行する）】を選択します。
- **インスタンス内のコンテナ**：実際のニーズに応じて、DaemonSetのPodに1つ又は複数の異なるコンテナを設定します。
  - **名称**：カスタマイズします。
  - **イメージ**：実際のニーズに応じて選択します。
  - **イメージバージョン（Tag）**：実際のニーズに応じて入力します。
  - **イメージプルポリシー**：次の3つのポリシーを提供し、ニーズに応じて選択してください。
    イメージプルポリシーが設定されていない場合は、イメージバージョンがnullまたは `latest` であるとき、Alwaysポリシーを使用し、そうでないとき、IfNotPresentポリシーを使用します。
    - **Always**：常にリモートからこのイメージをプルします。
    - **IfNotPresent**：デフォルトではローカルイメージを使用しますが、ローカルでこのイメージがない場合は、リモートからこのイメージをプルします。
    - **Never**：ローカルイメージのみを使用します。ローカルでこのイメージがない場合は、異常を報告します。
  - **CPU/メモリ制限**：Kubernetes リソース制限に基づいて、CPUおよびメモリの制限範囲を設定することができ、サービスのロバストを向上させます。
  - 詳細設定：「**作業ディレクトリ**」、「**実行コマンド**」、「**実行パラメータ**」、「**コンテナ健康診断**」、「**特権レベル**」などのパラメータを設定できます。

5. 【Create a workload】をクリックして、作成が完了します。

DaemonSetの更新

YAMLの更新

1. TKEコンソールにログインして、左側ナビゲーションバーの【Cluster】を選択してください。
2. YAMLの更新を必要とするクラスターIDをクリックして、YAMLが更新されるクラスター管理ページへ進みます。
3. 【Workload】>【DaemonSet】を選択して、DaemonSet情報ページへ進みます。
4. YAMLの更新を必要とするDaemonSet行から、【More 】>【Edit YAML】を選択して、DaemonSet更新ページへ進みます。
5. 「Update a DaemonSet」ページでYAMLを編集し、【 Finish】をクリックしてYAMLが更新されます。

Pod構成の更新

> DaemonSetローリング更新機能は、Kubernetes 1.6以降のバージョンでのみサポートされます。

1. クラスター管理ページでは、Pod構成の更新を必要とするDaemonSetのクラスターIDをクリックして、Pod構成が更新されるDaemonSetのクラスター管理ページへ進みます。
2. Pod構成の更新を必要とするDaemonSet行では、【 Update Pod Configurations】をクリックします。
3. 「 Update Pod Configurations」ページでは、実際のニーズに応じて更新方法を変更し、パラメータを設定します。
4. 【Finish 】をクリックして、Pod構成が更新されます。

KubectlによるDaemonSetのアクションガイド

YAMLの例

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
name: fluentd-elasticsearch
namespace: kube-system
labels:
k8s-app: fluentd-logging
spec:
selector:
matchLabels:
name: fluentd-elasticsearch
template:
metadata:
```

```
labels:
name: fluentd-elasticsearch
spec:
tolerations:
- key: node-role.kubernetes.io/master
effect: NoSchedule
containers:
- name: fluentd-elasticsearch
image: k8s.gcr.io/fluentd-elasticsearch:1.20
resources:
limits:
memory: 200Mi
requests:
cpu: 100m
memory: 200Mi
volumeMounts:
- name: varlog
mountPath: /var/log
- name: varlibdockercontainers
mountPath: /var/lib/docker/containers
readOnly: true
terminationGracePeriodSeconds: 30
volumes:
- name: varlog
hostPath:
path: /var/log
- name: varlibdockercontainers
hostPath:
path: /var/lib/docker/containers
```

上記のYAMLの例は `https://kubernetes.io/docs/concepts/workloads/controllers/daemonset` から引用されています。作成中にコンテナイメージのプルが失敗する場合があります。このドキュメントでDaemonSetの構成を紹介するためにのみ使用されます。

- **kind**：DaemonSetリソースタイプを標識します。
- **metadata**：DaemonSetの名称、Labelなどの基本情報です。
- **metadata.annotations**：DaemonSetに関する追加説明であり、このパラメータを使用してTencent Cloud TKEの追加の拡張機能を設定できます。
- **spec.template**：DaemonSetによって管理されるPodの詳細なテンプレート構成です。

詳細については、Kubernetes DaemonSet 公式ドキュメントで確認できます。

KubectlによるDaemonSetの作成

1. YAMLの例を参照して、StatefulSet YAMLファイルを準備します。
2. Kubectlをインストールして、クラスターに接続されます。アクションの詳細については、Kubectlによるクラスターへの接続をご参照ください。
3. 次のコマンドを実行して、DaemonSet YAMLファイルを作成します。

```
kubectl create -f DaemonSet YAMLファイル名称
```

例えば、ファイル名がfluentd-elasticsearch.yamlであるStatefulSet YAMLファイルを作成するには、次のコマンドを実行します：

```
kubectl create -f fluentd-elasticsearch.yaml
```

4. 次のコマンドを実行して、作成が成功したかどうかを確認します。

```
kubectl get DaemonSet
```

下記のような情報が返された場合は、作成が成功したことを示しています。

```
NAME DESIRED CURRENT READY UP-TO-DATE AVAILABLE NODE SELECTOR AGE
frontend 0 0 0 0 0 app=frontend-node 16d
```

KubectlによるDaemonSetの更新

次のコマンドを実行して、DaemonSetの更新ポリシータイプを確認します。

```
kubectl get ds/<daemonset-name> -o go-template='{{.spec.updateStrategy.type}}{{
"\n"}}'
```

DaemonSetには、次の2つの更新ポリシータイプが含まれています：

- OnDelete：デフォルトの更新ポリシー。この更新ポリシーでDaemonSetを更新した後、新しいDaemonSet Podを作成する前に、古いDaemonSet Podを手動で削除する必要があります。
- RollingUpdate：Kubernetes 1.6以降のバージョンをサポートします。この更新ポリシーでDaemonSetテンプレートを更新した後、古いDaemonSet Podが終了し、ローリング更新方式で新しいDaemonSet Podを作成します。

方法1

次のコマンドを実行して、DaemonSetを更新します。

```
kubectl edit DaemonSet/[name]
```

この方法は、単純なデバッグ確認に適しています。実稼働環境で直接使用することはお勧めしません。この方法を使用して、任意のDaemonSetパラメータを更新できます。

方法2

次のコマンドを実行して、指定されたコンテナのイメージを更新します。

```
kubectl set image ds/[daemonset-name][container-name]=[container-new-image]
```

DaemonSetの他のパラメータを変更せず、ビジネスが更新されたときに、コンテナイメージのみを更新することはお勧めします。

KubectlによるDaemonSetのロールバック

1. 次のコマンドを実行して、DaemonSetの更新履歴を確認します。

```
kubectl rollout history daemonset /[name]
```

2. 次のコマンドを実行して、指定されたバージョンの詳細を確認します。

```
kubectl rollout history daemonset /[name] --revision=[REVISION]
```

3. 次のコマンドを実行して、前のバージョンへロールバックします。

```
kubectl rollout undo daemonset /[name]
```

ロールバックバージョン番号を指定するには、次のコマンドを実行できます。

```
kubectl rollout undo daemonset /[name] --to-revision=[REVISION]
```

KubectlによるDaemonSetの削除

次のコマンドを実行して、DaemonSetを削除します。

```
kubectl delete DaemonSet [NAME]
```

Job管理

最終更新日：：2022-04-07 11:04:08

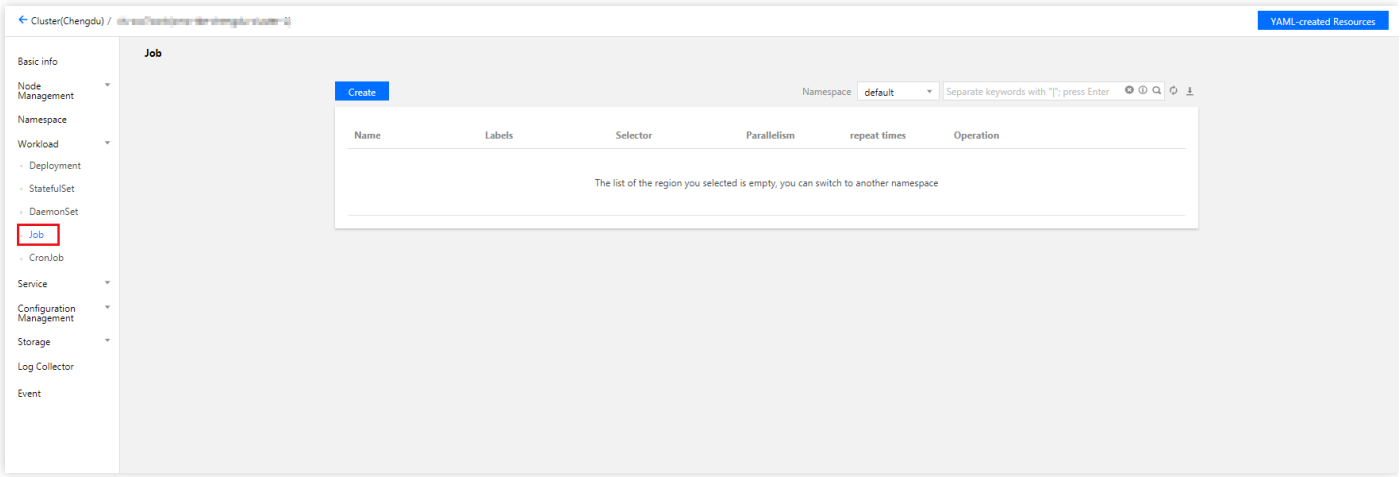
概要

Jobコンソールは、実行が終了するまで実行ルールに従って実行されている1-NのPodを作成します。Jobは、BatchComputeやデータ分析などのシナリオで使用できます。繰り返し実行の回数、並列性およびリスタートポリシーを設定することにより、ビジネス要件を満たします。
Jobが実行された後、新しいPodの作成またはPodの削除は行いません。「ログ」では、完了したPodのログを確認できます。Jobが削除されると、Jobによって作成されたPodも削除され、Jobによって作成されたPodのログを表示できなくなります。

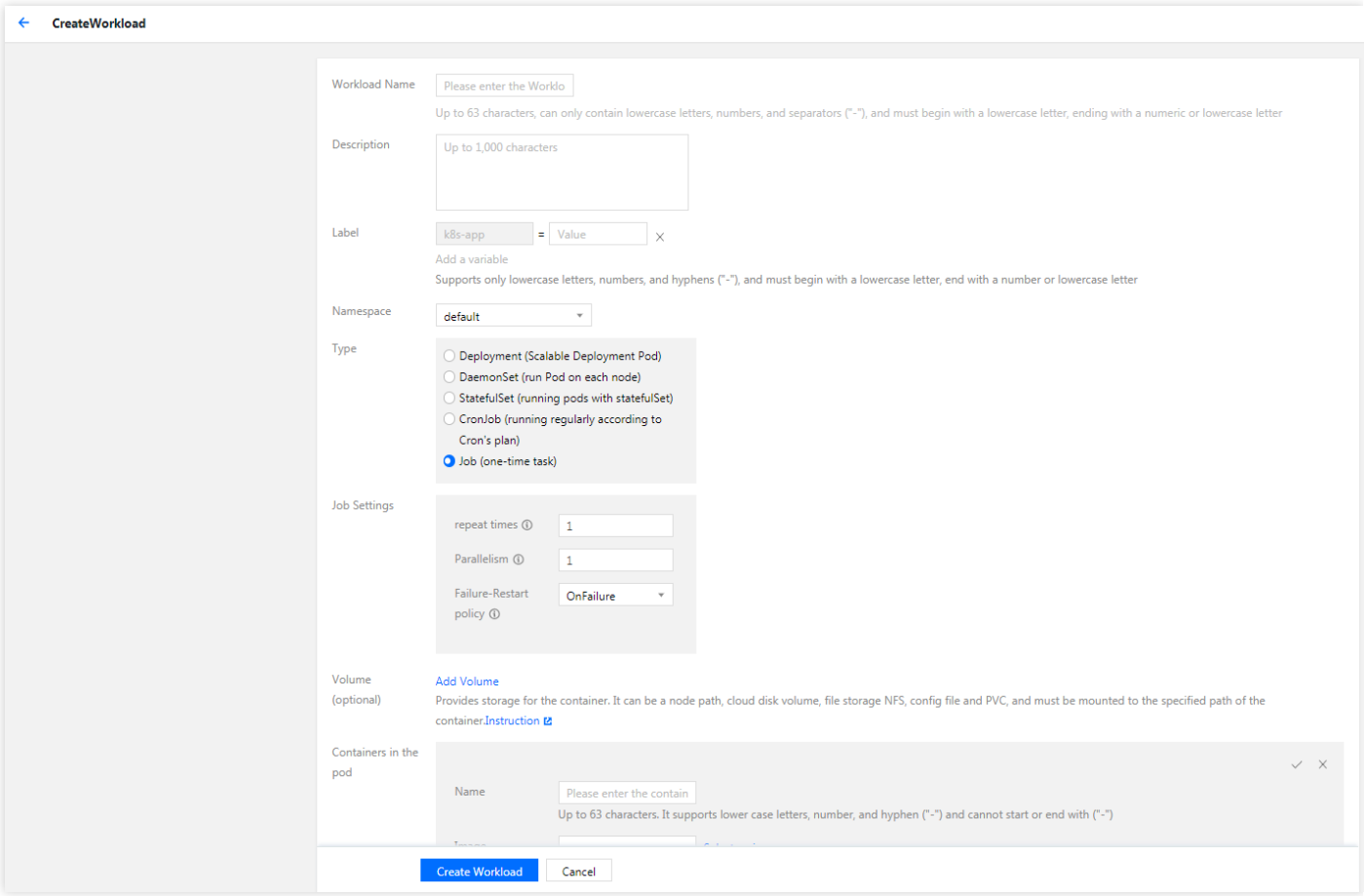
Jobコンソールのアクションガイド

Jobの作成

1. Tencent Kubernetes Engineコンソールにログインします。
2. 左側ナビゲーションバーでは、【 Cluster】をクリックして、クラスター管理ページへ進みます。
3. Jobの作成を必要とするクラスターIDをクリックして、Jobが作成されるクラスター管理ページへ進みます。
4. 「Workload」 > 「Job」を選択して、Job情報ページへ進みます。以下の通りです。



5. 【Create】をクリックして、「Create Workload」ページへ進みます。下図の通りです：



6. 実際のニーズに応じて、Jobパラメータを設定します。キーパラメータの情報は以下の通りです：

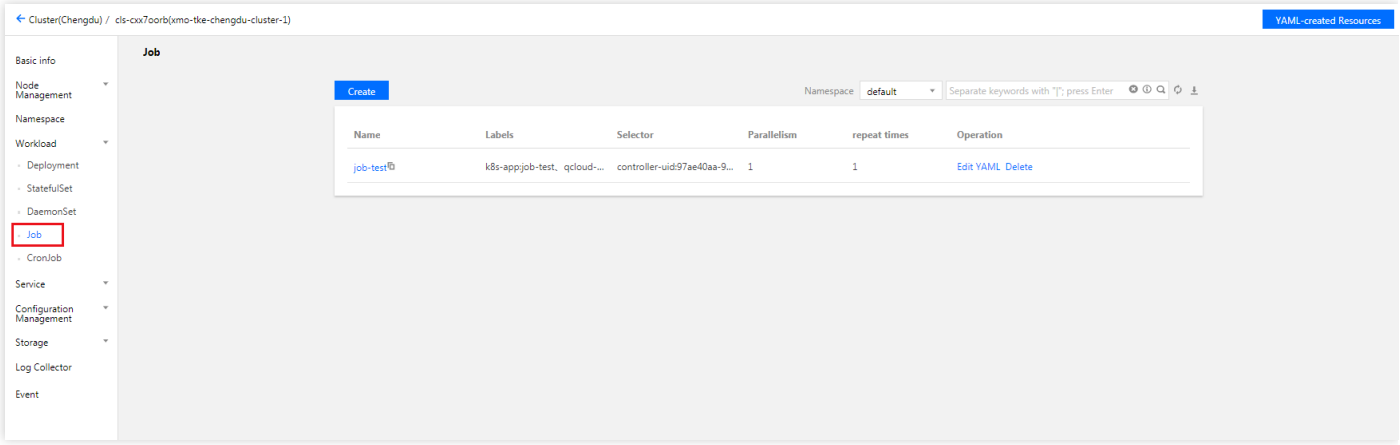
- ワークロード名：カスタマイズします。
- ネームスペース：実際のニーズに応じて選択します。
- タイプ：「Job（シングルタスク）」を選択します。
- Jobの設定：実際のニーズに応じて、JobのPodに1つ又は複数の異なるコンテナを設定します。
  - 繰り返し回数：Jobによって管理されるPodを繰り返し実行する必要がある回数を設定します。
  - 並列性：Jobを並列に実行するPodの数を設定します。
  - 失敗リスタートポリシー：Podではコンテナが異常終了した後のリスタートポリシーを設定します。
    - Neverオプション：Podではすべてのコンテナが終了するまで、コンテナがリスタートされません。
    - OnFailureオプション：Podが実行し続けて、コンテナがリスタートされます。
- インスタンス内のコンテナ：実際のニーズに応じて、JobのPodに1つ又は複数の異なるコンテナを設定します。
  - 名称：カスタマイズします。
  - イメージ：実際のニーズに応じて選択します。
  - イメージバージョン：実際のニーズに応じて入力します。

- CPU/メモリ制限：Kubernetes リソース制限に基づいて、CPUおよびメモリの制限範囲を設定することができ、サービスのロバストを向上させます。
- 詳細設定：「**作業ディレクトリ**」、「**実行コマンド**」、「**実行パラメータ**」、「**コンテナ健康診断**」、「**特権レベル**」などのパラメータを設定できます。

7. 【Create Workload】をクリックして、作成が完了します。

Job状態の確認

1. Tencent Kubernetes Engineコンソールにログインします。
2. 左側ナビゲーションバーでは、【Cluster】をクリックして、クラスター管理ページへ進みます。
3. Job状態の確認を必要とするクラスターIDをクリックして、Job状態が確認されるクラスター管理ページへ進みます。
4. 「Workload」 > 「Job」を選択して、Job情報ページへ進みます。下図の通りです：



5. 状態の確認を必要とするJob名称をクリックして、Jobの詳細を確認できます。

Jobの削除

Jobが実行された後、新しいPodの作成またはPodの削除は行いません。「ログ」では、完了したPodのログを確認できます。Jobが削除されると、Jobによって作成されたPodも削除され、Jobによって作成されたPodのログを表示できなくなります。

KubectlによるJobのアクションガイド

YAMLの例

```
apiVersion: batch/v1
kind: Job
metadata:
```

```
name: pi
spec:
completions: 2
parallelism: 2
template:
spec:
containers:
- name: pi
image: perl
command: ["perl", "-Mbignum=bpi", "-wle", "print bpi(2000)"]
restartPolicy: Never
backoffLimit: 4
```

- kind：Jobのリソースタイプを標識します。
- metadata：Jobの名称、Labelなどの基本情報です。
- metadata.annotations：Jobに関する追加説明であり、このパラメータを使用してTencent Cloud TKEの追加の拡張機能を設定できます。
- spec.completions：Jobによって管理されるPodを繰り返し実行する回数です。
- spec.parallelism：Jobを並列に実行するPodの数です。
- spec.template：Jobによって管理されるPodの詳細なテンプレート構成です。

Jobの作成

1. YAMLの例を参照して、Job YAMLファイルを準備します。
2. Kubectlをインストールして、クラスターに接続されます。アクションの詳細については、Kubectlによるクラスターへの接続をご参照ください。
3. Job YAMLファイルを作成します。

   ```
   kubectl create -f Job YAMLファイル名称
   ```

例えば、ファイル名がpi.yamlであるJob YAMLファイルを作成するには、次のコマンドを実行します。

```
kubectl create -f pi.yaml
```

4. 次のコマンドを実行して、作成が成功したかどうかを確認します。

   ```
   kubectl get job
   ```

下記のような情報が返された場合は、作成が成功したことを示しています。

```
NAME DESIRED SUCCESSFUL AGE
job 1 0 1m
```

Jobの削除

次のコマンドを実行して、Jobを削除します。

```
kubectl delete job [NAME]
```

CronJob管理

最終更新日：：2022-04-07 11:04:08

概要

CronJobオブジェクトは、crontab（cron table）ファイルの行に類似します。指定されたスケジュールに従って定期的にJobを実行して、フォーマットはCronをご参照ください。
Cronのフォーマットは次のように説明されます。

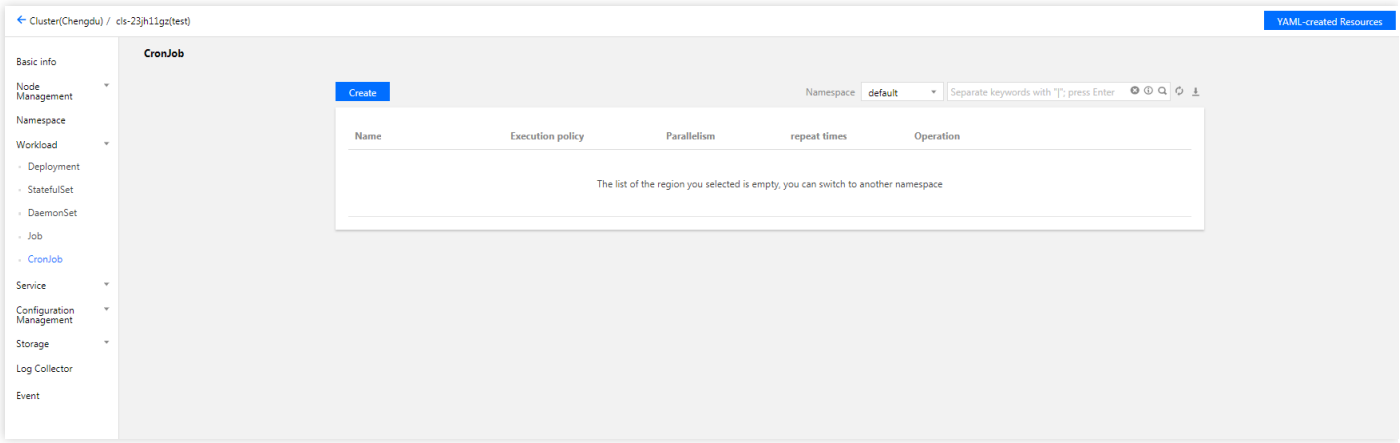
```
# ファイルフォーマットの説明
# ┌──分（0 - 59）
# │ ┌──時（0 - 23）
# │ │ ┌──日（1 - 31）
# │ │ │ ┌──月（1 - 12）
# │ │ │ │ ┌──曜日（0 - 6）
# │ │ │ │ │
# * * * * *
```

CronJobコンソールのアクションガイド

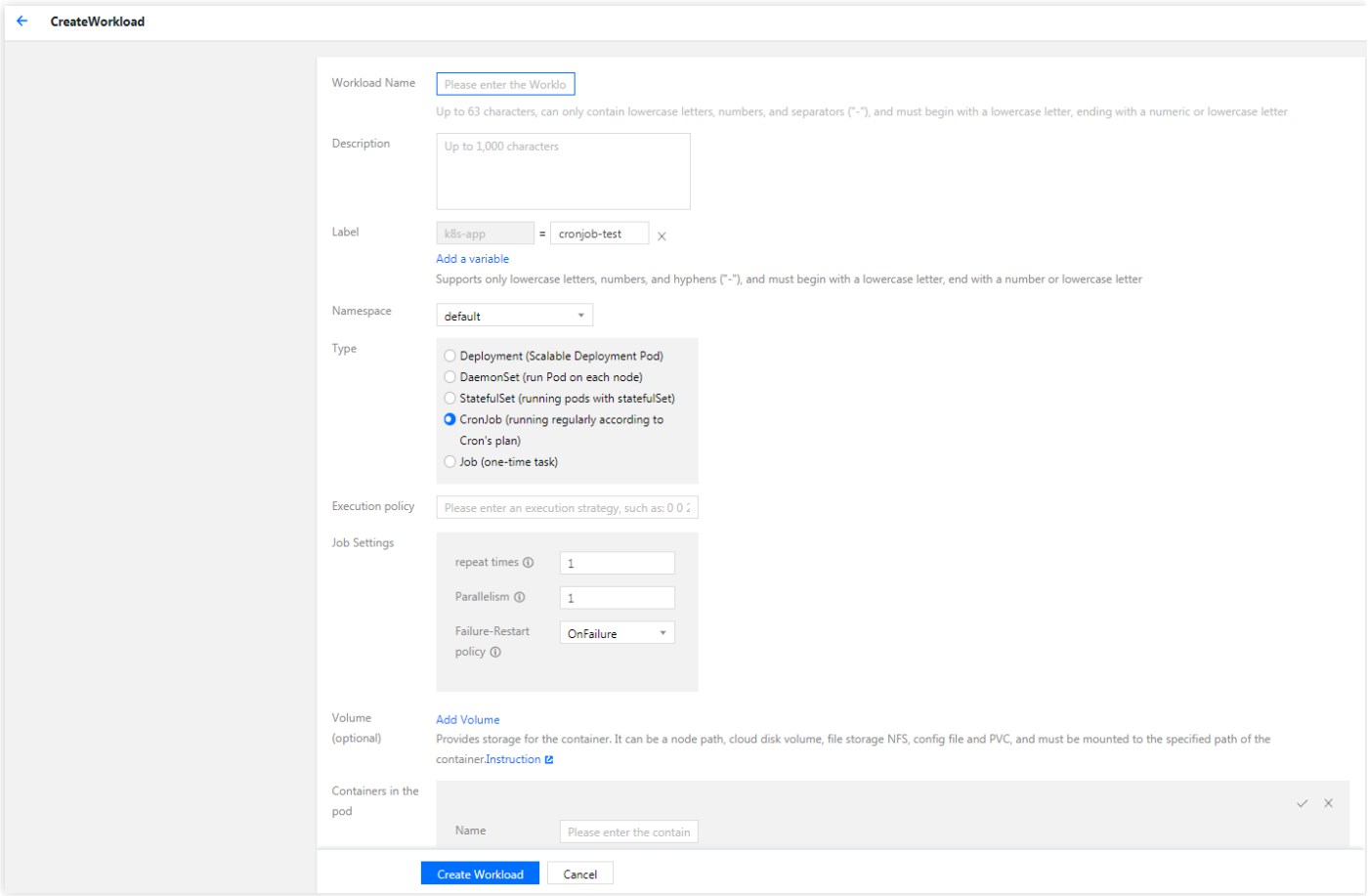
CronJobの作成

1. Tencent Kubernetes Engineコンソールにログインします。
2. 左側ナビゲーションバーでは、【Cluster】をクリックして、クラスター管理ページへ進みます。
3. CronJobの作成を必要とするクラスターIDをクリックして、CronJobが作成されるクラスター管理ページへ進みます。

4. 「Workload」 > 「CronJob」を選択して、CronJob情報ページへ進みます。以下の通りです：



5. 【Create】をクリックして、「Create workload」ページへ進みます。以下の通りです：



6. 実際のニーズに応じて、CronJobパラメータを設定します。キーパラメータの情報は以下の通りです：

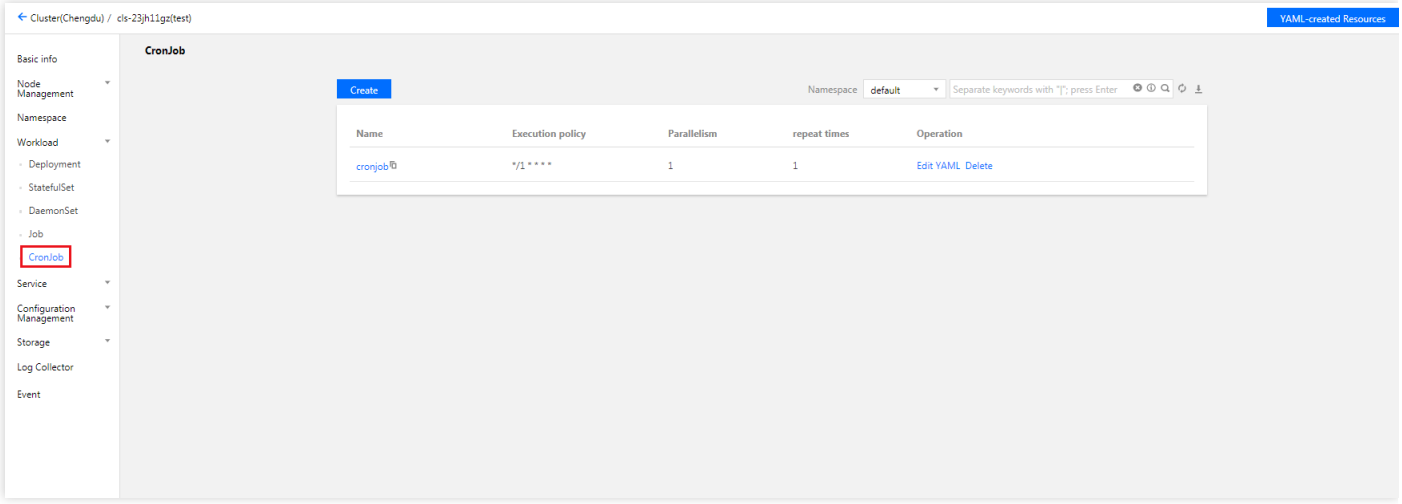
- ワークロード名：カスタマイズします。
- ネームスペース：実際のニーズに応じて選択します。
- タイプ：「CronJob（Cronのスケジュールに従って定時に実行する）」を選択します。

- 実行ポリシー：Cronフォーマットに基づいてタスクを設定する定期的な実行ポリシーです。
- Jobの設定
  - 繰り返し回数：Jobによって管理されるPodを繰り返し実行する必要がある回数です。
  - 並列性：Jobを並列に実行するPodの数です。
  - 失敗リスタートポリシー：Podではコンテナが異常終了した後のリスタートポリシーです。
    - Never：Podではすべてのコンテナが終了するまで、コンテナがリスタートされません。
    - OnFailure：Podが実行し続けて、コンテナがリスタートされます。
- インスタンス内のコンテナ：実際のニーズに応じて、CronJobのPodに1つ又は複数の異なるコンテナを設定します。
  - 名称：カスタマイズします。
  - イメージ：実際のニーズに応じて選択します。
  - イメージバージョン：実際のニーズに応じて入力します。
  - CPU/メモリ制限：Kubernetes リソース制限に基づいて、CPUおよびメモリの制限範囲を設定することができ、サービスのロバストを向上させます。
  - 詳細設定：「**作業ディレクトリ**」、「**実行コマンド**」、「**実行パラメータ**」、「**コンテナ健康診断**」、「**特権レベル**」などのパラメータを設定できます。

7. 【Create workload】をクリックして、作成が完了します。

CronJob状態の確認

1. Tencent Kubernetes Engineコンソールにログインします。
2. 左側ナビゲーションバーでは、【Cluster】をクリックして、クラスター管理ページへ進みます。
3. CronJob状態の確認を必要とするクラスターIDをクリックして、CronJob状態が確認されるクラスター管理ページへ進みます。
4. 「Workload」 > 「CronJob」を選択して、CronJob情報ページへ進みます。以下の通りです：



5. 状態の確認を必要とするCronJob名称をクリックして、CronJobの詳細を確認できます。

KubectlによるCronJobのアクションガイド

YAMLの例

```
apiVersion: batch/v1beta1
kind: CronJob
metadata:
name: hello
spec:
schedule: "*/1 * * * *"
jobTemplate:
spec:
template:
spec:
containers:
- name: hello
image: busybox
args:
- /bin/sh
- -c
- date; echo Hello from the Kubernetes cluster
restartPolicy: OnFailure
```

- kind：CronJobリソースタイプを標識します。
- metadata：CronJobの名称、Labelなどの基本情報です。
- metadata.annotations：CronJobに関する追加説明であり、このパラメータを使用してTencent Cloud TKEの追加の拡張機能を設定できます。
- spec.schedule：CronJobによって実行されるCronのポリシーです。
- spec.jobTemplate：Cronによって実行されるJobのテンプレートです。

CronJobの作成

方法1

1. YAMLの例を参照して、CronJob YAMLファイルを準備します。
2. Kubectlをインストールして、クラスターに接続されます。アクションの詳細については、Kubectlによるクラスターへの接続をご参照ください。
3. 次のコマンドを実行して、CronJob YAMLファイルを作成します。

```
kubectl create -f CronJob YAMLファイル名称
```

例えば、ファイル名がcronjob.yamlであるCronJob YAMLファイルを作成するには、次のコマンドを実行します。

```
kubectl create -f cronjob.yaml
```

方法2

1. `kubectl run` コマンドを速やかに実行することによって、CronJobを速やかに作成します。
   例えば、完全な構成情報を書き込む必要がないCronJobを速やかに作成するには、次のコマンドを実行します：

   ```
   kubectl run hello --schedule="*/1 * * * *" --restart=OnFailure --image=busybox
   -- /bin/sh -c "date; echo Hello"
   ```

2. 次のコマンドを実行して、作成が成功したかどうかを確認します。

   ```
   kubectl get cronjob [NAME]
   ```

下記のような情報が返された場合は、作成が成功したことを示しています。

```
NAME SCHEDULE SUSPEND ACTIVE LAST SCHEDULE AGE
cronjob * * * * * False 0 <none> 15s
```

CronJobの削除

注意：

- この削除コマンドを実行する前に、作成中のJobが存在するかいやかを確認してください。そうしないと、このコマンドを実行したら、作成中のJobを終了します。
- この削除コマンドを実行するとき、作成したJobおよび完了したJobの両方とも、終了または削除されません。
- CronJobによって作成されたJobを削除するには、手動で削除してください。

次のコマンドを実行して、CronJobを削除します。

```
kubectl delete cronjob [NAME]
```

ワークロードリソース制限の設定

最終更新日：：2022-03-31 11:40:58

リクエスト（Request）と制限（Limit）

Request：コンテナによって使用される最小リソース要件です。コンテナのスケジューリング中にリソースアサインを判断する依存関係として使用されます。ノード上のアサイン可能なリソースの量>=コンテナリソースのリクエストの数の場合にのみ、コンテナをノードにスケジューリングできます。ただし、Requestパラメータは、コンテナの使用可能な最大リソース値を制限しません。
Limit：コンテナの使用可能な最大リソース値です。

> 注意：
> **Limit**と**Request**のパラメータの詳細については、詳細についてをクリックしてください。

CPU 制限の説明

CPUリソースでは、CPUリクエストとCPU制限のリソース量を設定できます。単位はコア（U）で、小数が許可されます。

> 注意：
>
> - CPU Requestは、スケジューリングの根拠として、作成時にコンテナのノードにCPU使用リソースをアサインします。これは、「アサインされたCPU」リソースと呼ばれます。
> - CPU Limitはコンテナ CPUリソースの上限を制限します。設定されていない場合は、制限しないことを意味します（CPU Limit >= CPU Request）。

メモリ制限の説明

メモリリソースは、コンテナの使用可能なメモリの最大量を制限することのみが許可されています。単位はMiBで、小数が許可されます。

> 注意：

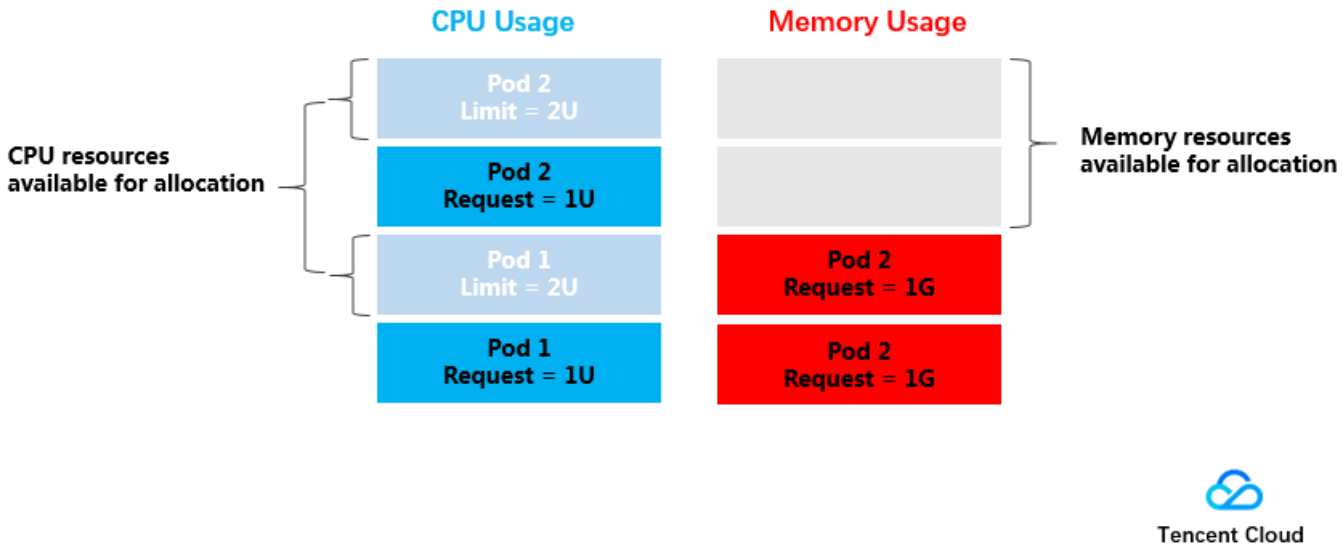
- メモリRequestは、スケジューリングの根拠として、作成時にコンテナのノードにメモリをアサインします。これは、「アサインされたメモリ」リソースと呼ばれます。
- メモリリソースはスケーリング不可のリソースです。ノード上のすべてのコンテナが過剰なメモリを使用する場合は、OOM（Out Of Memory、メモリオーバーフロー）のリスクがあります。したがって、Limitが設定されていない場合は、コンテナの通常の動作を保証するために、デフォルトではLimit = Requestです。

CPU使用量とCPU利用率

- CPU使用量は絶対値であり、実際に使用されているCPUの物理コア数を示します。CPUリソースリクエストとCPUリソース制限の両方とも、CPU使用量に基づいて判断されます。
- CPU利用率は相対値であり、CPUシングルコアに対するCPU使用量の比率（ノード上のCPUコアの総数に対する比率）を示します。

ユースケース

簡単な例でRequestとLimitの作用を説明します。テストクラスターには、1つの4U4Gノード、デプロイされた2つのPod（Pod1、Pod2）が含まれています。各Podのリソースは（CPU Request，CPU Limit，Memory Request，Memory Limit）=（1U，2U，1G，1G）に設定されています。（1.0G = 1000MiB）
ノード上のCPUとメモリのリソース使用状況は下図の通りです：



アサインされたCPUリソースは1U（Pod1をアサイン）＋1U（Pod2をアサイン）＝2Uで、アサイン可能な残りのCPUリソースは2Uです。
アサインされたメモリリソースは1G（Pod1をアサイン）＋1G（Pod2をアサイン）＝2Gで、アサイン可能な残りのメモリリソースは2Gです。
したがって、ノードは、別の1つの（CPU Request、Memory Request）=（2U、2G）のPodデプロイをデプロイするか、または2つの（CPU Request、Memory Request）=（1U、1G）のPodデプロイをデプロイできます。

リソース制限に関しては、各Pod1とPod2が使用するリソースの上限は（2U、1G）です。つまり、リソースがアイドル状態のとき、Podが使用するCPUの最大量は2Uに達する可能性があります。

サービスリソース制限の推奨事項

CCSは、現在のコンテナイメージの過去のロードに基づいて、リクエスト値と制限値を推奨します。推奨値を使用すると、コンテナがよりスムーズに実行され、異常の可能性が大幅に減少します。

推奨アルゴリズム：
まず、過去7日間の現在のコンテナイメージの分レベルのロードを取り出し、パーセンタイル統計の95%値を追加して、推奨されるRequestを最終的に決定します。LimitはRequestの2倍です。

```
Request = Percentile（実際ロード[7d]、0.95）
Limit = Request * 2
```

現在のサンプル数（実際ロード）が推奨される計算数の要件を満たしていない場合は、それに応じてサンプル値の範囲を拡大し、再計算を試みます。例えば、イメージtag，namespace，serviceNameなどのフィルタリング条件を削除します。複数の計算を行っても有効値が得られない場合、推奨値はNULLとします。

推奨値がNULL：
使用中に、一部の値が推奨されていない場合があります。これは、次の点が原因である可能性があります：

1. 現在のデータは計算のニーズを満たしていません。計算されるサンプル数（実際ロード）は1440、つまり1日のデータを超える必要があります。
2. 推奨値は現在のコンテナに構成されたRequestまたはLimitより小さくします。

注意：

1. 推奨値は過去のロードに基づいて計算されるため、原則として、コンテナイメージが実際のビジネスを実行する時間が長いほど、推奨値はより正確になります。
2. 推奨値を使用してサービスを作成する場合は、クラスターリソースが不十分なため、コンテナが正常にスケジューリングされない場合があります。保存するとき、現在のクラスターの残りのリソースを確認する必要があります。

3. 推奨値は提案された値であり、実際のサービス状況に応じて調整できます。

ワークロードスケジューリングルールの設定

最終更新日：：2023-05-25 11:22:38

概要

ワークロードの詳細設定のスケジューリングルールを設定することにより、ワークロードの下のPodがクラスター内でスケジューリングされるように指定します。次のユースケースが存在します：

- 指定されたノードでPodを実行します。
- 特定のスコープ（スコープ：Availability Zone、モデルなどの属性）を持つノードでPodを実行します。

利用方法

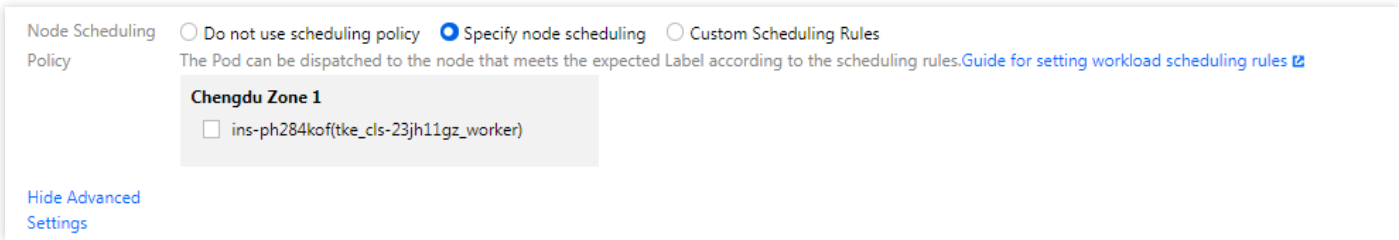
前提条件

- ワークロードの詳細設定でスケジューリングルールを設定します。クラスターのKubernetesバージョンは1.7以降である必要があります。
- Podを正常にスケジューリングするには、設定したスケジューリングルールが完了した後、ノードにはコンテナをスケジューリングするための空きリソースがあることを確認してください。
- カスタマイズされたスケジューリング機能を使用する場合は、ノードに対応するLabelを設定する必要があります。詳細については、ノードLabelの設定をご参照ください。

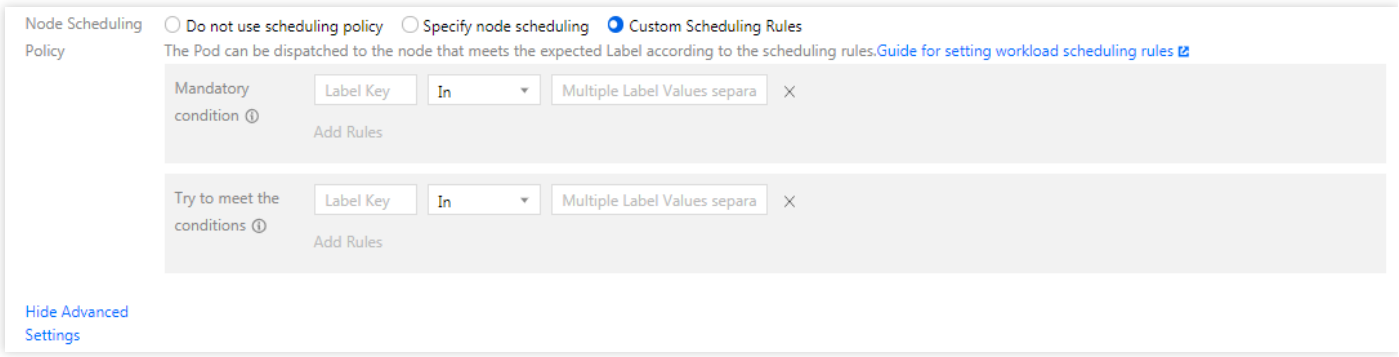
スケジューリングルールの設定

クラスターがバージョン1.7以降の場合は、ワークロードの作成でスケジューリングルールを設定できます。
実際のニーズに応じて、次の2つのスケジューリングタイプを選択できます：

- **指定されたノードによるスケジューリング**：インスタンス（Pod）は、指定されたルールを持つノードにスケジューリングされて、ノードタグに一致するように設定できます。



- **カスタマイズされたスケジューリングルールの**：インスタンス（Pod）のスケジューリングルールをカスタマイズして、インスタンスタグに一致するようにできます。



スケジューリングルールは、次の2つのモードがあります：

- 強制的に必要な条件を満たす：スケジューリング中にアフィニティ条件が満たされた場合、対応するnodeにスケジューリングされます。条件を満たすノードがない場合、スケジューリングは失敗します。
- 可能な限り必要な条件を満たす：スケジューリング中にアフィニティ条件が満たされた場合、対応するnodeにスケジューリングされます。条件を満たすノードがない場合、任意のノードにランダムにスケジューリングされます。

カスタマイズされたスケジューリングルールに複数のスケジューリングルールを追加できます。各ルールのオペレーターの意味は次の通りです。

- In：Labelのvalueはリストにあります。
- NotIn：Labelのvalueはリストにありません。
- Exists：Labelのkeyは存在します。
- DoesNotExits：Labelのkeyは存在しません。
- Gt：Labelの値がリストの値より大きい（文字列が一致する）。
- Lt：Labelの値がリストの値より小さい（文字列が一致する）。

原理の説明

サービスのスケジューリングルールは主に、YamlをKubernetesクラスターにデリバーすることで、KubernetesのAffinity and anti-affinityメカニズムにより、Podが特定のルールに従ってスケジューリングされます。KubernetesのAffinity and anti-affinityメカニズムの詳細については、詳細についてをご参照ください。

ワークロード健康診断の設定

最終更新日：：2022-04-07 11:04:08

Tencent Cloudコンテナのカーネルは Kubernetesに基づいています。Kubernetesは、コンテナを定期的にプローブし、プローブ結果に基づいてコンテナのヘルス状態を判断し、追加のアクションを実行することをサポートします。

健康診断のタイプ

健康診断は次のカテゴリに分類されます：

- **コンテナアクティブ診断**：コンテナがアクティブであるかどうかを検出するために使用されます。psコマンドを実行してプロセスが存在するかどうかを確認するとに類似しています。コンテナアクティブ診断が失敗した場合、クラスターはコンテナをリスタートします。コンテナアクティブ診断が成功した場合、いかなるアクションを実行しません。
- **コンテナ準備診断**：コンテナがユーザーのリクエストの処理を開始する準備ができているかどうかを検出するために使用されます。例えば、プログラムの起動に時間がかかる場合、サービスを提供する前に、ディスクデータをロードするか、外部モジュールに依存して起動を完了してください。この場合、コンテナ準備診断を使用してプログラムのプロセスを確認し、プログラムの起動が完了したかどうかを確認できます。コンテナ準備診断が失敗した場合、クラスターはコンテナへのアクセスリクエストをブロックします。コンテナ準備診断が成功した場合、コンテナへのアクセスが開放されます。

健康診断方法

TCPポートプローブ

TCPポートプローブの原理は次のとおりです：

TCP通信サービスを提供するコンテナの場合、クラスターはコンテナへのTCP接続を定期的に確立します。接続が成功した場合、プローブは成功します。それ以外の場合、プローブは失敗します。TCPポートプローブ方法を選択すると、コンテナがリッスンするポートを指定してください。
例えば、redisコンテナの場合、そのサービスポートは6379です。コンテナでTCPポートプローブを構成し、プローブポートを6379として指定する場合、クラスターはコンテナのポート6379でTCP接続を定期的に開始します。接続が成功した場合、診断は成功します。それ以外の場合、診断は失敗します。

HTTPリクエストプローブ

HTTPリクエストプローブは、HTTP/HTTPSサービスを提供するコンテナ用であり、クラスターはコンテナにHTTP/HTTPS GETリクエストを定期的に送信します。HTTP/HTTPS responseのリターンコードが200～399の範囲にある場合、プローブは成功します。それ以外の場合、プローブは失敗します。HTTPリクエストプローブを使用するには、コンテナがリッスンするポートとHTTP/HTTPSリクエストパスを指定する必要があります。
例えば、HTTPサービスを提供するコンテナの場合、サービスポートが80で、HTTPチェックパスが `/health-check` の場合、クラスターはコンテナに `GET http://containerIP:80/health-check` リクエストを定期的に送信します。

実行コマンドのチェック

実行コマンドのチェックは、ユーザーがコンテナ内の実行可能コマンドを指定する必要がある強力なチェック方法です。クラスターはコンテナ内でコマンドを定期的に実行します。コマンドの返された結果が0の場合、チェックは成功します。それ以外の場合、チェックは失敗します。
TCPポートプローブおよびHTTPリクエストプローブの両方とも、実行コマンドをチェックすることで置き換えることができます：

- TCPポートプローブの場合、コンテナのポートにconnectするプログラムを作成できます。connectが成功した場合、スクリプトは0が返されます。それ以外の場合、-1が返されます。
- HTTPリクエストプローブでは、コンテナをwgetしてresponseのリターンコードをチェックするスクリプトを生成できます。例えば、 `wget http://127.0.0.1:80/health-check` 。リターンコードが200～399の範囲にある場合、スクリプトは0が返されます。それ以外の場合、-1が返されます。

注意事項

- 実行する必要のあるプログラムは、コンテナのイメージに配置する必要があります。そうしないと、プログラムが見つからないため、実行が失敗します。
- 実行されるコマンドがshellスクリプトの場合、実行コマンドとしてスクリプトを直接指定できません。スクリプトのインタプリタを追加する必要があります。例えば、スクリプトが `/data/scripts/health_check.sh` の場合、実行コマンドを使用してチェックするとき、指定するプログラムは次のようになります：

```
sh
/data/scripts/health_check.sh
```

設定手順では、例としてTencent Kubernetes Engineコンソールを使用してDeploymentを作成します：

1. クラスター「Deployment」ページでは、** Create**をクリックします。
2. 「Create Workload」ページへ進み、「Containers in a pod 」モジュール下方の**Advanced Settings**を選択します。
3. 「 Container Health Check」では、例として**Liveness Check**を選択し、次のパラメータを設定します。

- **チェック方法**：「実行コマンドのチェック」を選択します。
- **実行コマンド**：次のように入力します。

```
sh
/data/scripts/health_check.sh
```

4. その他のパラメータ設定については、Deployment管理をご参照ください。

その他の共通パラメータ

- **起動レイテンシー**：単位は秒。コンテナを起動してからプローブを開始する時間を指定します。例えば、起動レイテンシーが5に設定されている場合、健康診断はコンテナの起動から5秒後に開始されます。
- **間隔時間**：単位は秒。健康診断の頻度を指定します。例えば、間隔が10に設定されている場合、クラスターは10秒ごとにチェックします。
- **レスポンスタイムアウト**：単位は秒。健康プローブのタイムアウト時間を指定します。TCPポートプローブ、HTTPリクエストプローブ、実行コマンドチェックの3つの方法に対応して、それぞれTCP接続のタイムアウト時間、HTTPリクエストレスポンスのタイムアウト時間、実行コマンドのタイムアウト時間を示します。
- **健康閾値**：単位は回。コンテナが正常であると判断する前に、健康診断が連続して成功する回数を指定します。例えば、健康閾値が3に設定されている場合、プローブが3回連続して成功した場合にのみ、コンテナが正常であると見なされることを意味します。

> 注意：
> 健康診断のタイプがアクティブ診断の場合、健康閾値は1のみであり、ユーザーが設定した他の値は無効と見なされます。

- **不健康閾値**：単位は回。コンテナが不健康であると判断する前に、健康診断が連続して失敗する回数を指定します。例えば、不健康閾値が3に設定されている場合、プローブが3回連続して失敗した場合にのみ、コンテナが不健康であると見なされることを意味します。

ワークロードの実行コマンドとパラメーターの設定

最終更新日：：2022-03-31 11:40:58

概要

ワークロードを作成する場合は、通常、イメージを使用して、インスタンス内のコンテナが実行するプロセスを指定します。デフォルトでは、イメージはデフォルトのコマンドを実行します。特定のコマンドを実行する必要がある場合、またはイメージのデフォルト値をオーバーライドする必要がある場合は、次の3つの設定を使用してください：

- ワークディレクトリ（workingDir）：現在のワークディレクトリを指定します。
- 実行コマンド（command）：イメージが実行する実際のコマンドをコントロールします。
- コマンドパラメータ（args）：実行コマンドに渡されるパラメータです。

ワークディレクトリの説明

WorkingDirは、現在のワークディレクトリを指定します。存在しない場合は、自動的に作成されます。指定されていない場合は、コンテナを実行するときのデフォルト値が使用されます。WORKDIRがイメージで指定されていなく、コンソールで指定されていない場合、workingDirはデフォルトで「/」になります。

コマンドとパラメータの使用

docker runコマンドをTencent Kubernetes Engine（TKE）に適応させる方法については、docker runパラメータ適応をご参照ください。

Dockerイメージには、イメージ情報の保存に関連するメタデータがあります。実行コマンドとパラメータが指定されていない場合、コンテナはイメージの作成時に提供されたデフォルトのコマンドとパラメータを実行します。Dockerによってネイティブに定義されるフィールドは、「Entrypoint」と「CMD」です。詳細については、DockerのEntrypoint説明およびCMD説明をご参照ください。
サービスの作成時にコンテナの実行コマンドとパラメータを入力すると、TKEはイメージの構築時のデフォルトのコマンド（すなわち、「Entrypoint」と「CMD」）をオーバーライドします。そのルールは次のとおりです：

イメージEntrypoint	イメージCMD	コンテナの実行コマンド	コンテナの実行パラメータ	最終実行

イメージEntrypoint	イメージCMD	コンテナの実行コマンド	コンテナの実行パラメータ	最終実行
[ls]	[/home]	未設定	未設定	[ls / home]
[ls]	[/home]	[cd]	未設定	[cd]
[ls]	[/home]	未設定	[/data]	[ls / data]
[ls]	[/home]	[cd]	[/data]	[cd / data]

- Docker entrypointはTKEコンソールの実行コマンドに対応し、Docker runのCMDパラメータはTKEコンソールの実行パラメータに対応します。複数の実行パラメータがある場合は、TKEの実行パラメータにパラメータを入力する必要があり、各パラメータは別々の行にあります。
- Tencent Kubernetes Engineコンソールによるコンテナの実行コマンドとパラメータの設定例については、CommandとArgsをご参照ください。

TCRエンタープライズ版インスタンスコンテンツのコンテナイメージを用いてワークロードを作成する

最終更新日：：2022-04-07 11:04:08

操作ケース

Tencent Container Registry（TCR）エンタープライズエデションは、厳格なデータセキュリティとコンプライアンスの要件、複数のリージョンに分散しているサービス、および大規模なクラスターを備えたエンタープライズクラスのコンテナの顧客に、エンタープライズクラスの専用イメージセキュリティホスティングサービスを提供します。パーソナルエデションと比較して、エンタープライズエデションは、コンテナイメージセキュリティスキャン、クロスリージョン自動同期、Helm Chartホスティング、ネットワークアクセスコントロールなどの特性をサポートしています。詳細については、Tencent Container Registryをご参照ください。

このドキュメントでは、TCRでホスティングされているプライベートイメージを使用して、Tencent Kubernetes Engine（TKE）でアプリケーションをデプロイする方法について説明します。

前提条件

TCRでホスティングされているプライベートイメージを使用してアプリケーションをデプロイする前に、次の準備を完了する必要があります：

- Tencent Container Registryでエンタープライズエデションインスタンスが作成されています。作成されていない場合、作成を完了するには、エンタープライズエデションインスタンスの作成をご参照ください。
- サブアカウントを使用して操作する場合、対応するインスタンスの操作権限をサブアカウントに事前に付与するには、エンタープライズエデションの権限付与の例をご参照ください。

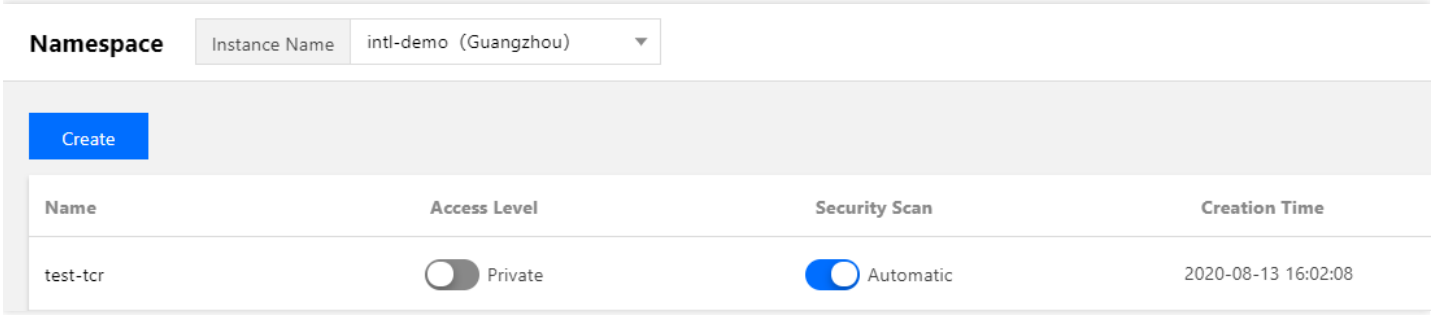
操作手順

コンテナイメージの準備

ネームスペースの作成

新規作成されたTCRエンタープライズエデションインスタンスにはデフォルトのネームスペースがなく、イメージをプッシュして自動的に作成できません。必要に応じて作成を完了するには、ネームスペースの作成をご参照

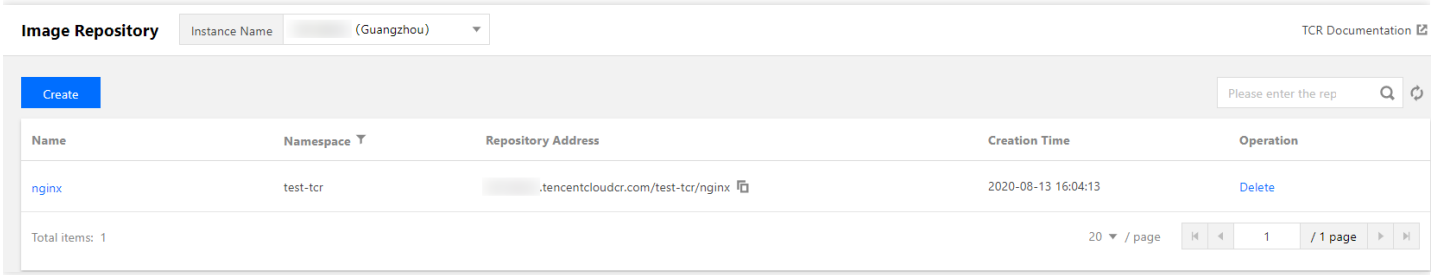
ください。
ネームスペース名はプロジェクト名またはチーム名を使用することをお勧めします。このドキュメントでは、例として `docker` を使用します。通常に作成されると、下図の通りになります：



イメージウエアハウスの作成（オプション）

コンテナイメージは特定のイメージウエアハウスでホスティングされます。必用に応じて作成を完了するには、イメージウエアハウスの作成をご参照してください。イメージウエアハウス名は、デプロイされるコンテナイメージの名前に設定されます。このドキュメントでは、例として `getting-started` を使用します。通常に作成されると、下図の通りになります：

> 説明：
> docker cliまたは他のイメージツール（例えば、jenkins）を使用して、イメージをエンタープライズエディションインスタンスにプッシュするとき、イメージウエアハウスが存在しない場合は、手動で事前に作成する必要がなく、自動的に作成されます。



コンテナイメージのプッシュ

docker cliまたは他のイメージ作成ツール（例えば、jenkins）を使用して、イメージを指定されたイメージウエアハウスにプッシュできます。このドキュメントでは、例としてdocker cliを使用します。この手順では、Dockerがインストールされたcloud Virtual Machine（CVM）または物理マシンを使用する必要があります。また、アクセスしたクライアントが、ネットワークアクセスポリシーの構成で定義されたパブリックネットワークまたはプライベートネットワークの許可されたアクセス範囲内にあることを確認してください。

1. ログインコマンドを取得し、Docker Loginを実行するには、インスタンスアクセス証明の取得をご参照ください。
2. 通常にログインすると、テストするために、新しいコンテナイメージをローカルで構築するか、DockerHubからパブリックイメージを取得できます。
   このドキュメントでは、DockerHubの公式Nginxの最新イメージを例として、コマンドラインツールで次のコマンドを順に実行してイメージをプッシュします。demo-tcr、docker、およびgetting-startedを、実際に作成したインスタンス名、ネームスペース名、およびイメージウエアハウス名に順に置き換えてください。

```
docker tag getting-started:latest demo-tcr.tencentcloudcr.com/docker/getting-started:latest
```

```
docker push demo-tcr.tencentcloudcr.com/docker/getting-started:latest
```

通常にプッシュすると、コンソールの「イメージウエアハウス」ページへ進んで、ウェアハウス名を選択して、詳細ページで確認できます。

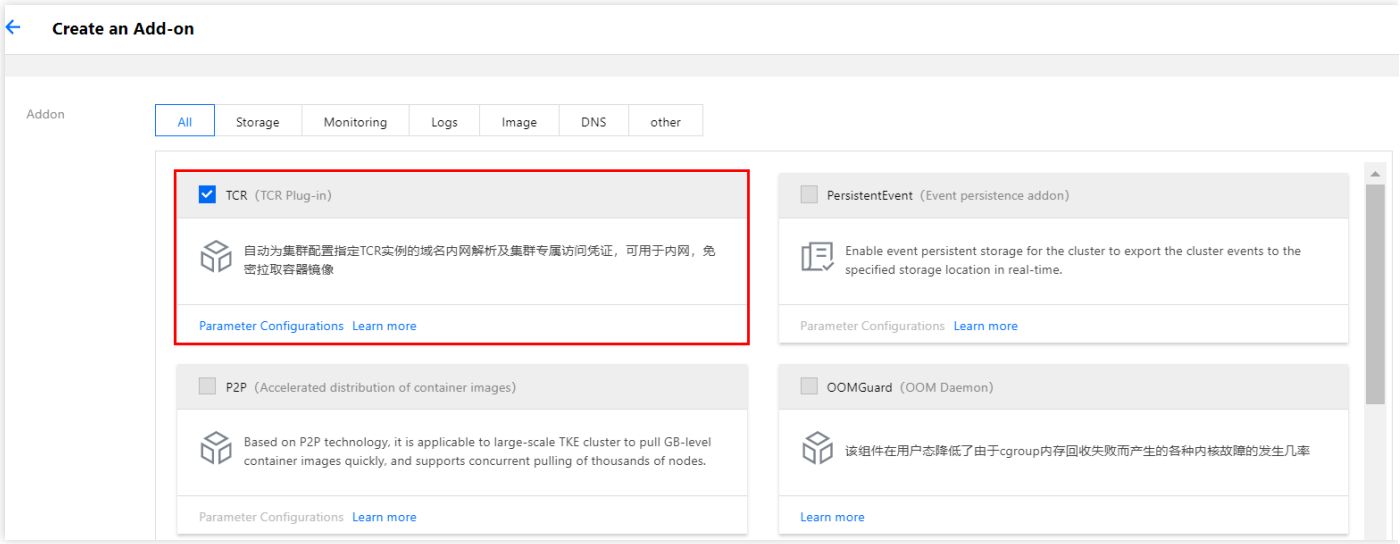
TCRインスタンスへのTKEクラスターのアクセスの構成

TCRエンタープライズエデションインスタンスはネットワークアクセスコントロールをサポートします。デフォルトでは、すべてのソースからの外部アクセスを拒否します。TKEクラスターのネットワーク構成に応じて、パブリックネットワークまたはプライベートネットワークを介して指定されたインスタンスにアクセスし、コンテナイメージをプルすることを選択できます。TKEクラスターとTCRインスタンスが同一地域にデプロイされている場合は、プライベートネットワークアクセスを介してコンテナイメージをプルすることをお勧めします。これにより、プル速度が向上し、パブリックネットワークトラフィックのコストを節約できます。

TCR拡張コンポーネントを使用したクイック構成（推奨）

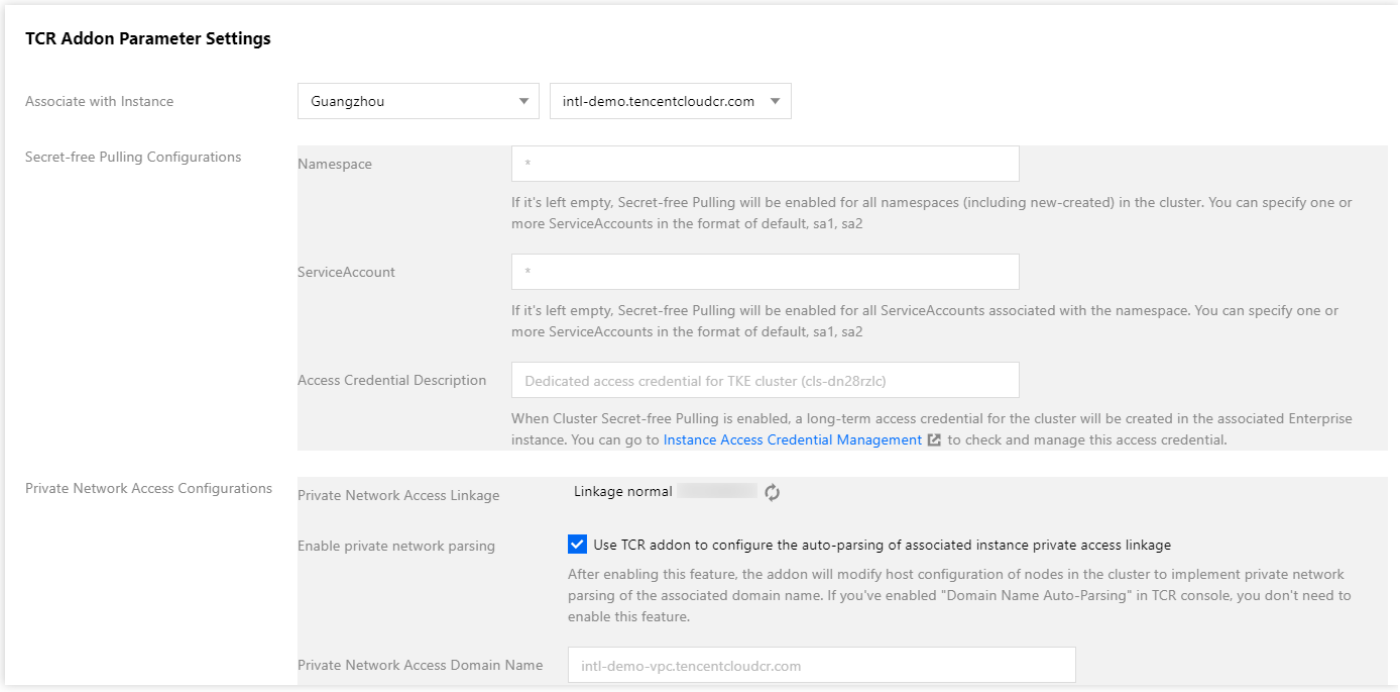
1. TKEコンソールにログインして、左側ナビゲーションバーの【Cluster】を選択してください。
2. 「Cluster Management」ページでは、クラスターIDを選択して、クラスターの詳細ページへ進みます。
3. クラスターの詳細ページでは、左側の【 Add-on Management】を選択して、「Add-on Management」ページへ進んで、【Create】をクリックします。
4. 「 Create an add-on」ページでは、「TCR」コンポーネントを選択します。下図の通りです：

> 説明：
> 現在のTCRコンポーネントは、K8Sのバージョンが1.14と1.16のクラスターのみをサポートします。クラスターのバージョンが現在サポートされていない場合は、手動構成を使用してください。



- 【View Details】をクリックして、コンポーネント機能と構成手順を確認します。
- 【Parameter Configurations】をクリックして、コンポーネントを構成します。

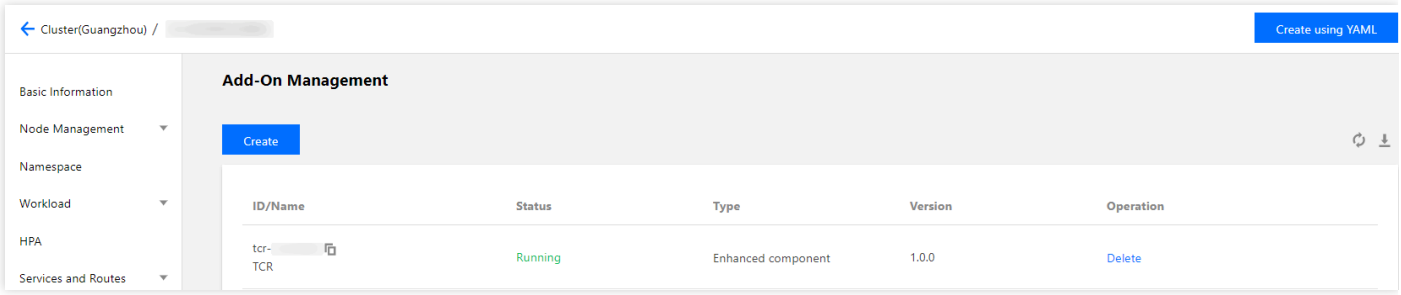
5. 「TCR Add-on Parameter Settings」ページでは、【 View Details】で説明されているコンポーネント構成を参照して、関連パラメータを設定します。下図の通りです：



- **アソシエイトインスタンス**：クラスターと同一地域のTCRインスタンスを選択します。
- **シークレットフリーによる構成プル**：デフォルトの構成を使用できます。
- **プライベートネットワークを介した構成アクセス**：プライベートネットワークアクセスリンクが「リンクは正常です」と表示されない場合、TCRインスタンスとTKEクラスターの存在するVirtual Private Cloud

（VPC）間のプライベートネットワークリンクを構成するには、プライベートネットワークアクセスコントロールをご参照ください。

6. 【OK】をクリックして、コンポーネント選択インターフェースに戻します。
7. コンポーネント選択インターフェースで【 Done】をクリックして、クラスターのTCR拡張コンポーネントをインストールします。
8. コンポーネントがインストールされると、クラスターには、追加の構成なしで、プライベートネットワーク上のアソシエイトインスタンス内のイメージをシークレットフリーでプルする機能があります。下図の通りです：



プライベートネットワークアクセスとアクセス証明の手動構成

1. プライベートネットワークアクセスの構成

1. TCRインスタンスとTKEクラスターの存在するVPC間のプライベートネットワークリンクを構成するには、プライベートネットワークアクセスコントロールをご参照ください。
2. TCRインスタンスとTKEクラスターの存在するVPC間のプライベートネットワークリンクが確立されると、TKEクラスターでは、TCRインスタンスのドメイン名の解析を構成する必要があります。実際の状況に応じて、次の方法を選択します：

- **クラスター作成時にノードHostを構成する**
  TKEクラスターを作成する「CVM構成」の手順では、【 Advanced Settings】を選択して、「Node Launch Configuration」で次のように入力します：

```
echo '172.21.17.69 demo.tencentcloudcr.com' >> /etc/hosts
```

- **既存クラスターのノードHostを構成する**
  クラスターの各ノードにログインして、次のコマンドを実行します：

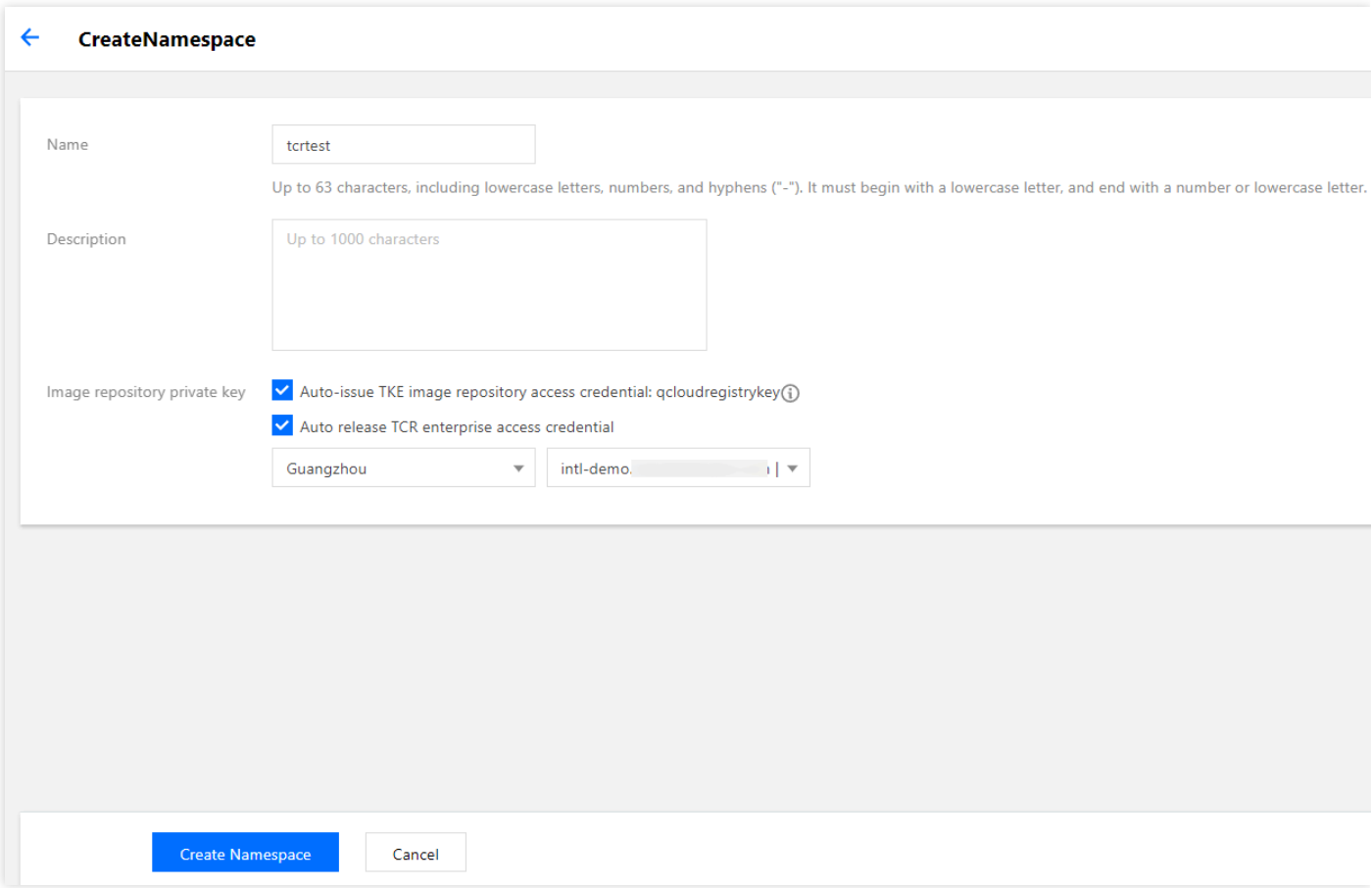
```
echo '172.21.17.69 demo.tencentcloudcr.com' >> /etc/hosts
```

`172.21.17.69` および `demo.tencentcloudcr.com` を、実際に使用されるプライベートネットワーク解析IPおよびTCRインスタンスのドメイン名に置き換えてください。

2. アクセス証明の構成

ネームスペースの新規作成時、アクセス証明をデリバーするには、次の手順をご参照ください。

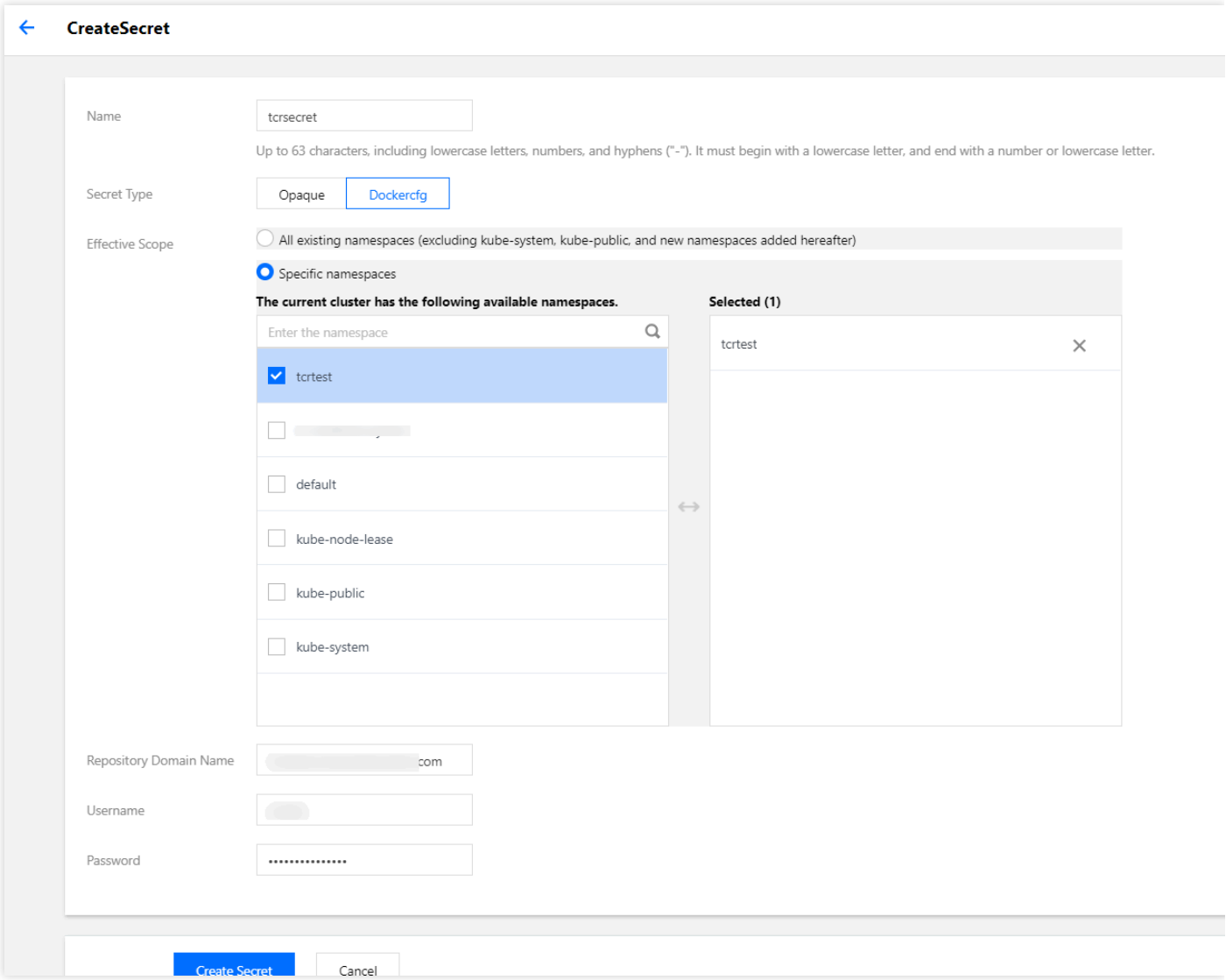
1. TKEコンソールにログインして、左側ナビゲーションバーの【Cluster 】を選択してください。
2. 「Cluster Management」ページでは、クラスターIDを選択して、クラスターの詳細ページへ進みます。
3. 左側の【Namespace】を選択して、「Namespace」ページへ進んで、【Create】をクリックします。
4. 「Create Namespace」ページへ進んで、「Auto-issue TKE image repository access credential」をチェックし、クラスタがアクセスする必要のあるTCRインスタンスを選択します。以下の通りです。



5. 【Create Namespace】をクリックして、作成します。
   作成が完了すると、インスタンスのアクセス証明がネームスペースに自動的にデリバーされます。左側の【Configuration Management】>【Secret】を選択し、「Secret」ページへ進んで、アクセス証明を確認できます。例えば、 `1000090225xx-tcr-m3ut3qxx-dockercfg` です。その中で、 `1000090225xx` は、ネームスペースを作成するためのサブアカウントUINで、 `tcr-m3ut3qxx` は、選択されたインスタンスのインスタンスIDです。

既存のネームスペースにアクセス証明をデリバーするには、次の手順をご参照ください。

1. ユーザー名とパスワードを取得するには、インスタンスアクセス証明の取得をご参照ください。
2. クラスターの詳細ページでは、左側の【 Configuration Management】>【Secret】を選択し、「Secret」ページへ進みます。
3. 「Secret」ページでは【 Create 】をクリックして、「 Create Secret 」ページへ進み、アクセス証明をデリバーするには、次の情報をご参照ください。下図の通りです：



主なパラメータの情報は以下の通りです：

- **Secretタイプ**：【Dockercfg】を選択します。
- **有効範囲**：証明をデリバーする必要のあるネームスペースをチェックします。
- **ウェアハウスのドメイン名**：TCRインスタンスのアクセスドメイン名を入力します。
- **ユーザー名とパスワード**：手順1で取得されたユーザー名とパスワードを入力します。

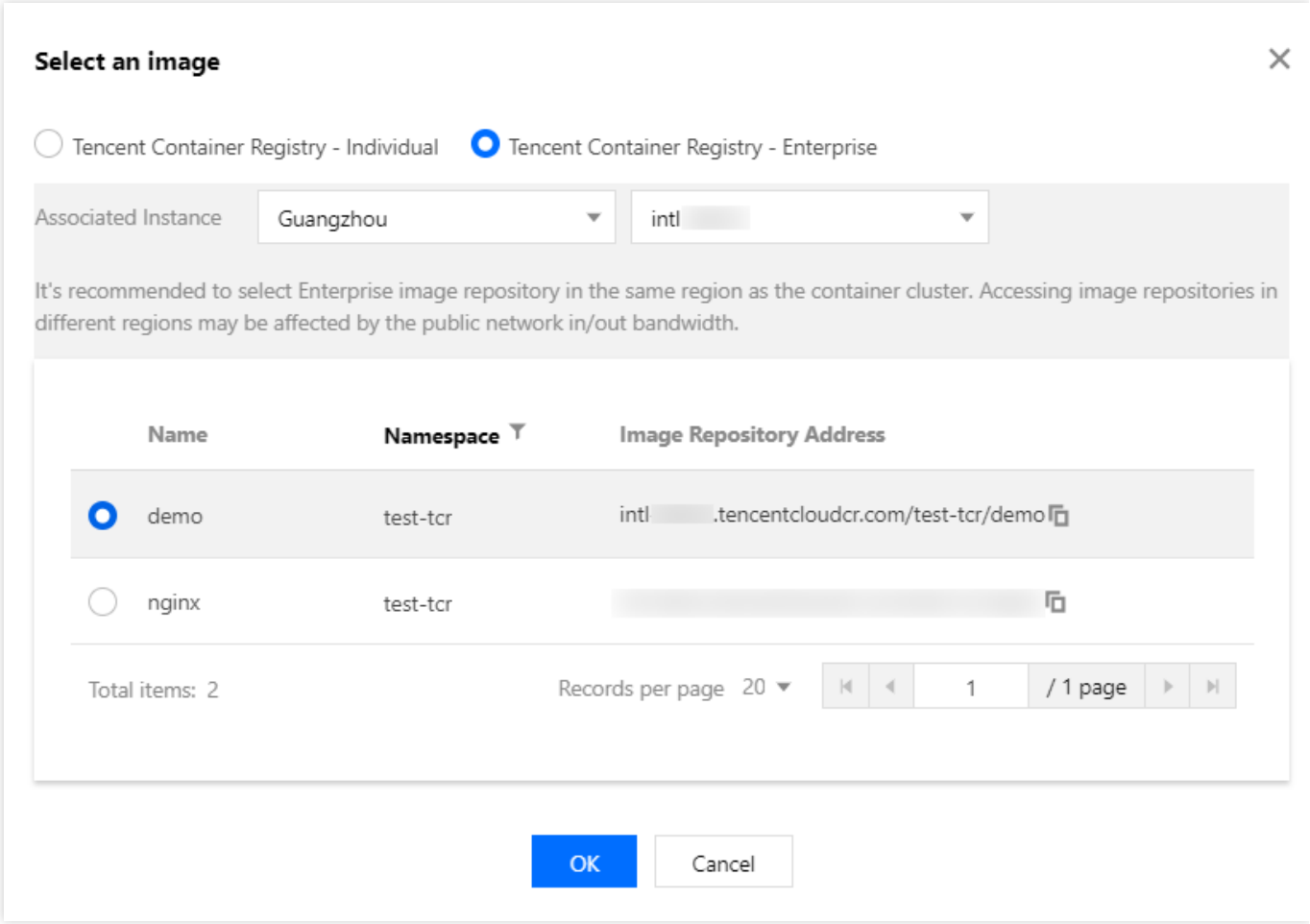
4. 【Create Secret 】をクリックして、デリバーを完了します。

TCRインスタンス内のコンテナイメージを使用したワークロードの作成

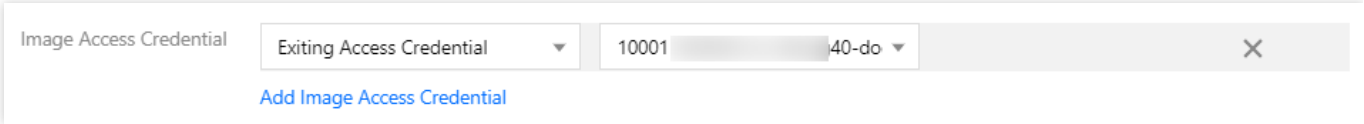
1. クラスターの詳細ページでは、左側の【Workload】>【Deployment】を選択します。

2. 「Deployment」ページへ進んで、【Create】をクリックします。
3. 「Create Workload」ページへ進んで、ワークロードを作成するには、次の情報をご参照ください。。
   主なパラメータの情報は以下の通りで、他のパラメータは必要に応じて設定してください。

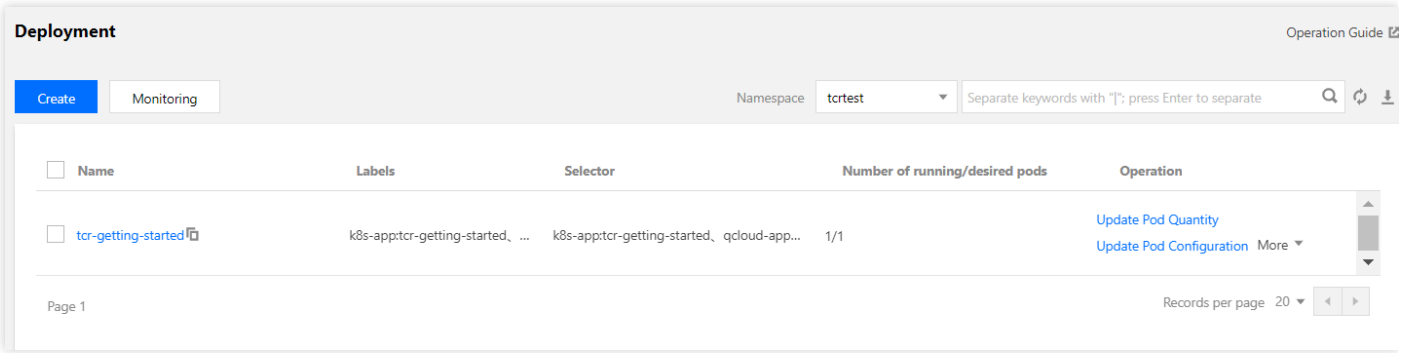
- **ネームスペース**：アクセス証明がデリバーされたネームスペースを選択します。
- **インスタンス内のコンテナ**：
  - **イメージ**：【Select Image】をクリックし、ポップアップ表示された「Select Image」ウィンドウでは、TCRインスタンス内のコンテナイメージを選択します。以下の通りです。



- **イメージアクセス証明**：
  - TCR拡張コンポーネントがインストールされているクラスター：構成する必要はありません。
  - TCR拡張コンポーネントがインストールされていないクラスター：【Add Image Access Credential 】を選択し、アクセス証明の構成手順でデリバーしたアクセス証明を選択します。以下の通りです。



4. 他のパラメータ設定が完了すると、【Create Workload】をクリックして、ワークロードのデプロイ進捗状況を確認します。
   正常にデプロイが完了すると、「Deployment」ページでは、ワークロードの「実行/期待Podの数」が「1/1」であると確認できます。下図の通りです：



Service管理
概要

最終更新日：：2023-04-28 11:08:19

Service基本概念

ユーザーはKubernetes内で各種コンテナをデプロイできます。そのうちの一部はHTTP、HTTPSプロトコルによって外部にレイヤー7ネットワークサービスを提供し、もう一部はTCP、UDPプロトコルによってレイヤー4ネットワークサービスを提供します。Kubernetesによって定義されたServiceリソースはクラスター内のレイヤー4ネットワークを管理するためのサービスアクセスです。
KubernetesのServiceTypesはServiceタイプを指定でき、デフォルトはClusterIPタイプです。ServiceTypesの値および行動は次のとおりです。

値	説明
ClusterIP	クラスターの内部IPによってサービスを公開します。サービスがクラスター内部でのみアクセスされる必要がある場合は、このタイプを使用してください。このタイプはデフォルトのServiceTypeです。
NodePort	各クラスターノード上のIPおよび静的ポート（NodePort）によってサービスを公開します。NodePortサービスはClusterIPサービスにルーティングされ、このClusterIPサービスは自動的に作成されます。 `<NodeIP>:<NodePort>` をリクエストすることによって、クラスターの外部からこのNodePortサービスにアクセスすることができます。テストおよび非本番環境を除き、本番環境内で直接クラスターノードによって外部またはパブリックネットワークにサービスを提供することは推奨していません。安全面から考慮すると、このタイプを使用して直接クラスターノードを公開すると、攻撃を受けやすくなります。通常クラスターノードは動的で、スケーラブルであると考えられ、このタイプを使用して外部にサービスを提供するアドレスおよびクラスターノードには結合が発生します。
LoadBalancer	Tencent CloudのCLBを使用し、パブリックネットワークまたはプライベートネットワークにサービスを公開することができます。CLBはNodePortサービスにルーティングするか、VPC-CNIネットワーク条件のコンテナ内に転送することができます。

ClusterIPおよびNodePortタイプの Serviceは、異なるクラウドサービスプロバイダまたは自作のクラスター内での行動が通常の状況で同じです。LoadBalancerタイプのServiceは、クラウドサービスプロバイダのCLBを使用してサービスの公開を行うため、クラウドサービスプロバイダはCLBを制御するネットワークタイプ、バックエンドバ

インディングの重み調整など、そのCLBの機能に関わるさまざまな追加機能を提供します。詳細はService機能ドキュメントをご参照ください。

サービスのアクセス方法

上記の `ServiceTypes` に基づいて定義されます。TKEが提供する以下の4種類のサービスアクセス方式を使用することができます。

アクセス方法	Serviceタイプ	説明
パブリックネットワーク	LoadBalancer	ServiceのLoadbalanceモードを使用し、パブリックネットワークIPはバックエンドのPodにアクセスし、Webフロントエンド類のサービスに適用されます。 作成が完了したサービスはクラスター外ではCLBドメイン名またはIP+サービスポートによってサービスにアクセスでき、クラスター内ではサービス名+サービスポートによってサービスにアクセスできます。 注意：Tencent Cloud Load Balancerインスタンスは2023年3月6日にアーキテクチャがアップグレードされました。アップグレード後、パブリックネットワークCLBは**ドメイン名**の形式でサービスを提供します。**VIP**は業務リクエストに応じて動的に変化し、コンソールにはVIPアドレスが表示されなくなります。ドメイン名型パブリックネットワークCLBリリースのお知らせをご参照ください。 新規登録したTencent Cloudユーザーはデフォルトでアップグレード後のドメイン名型CLBを使用します。 既存のユーザーは元のCLBを継続して使用することを選択でき、アップグレードの影響を受けません。CLBサービスをアップグレードする必要がある場合は、同時にTencent Cloud製品のCLBおよびTKEをアップグレードする必要があります。アップグレードしない場合、TKE内のすべてのパブリックネットワークタイプのService/Ingressの同期は影響を受ける可能性があります。CLBのアップグレード操作の詳細については、ドメイン名型CLBアップグレードガイドをご参照ください。TKEがService/Ingressコンポーネントのバージョンをアップグレードするには、チケットを提出してお問い合わせください。
VPCプライベートネットワーク	LoadBalancer	ServiceのLoadbalanceモードを使用し、 `service.kubernetes.io/qcloud-loadbalancer-internal-subnetid: subnet-xxxxxxxx` というアノテーションを指定します。す

		なわちプライベートIPによってバックエンドのPodに直接アクセスすることができます。 作成が完了したサービスはクラスター外ではCLBドメイン名またはIP+サービスポートによってサービスにアクセスでき、クラスター内ではサービス名+サービスポートによってサービスにアクセスできます。
ホストポートアクセス	NodePort	1つのホストポートをコンテナにマッピングするアクセス方式を提供し、TCP、UDP、Ingressをサポートしています。業務のカスタム上位階層LBをNodeに転送することができます。 作成が完了したサービスはCVM IP+ホストポートによってサービスにアクセスできます。
クラスター内のみアクセス	ClusterIP	ServiceのClusterIP モードを使用し、Serviceネットワークセグメント内のIPを自動的に割り当て、クラスター内のアクセスに使用します。MySQLのようなデータベース類などのサービスはクラスター内アクセスを選択でき、それによってサービスネットワークの隔離を保証します。 作成が完了したサービスはサービス名+サービスポートによってサービスにアクセスできます。

CLBの関連概念

Serviceの動作原理

Tencent Cloudコンテナクラスター内のService Controllerコンポーネントはユーザーのリソースの同期を行う役割を担います。ユーザーがServiceリソースを作成、変更または削除した場合、クラスターノードまたはService Endpointsに変化が発生した場合、コンポーネントコンテナにドリフト再起動が発生した場合、コンポーネントはいずれもユーザーのServiceリソースに同期を行います。

Service ControllerはユーザーのServiceリソースの記述によって対応するCLBリソースを作成し、リスナーおよびそのバックエンドを設定します。ユーザーがクラスターのServiceリソースを削除した場合、対応するCLBリソースを変更します。

Serviceのライフサイクル管理

Serviceが外部にサービスする機能はCLBが提供するリソースに依存します。サービスリソース管理もServiceの重要な作業のひとつです。Serviceはリソースのライフサイクル管理において以下のタグを使用します。

`tke-createdBy-flag = yes` ：このリソースがコンテナサービスによって作成されたことを示します。

このタグがあれば、Serviceは破棄される時に対応するリソースを削除します。

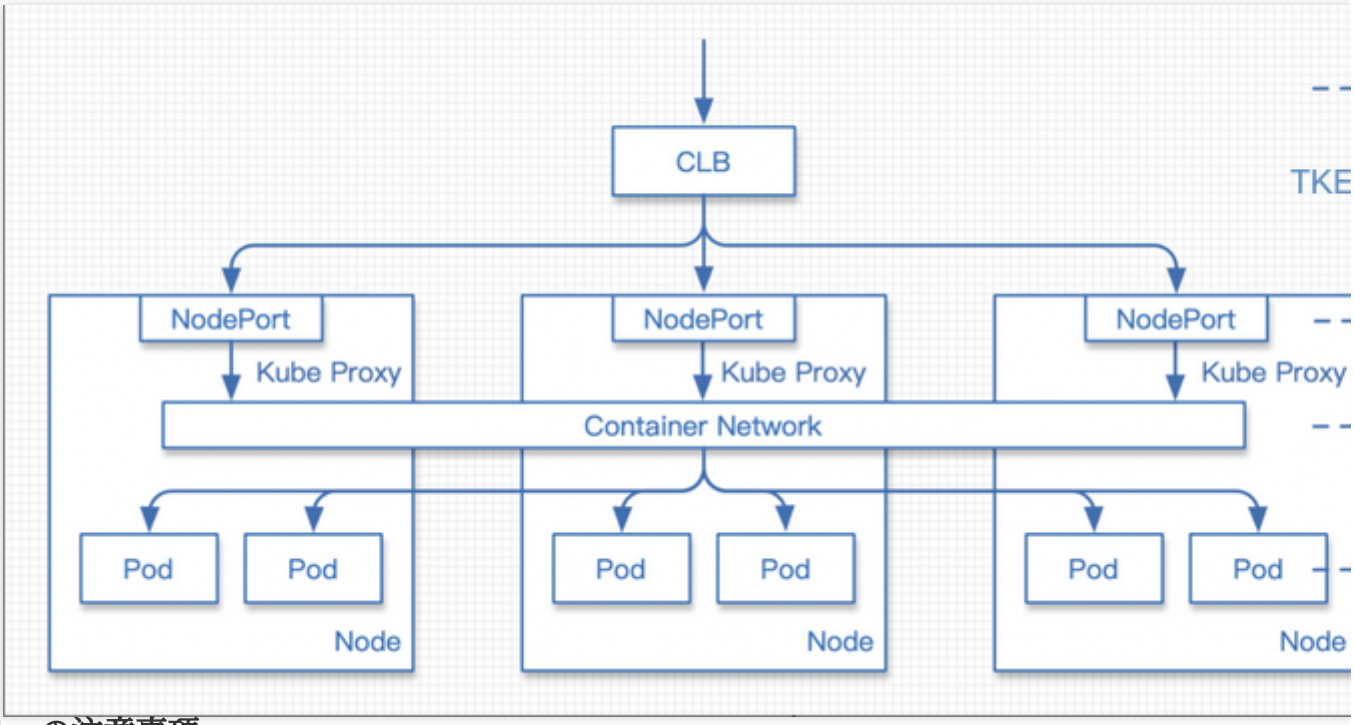
このタグがなければ、Serviceは破棄される時に、CLB内のリスナーリソースのみを削除します。CLB自体は削除されません。

`tke-clusterId = <ClusterId>` ：このリソースがどのClusterによって使用されているかを示します。

ClusterIdが正確であれば、Serviceが破棄された時に、対応するタグを削除します。

説明

ユーザーが既存のCLBを使用した場合、ServiceはこのCLBのみを使用し、このCLBを削除することはありません。
ユーザーがCLB上で削除保護を有効にしたか、プライベート接続を使用した場合、Serviceを削除する時に、このCLBは削除されません。
LoadBalancerタイプのServiceクラスターリソースが作成された時に、CLBに対応するライフサイクルが開始されます。Serviceリソースが削除されるかCLBが再構成された時に、CLBのライフサイクルが終了します。この期間、CLBはServiceリソースの記述に基づいて継続的に同期を行います。**ユーザーがServiceのネットワークアクセスに切り替えた場合、例えばパブリックネットワーク > VPCプライベートネットワーク、VPCプライベートネットワーク > パブリックネットワーク、VPCサブネットに切り替え、使用する既存のCLBを切り替えた場合、この操作はいずれもCLBの再構成または破棄に影響します。** `LoadBalancer` タイプのServiceの動作原理は下図に示すとおりです。



Serviceの注意事項

Serviceには `.spec.externalTrafficPolicy` というフィールドがあります。kube-proxyは、spec.internalTrafficPolicyの設定に基づいてルーティングされたターゲットサービスのエンドポイントをフィルタリングします。その値が `Local` に設定されている場合、ノードローカルのサービスのエンドポイントのみが選択されます。その値が `Cluster` に設定されているか省略されている場合、Kubernetesはすべてのサービスのエンドポイントを選択します。詳細については、Kubernetesドキュメントをご参照ください。
Serviceが `Local` 方式を使用している場合、PodがTKEノードからスーパーノードにスケジューリングされるか、またはスーパーノードからTKEノードにスケジューリングされる時に切断が発生するため、Serviceはローカルのサービスエンドポイントのみ選択できます。

Serviceの高リスク操作

従来型CLBを使用する（非推奨）。
コンテナサービスによって追加されたCLBタグを変更または削除し、新しいCLBを再購入してそのタグを復元します。
CLBコンソールによって、コンテナサービスがCLBを管理するリスナー名を変更します。

Service機能

Serviceの関連操作および機能は次のとおりです。以下のドキュメントをご参照ください。
Service基本機能
Service CLBの設定
Serviceに既存のCLBを使用する
Serviceのバックエンド選択
Serviceのクロスドメインバインド
Serviceのグレースフルシャットダウン
LoadBalancerのダイレクト接続Podモードを使用する
マルチServiceでCLBを再利用する
Service拡張プロトコル
Service Annotation説明

参考資料

オープンソースドキュメントKubernetes Serviceを参照することでServiceに関する詳細な情報を理解することができます。

Service CLB設定

最終更新日：：2023-04-27 18:15:01

TkeServiceConfig

TkeServiceConfigはTencent Kubernetes Engineが提供するカスタムリソースCRDです。TkeServiceConfigによってLoadBalancerタイプのServiceをより柔軟に設定できるようにし、CLBの各種設定を管理します。

ユースケース

Service YAMLのセマンティクスで定義できないCLBのパラメータおよび機能は、TkeServiceConfigによって設定することができます。

設定についての説明

TkeServiceConfigを使用すると迅速にロードバランサの設定を行うことができます。Serviceアノテーション `service.cloud.tencent.com/tke-service-config:<config-name>` によって、目標の設定を指定してServiceに適用することができます。

注意

TkeServiceConfigリソースとServiceが同じネームスペースにある必要はありません。

TkeServiceConfigはプロトコルおよびポートを直接設定して変更しないため、設定でプロトコルおよびポートを記述することによって送信するリスナーを指定する必要があります。TkeServiceConfig内で複数のリスナー設定を宣言でき、現在、主にCLBのヘルスチェックおよびバックエンドへのアクセスに対する設定を提供しています。

プロトコルおよびポートを指定することによって、設定は対応するリスナーに正確に送信されます。

`spec.loadBalancer.l4Listeners.protocol` ：レイヤー4プロトコル

`spec.loadBalancer.l4Listeners.port` ：リスニングポート

ServiceとTkeServiceConfigの関連行動

1. LoadbalancerモードのService時、アノテーション `service.cloud.tencent.com/tke-service-config-auto: "true"` に設定すると、<ServiceName>-auto-service-configが自動作成されます。

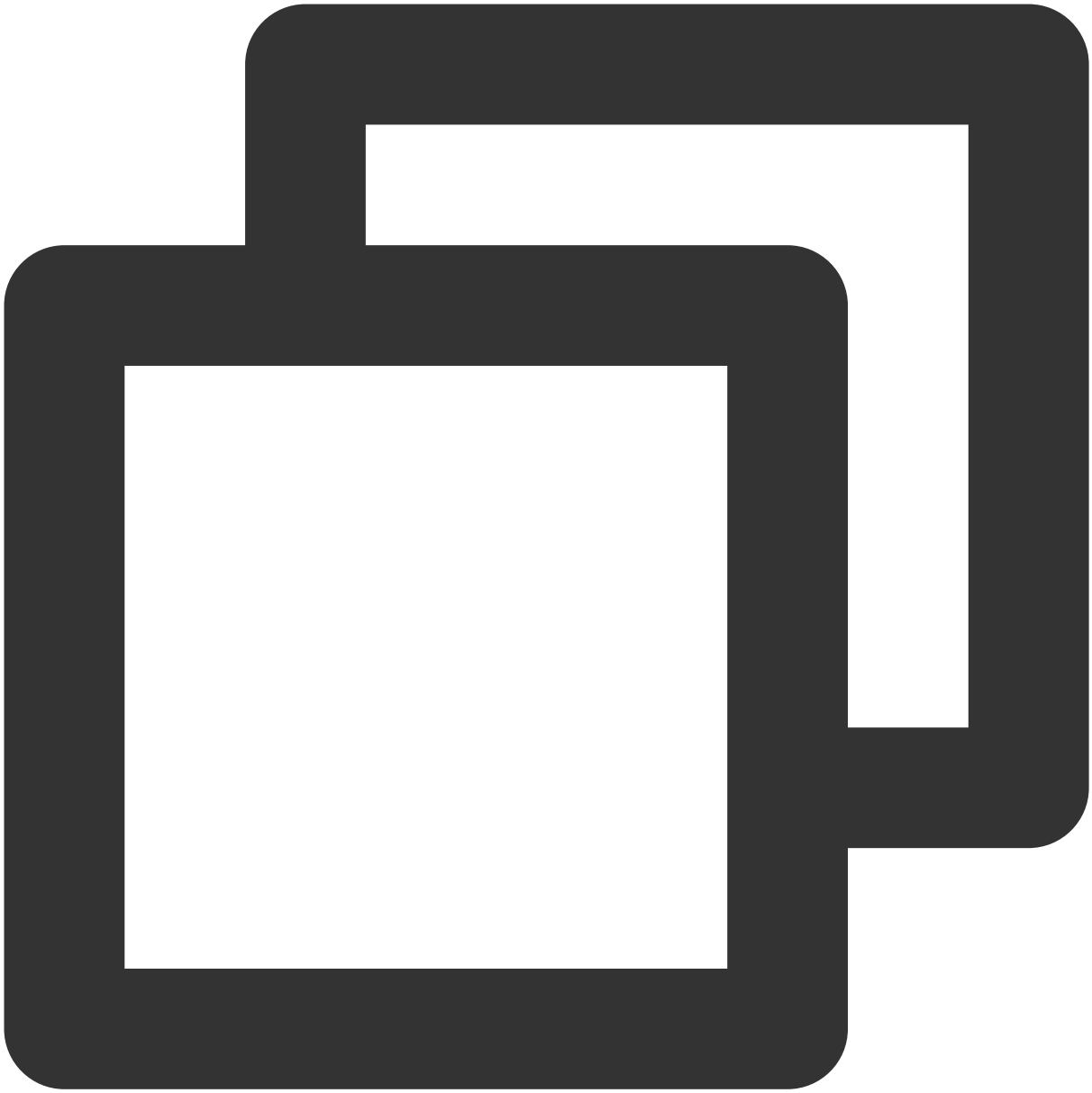
service.cloud.tencent.com/tke-service-config:<config-name>によってご自身が作成したTkeServiceConfigを直接指定することもできます。2つのアノテーションは同時に使用できません。

2. そのうち、自動で作成されたTkeServiceConfigには以下の同期行動が存在します。

Serviceリソースの更新時に、複数のレイヤー4リスナーを追加する場合、このリスナーまたは転送ルールに対応するTkeServiceConfig設定セグメントがない場合、Service-ControllerはTkeServiceConfigに対応するセグメントを自動的に追加します。

複数のレイヤー4リスナーを削除する場合、Service-controllerコンポーネントはTkeServiceConfigの対応セグメントを能動的に削除します。
Serviceリソースを削除すると、このTkeServiceConfigも削除されます。
ユーザーがServiceのデフォルトのTkeServiceConfigを変更すると、TkeServiceConfigの内容も同様にCLBに適用されます。
3. 以下のTkeServiceConfigの完全な設定リファレンスを参照して、ご自身で必要なCLB設定を作成することができます。Serviceはアノテーション：**service.cloud.tencent.com/tke-service-config:<config-name>**によってこの設定を参照します。
4. そのうち、手動で作成したTkeServiceConfigには以下の同期行動が存在します。
ユーザーがService内に設定アノテーションを追加した場合、CLBはすぐに同期を設定します。
ユーザーがService内で設定アノテーションを削除した場合、CLBは変更されません。
TkeServiceConfig設定を変更する場合、この設定ServiceのCLBを引用して新しいTkeServiceConfigに基づいて設定の同期を行います。
Serviceのリスナーが対応する設定を見つけられない場合、このリスナーは変更を行いません。
Serviceのリスナーが対応する設定を見つけた場合、設定内に宣言された属性がなければ、このリスナーは変更を行いません。

完全な設定リファレンス

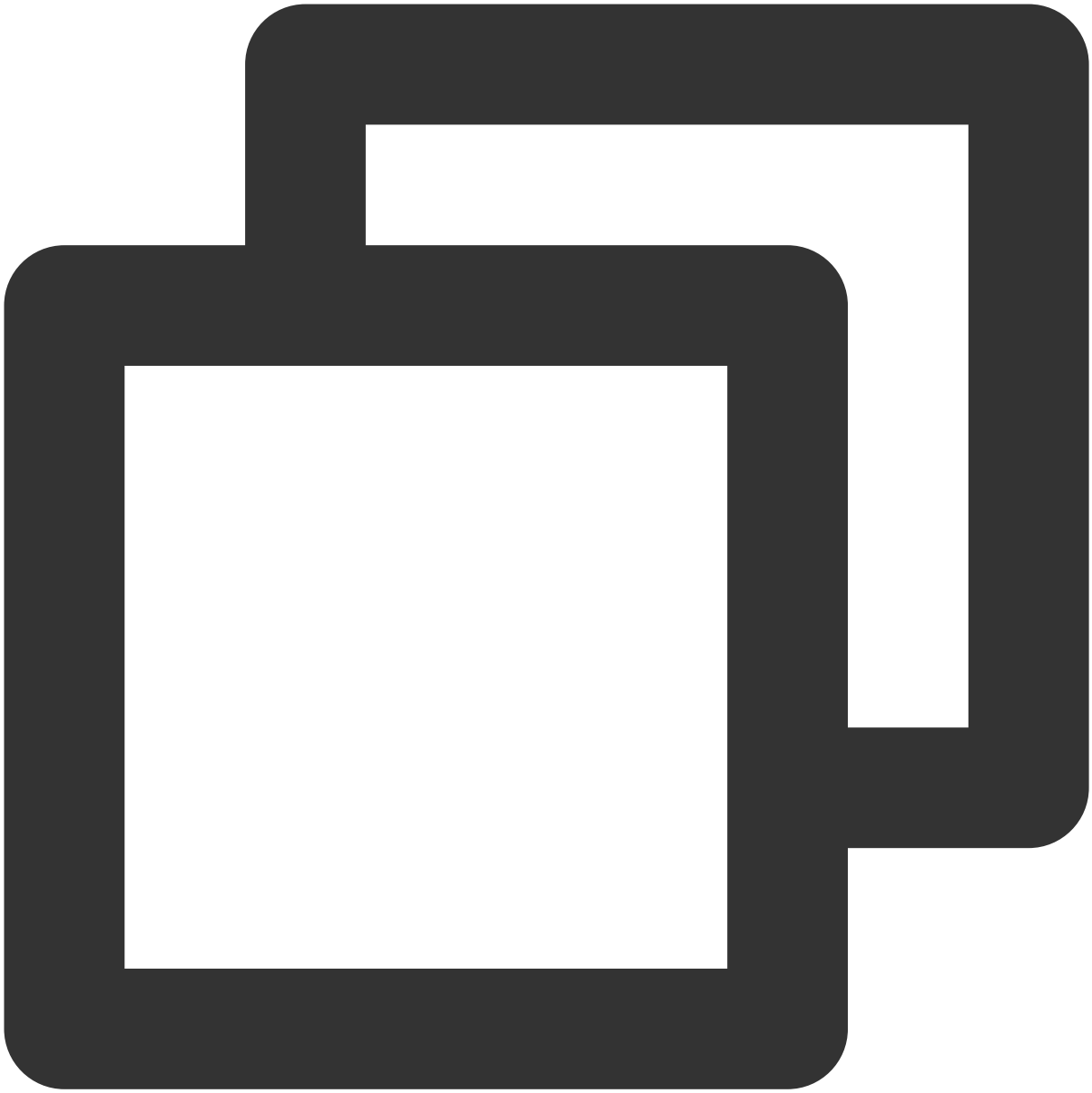


```
apiVersion: cloud.tencent.com/v1alpha1
kind: TkeServiceConfig
metadata:
  name: sample # 設定の名称
  namespace: default # 設定のネームスペース
spec:
  loadBalancer:
    l4Listeners: # レイヤー4のルール設定。Serviceのリスナー設定に適用されます。
    - protocol: TCP # プロトコルポートがServiceを固定するレイヤー4ルール。入力必須。列挙値：TC
      port: 80 # 入力必須。選択可能な値：1~65535。
      session: # セッション維持の関連設定。オプション
```

```
      enable: true # セッション維持を有効にするかどうか。入力必須。ブール値
      sessionExpireTime: 100 # セッション維持の時間。オプション。デフォルト値：30、選択可能
    healthCheck: # ヘルスチェック関連設定。オプション
      enable: true # セッション維持を有効にするかどうか。入力必須。ブール値
      intervalTime: 10 # ヘルスチェックの検出間隔時間。オプション。デフォルト値：5、選択可能
      healthNum: 2 # 健全なしきい値。数回連続して健全であることを検出した場合、この転送が正常
      unHealthNum: 3 # 不健全なしきい値。数回連続して健全であることを検出した場合、この転送が
      timeout: 10 # ヘルスチェックの応答タイムアウト時間。応答タイムアウト時間はチェック間隔時
    scheduler: WRR # 転送方法設定をリクエストします。WRR、LEAST_CONN、IP_HASHはそれぞれ重
```

事例

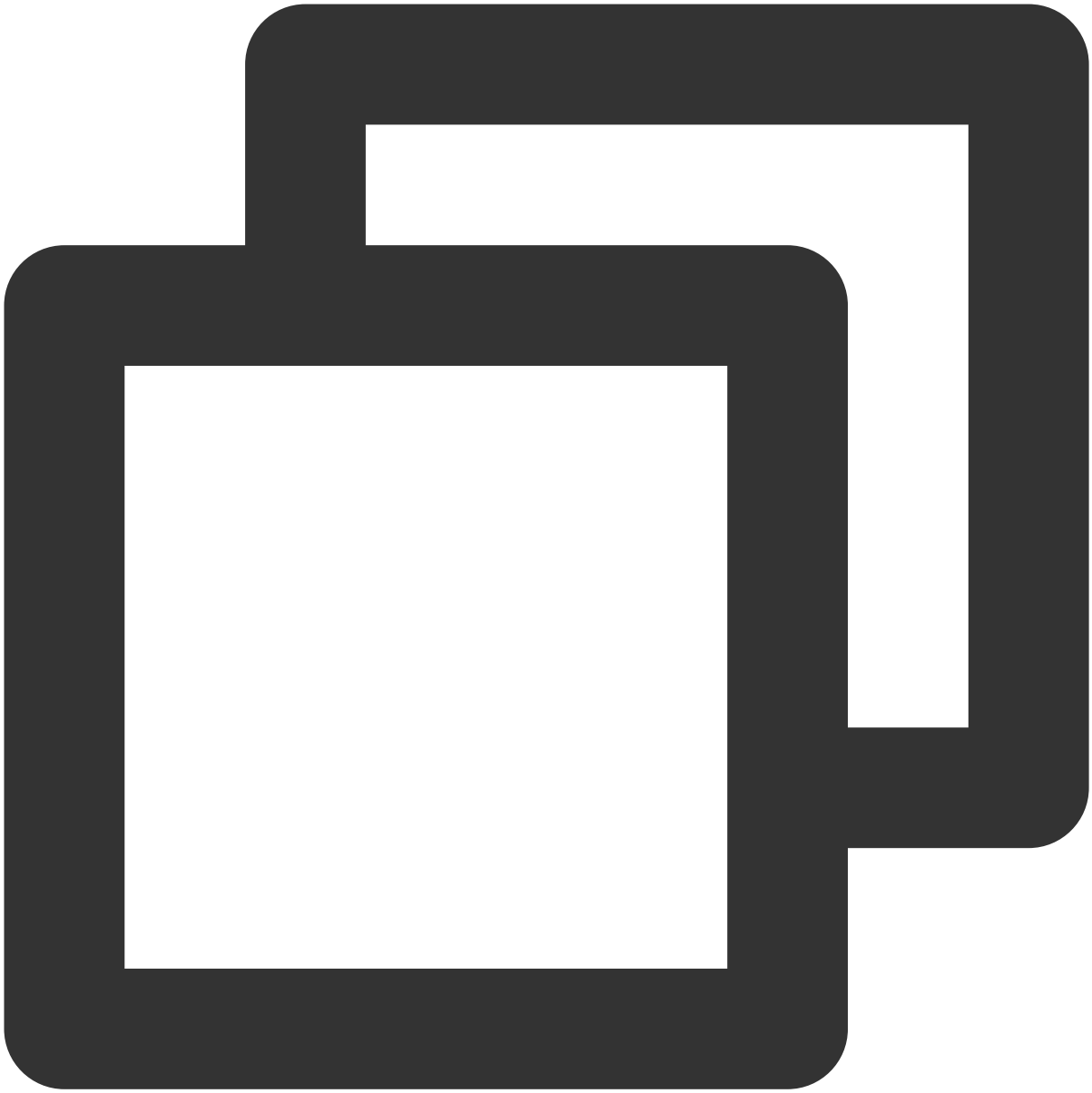
Deploymentの例：jetty-deployment.yaml



```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: jetty
  name: jetty-deployment
  namespace: default
spec:
  progressDeadlineSeconds: 600
  replicas: 3
  revisionHistoryLimit: 10
```

```
  selector:
    matchLabels:
      app: jetty
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: jetty
    spec:
      containers:
      - image: jetty:9.4.27-jre11
        imagePullPolicy: IfNotPresent
        name: jetty
        ports:
        - containerPort: 80
          protocol: TCP
        - containerPort: 443
          protocol: TCP
        resources: {}
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      terminationGracePeriodSeconds: 30
```

Serviceの例：jetty-service.yaml



```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.cloud.tencent.com/tke-service-config: jetty-service-config
    # 既存のtke-service-configを指定する
    # service.cloud.tencent.com/tke-service-config-auto: "true"
    # tke-service-configを自動作成する
  name: jetty-service
  namespace: default
spec:
```

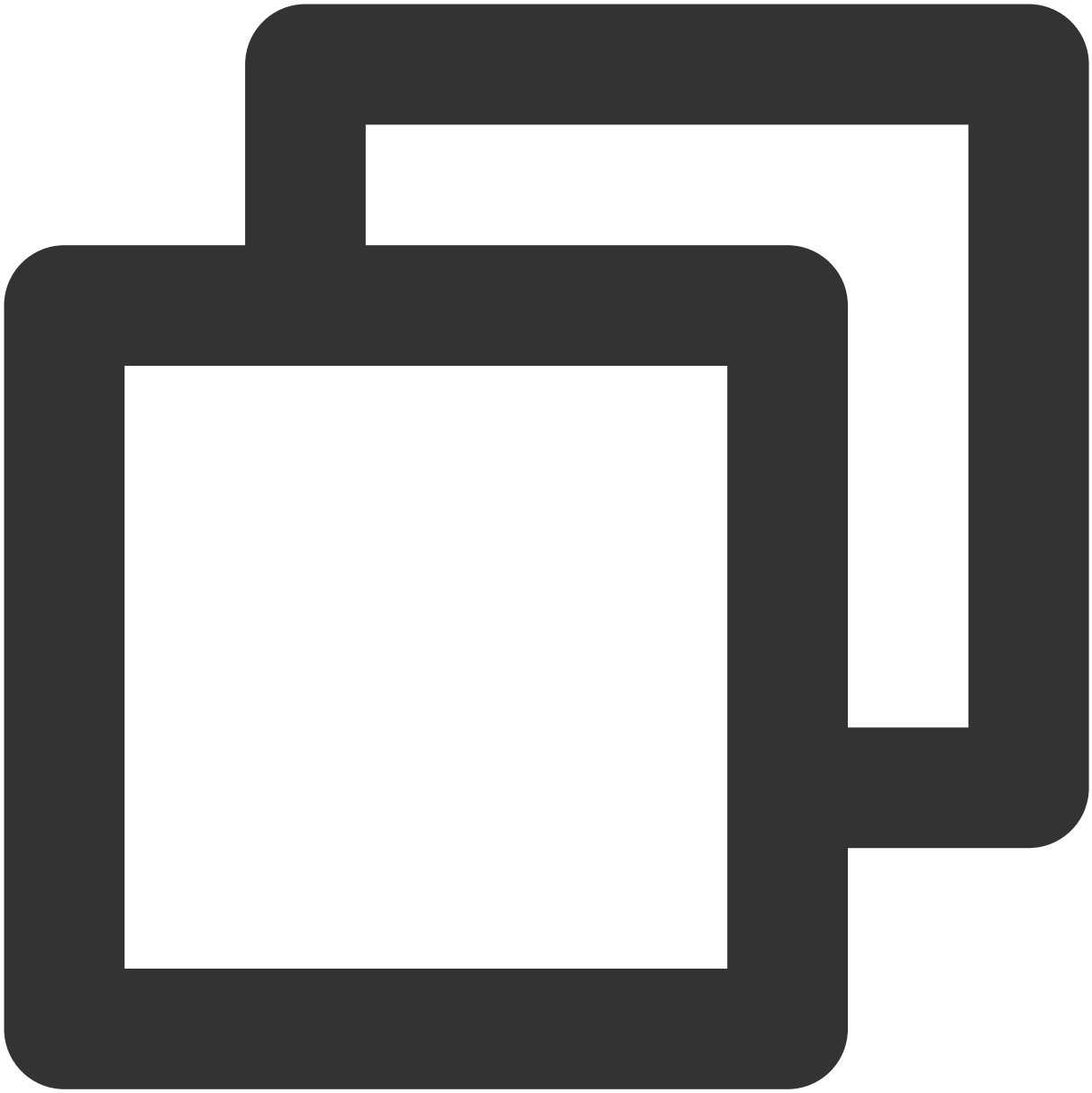
```
  ports:
  - name: tcp-80-80
    port: 80
    protocol: TCP
    targetPort: 80
  - name: tcp-443-443
    port: 443
    protocol: TCP
    targetPort: 443
  selector:
    app: jetty
  type: LoadBalancer
```

この例には以下の設定が含まれます。

Serviceはパブリックネットワークの LoadBalancer タイプです。2つのTCPサービスを宣言し、1つは80ポート、もう1つは443ポートです。

`jetty-service-config` というCLB設定を使用します。

TkeServiceConfigの例：jetty-service-config.yaml



```
apiVersion: cloud.tencent.com/v1alpha1
kind: TkeServiceConfig
metadata:
  name: jetty-service-config
  namespace: default
spec:
  loadBalancer:
    l4Listeners:
    - protocol: TCP
      port: 80
      healthCheck:
```

```
      enable: false
  - protocol: TCP
    port: 443
    session:
      enable: true
      sessionExpireTime: 3600
    healthCheck:
      enabled: true
      intervalTime: 10
      healthNum: 2
      unHealthNum: 2
      timeout: 5
    scheduler: WRR
```

この例には以下の設定が含まれます。

名称は `jetty-service-config` です。レイヤー4のリスナー設定では、次の2つの設定を宣言します。

1. 80ポートのTCPリスナーが設定されます。ヘルスチェックはオフになります。

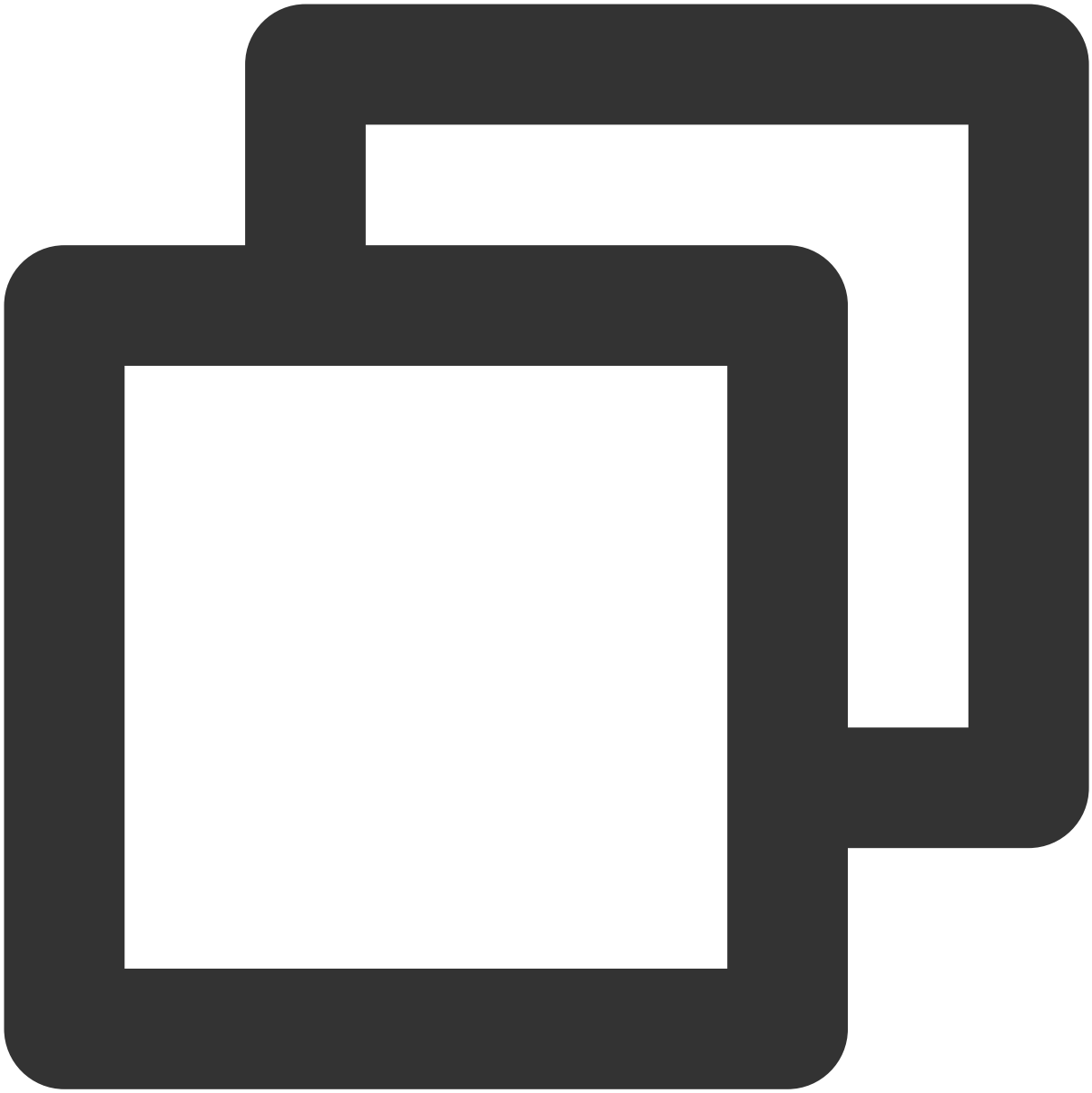
2. 443ポートのTCPリスナーが設定されます。

ヘルスチェックを開き、ヘルスチェック間隔を10s、健全なしきい値を2回、不健全なしきい値を2回、タイムアウトを5sに調整します。

セッション維持機能を開き、セッション維持のタイムアウト時間を3600sに設定します。

転送ポリシーの設定は以下のとおりです。重み付けによるラウンドロビン。

kubectl設定コマンド



```
➜ kubectl apply -f jetty-deployment.yaml
➜ kubectl apply -f jetty-service.yaml
➜ kubectl apply -f jetty-service-config.yaml

➜ kubectl get pods
NAME                                 READY   STATUS    RESTARTS   AGE
jetty-deployment-8694c44b4c-cxscn    1/1     Running   0          8m8s
jetty-deployment-8694c44b4c-mk285    1/1     Running   0          8m8s
jetty-deployment-8694c44b4c-rjrtm    1/1     Running   0          8m8s

➜ kubectl get service jetty
```

```
NAME     TYPE           CLUSTER-IP       EXTERNAL-IP       PORT(S)
jetty    LoadBalancer   10.127.255.209   150.158.220.237   80:31338/TCP,443:32373/TC

# TkeServiceConfig設定リストを取得する
➜ kubectl get tkeserviceconfigs.cloud.tencent.com
NAME                   AGE
jetty-service-config   52s

# TkeServiceConfig設定を更新して変更する
➜ kubectl edit tkeserviceconfigs.cloud.tencent.com jetty-service-config
TkeServiceConfig.cloud.tencent.com/jetty-service-config edited
```

LoadBalancerを使用してPodモードServiceにダイレクト接続します

最終更新日：：2023-04-26 18:43:22

ユースケース

ネイティブLoadBalancerモードのServiceは、Cloud Load Balancer(CLB)を自動的に作成でき、クラスターのNodeportを介してクラスター内に転送し、さらにiptableまたはipvsを介して2回転送します。このモードのServiceはほとんどのユースケースに対応できますが、以下のシナリオでは**ダイレクト接続PodモードのService**を使用することを推奨します。
ソースIPを取得する必要がある場合（非ダイレクト接続モードでは、Local転送を有効にする必要があります）。
より高い転送性能が求められる場合（非ダイレクト接続モードでは、CLBとServiceそのものとの間に2層のCLBがあるため、性能がある程度低下します）。
Podレベルまで完全なヘルスチェックやセッション維持が必要な場合（非ダイレクト接続モードでは、CLBとServiceそのものとの間に2層のCLBがあるため、ヘルスチェックやセッション維持機能の設定が困難になります）。

説明

クラスターがServerlessクラスターの場合、デフォルトでダイレクト接続Podモードになっているため、何も操作する必要はありません。
現在のGlobalRouterとVPC-CNIコンテナネットワークモードは、どちらもダイレクト接続Podモードをサポートしています。クラスターリストでクラスターIDをクリックして、クラスター詳細ページに進み、クラスターの「基本情報」ページで現在のクラスターが使用しているネットワークプラグインを確認できます。

コンテナネットワークモードをVPC-CNIにする

使用制限

クラスターKubernetesのバージョンは1.12以上である必要があります。
クラスターネットワークモードでは、VPC-CNI ENIモードを有効にする必要があります。
ダイレクト接続モードServiceが使用するワークロードは、VPC-CNI ENIモードを使用する必要があります。
デフォルトのCLBバックエンド数の制限は200です。バインドされたワークロードのコピーが200を超える場合、チケットを提出することによって、CLBのクォータを増やすことができます。
CLB自体にバインドされたENIの機能制限に対応します。詳細については、ENIのバインドをご参照ください。
ダイレクト接続Podモードのワークロード更新が有効になっている場合、CLBのヘルスチェック状況に基づいてローリングアップデートが行われるため、更新速度にある程度の影響が生じます。

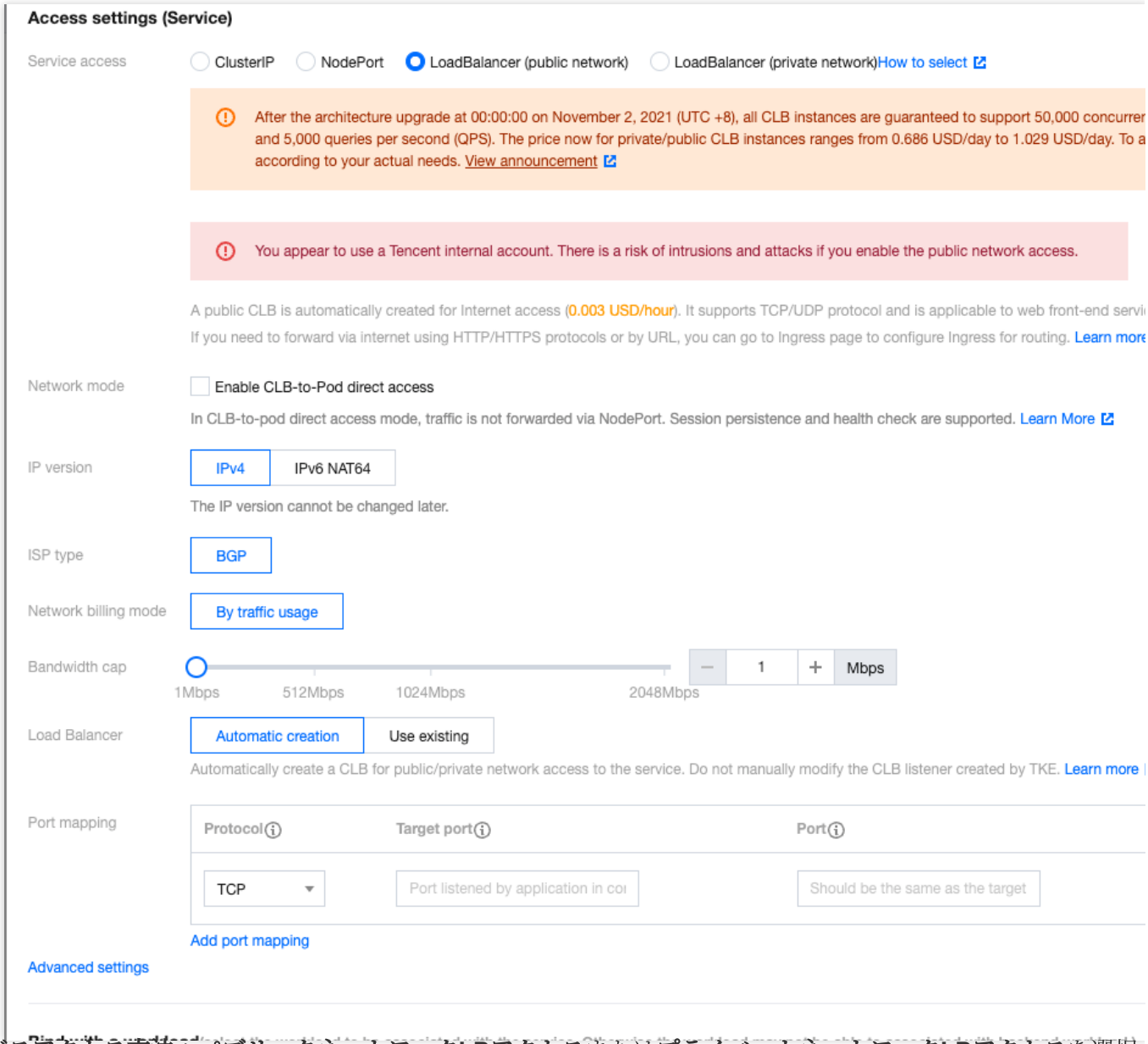
HostNetworkタイプのワークロードはサポートされていません。

操作手順

コンソール操作ガイド
YAML操作ガイド

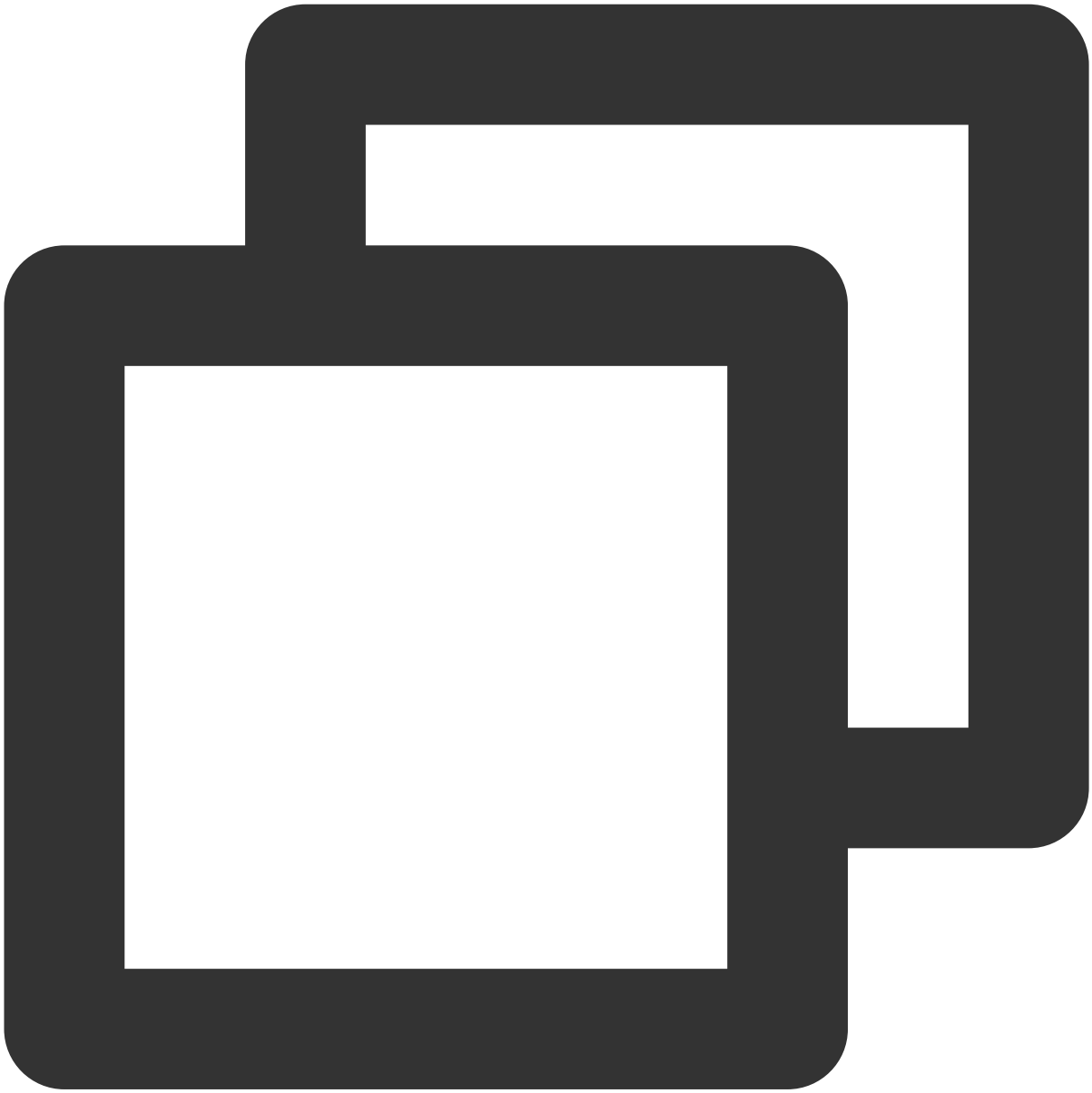
1. TKEコンソールにログインします。
2. コンソールでServiceを作成ステップを参照し、**Serviceの新規作成**ページに進み、実際のニーズに応じてServiceパラメータを設定します。

そのうち、いくつかの主要なパラメータ情報は、下の図のように設定する必要があります。



サービスアクセス方法：**パブリックネットワークLBアクセス**または**プライベートネットワークLBアクセス**を選択します。
ネットワークモード：**CLBダイレクト接続Podモードを使用**にチェックを入れます。

Workloadのバインド：**Workloadを参照**を選択します。
3. **サービスの作成**をクリックし、作成を完了します。
ダイレクト接続PodモードServiceのYAML設定は、通常のService YAML設定と同じです。例のannotationは、ダイレクト接続Podモードが有効かどうかを表します。

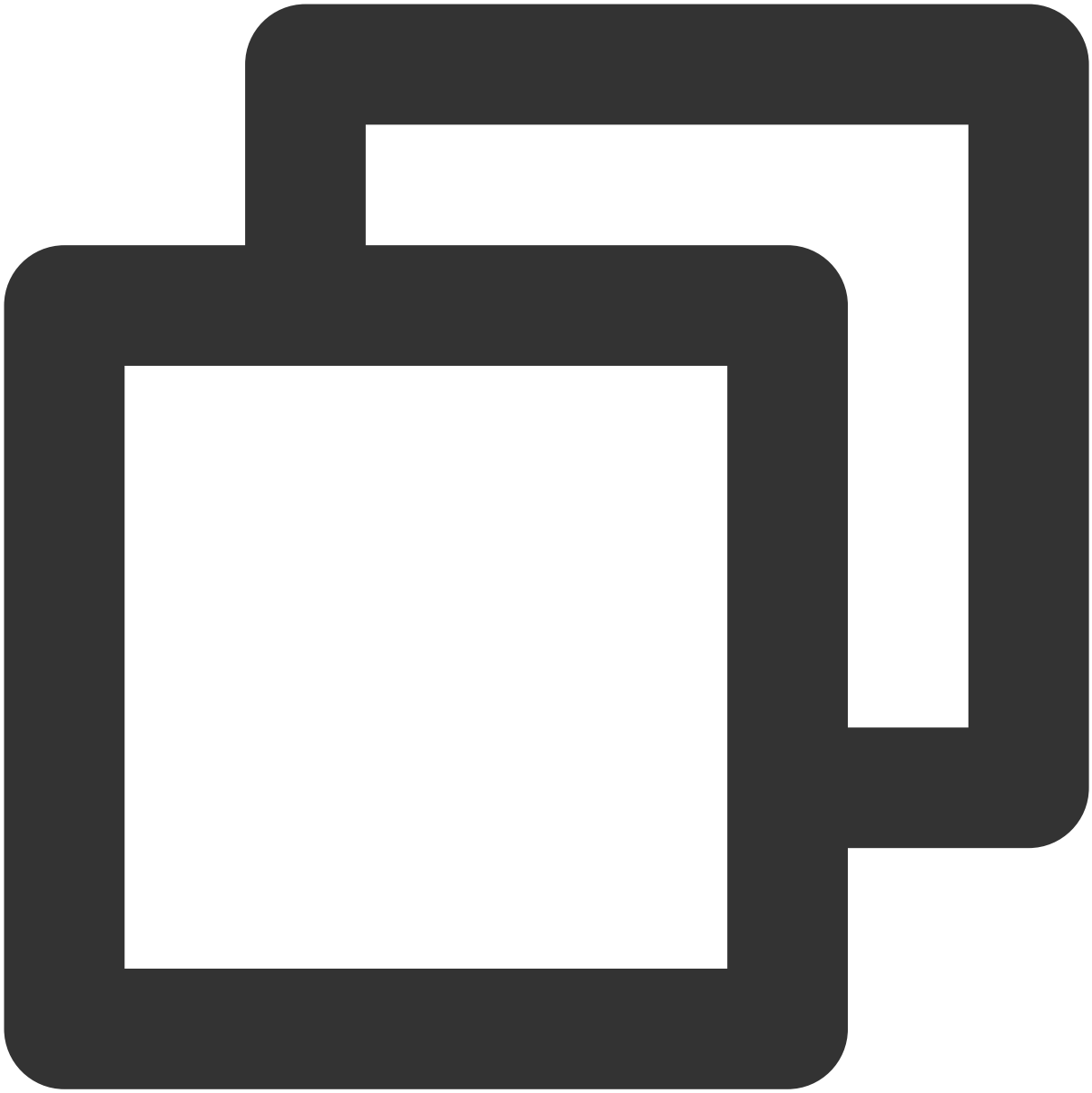


```
kind: Service
apiVersion: v1
metadata:
  annotations:
    service.cloud.tencent.com/direct-access: "true" ##ダイレクト接続Podモードの有効化
  name: my-service
```

```
spec:
  selector:
    app: MyApp
  ports:
  - protocol: TCP
    port: 80
    targetPort: 9376
  type: LoadBalancer
```

annotation拡張

CLBに関する設定については、TkeServiceConfigの説明をご参照ください。そのうちannotationに関する設定は次のとおりです。



```
service.cloud.tencent.com/tke-service-config: [tke-service-configName]
```

注意事項

ローリングアップデート時の可用性を保証する方法

Kubernetesが公式に提供しているReadinessGateは、主にPodのステータスを制御するために用いられる機能で、クラスターバージョンは1.12以上である必要があります。デフォルトでは、PodにはPodScheduled、Initialized、ContainersReadyというConditionがあります。これらのステータスがすべてReadyになったときに、Pod ReadyのConditionが成立します。ただし、クラウドネイティブのシナリオでは、Podのステータスが他のステータスを参照

する必要がある場合があります。ReadinessGateでは、Podのステータスの判断にフェンスを追加し、サードパーティが判断・制御できる仕組みを提供しています。これにより、Podのステータスはサードパーティにバインドされます。

ダイレクト接続モードでのローリングアップデートの変化

ユーザーがアプリケーションのローリングアップデートを開始すると、Kubernetesはアップデートポリシーに基づいてアップデートを行います。ただし、Podの起動を判断するためのマーカーにはPod自体のステータスだけが含まれ、このPodがCLBでのヘルスチェックをパスするように設定されているどうかは考慮されません。アクセス層のコンポーネントに高い負荷がかかっている際、そのようなPodを速やかにスケジューリングできない場合、ローリングアップデートに成功したPodが対外的にサービスを提供せず、その結果サービスの中断が生じる可能性があります。
TKEアクセス層コンポーネントは、ローリングアップデートやCLBのバックエンドステータスをバインドするために、Kubernetes 1.12で導入された新機能である `ReadinessGate` を導入しています。TKEアクセス層コンポーネントのバックエンドは、バインドが成功し、ヘルスチェックにパスしたことを確認した場合のみ、 `ReadinessGate` のステータスを設定することで、Podのステータスを`Ready`にして、ワークロード全体のローリングアップデートを進めるようになっています。

クラスターでのReadinessGateの使用

Kubernetesクラスターは、サービス登録の仕組みを提供しているため、サービスを MutatingWebhookConfigurations リソースという形でクラスターに登録するだけで利用できます。クラスターは `Pod`が作成されると、設定されたコールバックパスに従ってクラスターに通知されます。この際、作成前の操作として、`Pod`に ReadinessGate を追加することができます。このコールバックプロセスは`HTTPS`である必要があります。つまり、 MutatingWebhookConfigurations`でリクエストを発行するCAを設定し、そのCAが発行する証明書をサーバー側で設定する必要があります。

ReadinessGateのメカニズムのディザスタリカバリ

ユーザーのクラスター内のサービス登録や証明書は、ユーザーによって削除される可能性があります。これらのシステムコンポーネントリソースはユーザーが変更や破棄をすべきではありませんが、ユーザーがクラスターを検索したり、操作を誤ったりすると、このような問題は避けて通れません。そのため、これらのリソースの整合性は起動時にアクセス層コンポーネントによってチェックされ、整合性が損なわれた場合には再構築され、システムの堅牢性を高めます。詳細については、Kubernetes Pods ReadinessGateの特性をご参照ください。

コンテナのネットワークモードをGlobalRouterにする

使用制限

個々のワークロードは1つのネットワークモードでのみ実行できます。ENIまたはGlobalRouteとのいずれかのダイレクト接続を選択できます。

帯域幅でのアカウント転送のみをサポートしています。
デフォルトのCLBバックエンド数の制限は200です。バインドされたワークロードのコピーが200を超える場合、チケットを提出することによって、CLBのクォータを増やすことができます。
CLBを使用してPodにダイレクト接続する場合は、ネットワークリンクがCVMのセキュリティグループによる制限を受けることに注意し、セキュリティグループの設定が対応するプロトコルやポートに対して開放されていること、**CVM上のワークロードに対応するポートが有効になっていること**を確認する必要があります。
ダイレクト接続を有効にすると、デフォルトでReadinessGateReadinessチェックが開始され、Podをローリングアップデートする際に、CLBのトラフィックが正常かどうかをチェックします。ビジネスサイドで正しいヘルスチェックの設定を行う必要があります。詳細については、TkeServiceConfigの説明をご参照ください。
ダイレクト接続Globalrouterモードの Podは、次の2つの方法で使用できます。
****CCN**経由で使用します。**CCNがバインドされたIPアドレスを検証し、バインドエラーやアドレスループバックなどのよくある問題を防ぐことができるため、この方法を使用することを推奨します。操作手順は次のとおりです。
a. CCNインスタンスを作成済みであること。詳細については、CCNインスタンスの新規作成をご参照ください。
b. クラスターが配置されているVPC を、作成済みのCCNインスタンスに追加します。
c. コンテナネットワークセグメントをCCNに登録し、クラスターの**基本情報**ページでCCNを有効にします。
****チケットを提出**して申請できます。**この方法は、CCNのIP検証機能を備えていないため、推奨しません。

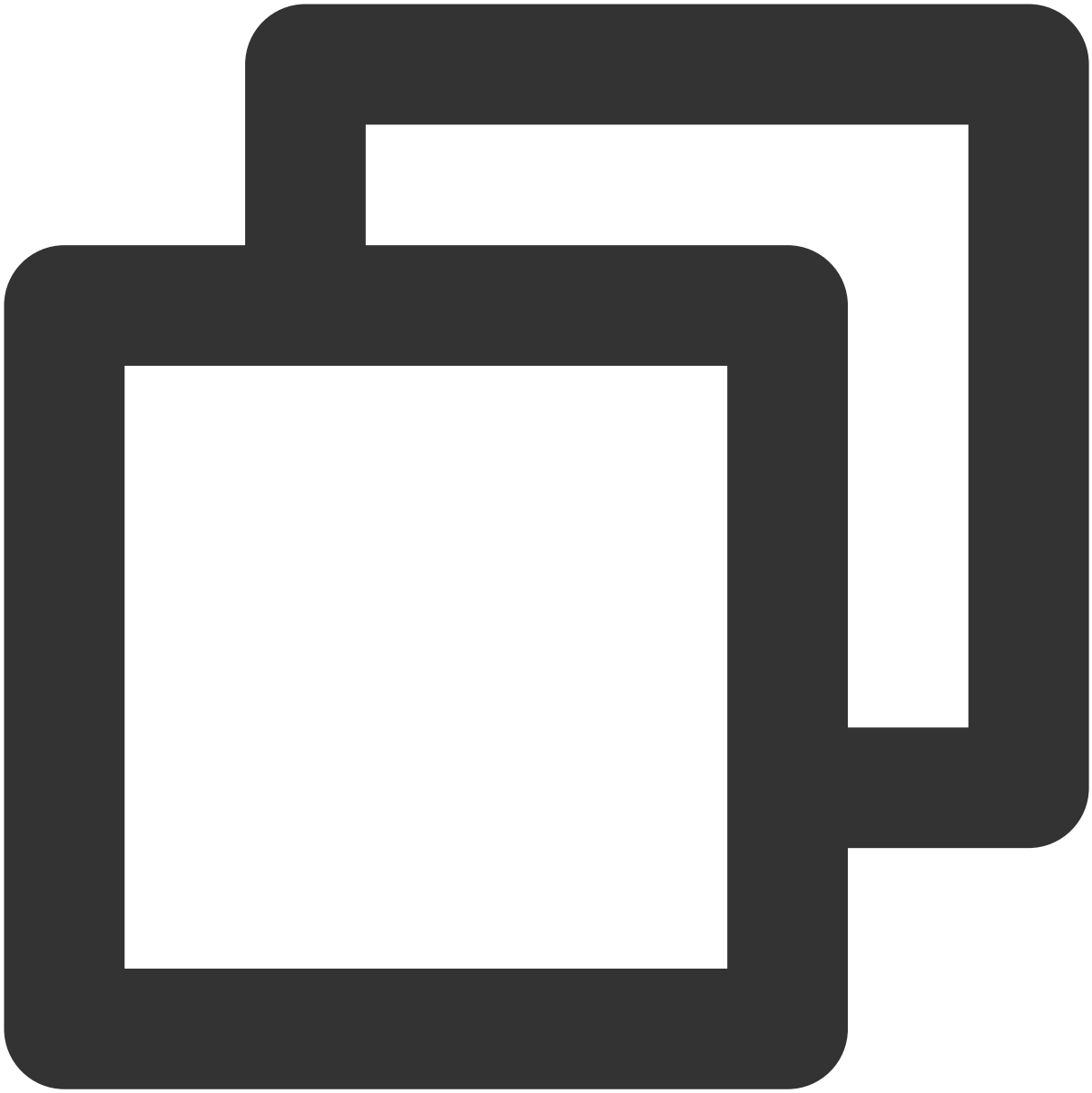
YAML操作ガイド

ダイレクト接続PodモードServiceのYAML設定は、通常のService YAML設定と同じです。例のannotationは、ダイレクト接続Podモードが有効かどうかを表します。

使用の前提条件

`kube-system/tke-service-controller-config` ConfigMapに `GlobalRouteDirectAccess: "true"` を追加して、GlobalRouteダイレクト接続機能を有効にします。

Service YAMLでのダイレクト接続モードの有効化

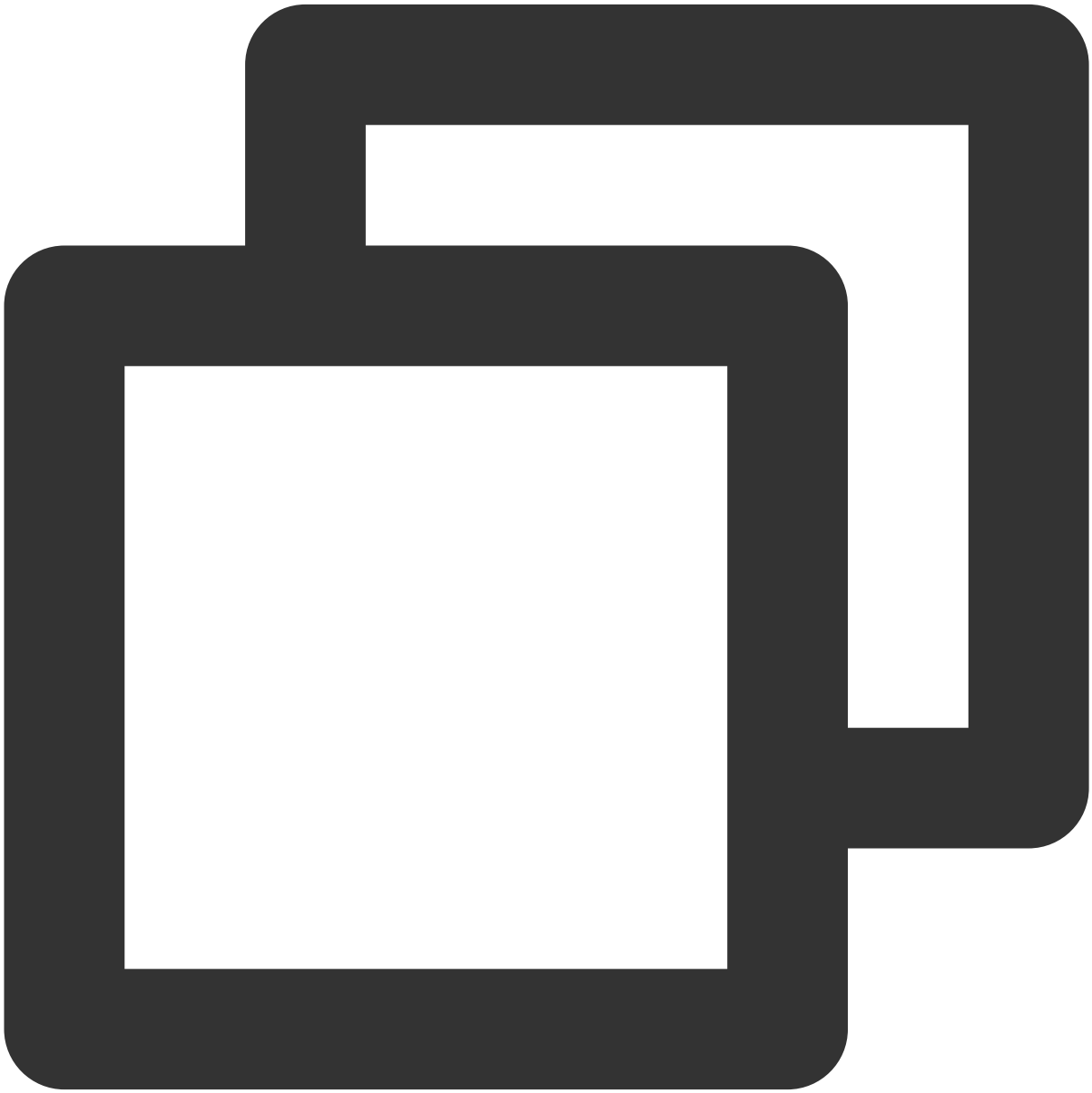


```
kind: Service
apiVersion: v1
metadata:
  annotations:
    service.cloud.tencent.com/direct-access: "true" ##ダイレクト接続Podモードの有効化
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
  - protocol: TCP
```

```
    port: 80
    targetPort: 9376
  type: LoadBalancer
```

annotation拡張

CLBに関する設定については、TkeServiceConfigの説明をご参照ください。そのうちannotationに関する設定は次のとおりです。



```
service.cloud.tencent.com/tke-service-config: [tke-service-configName]
```

マルチServiceによるCLB再利用

最終更新日：：2023-04-27 16:40:34

ユースケース

複数のServiceで同じCLBの機能を再利用することによって、同一のVIPでTCPおよびUDPが同じポートを同時に公開することをサポートします。

注意

その他のケースではいずれも複数のServiceを使用して同じCLBを再利用することをお勧めしません。

説明事項

2020年8月17日より前に作成したTKEクラスターについて、そのServiceが作成したCLBはデフォルトで同じCLBの再利用をサポートします。

2020年8月17日以降に作成した TKEクラスターは、デフォルトで複数のServiceが同じCLBを再利用する機能がオフになっています。

チケットを提出することによって複数のServiceを使用して同じCLBを再利用する機能を有効にする必要があります。

クラスターがTKE Serverlessクラスターの場合、クラスターはデフォルトでCLBの再利用機能が有効になりますが、以下の内容に注意が必要です。

1.1 再利用に使用されるCLBはユーザーが手動で購入する必要があり、Serverlessクラスターは自動で購入しません。Serverlessクラスターが自動購入するCLBを再利用する時にエラーが発生するのは、CLBを再利用するServiceのCLBがServerlessクラスターによって回収されるのを防ぐためです。

1.2 CLBの購入に成功したら、Service内で2つのAnnotationを追加する必要があります。

service.kubernetes.io/qcloud-share-existed-lb:"true"

service.kubernetes.io/tke-existed-lbid:lb-xxx

ServiceとCLBの間に設定された管理および同期はCLB IDを名前とするLoadBalancerResourceタイプのリソースオブジェクトです。このCRDにはいかなる操作もしないでください。そうしない場合、Serviceの無効化が発生しやすくなります。

使用制限

Serviceを再利用するケースでは、単一のCLB管理のリスナー数量はCLBのTOTAL_LISTENER_QUOTAによって制限されます。詳細については、ドキュメントを見るをご参照ください。

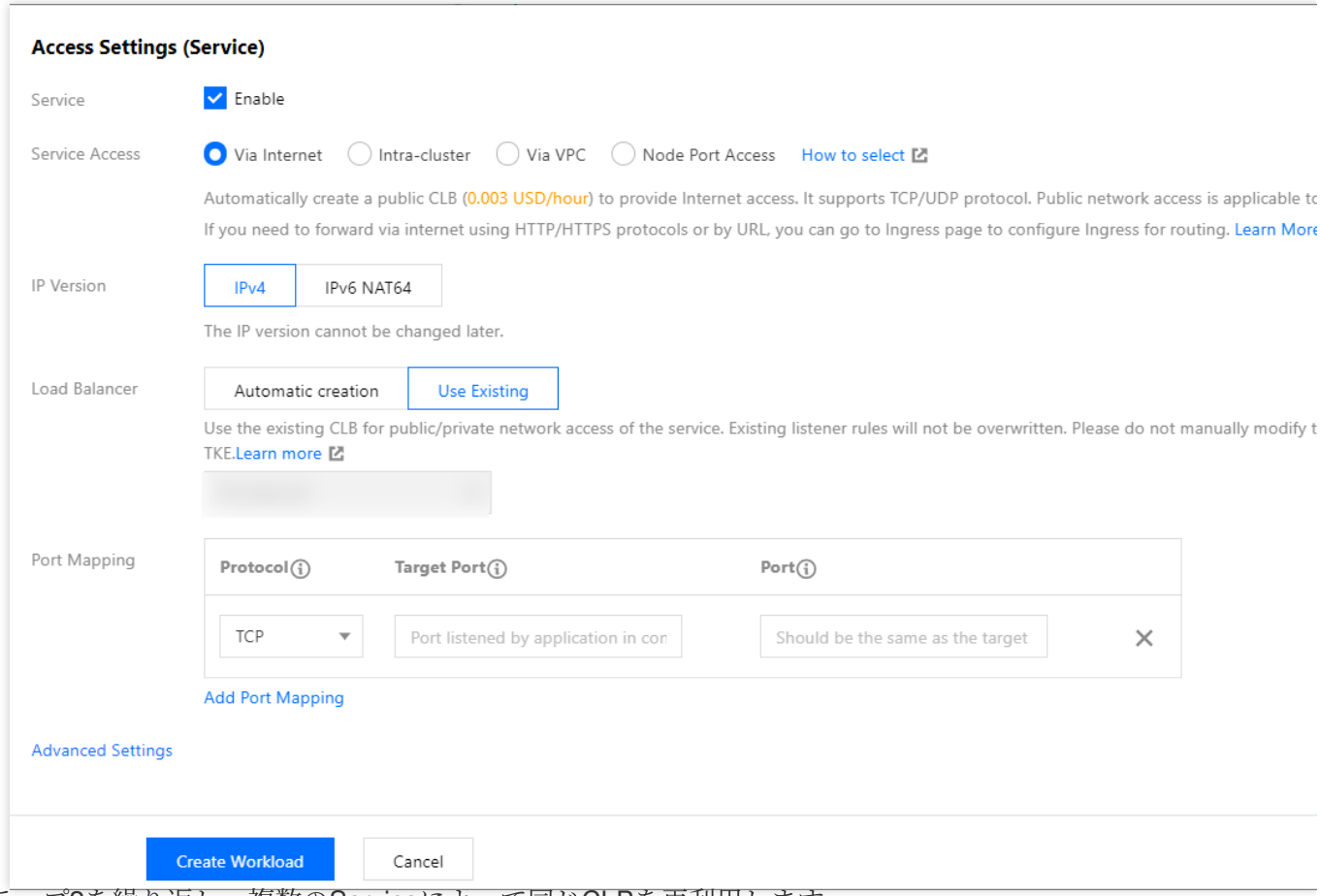
Serviceを再利用するケースでは、ユーザーが自ら作成したCLBのみ使用することができます。TKEクラスターが作成したCLBが再利用される状況では、CLBリソースがリリースできず漏洩する可能性があるためです。

注意

現在TKEが作成したCLBリソースを使用して再利用を行うと、タグが不足しているため、このCLBのライフサイクルはTKE側によって制御されないので、ご自身で管理する必要があります。慎重に操作してください。

操作手順

1. CLBインスタンスの作成を参照して、クラスターがVPCにあるパグリックネットワークまたはプライベートネットワークタイプのCLBを作成します。
2. Deploymentの作成またはServiceの作成を参照して、Loadbalancerタイプの Serviceを作成します。**既存**のCLBを使用するを選択し、ステップ1 で作成したCLBインスタンスを選択します。下図のように表示されます。



3. ステップ2を繰り返し、複数のServiceによって同じCLBを再利用します。

Service拡張プロトコル

最終更新日：：2023-05-06 19:41:07

Serviceがデフォルトでサポートするプロトコル

Serviceとは、Kubernetesがアプリケーションプログラムをクラスター外に公開する一種のメカニズムであり抽象的なものです。Serivceを介してクラスター内のアプリケーションプログラムにアクセスすることができます。

注意

シナリオへのダイレクト接続 からアクセスします。拡張プロトコルを使用する場合、時間の制限はありません。
TCPおよびUDPプロトコルの併用をサポートします。
ダイレクト接続ではないシナリオで、ClusterIPおよびNodePortモードは併用をサポートします。ですがコミュニティのLoadBalanceタイプのServiceには制限があります。現在使用できるのは同じタイプのプロトコルのみです。
LoadBalance表明がTCPの場合、ポートは拡張プロトコルの機能を利用できます。Cloud Load Balancer（CLB）のプロトコルをTCP_SSL、HTTP、HTTPSに変更します。
LoadBalance表明がUDPの場合、ポートは拡張プロトコルの機能を利用できます。CLBのプロトコルをUDPに変更します。

TKEのService転送プロトコルの拡張

ネイティブServiceがサポートするプロトコルのルールでは、一部のシナリオにおいてService上でTCPとUDPの併用を同時にサポートする必要があり、かつ、ServiceはTCP SSL、HTTP、HTTPSプロトコルをサポートしなければなりません。TKEはLoadBalancerモードを拡張し、 さまざまなプロトコルへのサポートを行います。

前提についての説明

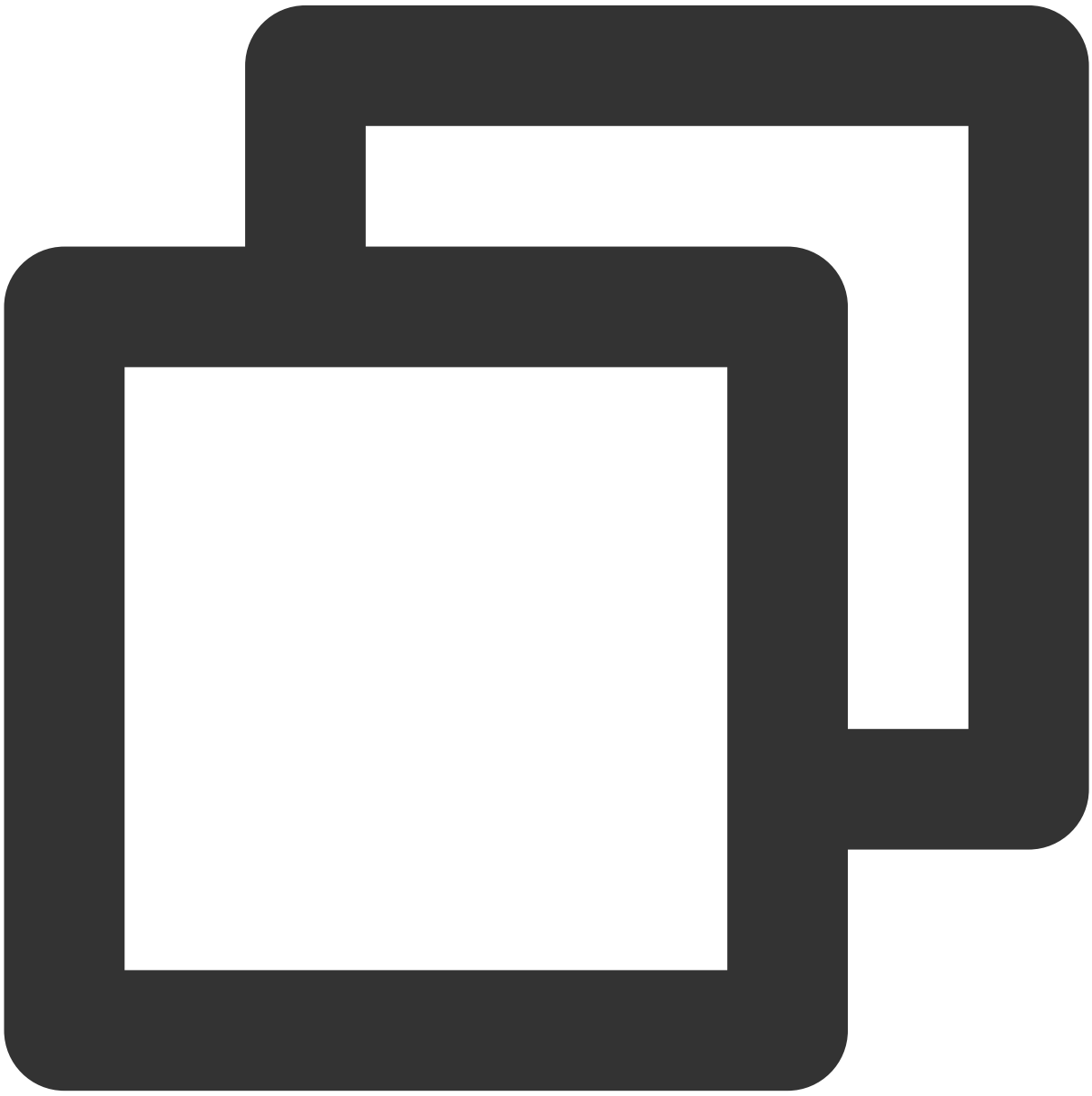
拡張プロトコルはLoadBalancerモードのServiceに対してのみ有効です。
拡張プロトコルではAnnotationの形式でプロトコルとポートの関係を説明します。
拡張プロトコルとAnnotationの関係は以下のとおりです。
拡張プロトコルがアノテーションの中で、Service Specで説明されているポートをカバーしなかった場合、Service Specはユーザー説明に従って設定されます。
拡張プロトコルがアノテーションの中で説明したポートがService Specに存在しない場合は、この設定を無視してください。
拡張プロトコルがアノテーションの中で説明したポートがService Specに存在する場合は、ユーザーがService Specで表明したプロトコル設定をカバーします。

アノテーションの名称

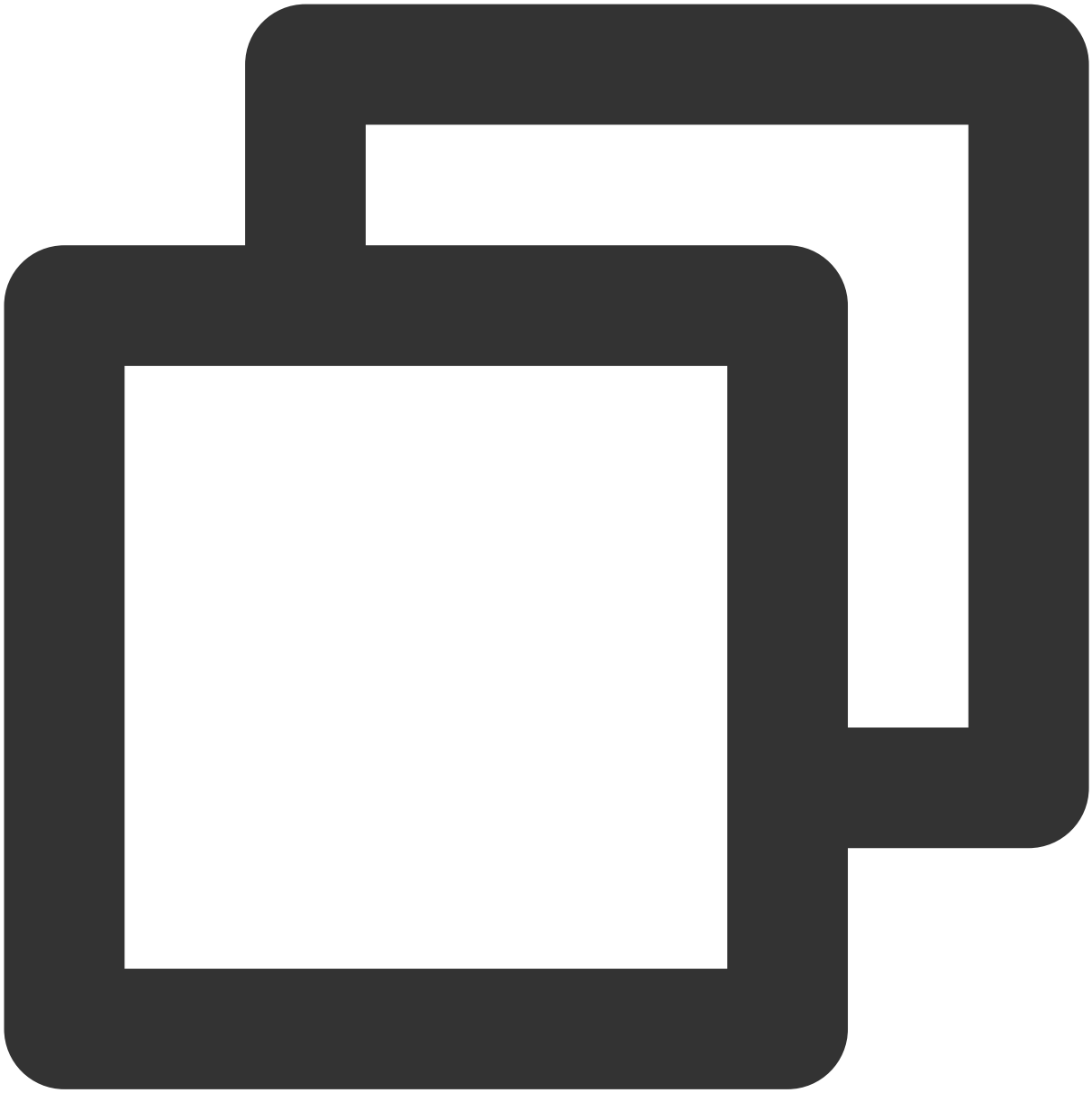
service.cloud.tencent.com/specify-protocol

プロトコルアノテーション拡張の事例

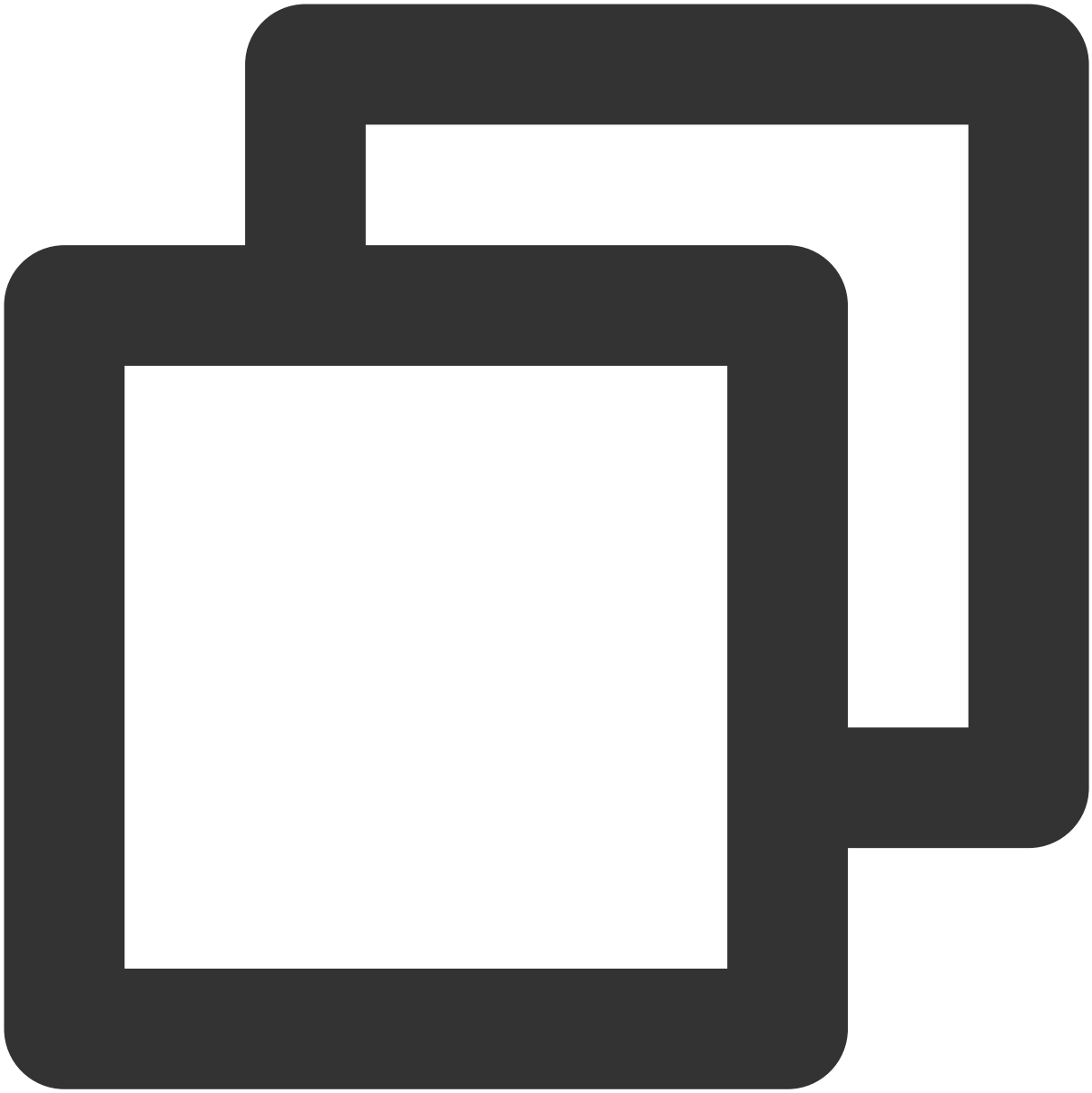
TCP_SSLの事例
HTTPの事例
HTTPSの事例
TCP/UDPハイブリッドの事例
ハイブリッドの事例
QUIC



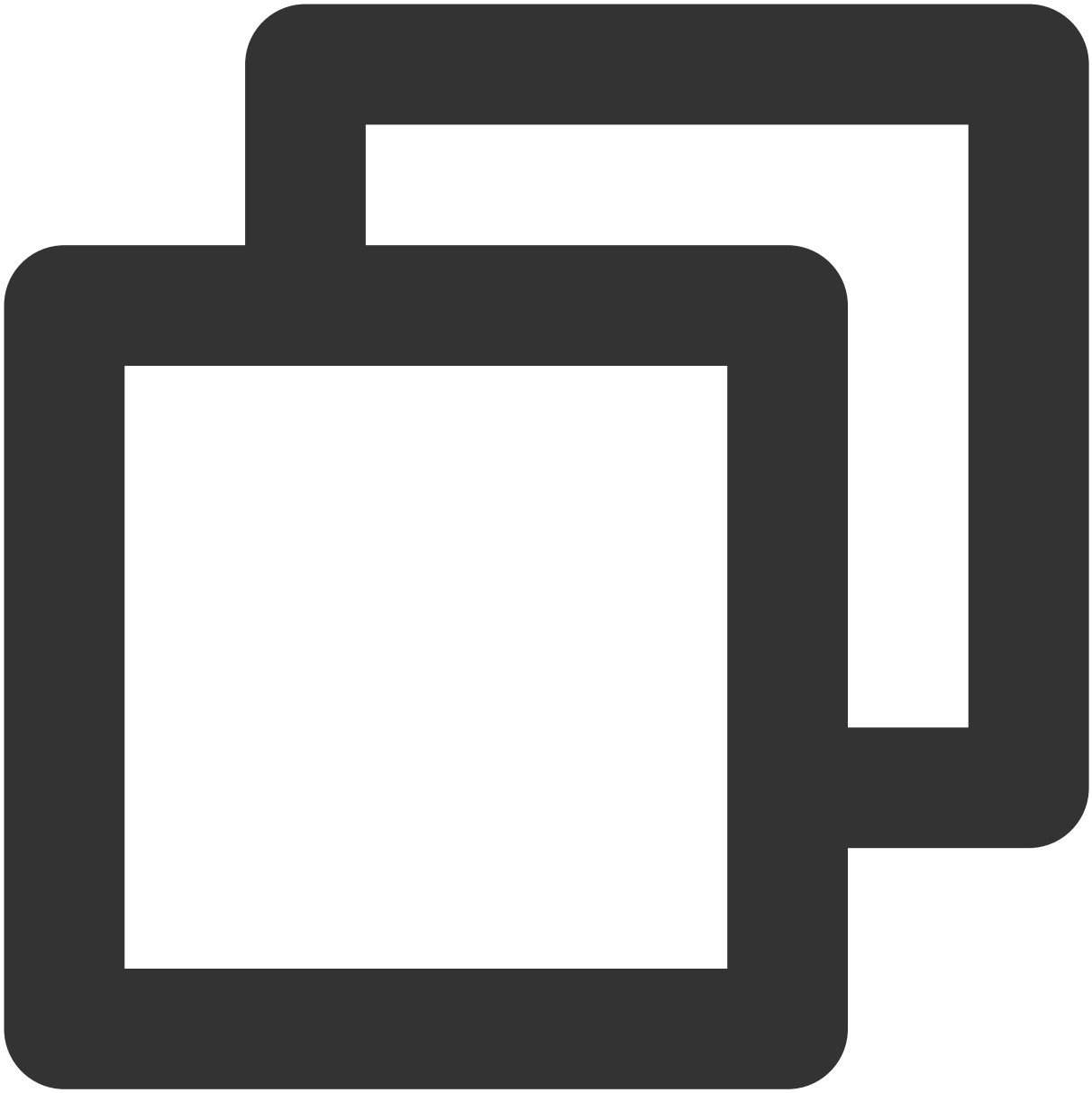
```
{"80":{"protocol":["TCP_SSL"],"tls":"cert-secret"}}
```



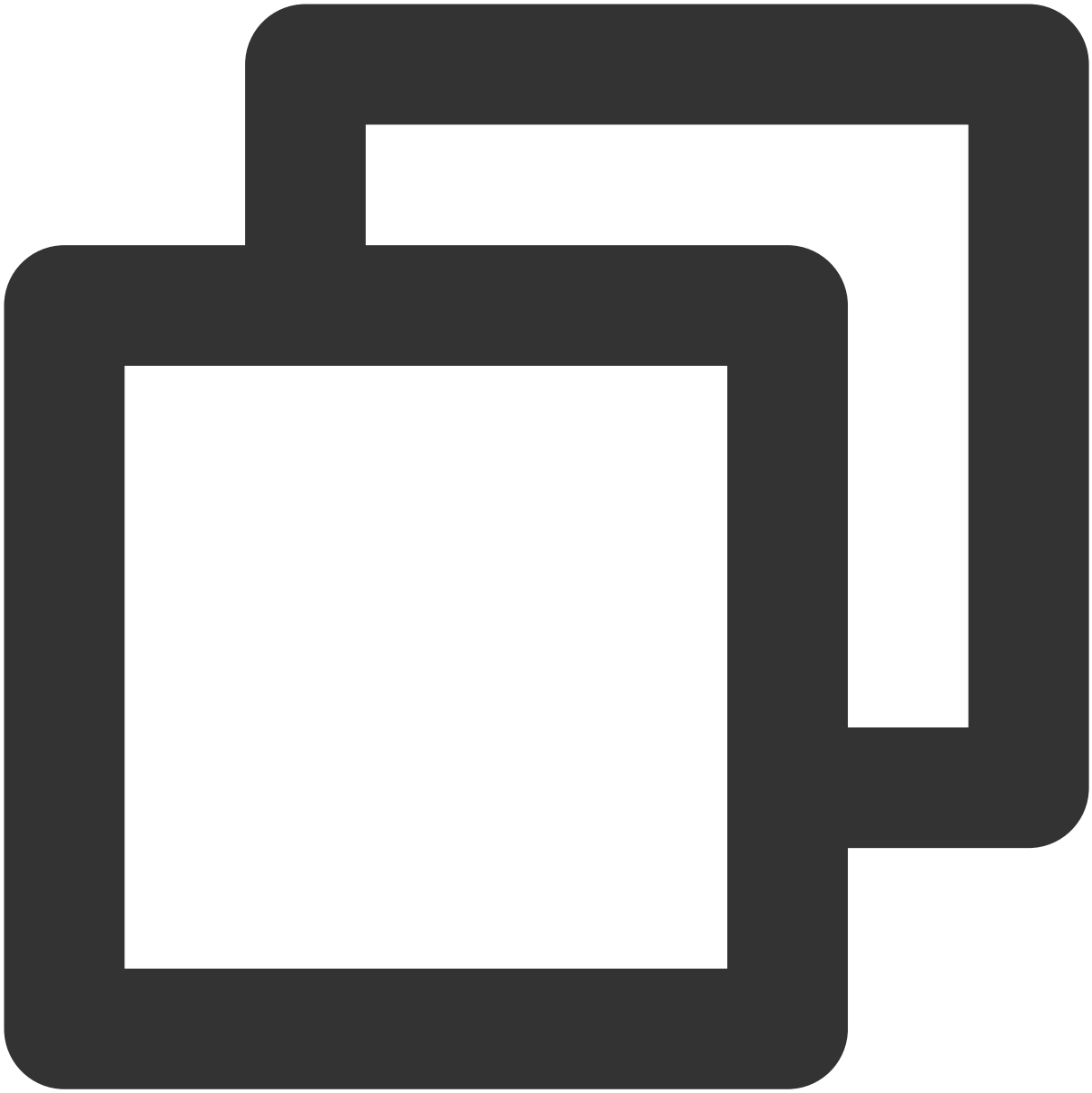
```
{"80":{"protocol":["HTTP"],"hosts":{"a.tencent.com":{},"b.tencent.com":{}}}}
```



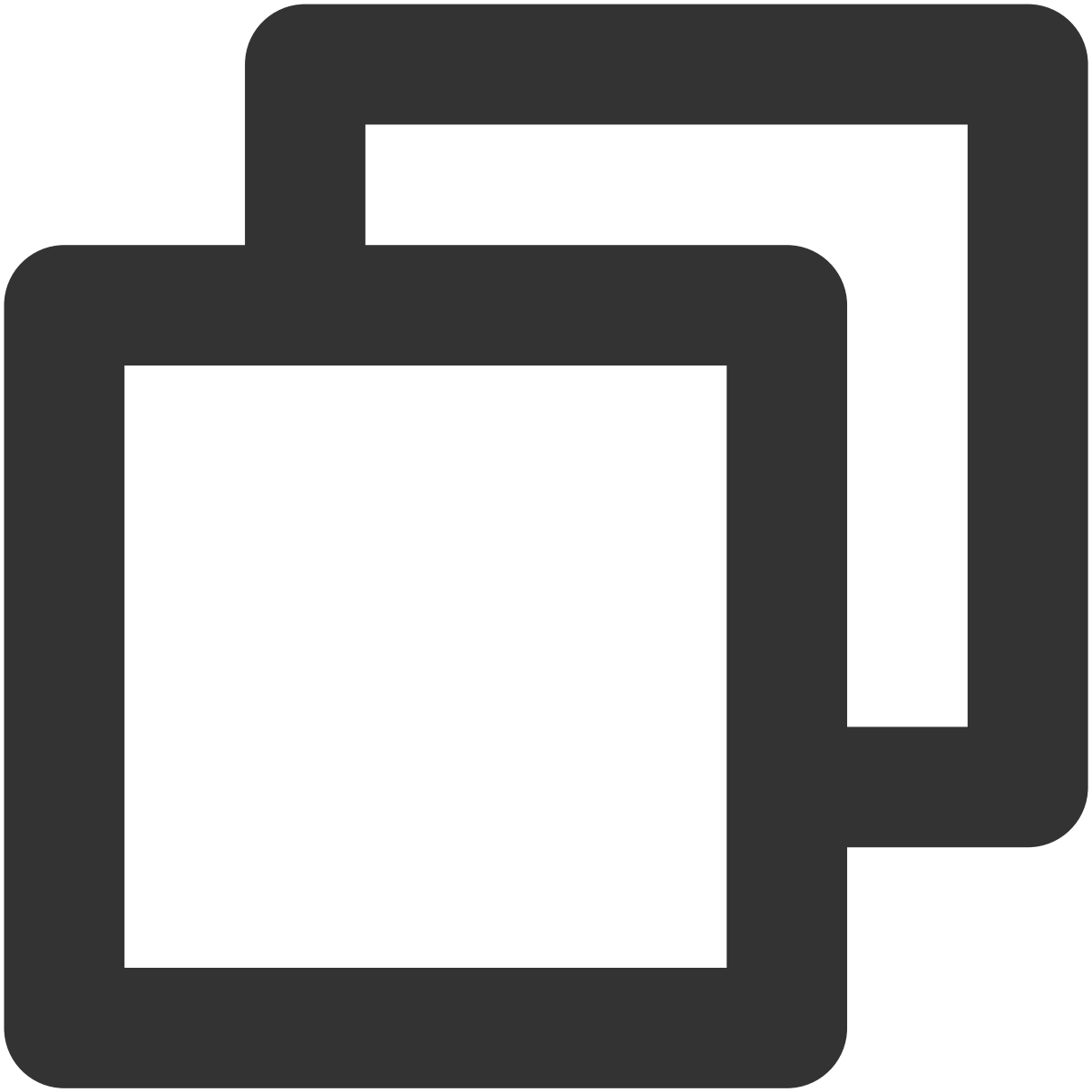
```
{"80":{"protocol":["HTTPS"],"hosts":{"a.tencent.com":{"tls":"cert-secret-a"},"b.te
```



```
{"80":{"protocol":["TCP","UDP"]}} # ダイレクト接続モードのみのサポートです。詳しくは、https:
```



```
{"80":{"protocol":["TCP_SSL","UDP"],"tls":"cert-secret"}} # ダイレクト接続モードのみの
```



```
{"80":{"protocol":["QUIC"],"tls":"cert-secret"}}
```

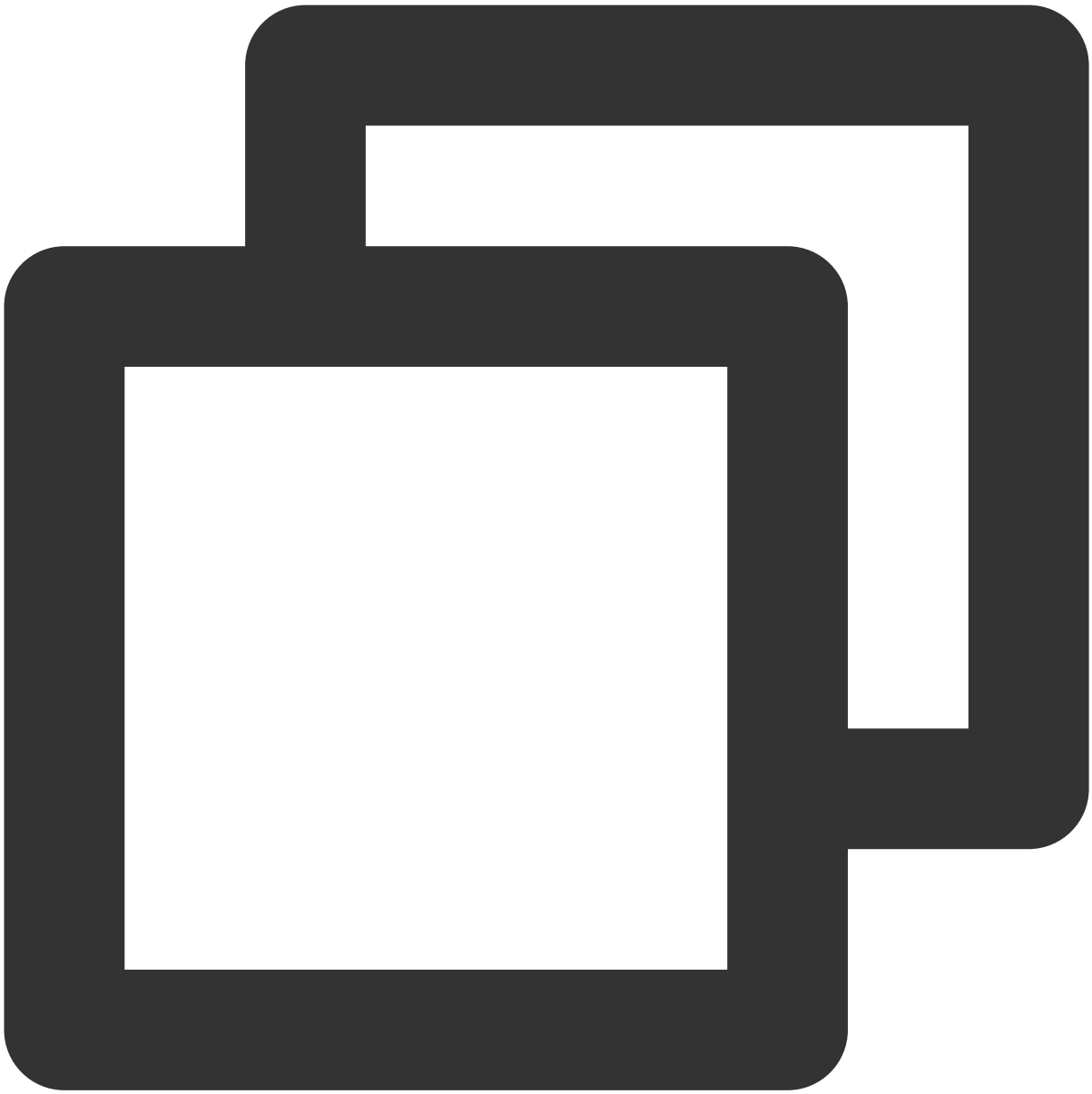
注意

TCP_SSLおよびHTTPS内のフィールド `cert-secret` は、そのプロトコルを使用するには証明書を1つ指定する必要があることを表しています。証明書はOpaqueタイプのSecretで、SecretのKeyはqcloud_cert_idです。Valueは証明書のIDです。詳しい内容については Ingress証明書の設定をご参照ください。

拡張プロトコルの使用説明

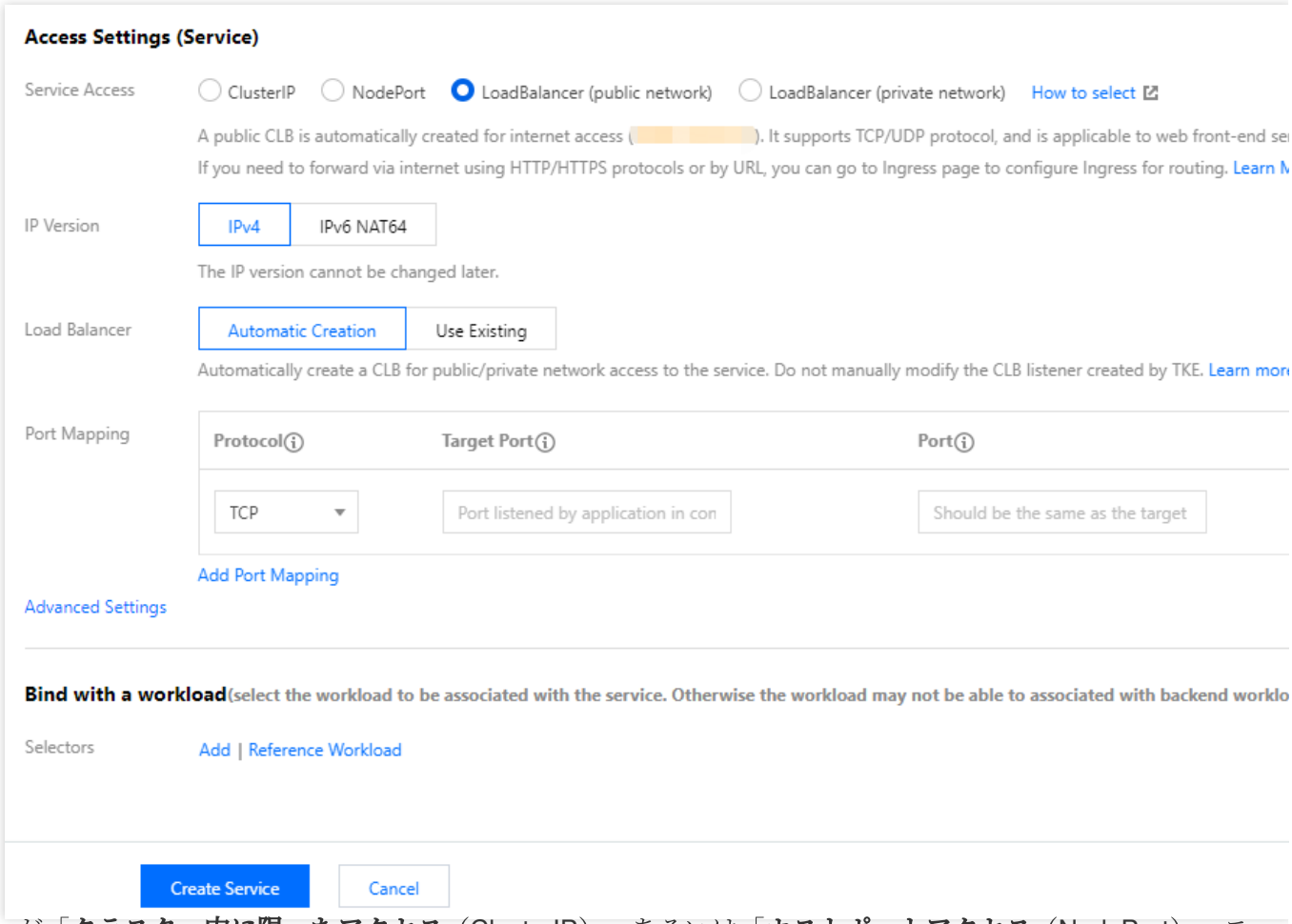
拡張プロトコルYAMLの使用説明

拡張プロトコルコンソールのご利用説明



```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.cloud.tencent.com/specify-protocol: '{"80":{"protocol":["TCP_SSL"],"tls
  name: test
   ....
```

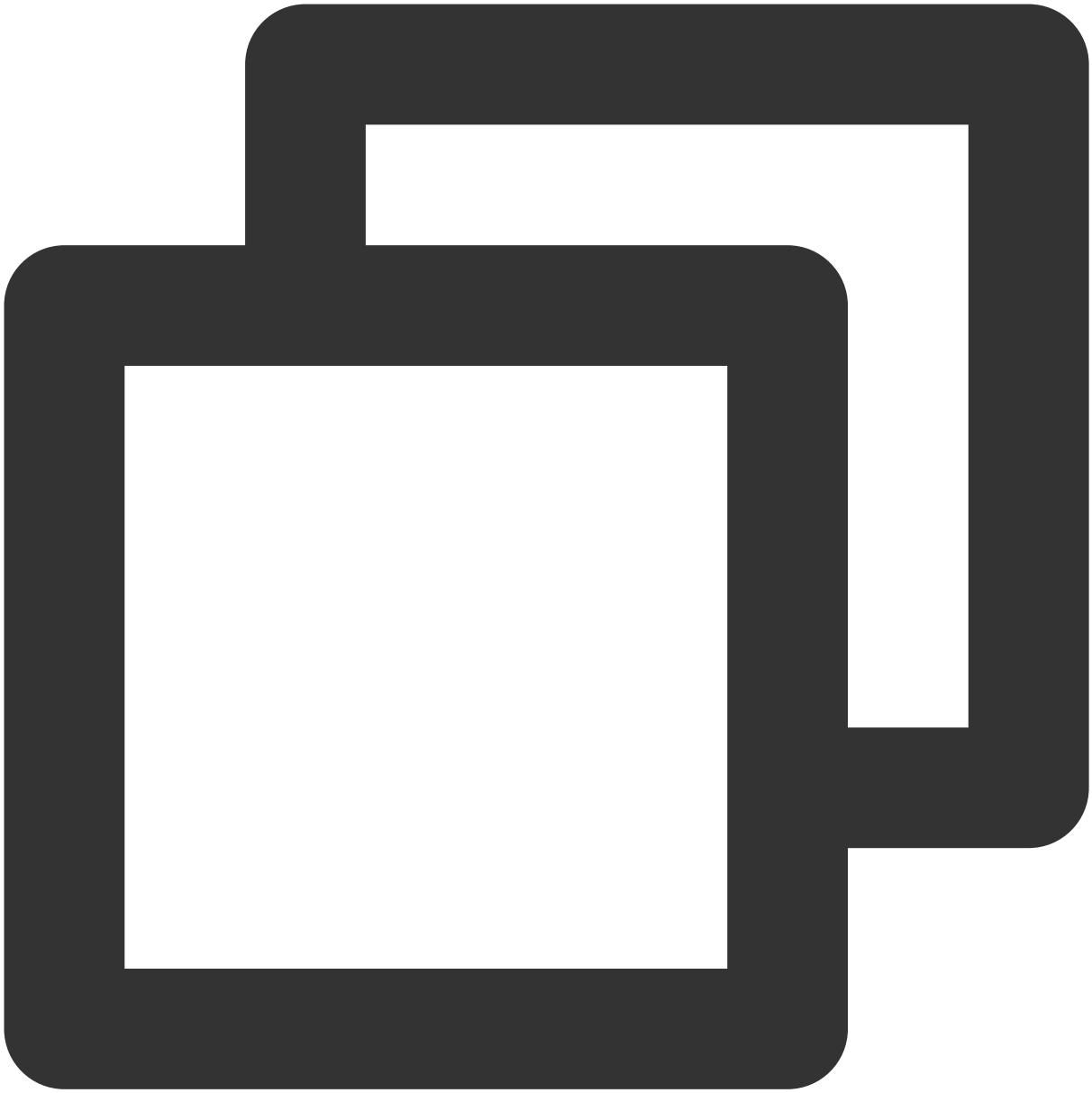
Service作成時に、「**パブリックネットワークLB**」あるいは「**プライベートネットワークLB**」の形式でサービスを公開する場合は、 直接接続モード ではない状況下の「ポートマッピング」の中で、TCPとTCP SSLの同時使用のみをサポートします。下図に示すとおりです。



Serviceが「**クラスター内に限ったアクセス**（ClusterIP）」あるいは「**ホストポートアクセス**（NodePort）」モードの場合は、任意のプロトコルの併用をサポートします。
直接接続モードは、任意のプロトコルの併用をサポートします。

事例の説明

ネイティブServiceがプロトコルの併用をサポートしていない場合、TKEは特殊な改造後に、 シナリオへのダイレクト接続 にて、ハイブリッドプロトコルの使用をサポートします。
注意が必要なのは、YAMLの中ではまだ同じプロトコルが使用されている点ですが、各ポートのプロトコルタイプはAnnotationによって明らかになっています。次に、80ポートがTCPプロトコルを使用し、8080ポートがUDPプロトコルを使用している事例を示します。



```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.cloud.tencent.com/direct-access: "true"  #TKE Serverlessクラスターのデフォ
    service.cloud.tencent.com/specify-protocol: '{"80":{"protocol":["TCP"]},"8080":
  name: nginx
spec:
  externalTrafficPolicy: Cluster
  ports:
  - name: tcp-80-80
```

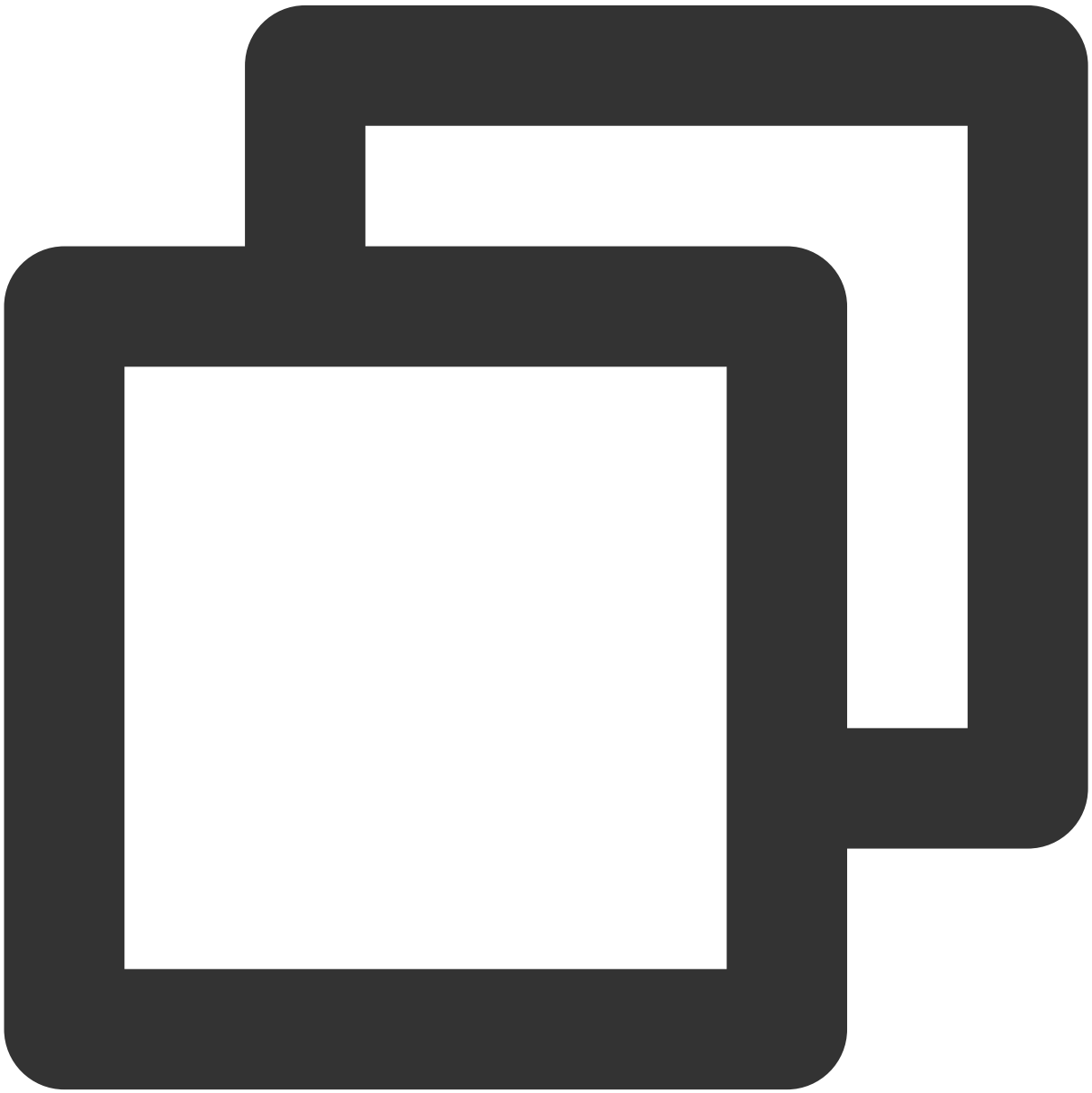
```
    nodePort: 32150
    port: 80
    protocol: TCP
    targetPort: 80
  - name: udp-8080-8080
    nodePort: 31082
    port: 8080
    protocol: TCP # Kubernetes Service Controllerには制限があり、同じタイプのプロトコルしか
    targetPort: 8080
  selector:
    k8s-app: nginx
    qcloud-app: nginx
  sessionAffinity: None
  type: LoadBalancer
```

Service Annotationの説明

最終更新日：：2023-04-27 18:15:01

以下のAnnotationによってServiceを設定することで、より豊富なCLBの機能を実現します。

アノテーションの使用方法



```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.kubernetes.io/tke-existed-lbid: lb-6swtxxxx
  name: test
........
```

Annotationセット

service.kubernetes.io/loadbalance-id

説明：

読み取り専用アノテーションは、現在Serviceが引用しているCLB LoadBalanceIdを提供します。Tencent Cloud CLBコンソールでクラスターと同じVPC下のCLBインスタンスIDを確認することができます。

service.kubernetes.io/qcloud-loadbalancer-internal-subnetid

説明：

このAnnotationによってプライベートネットワークタイプを作成するCLBを指定します。値はサブネットIDです。

使用例：

`service.kubernetes.io/qcloud-loadbalancer-internal-subnetid: subnet-xxxxxxxx`

service.kubernetes.io/tke-existed-lbid

説明：

既存のCLBを使用します。さまざまな使用方法がTencent Cloudタグに与える影響にご注意ください。

使用例：

使用方法の詳細については、Serviceに既存のCLBを使用するをご参照ください。

service.kubernetes.io/local-svc-only-bind-node-with-pod

説明：

Service Localモードではpodが存在するノードのみバインドします。

使用例：

使用方法の詳細については、Service Localモードをご参照ください。

service.cloud.tencent.com/local-svc-weighted-balance

説明：

Annotation `service.kubernetes.io/local-svc-only-bind-node-with-pod` と組み合わせて使用します。
CLBバックエンドの重みはノード上のワークロードの数量によって決定されます。
使用例：
使用方法の詳細については、Service Localモードをご参照ください。

service.kubernetes.io/qcloud-loadbalancer-backends-label

説明：
タグを指定してCLBバックエンドにバインドするノードを設定します。
使用例：
使用方法の詳細については、アクセス層のバックエンドを指定するをご参照ください。

service.cloud.tencent.com/direct-access

説明：
CLBを使用してPodに直接接続する。
使用例：
使用方法の詳細については、LoadBalancerを使用してPodモードServiceに直接接続するをご参照ください。

service.cloud.tencent.com/tke-service-config

説明：
tke-service-configによってCLBを設定します。
使用例：
使用方法の詳細については、Service CLBの設定をご参照ください。

service.cloud.tencent.com/tke-service-config-auto

説明：
このアノテーションによってTkeServiceConfigを自動作成します。
使用例：
使用方法の詳細については、ServiceとTkeServiceConfigの関連行動をご参照ください。

service.kubernetes.io/loadbalance-nat-ipv6

説明：
読み取り専用アノテーションは、NAT64 IPv6 CLBを作成した場合、CLBのIPv6アドレスがアノテーション内に表

示されます。

使用例：

```
service.kubernetes.io/loadbalance-nat-ipv6: "2402:4e00:1402:7200:0:9223:5842:2a44"
```

service.kubernetes.io/loadbalance-type（間もなく破棄されます）

説明：

自動作成されたCLBタイプ、従来型CLB、アプリケーション型CLBを制御します。

選択可能な値：yunapi_clb（従来型）、classic（従来型）、yunapiv3_forward_clb（アプリケーション型）

デフォルト値：yunapiv3_forward_clb（アプリケーション型）

注意

特別な理由がない限り、従来型CLBを使用することはお勧めしません。従来型CLBはイテレーションを停止してオフラインにする準備をしており、多くの機能が不足しています。

service.cloud.tencent.com/specify-protocol

説明：

アノテーションよって指定のリスニングポートにTCP、UDP、TCP SSL、HTTP、HTTPSを設定することをサポートしています。

使用例：

使用方法の詳細については、Service拡張プロトコルをご参照ください。

service.kubernetes.io/service.extensiveParameters

説明：

このAnnotationが使用するのはCLB作成時のパラメータです。現在は作成時のみ設定をサポートし、作成後は変更をサポートしておらず、作成後、本アノテーションの変更は無効です。

CLBインスタンスの作成を参照してCLBに追加のカスタムパラメータを作成します。

使用例：

NAT64 IPv6インスタンスの作成：

service.kubernetes.io/service.extensiveParameters: '{"AddressIPVersion":"IPV6"}'

チャイナテレコムCLBを購入する：

service.kubernetes.io/service.extensiveParameters: '{"VipIsp":"CTCC"}'

作成時にCLB名をカスタマイズする：

service.kubernetes.io/service.extensiveParameters: '{"LoadBalancerName":"my_cutom_lb_name"}'

service.cloud.tencent.com/enable-grace-shutdown

説明：
CLB直接接続モードのグレースフルシャットダウンをサポートしています。Podは削除され、このときPod内にはDeletionTimestampがあり、ステータスがTerminatingに設定されます。このときCLBはこのPodの重みが0になるように調整されます。
使用例：
直接接続モードでのみサポートされ、`service.cloud.tencent.com/direct-access`を組み合わせて使用する必要があります。使用方法の詳細については、Serviceのグレースフルシャットダウンをご参照ください。

service.cloud.tencent.com/enable-grace-shutdown-tkex

説明：
CLB直接接続モードのグレースフルシャットダウンをサポートしています。Endpointオブジェクト内のendpointsがnot-readyかどうかによって、not-readyのCLBバックエンドの重みを0に設定します。
使用例：
直接接続モードでのみサポートされ、`service.cloud.tencent.com/direct-access`を組み合わせて使用する必要があります。使用方法の詳細については、Serviceのグレースフルシャットダウンの関連機能をご参照ください。

service.kubernetes.io/qcloud-loadbalancer-internet-charge-type

説明：
CLBの支払いタイプです。現時点では作成時のみ設定をサポートし、作成後は支払いタイプの変更をサポートしていません。作成後、そのアノテーションの変更は無効です。
CLBを作成する時に、CLBの支払いタイプを指定します。`service.kubernetes.io/qcloud-loadbalancer-internet-max-bandwidth-out`アノテーションと組み合わせて使用してください。
選択可能な値：
`BANDWIDTH_POSTPAID_BY_HOUR` 帯域幅1時間単位で後払い
`TRAFFIC_POSTPAID_BY_HOUR` トラフィック1時間単位で後払い
使用例：
service.kubernetes.io/qcloud-loadbalancer-internet-charge-type `: "TRAFFIC_POSTPAID_BY_HOUR"`

service.kubernetes.io/qcloud-loadbalancer-internet-max-bandwidth-out

説明：
CLB帯域幅設定は、現時点では作成時のみ設定をサポートし、作成後は帯域幅の変更はサポートしていません。作成後、そのアノテーションの変更は無効です。
CLBを指定して作成する場合、CLBの最大ダウンストリーム帯域幅は、パブリックネットワーク属性のLBにのみ

有効です。 `service.kubernetes.io/qcloud-loadbalancer-internet-charge-type` アノテーションを組み合わせて使用する必要があります。

選択可能な値：

範囲は1から2048をサポート。単位はMbps。

使用例：

```
service.kubernetes.io/qcloud-loadbalancer-internet-max-bandwidth-out: "2048"
```

service.cloud.tencent.com/security-groups

説明：

このAnnotationによってCLBタイプのServiceにセキュリティグループをバインドすることができます。1つのCLBに最大5個のセキュリティグループをバインドすることができます。

注意：

CLBが使用するセキュリティグループの使用制限をご参照ください。

通常、セキュリティグループのデフォルト開放の機能を組み合わせる必要があります。CLBとCVMの間はデフォルトで開放され、CLBからのトラフィックはCLB上のセキュリティグループによってのみ検証されます。対応するAnnotationは次のとおりです。 `service.cloud.tencent.com/pass-to-target`

Serviceに既存のCLBを使用するシナリオについて、複数のServiceが異なるセキュリティグループを宣言する場合、ロジックが競合するという問題が発生する可能性があります。

使用例：

```
service.cloud.tencent.com/security-groups: "sg-xxxxxx,sg-xxxxxx"
```

service.cloud.tencent.com/pass-to-target

説明：

このAnnotationによってCLBタイプServiceのセキュリティグループのデフォルト開放機能を設定できます。CLBとCVMの間はデフォルトで開放され、CLBからのトラフィックはCLBを通過することによってセキュリティグループの検証をする必要があります。

注意：

CLBが使用するセキュリティグループの使用制限をご参照ください。

通常はセキュリティグループをバインドする機能と組み合わせる必要があります。対応するAnnotationは次のとおりです。 `service.cloud.tencent.com/security-groups`

Serviceに既存のCLBを使用するシナリオについて、複数のServiceが異なる開放設定を宣言する場合、ロジックが競合するという問題が発生する可能性があります。

使用例：

```
service.cloud.tencent.com/pass-to-target: "true"
```

Ingressの管理
Ingress Controllersの説明

最終更新日：：2023-05-06 19:41:07

各タイプのIngress Controllersの紹介

アプリケーション型CLB

アプリケーション型CLBはCLBに基づいて実現されるTKE Ingress Controllerです。異なるURLがクラスター内の異なるServiceにアクセスすることを実現するように設定することができます。CLBは直接トラフィックをNodePortによってPod（CLBがPodに直接接続されている場合は直接Podに転送）に転送し、1つのIngress設定が1つのCLBインスタンス（IP）にバインドされ、簡単なルーティング管理のみで、IPアドレス集約にセンシティブではないシナリオに適しています。詳細については、CLBタイプのIngressをご参照ください。

Istio Ingress Gateway

CLBおよびIstio Ingress Gateway（Tencent Cloud Mesh（TCM）によって提供）に基づくIngress Controllerで、コントロールプレーンと関連するサポートコンポーネントはTencent Cloudによって運用保守されます。クラスター内はトラフィック転送を実行するデータプレーンのみをコンテナ化してデプロイする必要があり、ネイティブKubernetes Ingressまたはより精密なトラフィック管理能力を提供するIstio APIを使用することができます。CLBの後にプロキシ（envoy）を追加し、アクセス層のルーティング管理に対するより多くのニーズに適し、IPアドレス集約のニーズ、クラスター、異種デプロイサービスに跨がるポータルのトラフィック管理のニーズというシナリオがあります。

専用型API Gateway

専用型API GatewayはTencent Cloud API Gateway専用インスタンスに基づいて実現されるTKE Ingress Controllerで、複数のTKEクラスターがあり、アクセス層を統一する必要があるシナリオ、およびアクセス層に認証、トラフィック制御のニーズがあるシナリオに適用されます。詳細については、API Gatewayタイプ Ingressをご参照ください。API Gateway Ingressは主に次のような優位性があります。
API GatewayはTKEクラスターのPodに直接接続され、中間ノードはありません。
1つのAPI GatewayのTKEチャネルは同時に複数のTKEサービスに接続でき、複数のサービス間は重み付けラウンドロビンアルゴリズムを採用してトラフィックを割り当てます。
認証、トラフィックの制御、カナリアスケール配信、キャッシュ、ヒューズのダウングレードなどのAPI Gatewayが提供する高度な機能拡張をサポートします。
API Gateway専用インスタンスのサポートを採用し、下層物理リソースはユーザーによって占有され、パフォーマンスが安定し、SLAが高くなります。

Nginx Ingress Controller

Nginx Ingress ControllerはCLBおよびNginxリバースプロキシ（コンテナ化してクラスター内にデプロイ）に基づくIngress Controllerです。Annotationsによってネイティブ Kubernetes Ingressを拡張する機能です。CLBの後にプロキシ（nginx）を追加し、アクセス層のルーティング管理に対するより多くのニーズ、およびIPアドレス集約のニーズがあるシナリオに適します。詳細については、NginxタイプのIngressをご参照ください。

各タイプのIngress Controllers機能の比較

モジュール	機能	アプリケーション型CLB	Istio Ingress Gateway（TCMによって提供）	専用型API Gateway	Nginx Ingress Controller
トラフィック管理	プロトコルをサポート	http,https	http,https,http2,grpc,tcp,tcp + tls	http,https,http2,grpc	http,https,http2,grpc,tcp
	IP管理	1つのIngressルールが1つのIP（CLB）に対応します	複数のIngressルールが1つのIP（CLB）に対応し、IPアドレスが集約されます	複数のIngressルールが1つのIP（専用型API Gateway）に対応し、IPアドレスが集約されます	複数のIngressルールがのIP（CLB）に対応し、アドレスが集約されま
	特徴のルーティング	host,URL	その他の特徴をサポートしています：header、method、query parameterなど	その他の特徴をサポートしています：header、method、query parameterなど	その他の特徴をサポートしています：header、coo など
	トラフィック行動	サポートしません。	リダイレクト、書き換えなどをサポートしています	リダイレクト、カスタムリクエスト、カスタム応答をサポートしています	リダイレクト、書き換 どをサポートしていま
	リージョン感知CLB	サポートしません。	サポート	サポートしません。	サポートしません。

アプリケーションアクセスのアドレッシング	サービスディスカバリ	単一のKubernetesクラスター	複数のKubernetesクラスター + 異種サービス	複数のKubernetesクラスター	単一のKubernetesクラスター
セキュリティ	SSL設定	サポート	サポート	サポート	サポート
	認証の権限承認	サポートしません。	サポート	サポート	サポート
監視可能性	監視指標	サポートしています（CLB内で確認する必要があります）	サポートしています（クラウドネイティブ監視、Tencent Cloud監視可能プラットフォーム）	サポート（API Gateway内で確認する必要があります）	サポート（クラウドネイティブ監視）
	呼び出しの追跡	サポートしません。	サポート	サポートしません。	サポートしません。
	コンポーネントの運用保守	関連するCLBがホスティングされ、クラスター内でTKE Ingress Controllerのみを実行する必要があります	コントロールプレーンがホスティングされ、クラスター内でデータプレーンのIngress Gatewayを実行する必要があります	Kubernetesクラスター内で管理プレーンを実行する必要がなく、クラスター内でネットワークアクセス機能を有効化するだけで完了します	クラスター内でNginx Ingress Controller（コントロールプレーン + データプレーン）を実行する必要があります

CLBタイプのIngress
Ingress基本機能

最終更新日：：2023-05-06 19:41:07

概要

Ingressはクラスター内のServiceのルールの集合にアクセスすることを可能にします。転送ルールを設定することによって、さまざまなURLがクラスター内の異なるServiceにアクセスすることを実現します。
Ingressリソースを正常に動作させるために、クラスターはIngress-controllerを実行する必要があります。TKEサービスはクラスター内ではデフォルトでTencent Cloud Load Balancerに基づいて実現される `l7-lb-controller` を有効化し、HTTP、HTTPSをサポートします。同時にクラスター内でその他のIngressコントローラを自作することもサポートし、業務上のニーズに応じてさまざまなIngressタイプを選択することができます。

注意事項

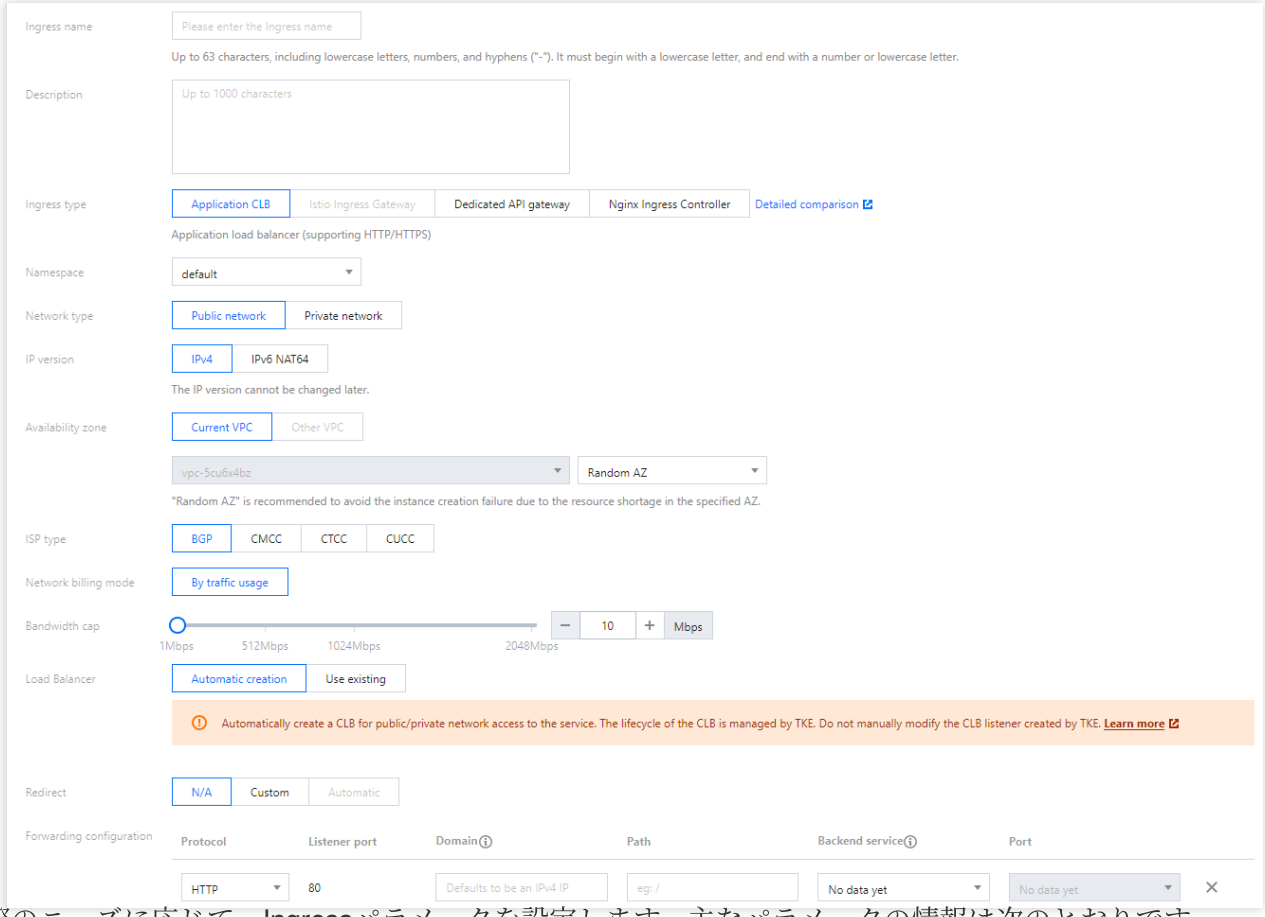
Tencent Cloud Load Balancerインスタンスは2023年3月6日にアーキテクチャがアップグレードされます。アップグレード後、パブリックネットワークCLBはドメイン名の方式でサービスを提供します。VIPは業務リクエストに応じて動的に変化し、コンソールはVIPアドレスを表示しなくなります。ドメイン名型パブリックネットワークCLBリリースのお知らせをご参照ください。
新規登録したTencent Cloudユーザーはデフォルトでアップグレード後のドメイン名型CLBを使用します。
既存のユーザーは元のCLBを継続して使用することを選択でき、アップグレードの影響を受けません。CLBサービスをアップグレードする必要がある場合は、同時にTencent Cloud製品のCLBおよびTKEをアップグレードする必要があります。アップグレードしない場合、TKE内のすべてのパブリックネットワークタイプの Service/Ingressの同期は影響を受ける可能性があります。CLBのアップグレード操作の詳細については、ドメイン名型CLBアップグレードガイドをご参照ください。TKEがService/Ingressコンポーネントのバージョンをアップグレードするには、チケットを提出してお問い合わせください。
Ingress apiVersionのサポート状況：extensions/v1beta1およびnetworking.k8s.io/v1beta1 APIバージョンのIngressは、v1.22バージョンでは継続して提供しません。networking.k8s.io/v1 APIは、v1.19（TKEシナリオでは偶数バージョンのみサポートします。そのためTKEのv1.20からとなります）バージョンから使用開始できます。詳細情報については Kubernetesドキュメントをご参照ください。
コンテナ業務がCVM業務と1つのCLBを共用しないことを確認してください。
CLBコンソールでTKEが管理するCLBのリスナー、転送パス、証明書およびバックエンドにバインドされたサーバーの操作はサポートしていません。変更はTKEによって自動的に上書きされます。
既存のCLBを使用する場合：

CLBコンソールによって作成されたロードバランサのみ使用できます。TKEによって自動的に作成されたCLBの再利用はサポートしていません。
複数のIngressによるCLBの再利用はサポートしていません。
IngressおよびServiceによるCLBをの共用はサポートしていません。
Ingressを削除すると、CLBバインドを再利用するバックエンドクラウドサーバーはご自身でバインドを解除する必要があり、同時に `tag tke-clusterId: cls-xxxx` を保持し、ご自身でクリーンアップする必要があります。
デフォルトのCLBの転送ルールの制限は50個です。Ingressの転送ルールが50を超える場合、チケットを提出することによってCLBのクォータを増やすことができます。
IngressとCLBの間に設定された管理および同期はCLB IDを名前とするLoadBalancerResourceタイプのリソースオブジェクトです。このCRDにはいかなる操作もしないでください。そうしない場合、Ingressの無効化が発生しやすくなります。

Ingressコンソール操作ガイド

Ingressの作成

1. TKEコンソールにログインします。
2. 左側ナビゲーションバーで、**クラスター**をクリックしてクラスター管理ページに進みます。
3. Ingressを作成する必要があるクラスターIDをクリックし、Ingress作成待ちのクラスター管理ページに進みます。
4. **サービス** > **Ingress**を選択し、Ingress情報ページに進みます。
5. **新規作成**をクリックし、「Ingressの新規作成」ページに進みます。下図のように表示されます。



6. 実際のニーズに応じて、Ingressパラメータを設定します。主なパラメータの情報は次のとおりです。

Ingress名称：カスタマイズします。

ネットワークタイプ：デフォルトは「パブリックネットワーク」です。実際のニーズに応じて選択します。

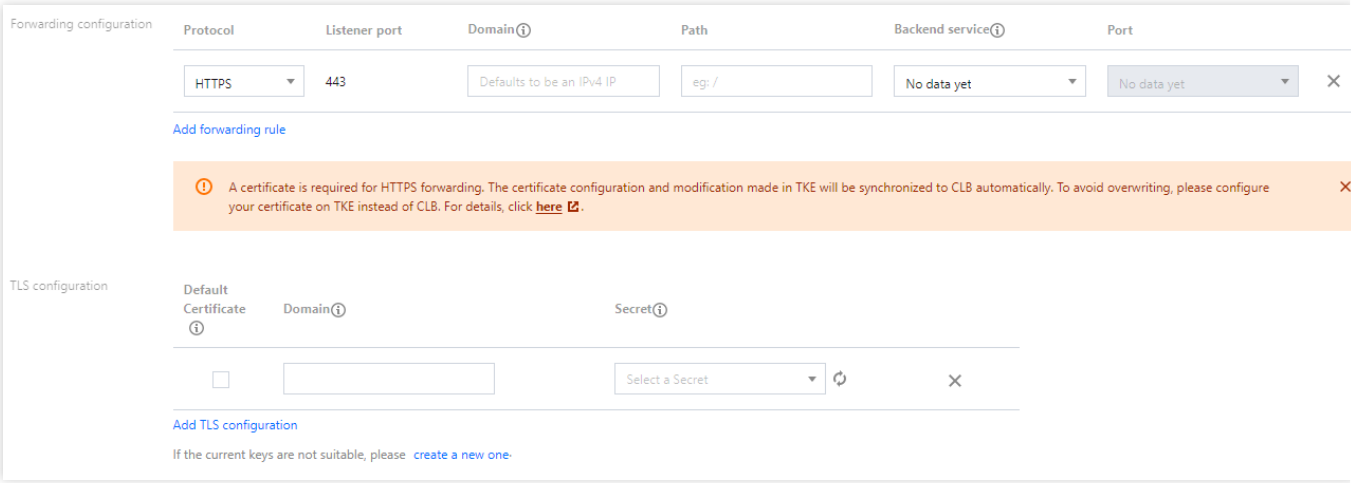
IPバージョン：IPv4およびIPv6 NAT64の2種類のバージョンを提供しています。実際のニーズに応じて選択します。

ロードバランサ：自動作成または既存のものを使用できます。

ネームスペース：実際のニーズに応じて選択します。

転送設定：「プロトコル」はデフォルトで**Http**です。実際の状況に応じて選択してください。

プロトコルに“”**Https**を選択した場合、下図に示すとおり、サーバー証明書をバインドすることによってアクセスのセキュリティを保証する必要があります。



詳細については、 SSL証明書形式の要件および形式変換の説明をご参照ください。

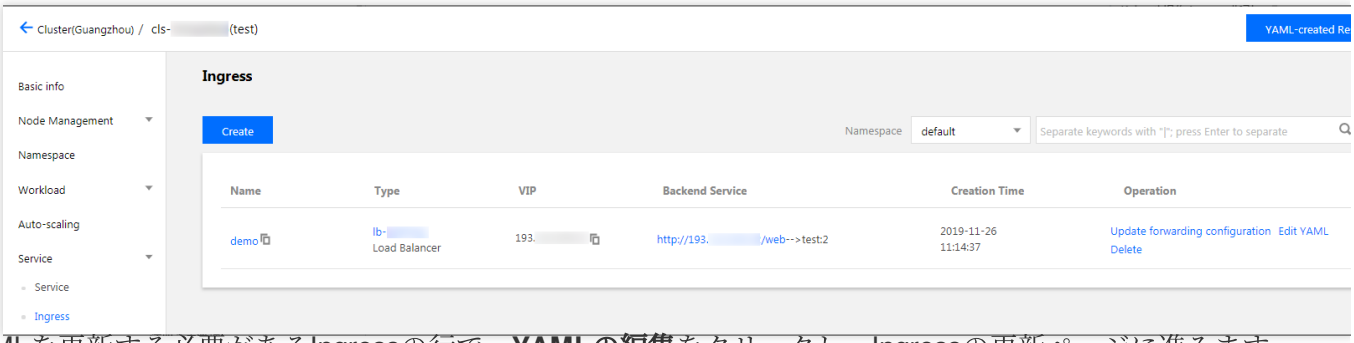
転送設定：実際のニーズに応じて設定します。

7. **Ingressの作成**をクリックすると、作成が完了します。

Ingressの更新

YAMLの更新

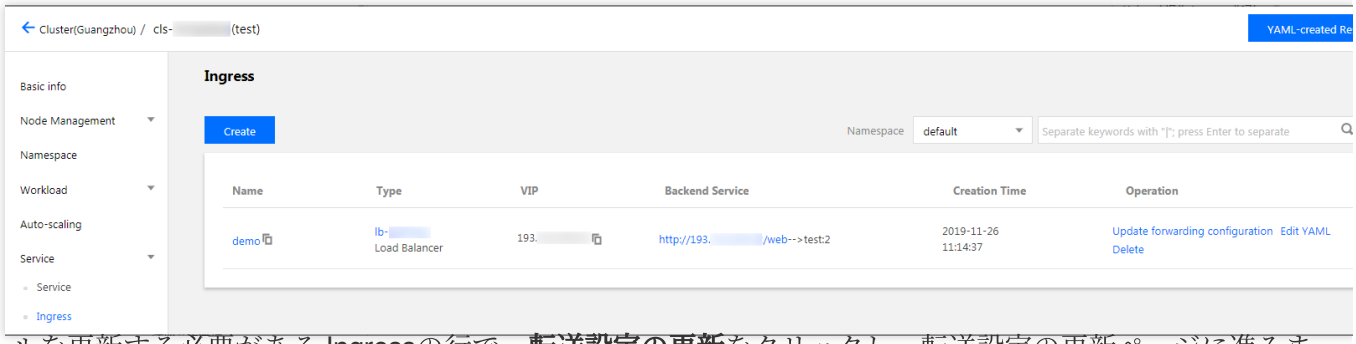
1. TKEコンソールにログインします。
2. 左側ナビゲーションバーで、**クラスター**をクリックし、クラスター管理ページに進みます。
3. YAMLを更新する必要があるクラスターIDをクリックし、更新待ちのYAMLのクラスター管理ページに進みます。
4. **サービス** > **Ingress**を選択し、Ingress情報ページに進みます。下図のように表示されます。



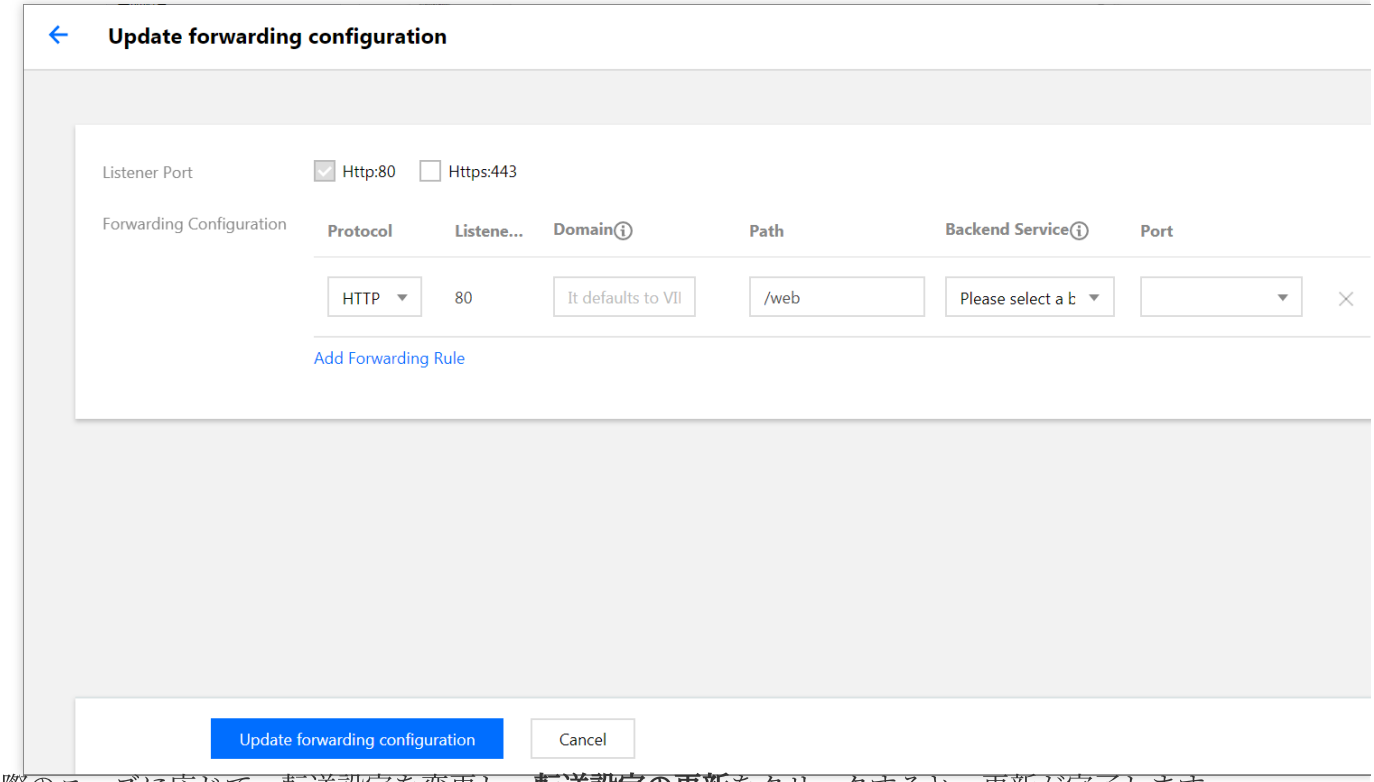
5. YAMLを更新する必要があるIngressの行で、**YAMLの編集**をクリックし、Ingressの更新ページに進みます。
6. 「Ingressの更新」ページで、YAMLを編集し、**完了**をクリックすると、YAMLの更新が完了します。

転送ルールを更新する

1. クラスター管理ページで、YAMLを更新する必要があるクラスターIDをクリックし、更新待ちのYAMLのクラスター管理ページに進みます。
2. **サービス** > **Ingress**を選択し、Ingress情報ページに進みます。下図のように表示されます。



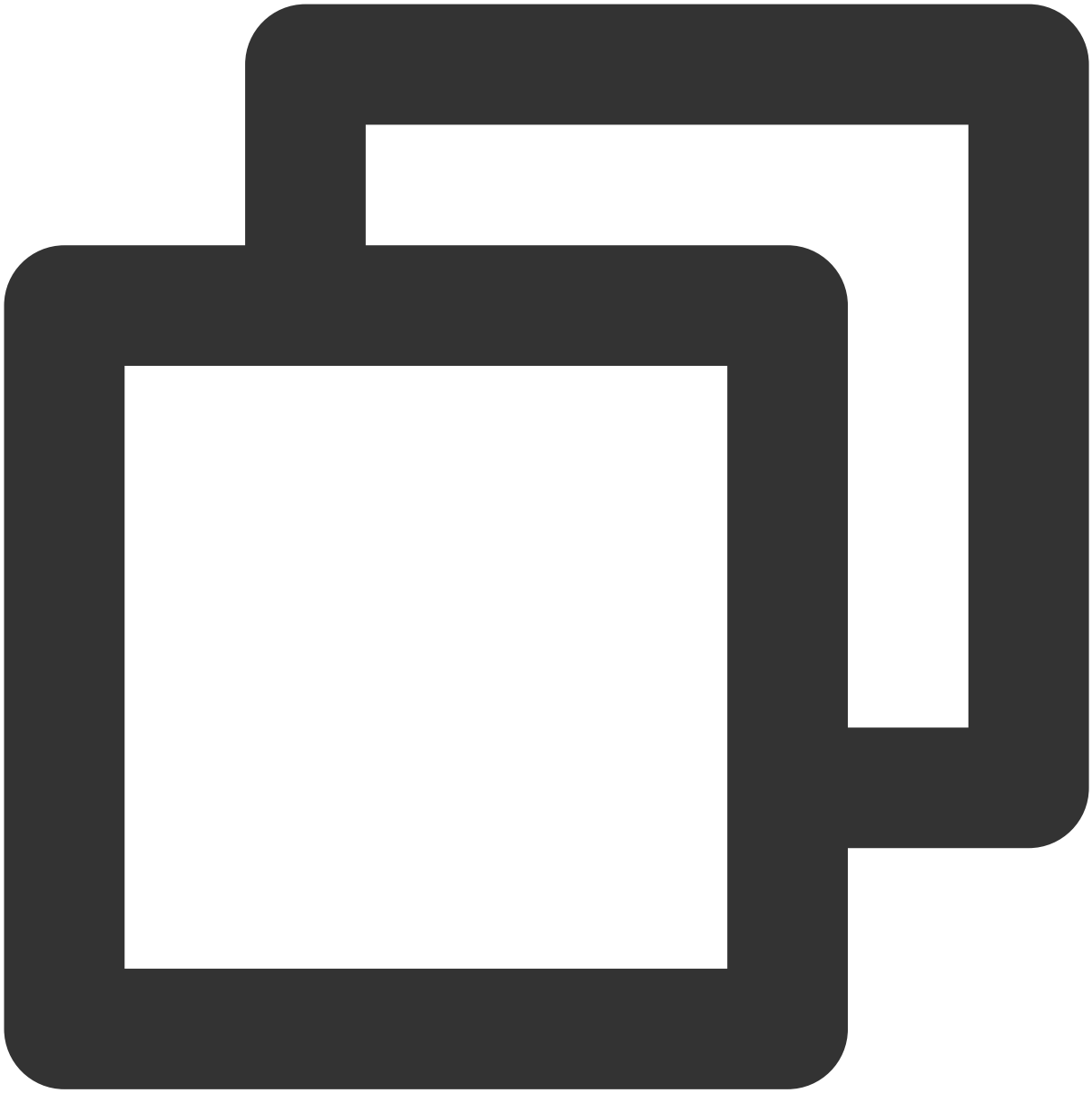
3. ルールを更新する必要がある Ingressの行で、**転送設定の更新**をクリックし、転送設定の更新ページに進みます。下図のように表示されます。



4. 実際のニーズに応じて、転送設定を変更し、**転送設定の更新**をクリックすると、更新が完了します。

KubectlによるIngress操作ガイド

YAMLの例



```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/ingress.class: qcloud ## 選択可能な値：qcloud（CLBタイプingress），ng
    ## kubernetes.io/ingress.existLbId： lb-xxxxxxxx       ##既存のロードバランサを使用し
    ## kubernetes.io/ingress.subnetId: subnet-xxxxxxxx  ##CLBタイプのプライベートネット
  name: my-ingress
  namespace: default
spec:
  rules:
```

```
  - host: localhost
    http:
      paths:
      - backend:
          serviceName: non-service
          servicePort: 65535
        path: /
```

kind：Ingressを識別するリソースタイプ。

metadata：Ingressの名称、Labelなどの基本情報です。

metadata.annotations：Ingressの追加説明。このパラメータによってTencent Cloud TKEの追加の拡張機能を設定できます。

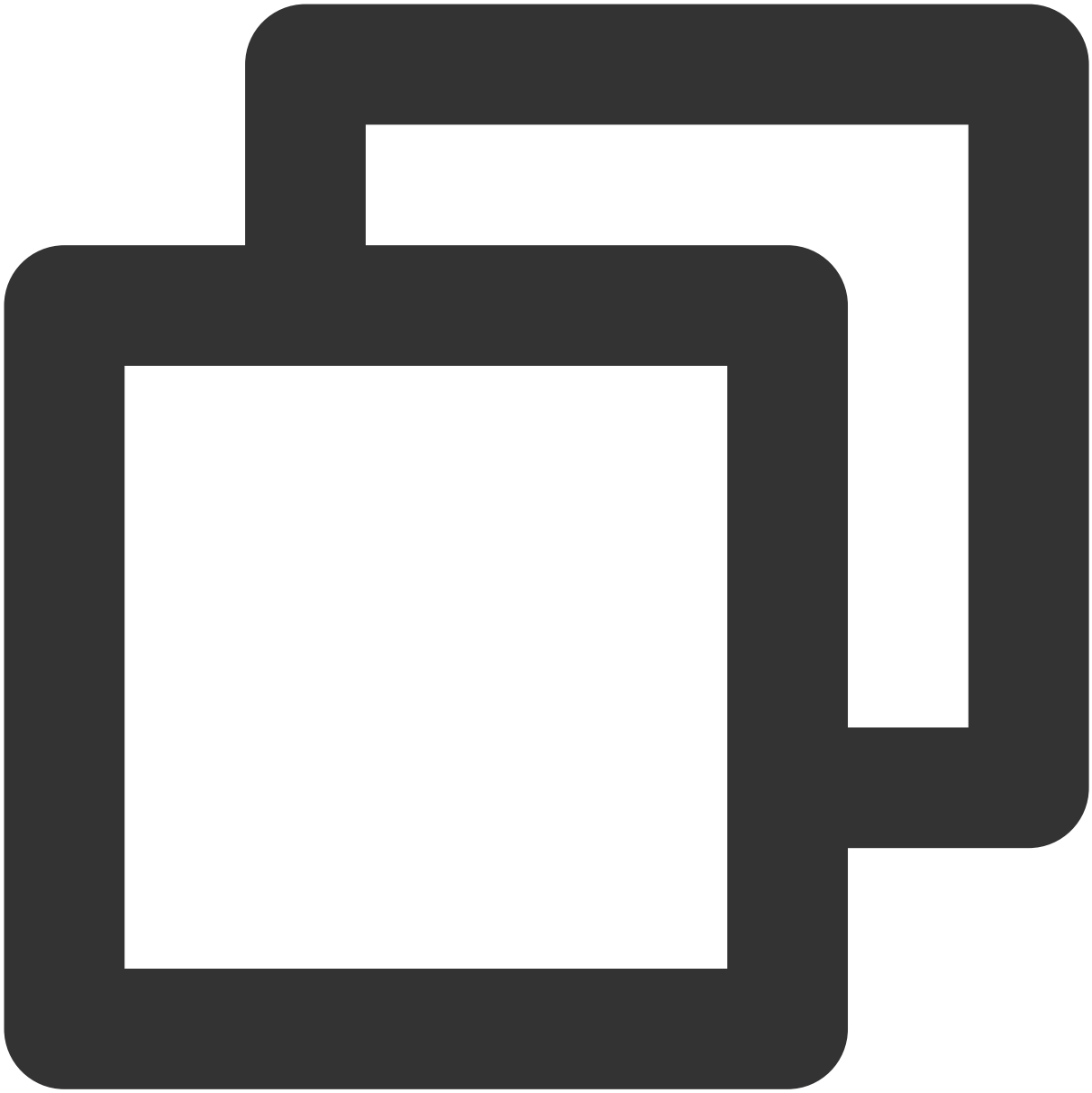
spec.rules：Ingressの転送ルール。このルールを設定するとシンプルルーティングサービス、ドメイン名に基づくシンプルファンアウトルーティング、シンプルルーティングのデフォルトドメイン名、安全なルーティングサービスの設定などを実現することができます。

annotations: 既存のロードバランサを使用してパブリックネットワーク/プライベートネットワークアクセスを作成するIngressを指定する

既存のアプリケーション型CLBがアイドル状態で、TKEに作成されたngressを提供して使用するか、クラスター内で同じCLBを使用することが想定される場合、以下のannotationsによって設定します。

説明

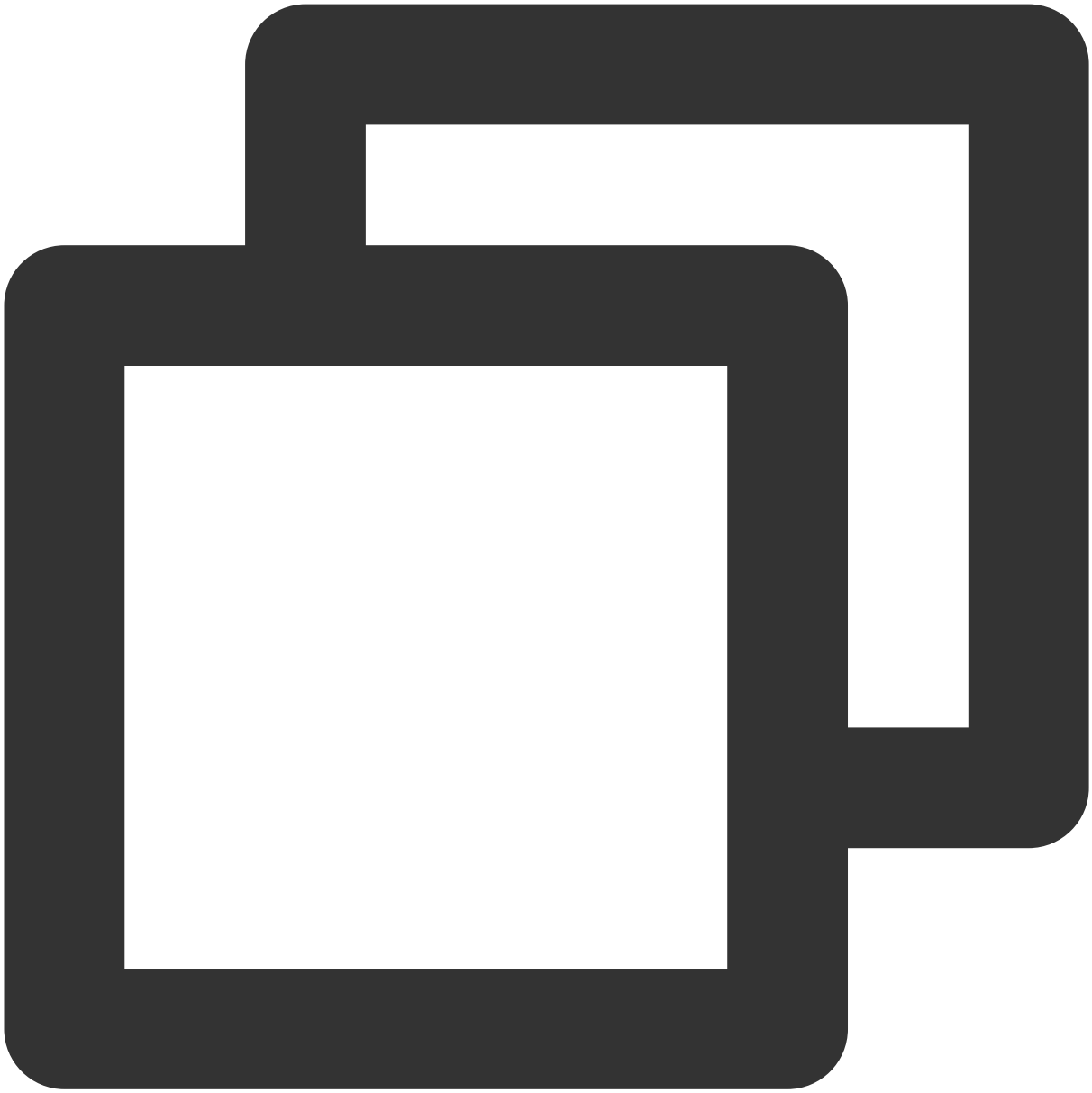
注意事項を理解してから使用を開始してください。



```
metadata:
  annotations:
    kubernetes.io/ingress.existLbId： lb-6swtxxxx
```

annotations:CLBタイプのプライベートネットワークIngressを作成する

プライベートネットワークCLBを使用する必要がある場合、以下によってannotationsを設定します。



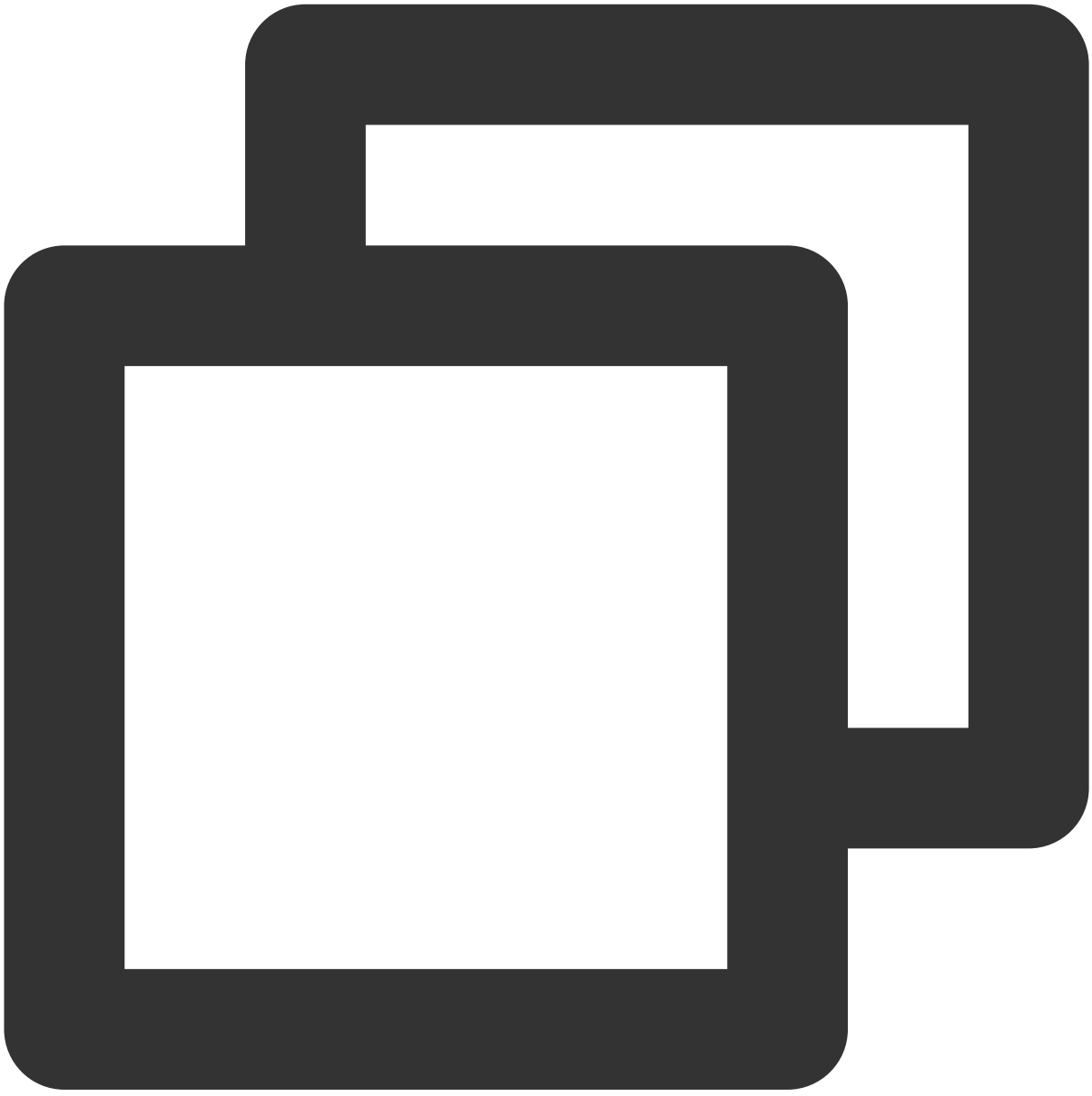
```
metadata:
  annotations:
    kubernetes.io/ingress.subnetId: subnet-xxxxxxxx
```

説明事項

使用するのが**IP帯域幅パッケージ**アカウントの場合、パブリックネットワークアクセス方式のサービスを作成する時に、以下の2つのannotations項目を指定する必要があります。

`kubernetes.io/ingress.internetChargeType` パブリックネットワークの帯域幅課金方式。選択可能な値は以下のとおりです。

TRAFFIC_POSTPAID_BY_HOUR（使用トラフィックに応じて課金）
BANDWIDTH_POSTPAID_BY_HOUR（帯域幅課金）
`kubernetes.io/ingress.internetMaxBandwidthOut` 帯域幅上限です。範囲：[1,2000] Mbps。
例：

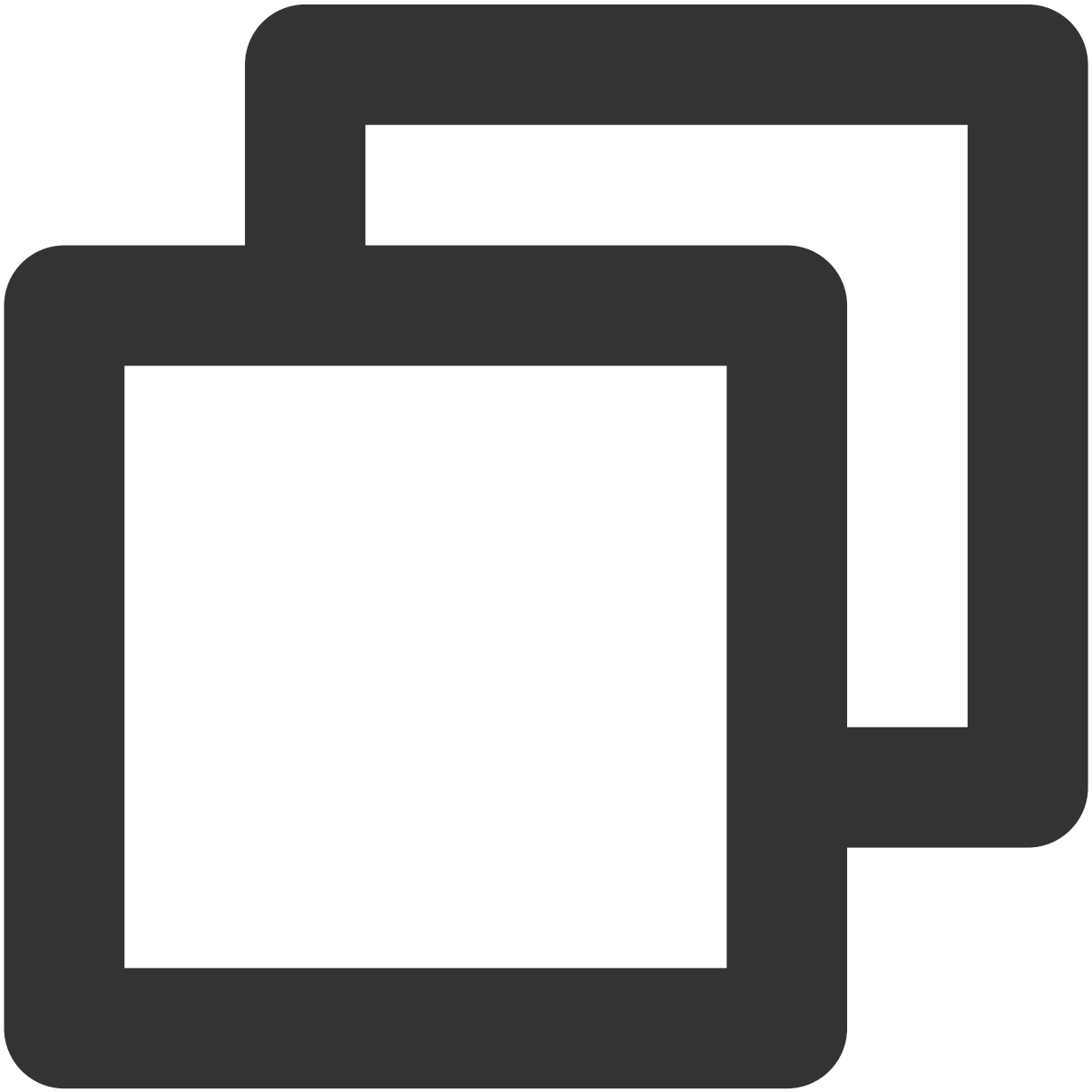


```
metadata:
annotations:
  kubernetes.io/ingress.internetChargeType: TRAFFIC_POSTPAID_BY_HOUR
  kubernetes.io/ingress.internetMaxBandwidthOut: "10"
```

IP帯域幅パッケージの詳しい情報については、 ドキュメントBandwidth Package製品のカテゴリーを、ぜひご参照ください。

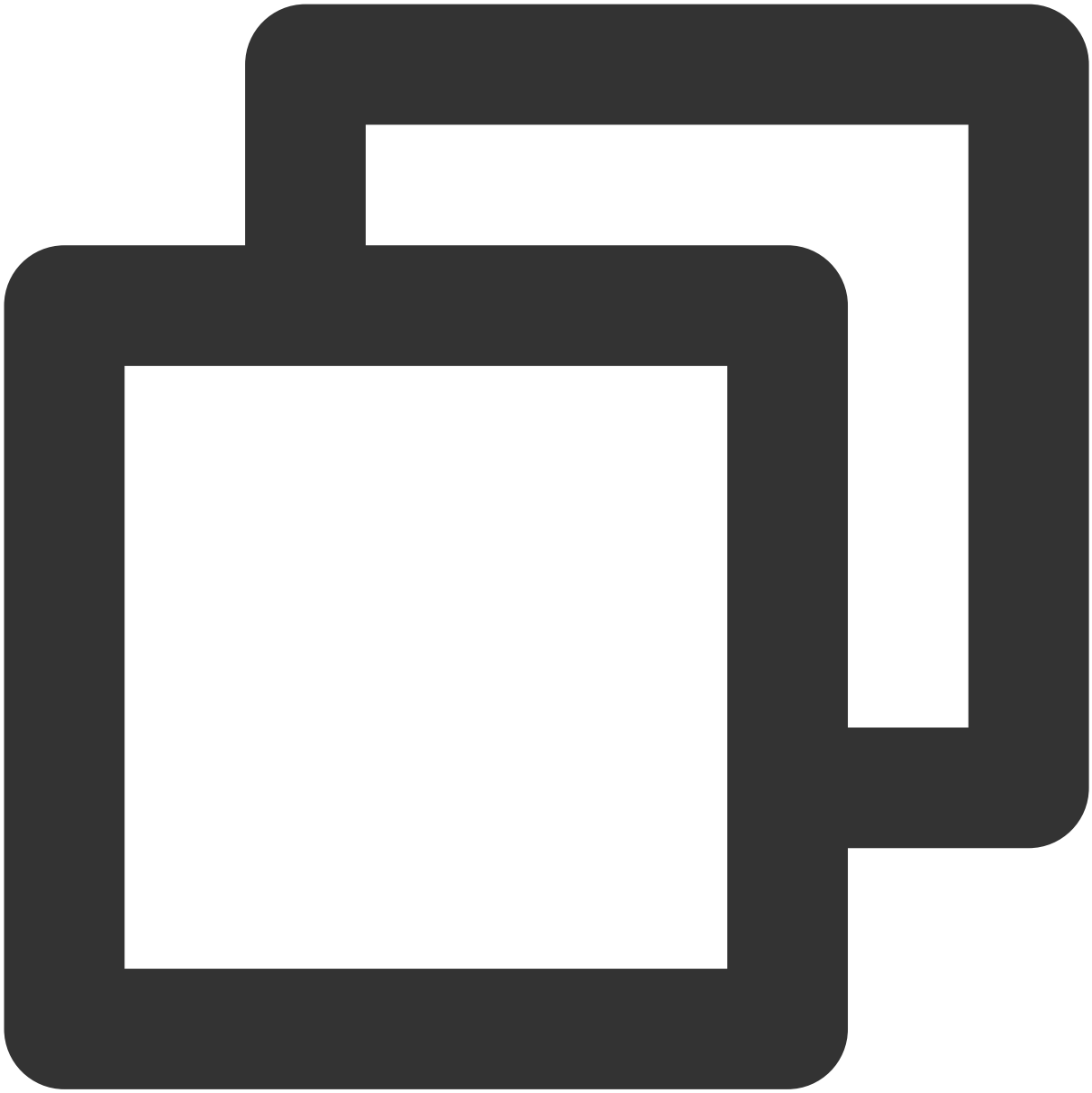
Ingressの作成

1. YAMLの例を参考に、 Ingress YAMLファイルを準備します。
2. Kubectlをインストールし、クラスターを接続します。操作の詳細については、Kubectlによってクラスターに接続するをご参照ください。
3. 以下のコマンドを実行し、Ingress YAMLファイルを作成します。



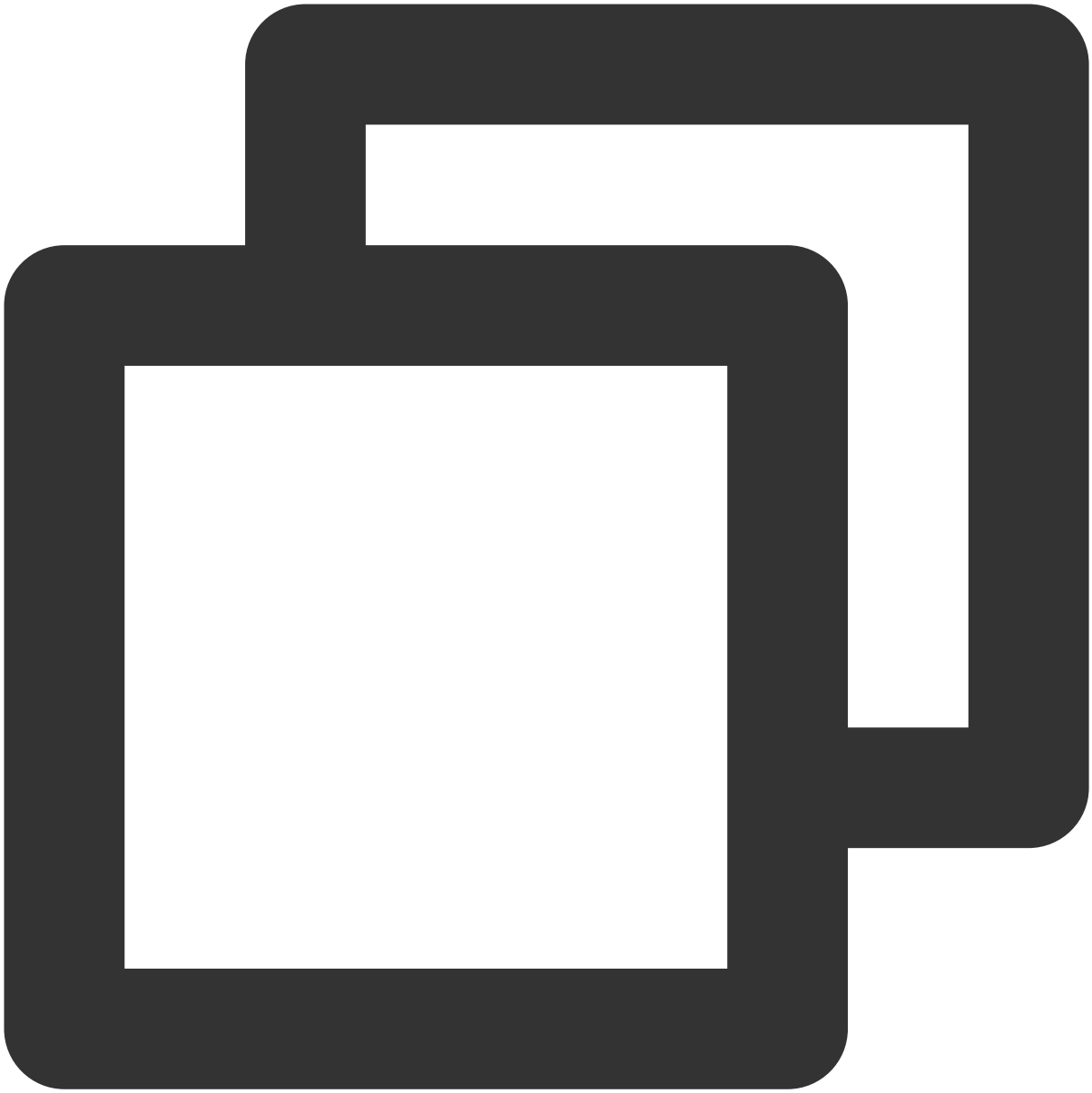
```
kubectl create -f Ingress YAMLファイル名称
```

例えば、ファイル名がmy-ingress.yamlのIngress YAMLファイルを作成するには、以下のコマンドを実行します。



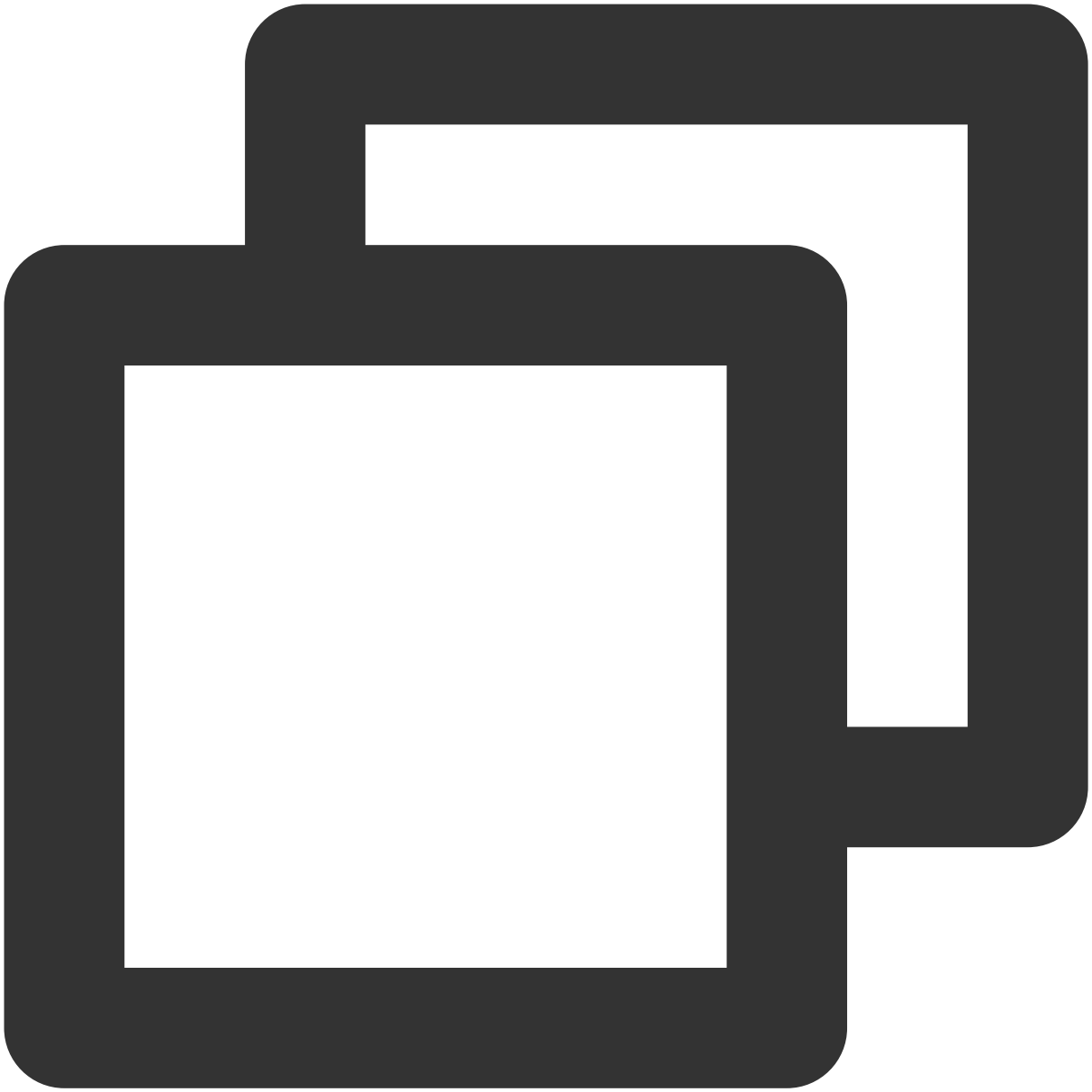
```
kubectl create -f my-ingress.yaml
```

4. 次のコマンドを実行して、変更が成功したかどうかをチェックします。



```
kubectl get ingress
```

下記のような情報が返された場合は、作成が成功したことを示しています。

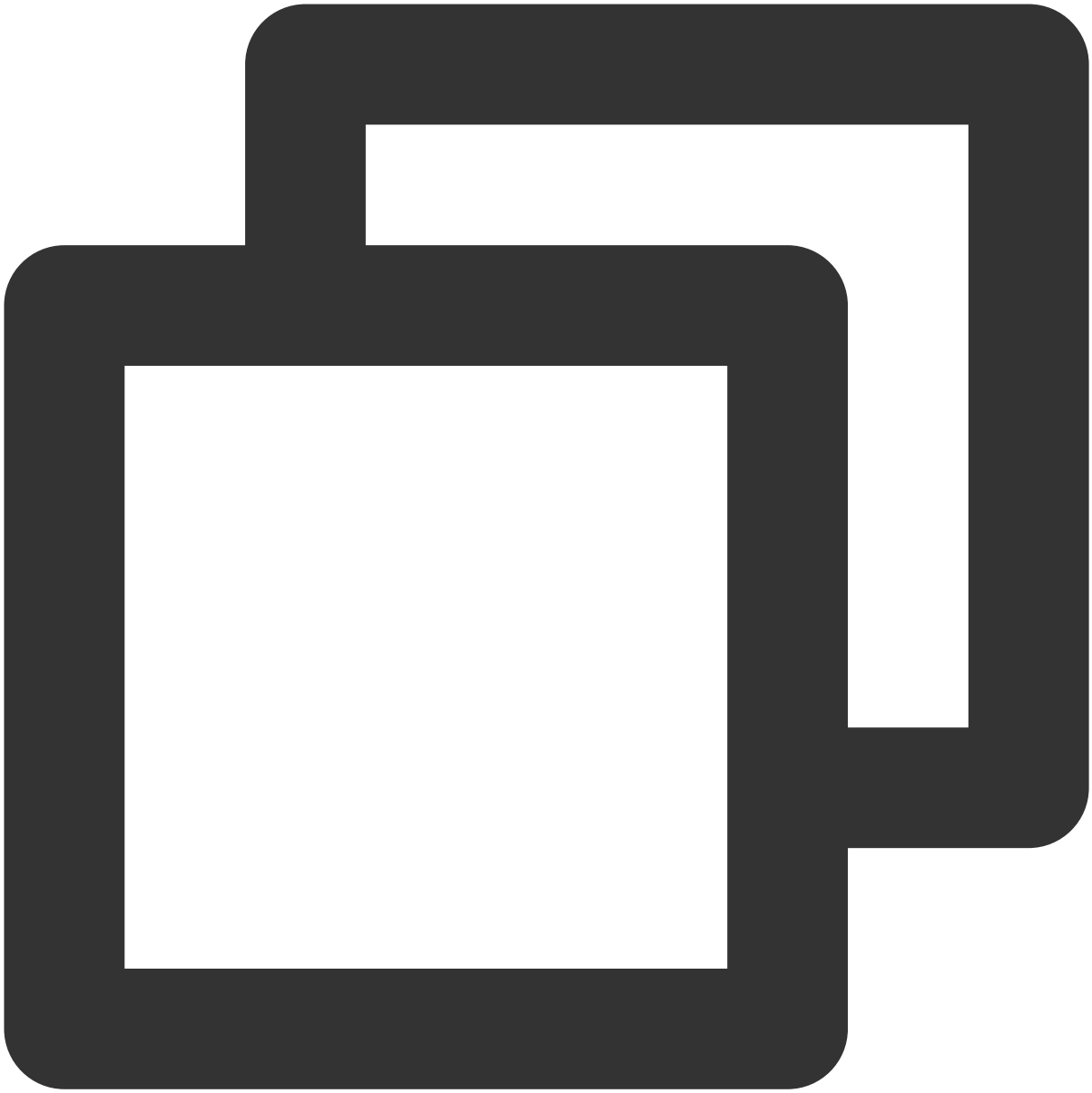


```
NAME           HOSTS       ADDRESS   PORTS     AGE
clb-ingress    localhost             80        21s
```

Ingressの更新

方法1

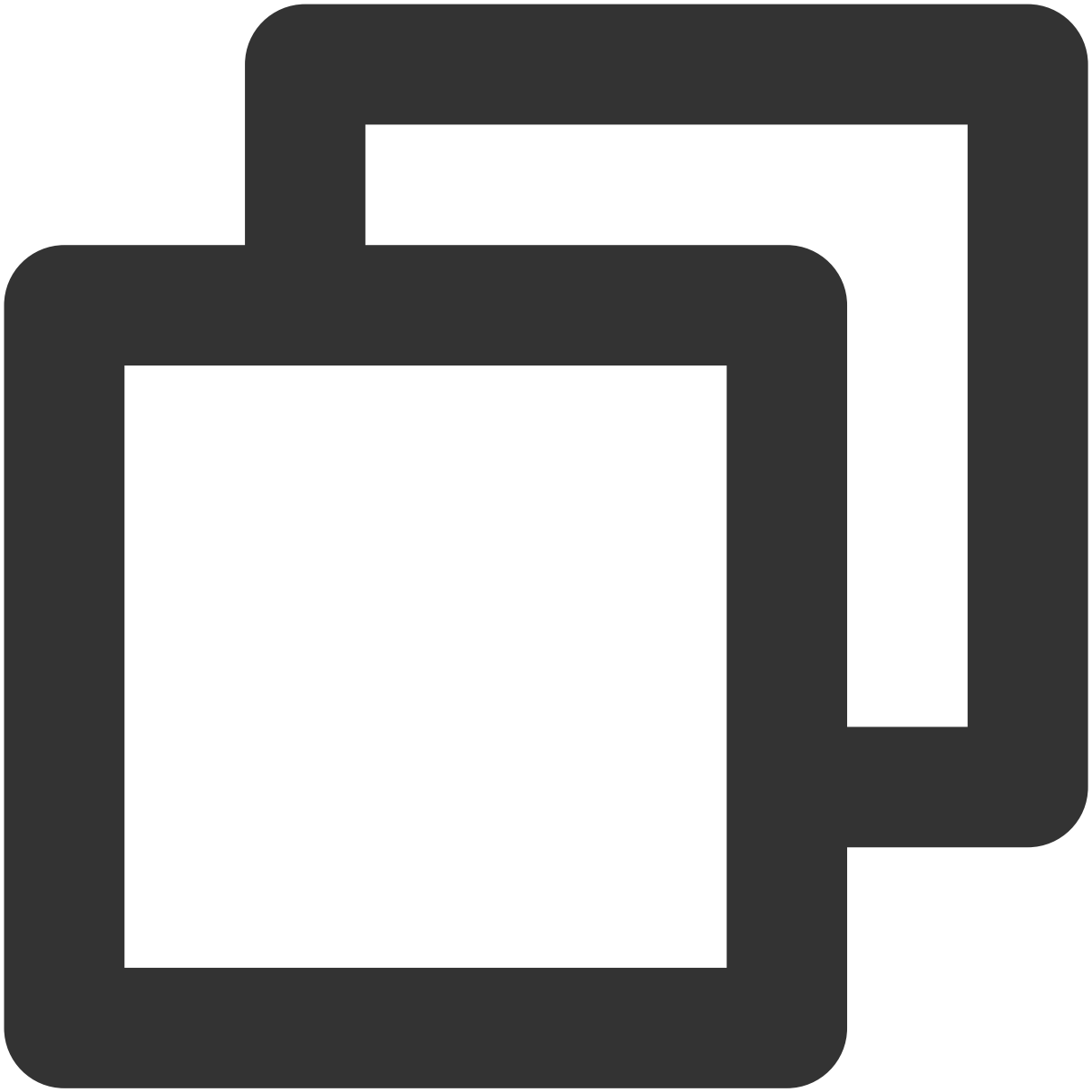
次のコマンドを実行して、Ingressを更新します。



```
kubectl edit  ingress/[name]
```

方法2

1. 手動で古いIngressを削除します。
2. 以下のコマンドを実行し、Ingressを再度作成します。



```
kubectl create/apply
```

Ingress証明書設定

最終更新日：：2023-04-26 17:02:26

ユースケース

ここではIngress証明書の使用に関する内容をご紹介します。次のシナリオでIngress証明書の設定を行うことができます。
Ingress作成の際にHTTPSリスニングプロトコルを選択した場合、適切なサーバー証明書を選択することでアクセスのセキュリティを保証できます。
すべてのHTTPSドメイン名に同一の証明書をバインドすることで、Ingress下のすべてのHTTPSルールの証明書の設定が簡略化され、更新操作がよりスピーディーになります。
異なるドメイン名に異なる証明書をバインドすることで、サーバーとクライアントのSSL/TLSが改善します。

注意事項

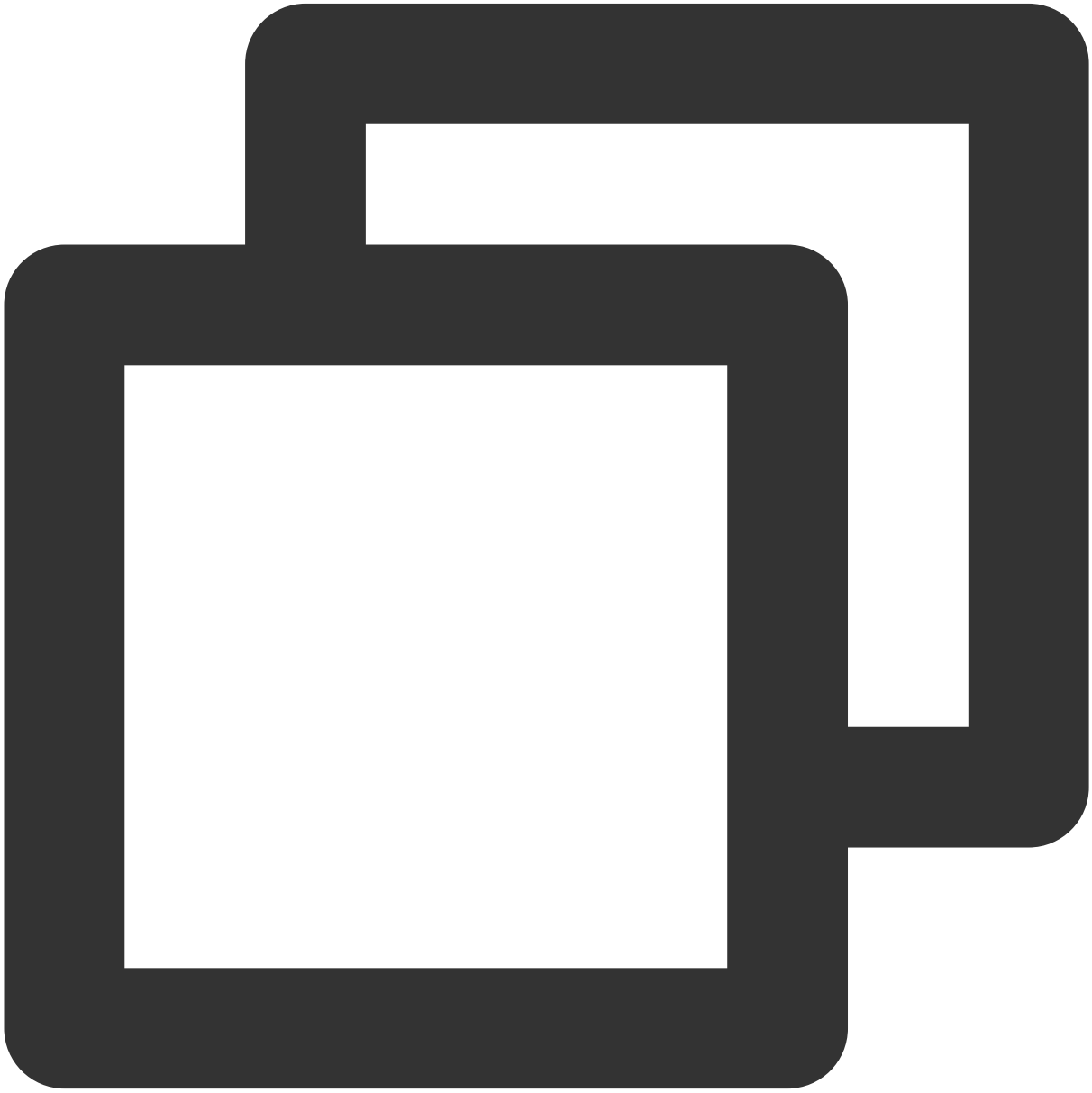
設定したい証明書を事前に作成しておく必要があります。詳細についてはコンソールでのサーバー証明書の新規作成をご参照ください。
Ingress証明書の設定はSecret形式で行う必要があります。Tencent Kubernetes Engine（TKE） Ingressはデフォルトで同名のSecretを作成し、その中に証明書IDが含まれています。
証明書を置き換えたい場合は、証明書プラットフォームで証明書を新規作成した後、Secretの証明書IDを更新することをお勧めします。クラスター内のコンポーネントの同期はSecretのステートメントに準じるため、他の証明書サービス、CLBサービス上で証明書を直接更新した場合、Secret内の内容に復元されます。
Secret証明書のリソースはIngressリソースと同一のNamespace下に配置する必要があります。
コンソールはデフォルトで同名のSecret証明書リソースを作成するため、同名のSecretリソースがすでに存在する場合はIngressを作成できません。
通常、Ingress作成の際には、Secretにバインドされた証明書リソースは再利用されません。ただし、Ingress作成時にSecretにバインドされた証明書リソースを再利用することはサポートされており、Secretの更新の際に、このSecretを参照するすべてのIngress証明書が同期更新されます。
ドメイン名に一致する証明書を追加すると、Cloud Load Balancer（CLB）のSNI機能が同期的に有効化されます（無効にすることはできません）。証明書に対応するドメイン名を削除すると、この証明書はIngressに対応するHTTPSドメイン名にデフォルトで一致します。
従来型CLBではドメイン名およびURLベースの転送はサポートしていません。従来型CLBで作成したIngressは複数の証明書の設定をサポートしていません。

事例

TKEではIngress内の `spec.tls` フィールドによって、Ingress用に作成したCLB HTTPSリスナーに証明書を設定することができます。このうち、secretNameはTencent Cloud証明書IDを含むKubernetes Secretリソースです。次に例を示します。

Ingress

YAMLで作成します。

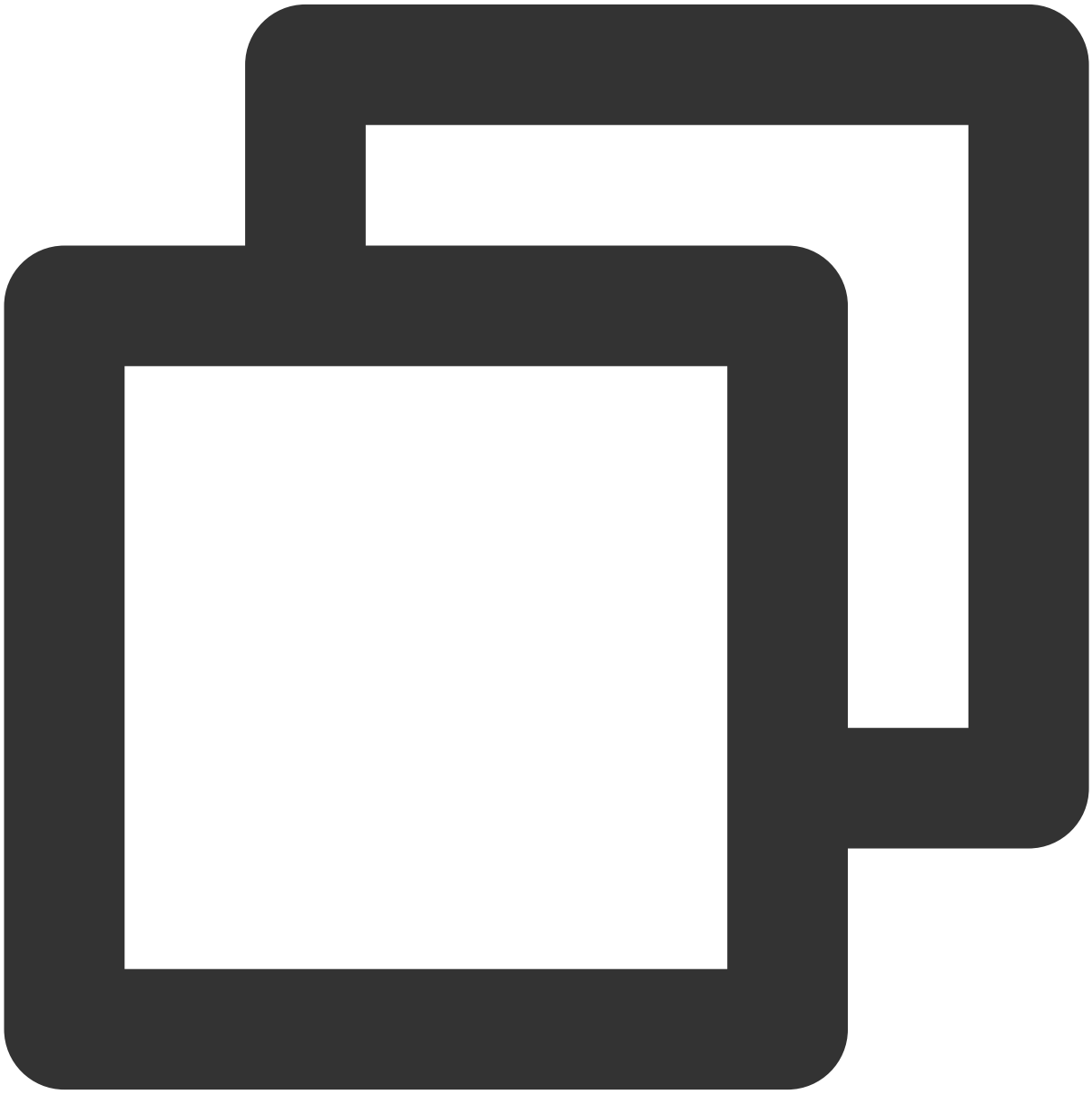


```
spec:
```

```
    tls:
    - hosts:
      - www.abc.com
      secretName: secret-tls-2
```

Secret

YAMLで作成します。
TKEコンソールで作成できます。



```
apiVersion: v1
```

```
stringData:
    qcloud_cert_id: Xxxxxxxx ## 証明書IDをXxxxxxxxに設定
    qcloud_ca_cert_id: Xxxxxxxx ## 証明書IDをXxxxxxxxに設定します。双方向証明書を設定する場
kind: Secret
metadata:
    name: tencent-com-cert
    namespace: default
type: Opaque
```

TKEコンソールで作成できます。操作の詳細についてはSecretの作成をご参照ください。

「Secretの新規作成」ページで、Secretの主要パラメータを次のように設定します。

名前：ご自身で定義します。ここではcos-secretとします。

Secretタイプ：**Opaque**を選択します。このタイプはキー証明書およびプロファイルの保存に適しています。ValueはBase64形式でエンコードされます。

効力の範囲：必要に応じて選択します。必ずIngressと同一のNamespace下にします。

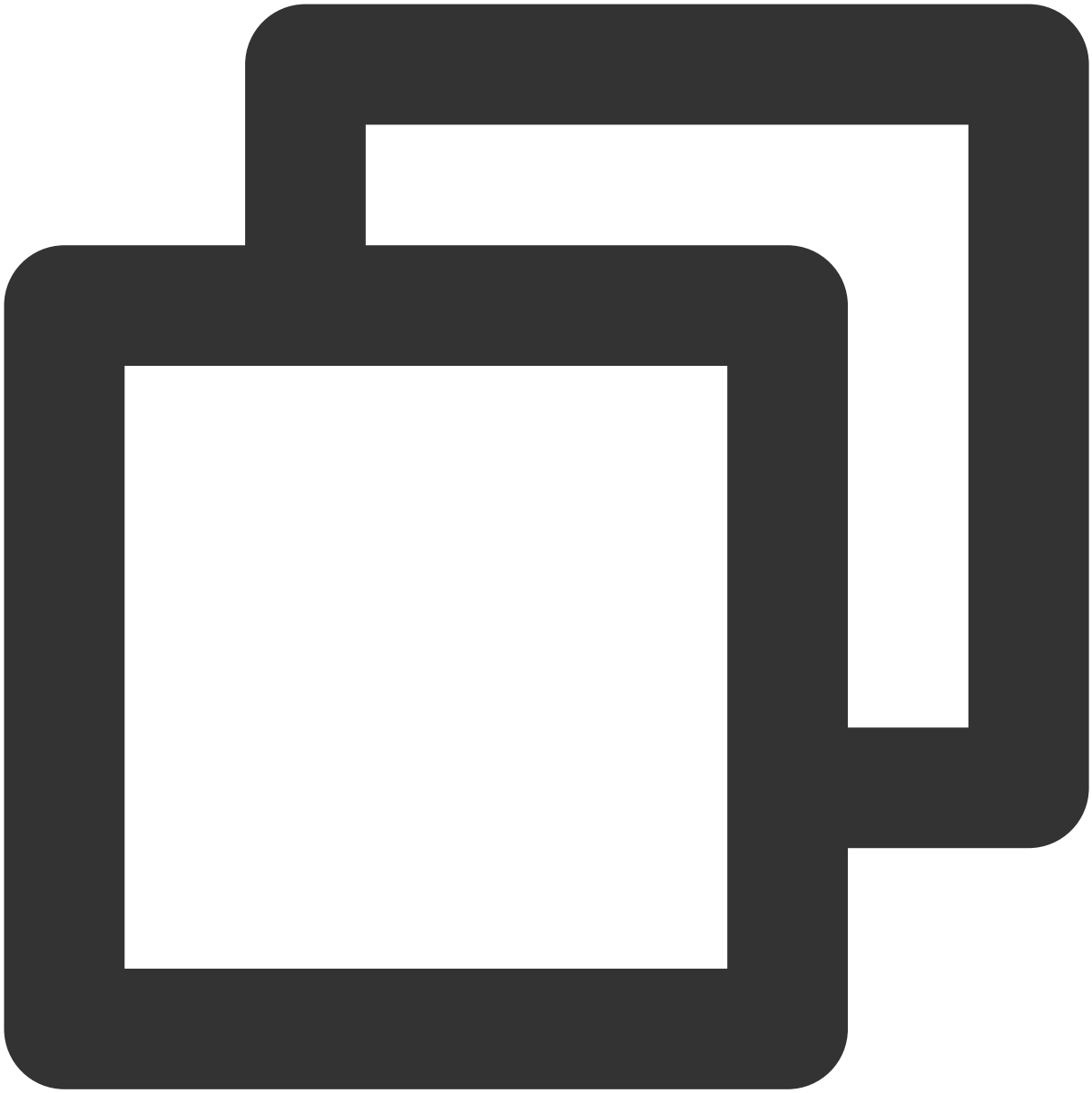
内容：変数名は `qcloud_cert_id` と設定し、変数値にはサーバー証明書の対応する証明書IDを設定します。

注意

双方向証明書を設定したい場合、Secretには「サーバー証明書」のほかに「クライアントCA証明書」も追加する必要があります。この場合、このSecretにはさらにキーバリューペアも1つ追加する必要があり、その変数名は `qcloud_ca_cert_id` とし、変数値には「クライアントCA証明書」の対応する証明書IDを設定します。

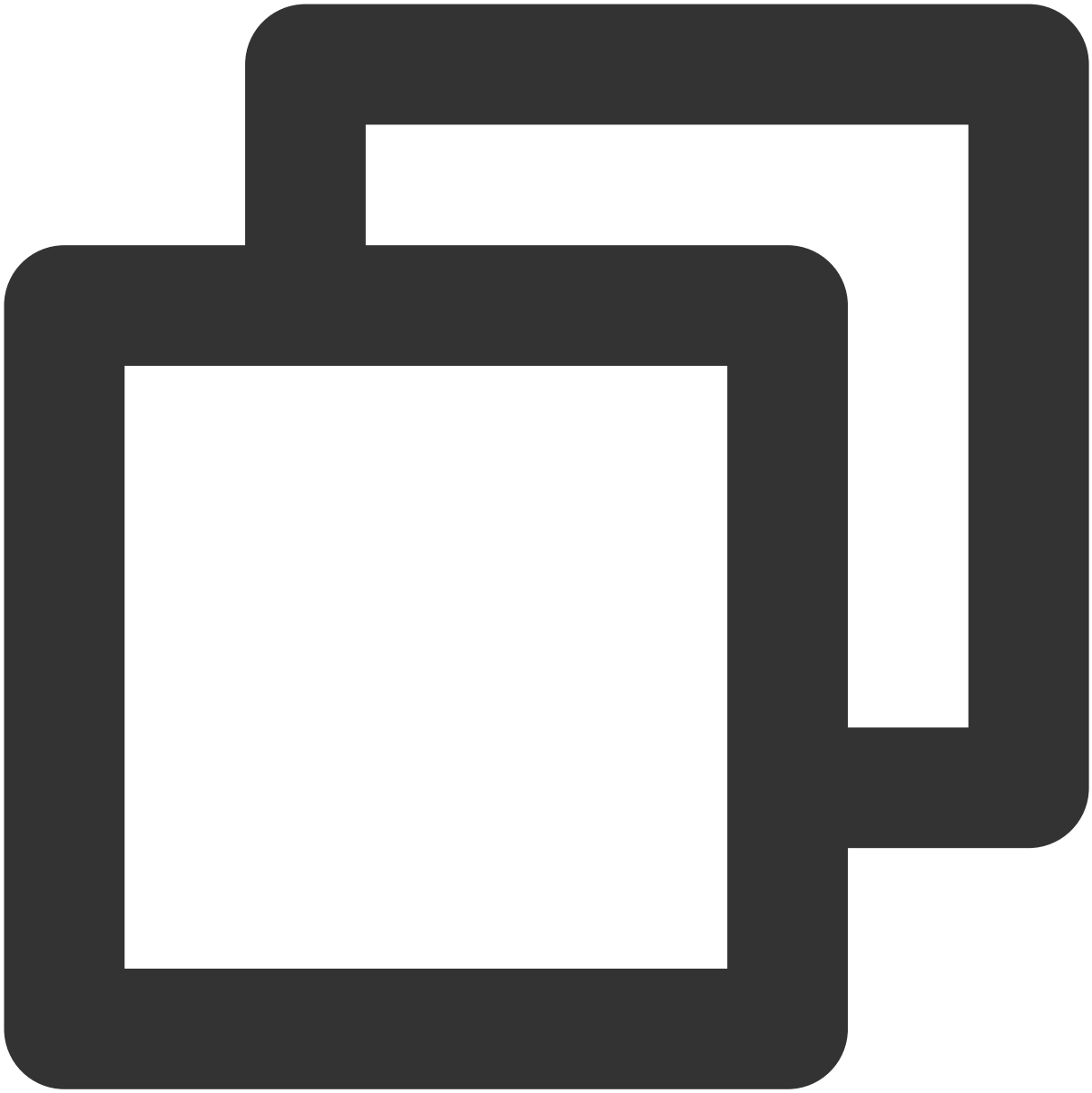
Ingress証明書設定動作

単一の `spec.secretName` のみ設定し、hostsを設定していない場合は、すべてのHTTPSの転送ルールにこの証明書が設定されます。次に例を示します。



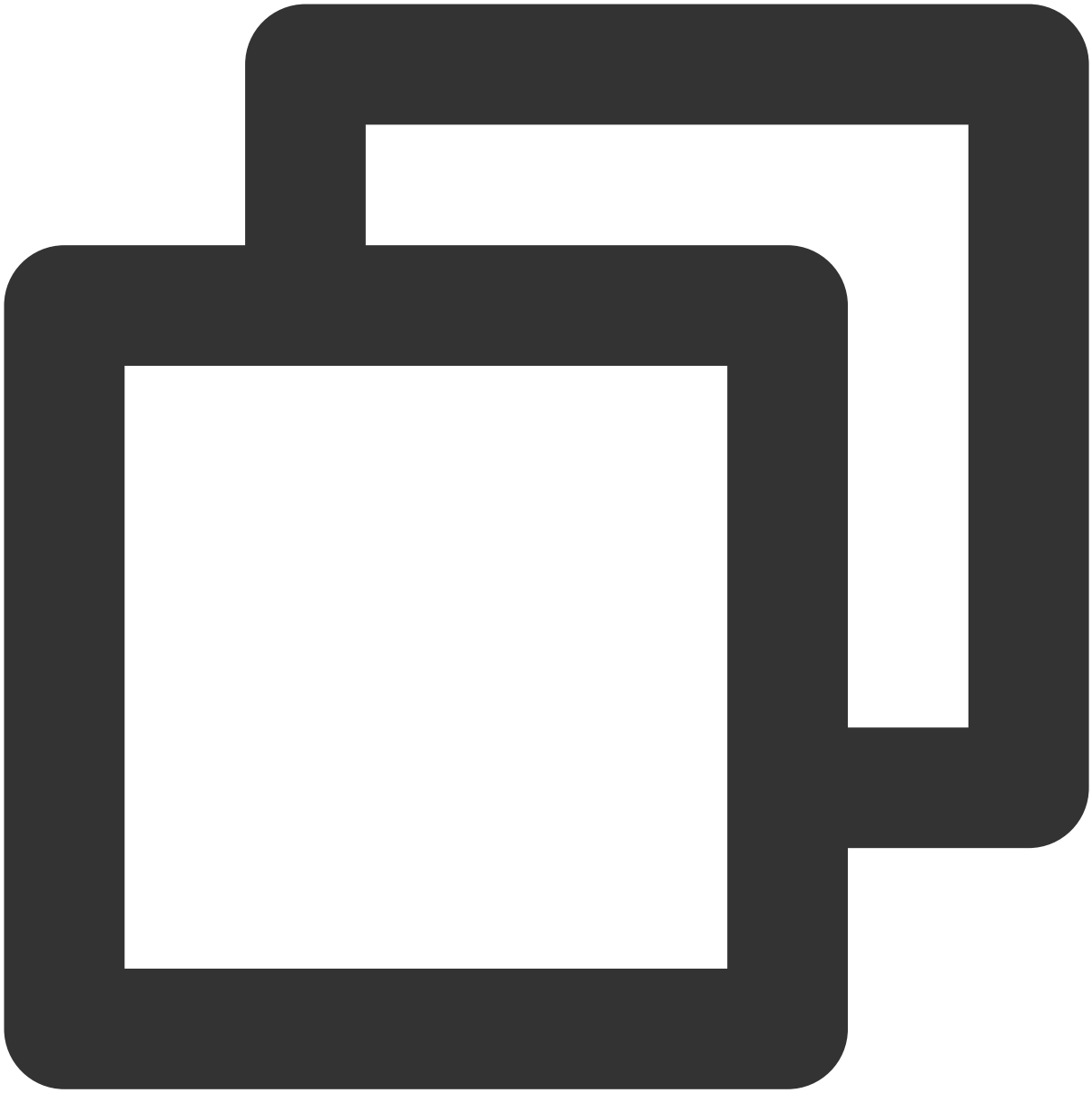
```
spec:
    tls:
    - secretName: secret-tls
```

第1レベル汎用ドメイン名の統一設定をサポートしています。次に例を示します。



```
spec:
    tls:
    - hosts:
      - *.abc.com
      secretName: secret-tls
```

証明書と汎用ドメイン名証明書が同時に設定されている場合、1つの証明書を優先的に選択します。次の例では、 `www.abc.com` は `secret-tls-2` に記述された証明書を使用します。



```
spec:
    tls:
    - hosts:
      - *.abc.com
      secretName: secret-tls-1
    - hosts:
      - www.abc.com
      secretName: secret-tls-2
```

複数の証明書をすでに使用しているIngressを更新する際、TKE Ingress controllerは以下の動作を行って判断します。
HTTPSのrules.hostに一致するものがない場合、不承認と判断されると更新は送信できません。
HTTPSのrules.hostが単一のTLSと一致する場合、更新を送信でき、かつこのhostにSecretに対応する証明書を設定することができます。
TLSのSecretNameを変更した場合はSecretNameが存在するかどうかのみを検証し、Secretの内容は検証しません。Secretが存在すれば更新を送信できます。
注意
Secret内の証明書IDが要件に適合することを確認してください。

操作手順

コンソールでのサーバー証明書の新規作成

説明
設定したい証明書をすでにお持ちの場合はこの手順をスキップしてください。
1. CLBコンソールにログインし、左側ナビゲーションバーの証明書管理を選択します。
2. 「証明書管理」ページで、**新規作成**をクリックします。
3. ポップアップした「証明書の新規作成」ウィンドウで、次の情報を参照して設定します。
証明書名：カスタム設定します。
証明書タイプ：「サーバー証明書」を選択します。**サーバー証明書**とはSSL証明書（SSL Certificates）のことです。SSL証明書に基づいてサイトを HTTP（Hypertext Transfer Protocol）から HTTPS（Hyper Text Transfer Protocol over Secure Socket Layer）に切り替えることができます。つまり、SSL（Secure Socket Layer）に基づいてセキュアなデータ伝送を行うための暗号化版 HTTPプロトコルです。
証明書内容：実際の状況に応じて証明書の内容を入力します。証明書形式の要件については、ドキュメント SSL証明書形式の要件および形式変換の説明をご参照ください。
キー内容：証明書タイプで「サーバー証明書」を選択した場合のみ、このオプションが表示されます。ドキュメント SSL証明書形式の要件および形式変換の説明をご参照の上、関連のキー内容を追加してください。
4. **送信**をクリックすると作成が完了します。

証明書を使用したIngressオブジェクトの作成

注意事項：

コンソールで作成したIngressでHTTPSサービスを有効化すると、証明書IDを保存するための同名のSecretリソースがまず作成されます。IngressではこのSecretが使用され、リスニングされます。
TLS設定ドメイン名と証明書の対応関係は次のとおりです。
第1レベル汎用ドメイン名の統一設定を使用できます。

ドメイン名に一致する複数の異なる証明書がある場合は、1つの証明書がランダムに選択されます。同一のドメイン名に異なる証明書を使用することは推奨しません。
作成が承認されるには、すべてのHTTPSドメイン名に証明書が設定されている必要があります。

操作手順：

Ingressの作成を参照してIngressの新規作成を行います。リスニングポートは**Https:443**にチェックを入れます。

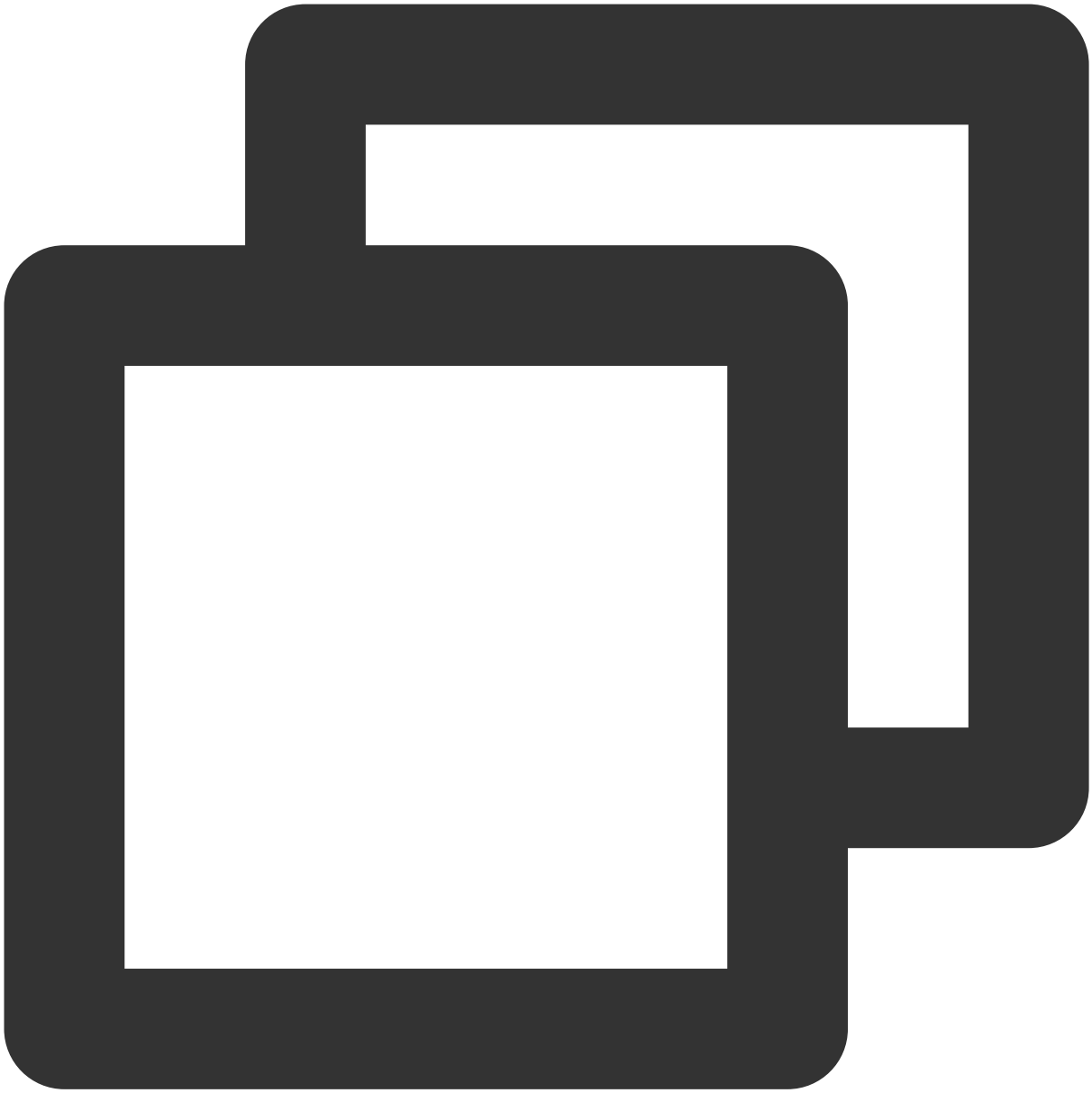
証明書の変更

注意事項：

証明書を変更したい場合、この証明書を使用しているすべてのIngressを確認してください。ユーザーの複数のIngress設定に同一のSecretリソースが使用されている場合、これらのIngressが対応するCLBの証明書が同期的に変更されます。
証明書はSecretの変更によって変更する必要があります。Secretの内容にはご使用中のTencent Cloud証明書のIDが含まれます。

操作手順：

1. 次のコマンドを実行し、デフォルトエディタを使用して変更したいSecretを開きます。`[secret-name]` は変更したいSecretの名前に置き換える必要があります。



```
kubectl edit secrets [secret-name]
```

2. Secretリソースを変更し、 `qcloud_cert_id`の値を新しい証明書`ID`に変更します。

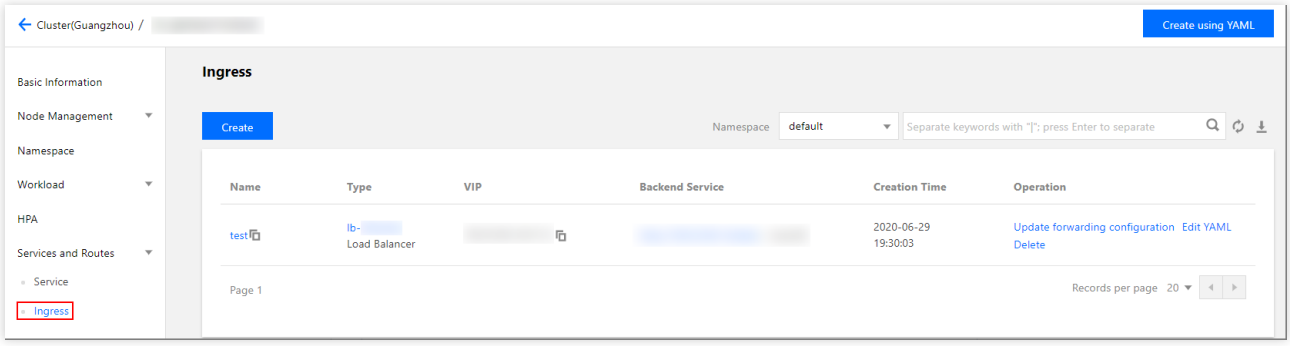
`Secret`の作成と異なり、`Secret`証明書`ID`の変更には`Base64`エンコードが必要です。実際のニーズに応じて、`Base64`手動エンコードか、 stringData`を指定して行うBase64自動エンコードのどちらかを選択します。

Ingressオブジェクトの更新

TKE コンソールで更新できます

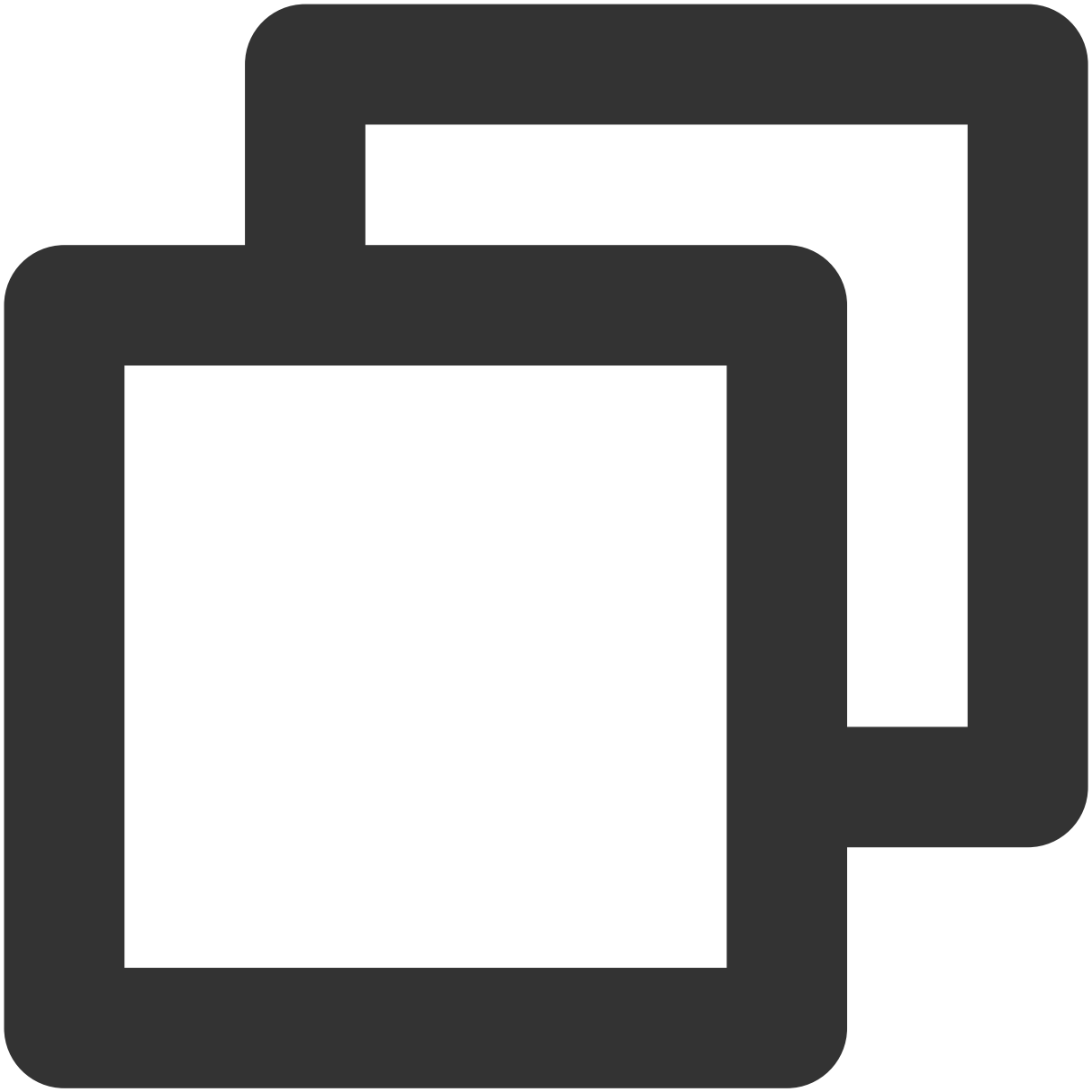
YAMLでの更新

1. TKEコンソールにログインし、左側ナビゲーションバーの**クラスター**を選択します。
2. 「クラスター管理」ページで、変更したいIngressのクラスターIDを選択します。
3. クラスター詳細ページで、左側の**サービスとルーティング** > **Ingress**を選択します。下図をご参照ください。



4. ターゲットのIngressがある行の右側の**転送設定の更新**をクリックします。
5. 「転送設定の更新」ページで、実際の状況に応じて転送設定ルールの更新を行います。
6. **転送設定の更新**をクリックすると更新操作が完了します。

次のコマンドを実行し、デフォルトエディタを使用して変更したいingressを開き、yamlファイルを変更して保存すれば更新操作は完了です。



```
kubectl edit ingress <ingressname> -n <namespaces>
```

API GatewayタイプのIngress
API GatewayのTKEチャネル設定

最終更新日：：2023-04-27 15:34:59

操作シナリオ

CLBを経由せずに、API Gatewayから直接Tencent Kubernetes Engine（TKE）クラスターのpodにアクセスできます。ここでは、コンソールを介してTKEチャネルを作成する方法について説明します。APIのバックエンドにて、バックエンドタイプをTKEチャネルに設定すると、API Gatewayからのリクエストが、直接TKEチャネルの対応するPod上に移動します。

機能のメリット

API Gatewayは直接TKEクラスターのPodに接続し、中間ノードを減らします（CLBなど）。

TKEチャネル1つで、同時に複数のTKEクラスターに接続可能です。

説明

現時点では**専用**タイプのAPI Gatewayでのみ、TKEチャネルをサポートしています。

前提条件

1. すでに専用タイプのサービスを有していること。
2. すでにTKEのクラスターがあり、クラスターadminロールを取得済みであること。

操作手順

ステップ1：TKEチャネルの作成

1. API Gatewayコンソールにログインします。
2. 左側のナビゲーションバーで**バックエンドチャネル**を選択し、**新規作成**をクリックします。
3. 新規作成したバックエンドチャネルページに以下の情報を入力します。

バックエンドチャネル名：バックエンドチャネル名を入力します

チャネルタイプ：**TKEチャネル**を選択します

Virtual Private Cloud：Virtual Private Cloud（VPC)を選択します

サービスリスト：サービスリストで複数のサービスを設定します。サービスの上限数は20個です。複数のPod間では、重み付けラウンドロビンアルゴリズムを採用しトラフィックをアサインします。個別サービスの設定ステップは以下のとおりです。

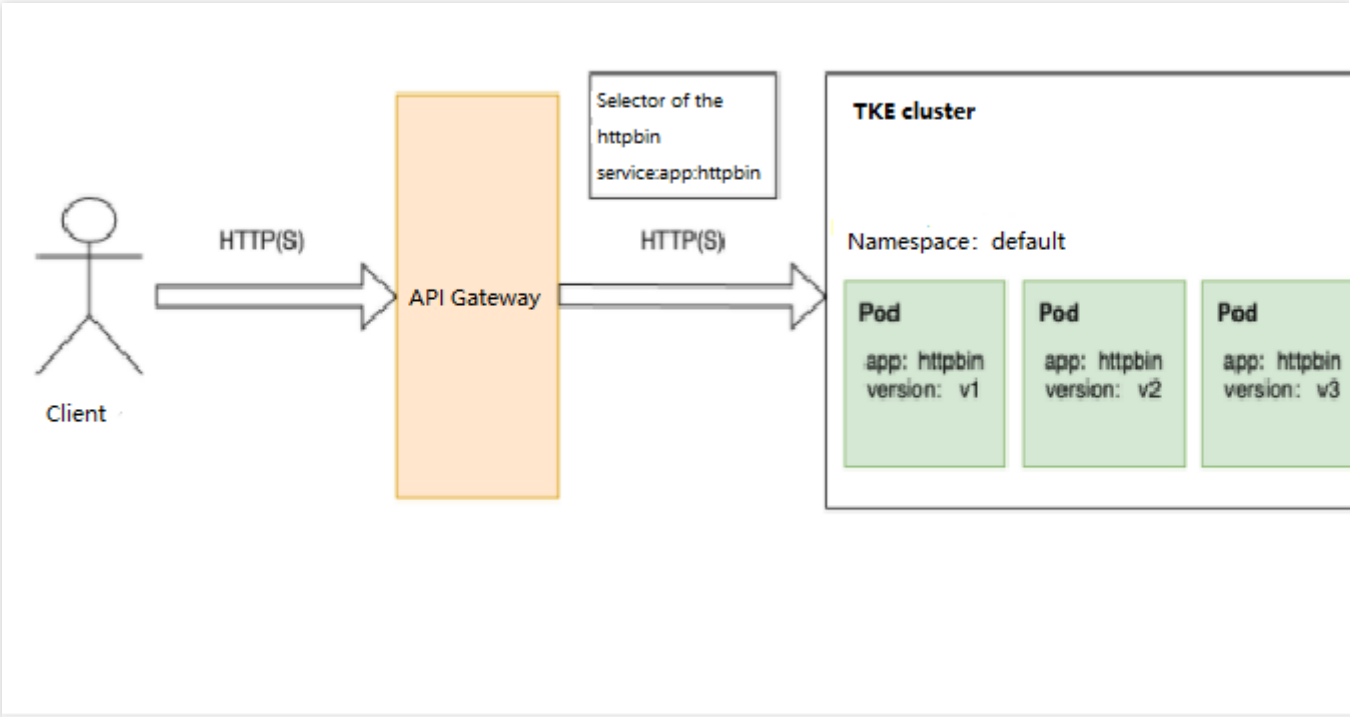
3.1.1 サービスの各Podの重み比率を入力します。
3.1.2 クラスターを選択します。クラスターにまだ権限承認がない場合は、API Gatewayが権限承認をリクエストします。
3.1.3 クラスター内のネームスペースを選択します。
3.1.4 サービスおよびサービスのポートを選択します。
3.1.5 高度なオプション：追加ノードLabelです。
バックエンドタイプ：HTTPまたはHTTPSを選択します。
Host Header：オプションです。Host Headerとは、API Gatewayがバックエンドサービスにアクセスする際、HTTP/HTTPSリクエストに含まれる、リクエストHEADERのHostの値のことです。
タグ：オプションです。タグは、さまざまなディメンションからリソースをカテゴリー管理するために用いられます。

ステップ2：APIバックエンドをTKEチャネルに接続

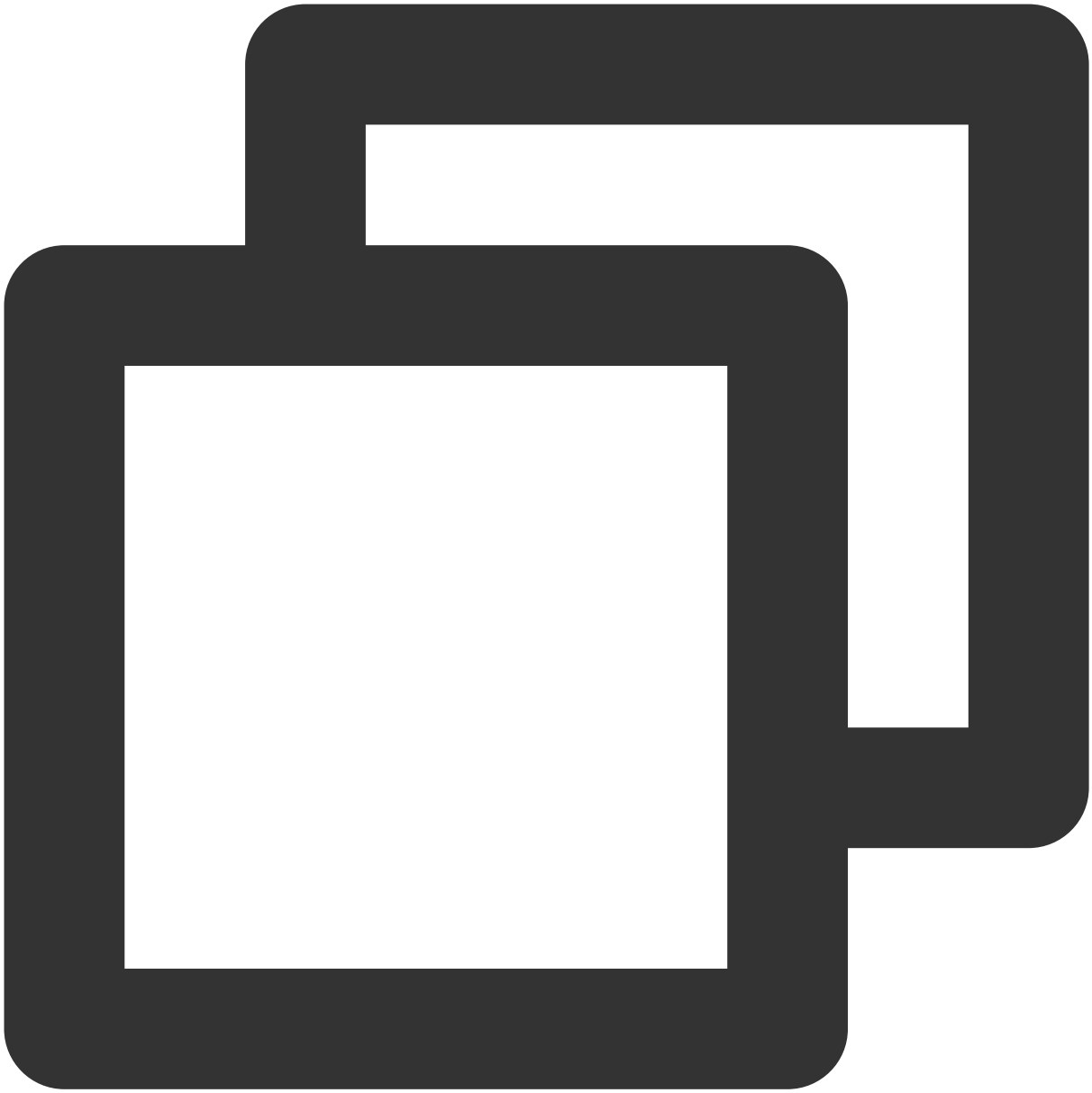
1. API Gatewayコンソールの サービスページで、目的のサービスの「ID」をクリックし、API管理ページに進みます。
2. **新規作成**をクリックし、汎用APIを作成します。
3. フロントエンド設定を入力の上、**次へ**をクリックします。
4. バックエンドタイプの選択は **VPC内リソース**で行います。バックエンドチャネルタイプは **TKEチャネル**を選択し、**次へ**をクリックします。
5. レスポンス結果を設定し、**完了**をクリックします。

ネットワークアーキテクチャ

TKEチャネルがAPIにバインドされると、ネットワーク全体のアーキテクチャは以下のようになります。



API Gatewayは、CLBを経由せず、直接TKEクラスターのPodにアクセスします。TKEクラスターにおけるhttpbinのサービス設定ファイルYAMLは、以下のとおりです。その中のselectorでは、タグキーのついたappを選択することを表しており、タグ値は、httpbinのPodをTKEチャネルのノードとしています。そのため、versionがv1/v2/v3のPodもすべて、TKEチャネルのノードとなります。



```
apiVersion: v1
kind: Service
metadata:
  name: httpbin
  labels:
    app: httpbin
    service: httpbin
spec:
  ports:
  - name: http
    port: 8000
```

```
    targetPort: 80
  selector:
    app: httpbin
```

注意事項

TKEチャネル1つが接続できるTKEサービスは、最大でも20個です。
ユーザーは、TKEクラスターのadminロールを有する必要があります。
TKEチャネルおよびAPI Gatewayの専用については、同じVPCにある場合でのみ利用可能です。現時点では、API Gatewayは直接のクロスVPCをサポートしていません。

API GatewayでのTKEクラスター権限承認取得

最終更新日：：2023-04-27 15:34:59

操作シナリオ

ここでは、API GatewayがTencent Kubernetes Engine（TKE）クラスターのAPI Serverにアクセスする権限承認の方法と、権限承認に関するご質問についてのソリューションをご紹介します。最後にYAMLファイルにより、API Gatewayが取得した権限リストについてご説明します。

前提条件

1. API Gatewayコンソールにログイン済みであること。
2. すでにTKEのクラスターがあり、クラスターadminロールを取得済みであること。

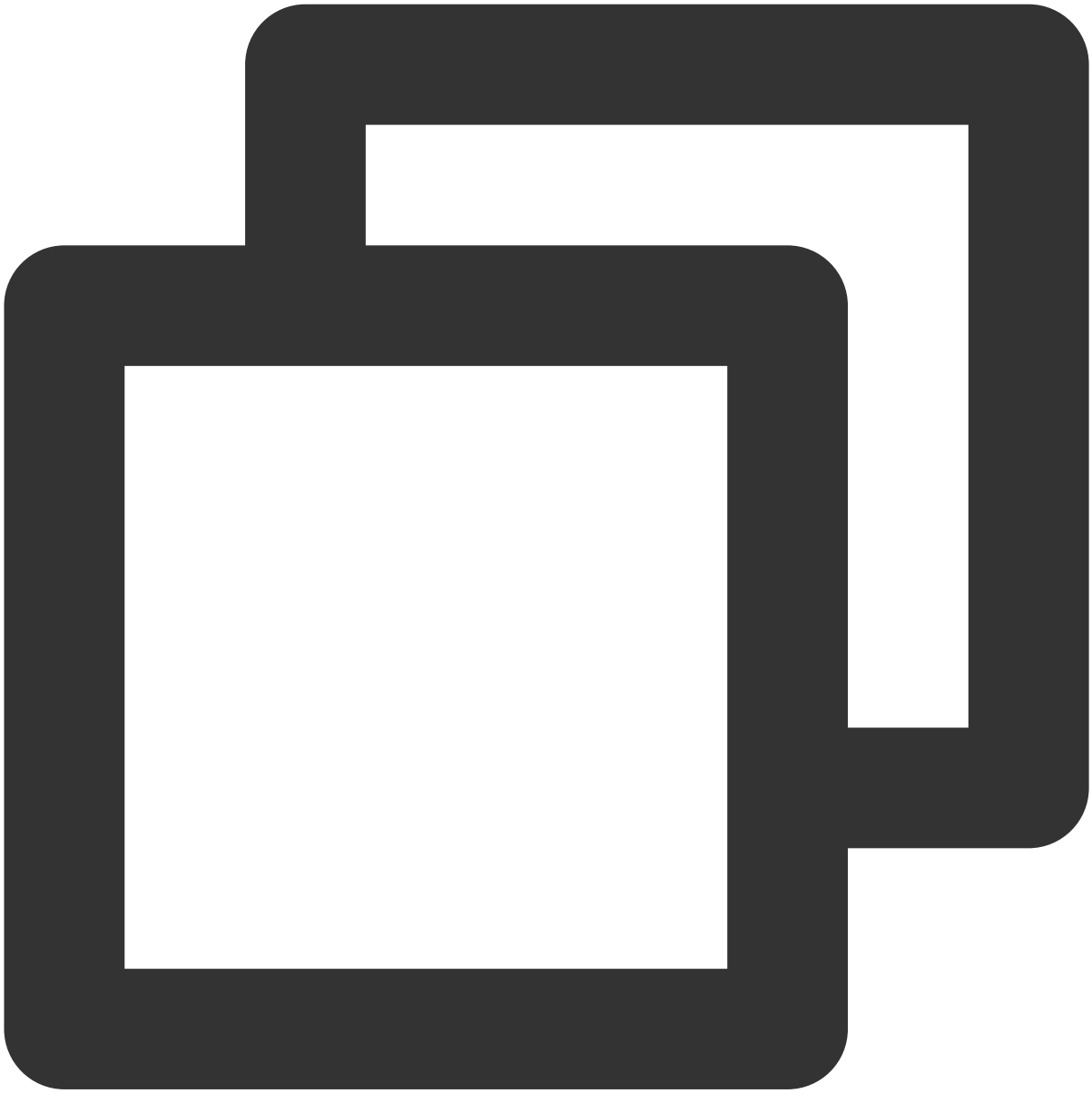
操作手順

API GatewayのTKEチャネル設定で、任意のTKEクラスターを初めて引用する場合、API Gatewayは、そのTKEクラスターAPI Serverの権限を取得し、TKEクラスターがすでにプライベートネットワークへのアクセスを有効化していることを保証する必要があります。
権限承認の操作について、TKEチャネルを設定する際、システムはクラスターが権限承認されているかどうかを自動識別します。権限が承認されていない場合は、API Gatewayがユーザー権限承認を促します。
クラスターがAPI Gatewayアクセスの権限を承認済みの場合は、**API Gateway権限承認済み**と表示されます。各クラスターは、一度権限を承認すれば、その後の利用時に繰り返し権限承認をする必要はありません。

原理の説明

API Gatewayでのユーザー権限承認取得フローは以下のとおりです。
1. ネームスペースkube-system下で、作成する名称はapigw-ingressのServiceAccountおよびapigw-ingress-clusterroleのClusterRoleです。
2. apigw-ingressおよびapigw-ingress-clusterroleを、ClusterRoleBindingでまとめてバインドします。 続けて、apigw-ingressのServiceAccountの権限がAPI Gatewayで取得され、クラスターのAPIServerのアクセスに用いられます。

そのうち、apigw-ingressという名称のServiceAccount権限は、apigw-ingress-token-をプレフィックスとするSecretの中に保存します。
API Gatewayが取得する権限の明細と具体的な方法については、作成関連リソースのYAMLファイルをご参照ください。



```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: apigw-ingress-clusterrole
rules:
  - apiGroups:
```

```
      - ""
    resources:
      - services
      - namespaces
      - endpoints
      - nodes
      - pods
    verbs:
      - get
      - list
      - watch
  - apiGroups:
      - apps
    resources:
      - deployments
      - replicasets
    verbs:
      - get
      - list
      - watch
  - apiGroups:
      - ""
    resources:
      - configmaps
      - secrets
    verbs:
      - "*"
  - apiGroups:
      - extensions
    resources:
      - ingresses
      - ingresses/status
    verbs:
      - "*"
  - apiGroups:
      - ""
    resources:
      - events
    verbs:
      - create
      - patch
      - list
      - update
  - apiGroups:
      - apiextensions.k8s.io
    resources:
      - customresourcedefinitions
```

```
    verbs:
      - "*"
  - apiGroups:
      - cloud.tencent.com
    resources:
      - tkeserviceconfigs
    verbs:
      - "*"
---
apiVersion: v1
kind: ServiceAccount
metadata:
  namespace: kube-system
  name: apigw-ingress
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: apigw-ingress-clusterrole-binding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: apigw-ingress-clusterrole
subjects:
  - kind: ServiceAccount
    name: apigw-ingress
    namespace: kube-system
```

注意事項

ユーザーがAPI GatewayのTKEクラスターのアクセス権限承認に成功すると、API Gatewayが利用を保留しているリソースの変更ができなくなります。リソースリストは以下のとおりです。
kube-systemネームスペース下の、apigw-ingressという名称のServiceAccount。
kube-systemネームスペース下の、apigw-ingress-clusterroleという名称のClusterRole。
kube-systemネームスペース下の、apigw-ingress-clusterrole-binding という名称の ClusterRoleBinding。
kube-systemネームスペース下の、apigw-ingress-token-がプレフィックスのSecret。

よくあるご質問

ご質問の説明：権限承認の際、TKEクラスターがプライベートネットワークへのアクセス機能を有効化していないことに気づきました。

解決方法：能動的に TKEクラスタープライベートネットワークへのアクセス機能を有効化し、クリックしてリトライします。

追加ノードLabelの利用

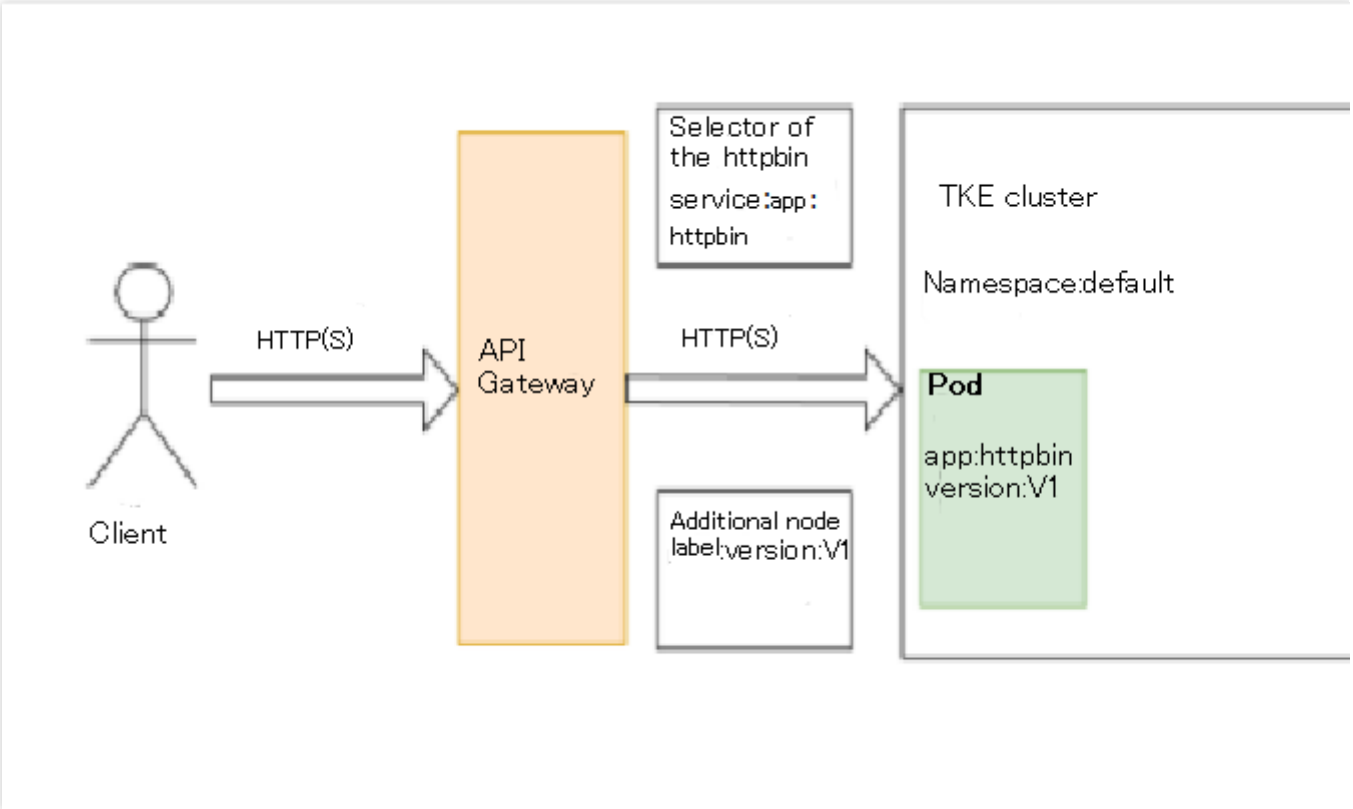
最終更新日：：2023-04-27 15:34:59

ユースケース

追加ノードLabelを利用することで、リクエストを任意のサービス下の指定LabelのあるPodに直接転送できます。転送にコントロールが必要な精密なPodです。
例：defaultネームスペースにおいて、Labelがapp: httpbinおよびversion: v1のPodと、app: httpbinおよびversion: v2のPodがあり、httpbinサービス（selectorの選択はapp:httpbin）が1つあるとします。API GatewayをLabelがapp: httpbinおよびversion: v1のPodのみに転送したい場合、追加ノードLabelに加えて、version: v1の設定をすれば、それが可能になります。

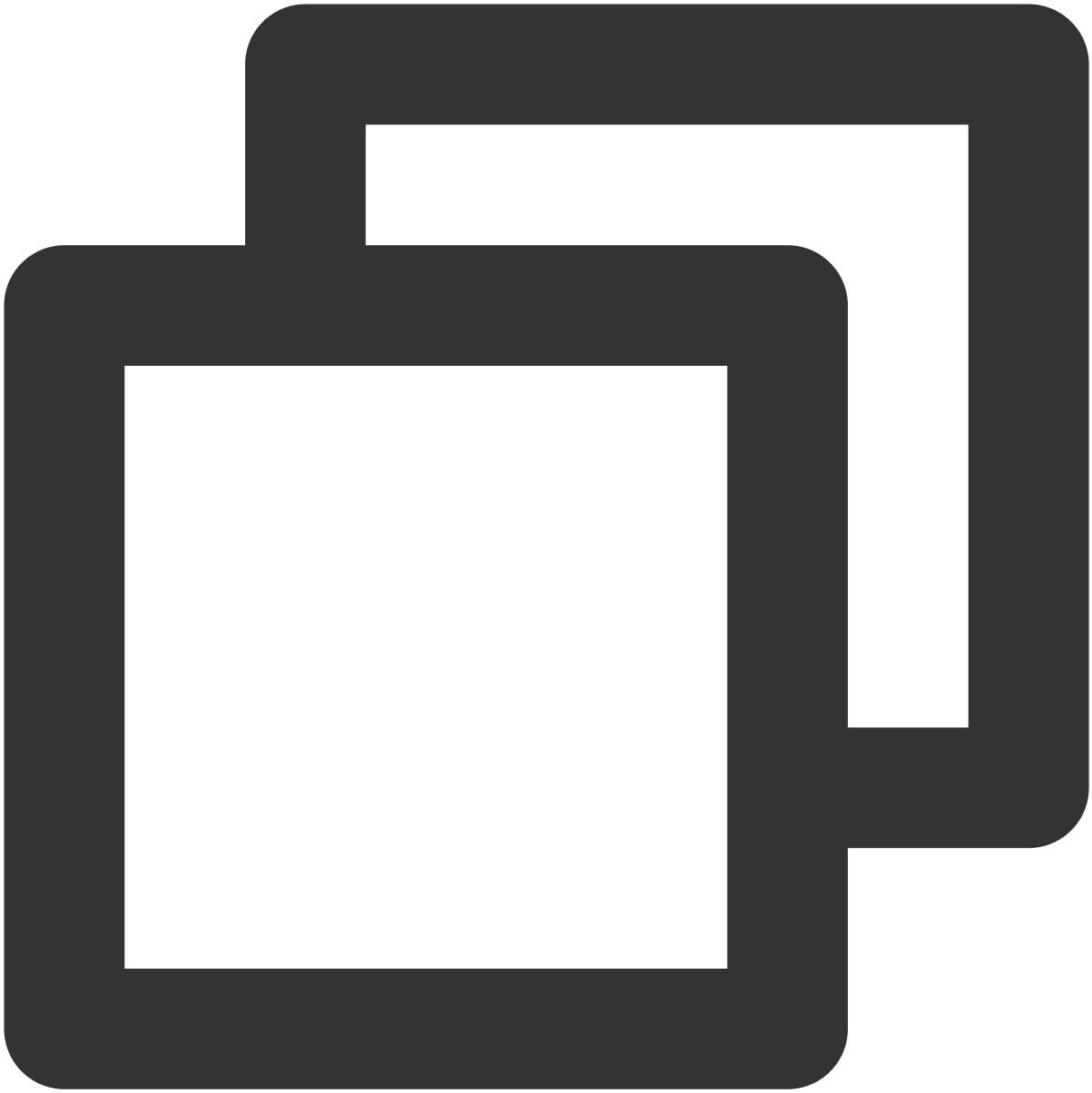
操作手順

1. TKEチャネルのサービスを設定する前提で、更に手動で追加ノードLabelを入力します。
2. **保存**をクリックし、新規作成あるいはTKEチャネルを変更します。
最終的な転送の効果は、以下のとおりです。



原理の説明

TKEクラスターでは、サービス自体にselectorの設定があります。例えば、httpbinサービスでは、selectorの設定はapp: httpbinです。しかし、API Gatewayが提供する追加ノードLabelは、httpbinサービスのselectorと結合し、組み合わせたLabelが、app: httpbinおよびversion: v1となります。これにより、TKEチャネルノードを変更すると、 発生するのはversion: v1のhttpのPodのみとなります。
追加ノードLabelで、httpbinサービスにすでに存在するLabelのキーを入力すると、追加ノードで入力したLabelが無視され、selectorに存在するLabelの値が基準になります。例えば、追加Labelでapp: not-httpbinを入力すると、このLabelとサービスhttpbinのselectorで競合が起こり、app: not-httpbinは、無視されます。
httpbinサービスのYAMLは、以下のとおりです。



```
apiVersion: v1
kind: Service
metadata:
  name: httpbin
  labels:
    app: httpbin
    service: httpbin
spec:
  ports:
  - name: http
    port: 8000
```

```
    targetPort: 80
  selector:
    app: httpbin
```

注意事項

追加ノードLabelは、高度な機能です。ユーザー入力の際は、Labelの存在の確認が必要です。間違ったLabelを入力すると、TKEチャネルのノード数量が0になります。
サービスのselectorと追加ノードLabelで同じキーが現れた場合は、selectorの設定が基準となります。
サービスのポート（port）で変更（80から8080への変更など）が発生した場合、API Gatewayでも同期変更が必要です。ポート（port）を変更せずに、ターゲットポート（target port）のみを変更した場合は、API Gatewayが自動同期します。API Gatewayで変更する必要はありません。

NginxタイプのIngress
Nginxのインストール-ingressインスタンス

最終更新日：：2023-05-23 15:54:35

NginxIngressコンポーネントをインストールする

1. TKEコンソールにログインし、左側ナビゲーションバーから**クラスター**を選択します。
2. 「クラスター管理」ページで目標のクラスターIDをクリックし、クラスター詳細ページに進みます。
3. 左側メニューバーの**コンポーネント管理**を選択し、「コンポーネントリスト」ページに進みます。
4. 「コンポーネントリスト」ページで**新規作成**を選択し、「コンポーネントの新規作成」ページでNginxIngressにチェックを入れます。
5. **完了**をクリックしてコンポーネントをインストールします。**サービスとルーティング > NginxIngress**でコンポーネントの詳細を確認できます。

注意事項

Tencent Cloud Load Balancerインスタンスは2023年3月6日にアーキテクチャがアップグレードされます。アップグレード後、パブリックネットワークCLBはドメイン名の方式でサービスを提供します。VIPは業務リクエストに応じて動的に変化し、コンソールはVIPアドレスを表示しなくなります。ドメイン名型パブリックネットワークCLBリリースのお知らせをご参照ください。
新規登録したTencent Cloudユーザーはデフォルトでアップグレード後のドメイン名型CLBを使用します。
既存のユーザーは元のCLBを継続して使用することを選択でき、アップグレードの影響を受けません。CLBサービスをアップグレードする必要がある場合は、同時にTencent Cloud製品のCLBおよびTKEをアップグレードする必要があります。アップグレードしない場合、TKE内のすべてのパブリックネットワークタイプの Service/Ingressの同期は影響を受ける可能性があります。CLBのアップグレード操作の詳細については、ドメイン名型CLBアップグレードガイドをご参照ください。TKEがService/Ingressコンポーネントのバージョンをアップグレードするには、チケットを提出してお問い合わせください。

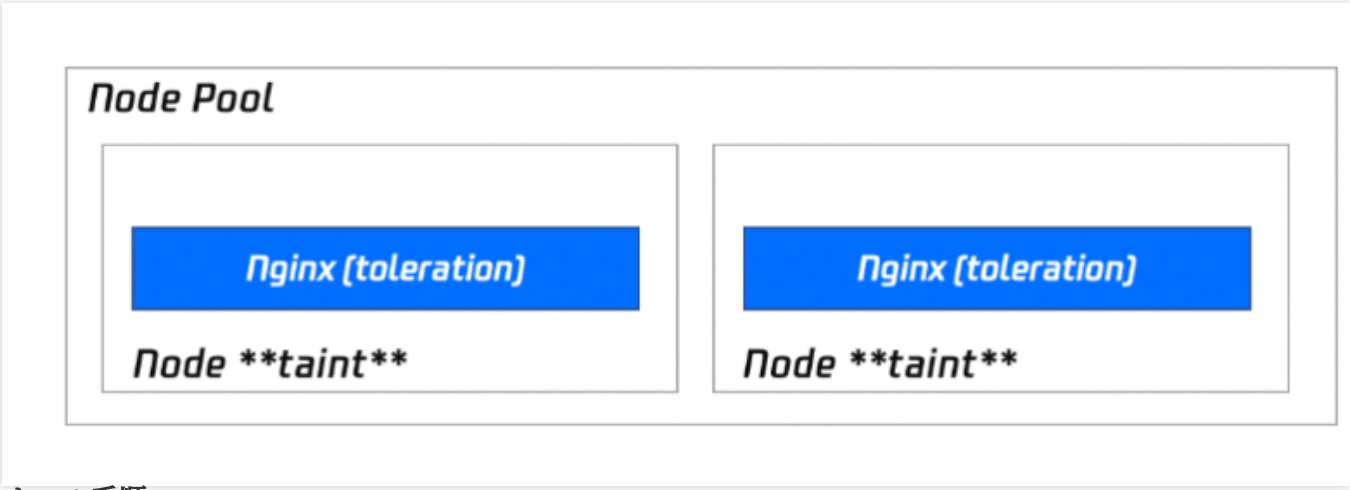
インストール方法

さまざまな業務シナリオのニーズに応じて、以下のいくつかのインストール方法を使用してTencent Kubernetes Engine（TKE）にNginx-ingressをインストールすることができます。
DaemonSet形式によってノードプールでデプロイを指定する
Deployment+HPA形式によってスケジューリングルールを指定してデプロイする

Nginxフロントエンドがlbデプロイにアクセスする方法

DaemonSet形式によって指定ノードプールにデプロイする（推奨）

Nginxをキーとするトラフィックはゲートウェイにアクセスするため、Nginxとその他の業務を同じノード内にデプロイすることはお勧めしません。指定したノードプールを使用してNginx-ingressをデプロイすることをお勧めします。デプロイアーキテクチャは下図のように表示されます。

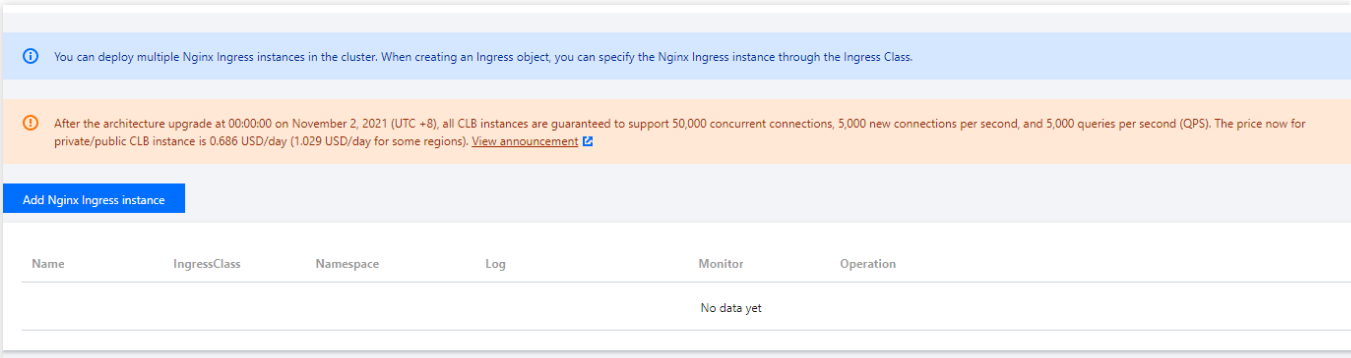


インストール手順

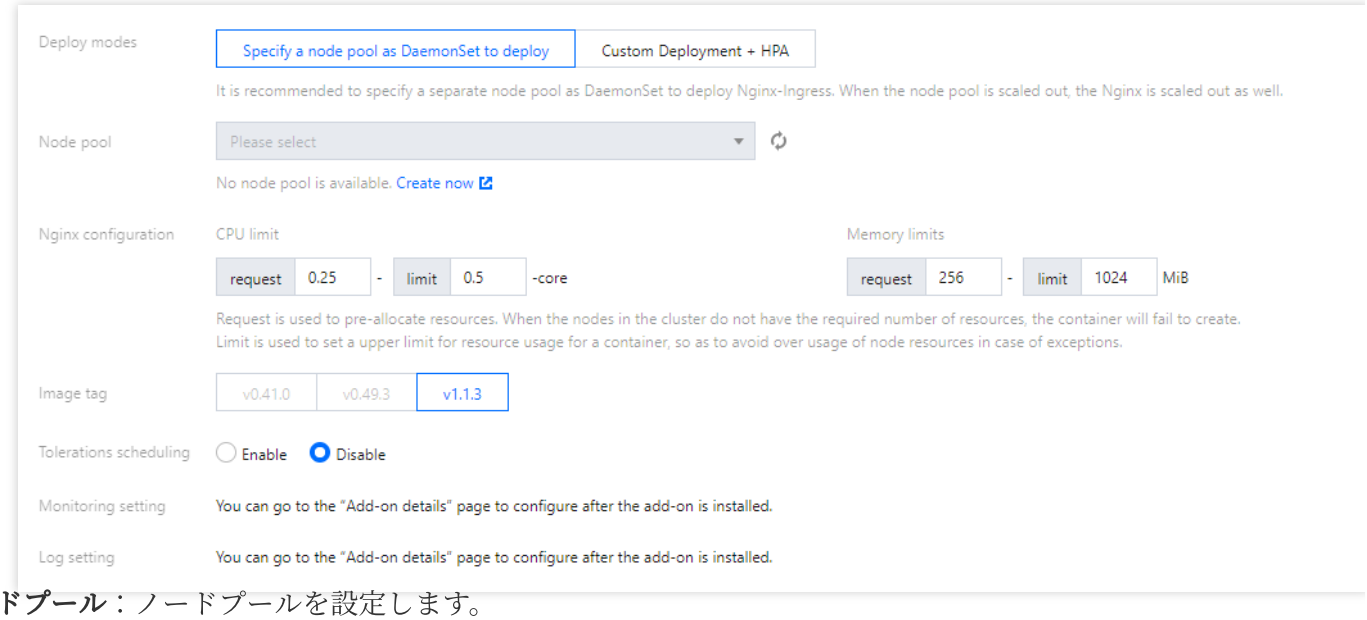
説明

このインストール方法を使用すると、ノードプールの高速スケーリングの機能をすべて使用でき、その後ノードプールの数量を調整するだけで、Nginxのレプリカをスケーリングすることができます。

1. Nginx-ingressのデプロイに使用するノードプールを準備すると同時に、taint（その他のPodがこのノードプールにスケジューリングすることを防止する）を設定します。デプロイノードプールの詳細については、ノードプール関連説明をご参照ください。
2. クラスター内でNginxIngressコンポーネントをインストールする。
3. クラスター情報ページで、**サービスとルーティング > NginxIngress**を選択し、**Nginx Ingressインスタンスの追加**をクリックします（1つのクラスター内には同時に複数のNginxを存在させることができます）。



4. 「NginxIngressの新規作成」で、デプロイオプションの**DaemonSetノードプールを指定してデプロイ**を選択し、必要に応じてその他のパラメータを設定します。下図のように表示されます。



ノードプール：ノードプールを設定します。

Nginx設定：Requstはノードプールのモデルコンフィグレーションより小さく設定する必要があります（ノード自体にリソースの予約があります）。Limitを設定しなくてもかまいません。

イメージバージョンの説明：

Kuberentesのバージョン範囲	Nginx Ingressコンポーネントがインストールをサポートしているバージョン	Nginxインスタンスがサポートするイメージバージョン
<=1.18	1.1.0、1.2.0	v0.41.0、v0.49.3
	1.0.0	v0.41.0
1.20	1.1.0、1.2.0	v1.1.3
	1.0.0	v0.41.0
>=1.22	1.1.0、1.2.0	v1.1.3

説明

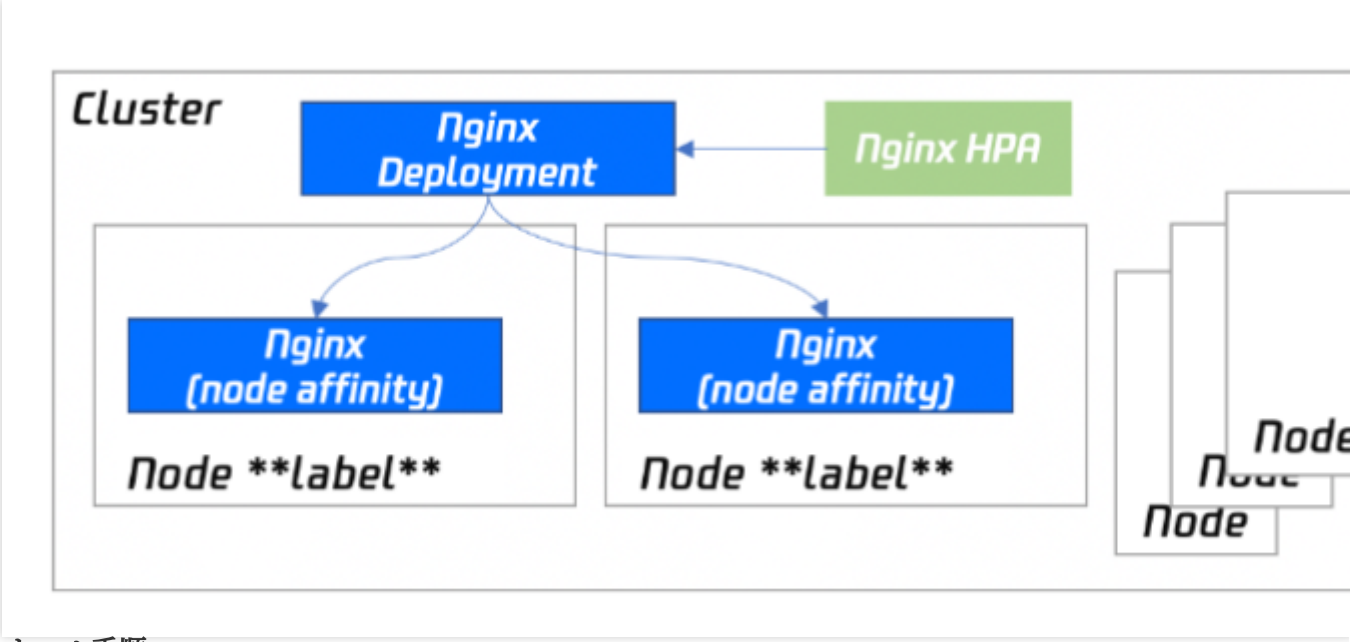
1. **コンポーネントで使用するEIPの説明**：Nginx Ingressコンポーネントのバージョン1.0.0および1.1.0はTencent CloudEIPサービス（EIP）に依存します。バージョンv1.2.0ではコンポーネントはEIPに依存しません。EIPの使用を制限する必要がある場合、Nginx Ingressコンポーネントをアップグレードすることをお勧めします。コンポーネントのアップグレードは既存のNginx Ingressインスタンスへの影響がなく、業務アクセスやデータセキュリティへの影響もありません。

2. **アップグレードの説明**：Nginxインスタンスのバージョン説明はingress-nginxドキュメントをご参照ください。クラスターのアップグレードはクラスターのアップグレード操作手順をご参照ください。Nginx Ingressコンポーネントのアップグレードはコンポーネントのアップグレード操作手順をご参照ください。

5. **OK**をクリックすると、インストールが完了します。

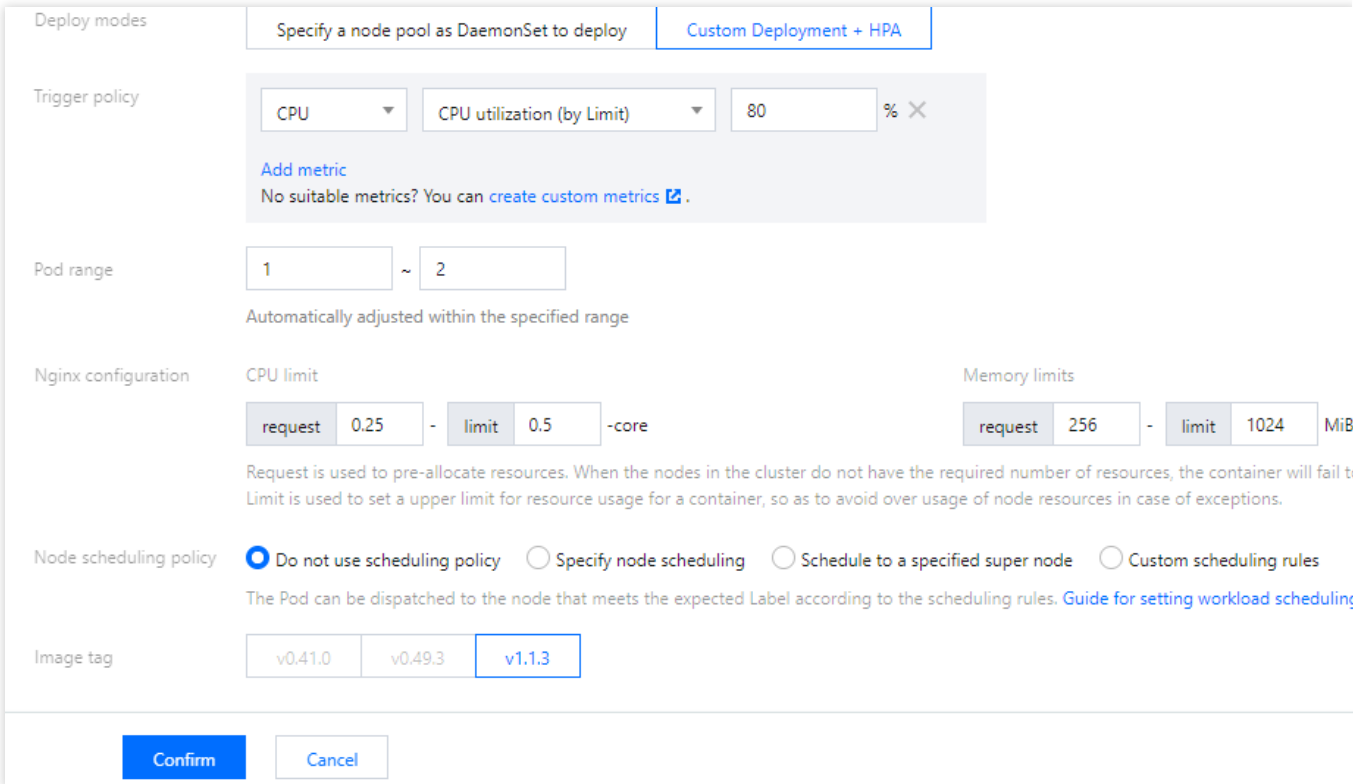
Deployment+HPA形式によってスケジューリングルールを指定してデプロイする

Deployment+HPAの形式を使用してNginx-ingressをデプロイします。業務上のニーズに応じてテイントとトレランスを設定してNginxおよび業務Podを分散してデプロイします。同時にHPAを組み合わせ、Nginxを設定してCPU/メモリなどの指標に基づいて自動スケーリングすることができます。デプロイアーキテクチャは下図のように表示されます。



インストール手順

1. Nginx-ingressのデプロイに使用するノードプールを準備すると同時に、taint（その他のPodがこのノードプールにスケジューリングすることを防止する）を設定します。デプロイノードプールの詳細については、ノードプール関連説明をご参照ください。
2. クラスター内でNginxIngressコンポーネントをインストールする。
3. クラスター情報ページで、**サービスとルーティング > NginxIngress**を選択し、**Nginx Ingressインスタンスの追加**をクリックします（1つのクラスター内には同時に複数のNginxが存在することができます）。
4. 「NginxIngressの新規作成」で、デプロイオプションの**カスタムDeployment+HPAのデプロイ**を選択し、必要に応じてその他のパラメータを設定します。下図のように表示されます。



Nginx設定：Requstはノードプールのモデルコンフィグレーションより小さく設定する必要があります（ノード自体にリソースの予約があります）。Limitを設定しなくてもかまいません。

ノードスケジューリングポリシー：ご自身で指定する必要があります。

イメージバージョンの説明：

Kubernetesが1.20以下のバージョンのクラスターは、Nginx Ingressコンポーネントのバージョンが1.0.0で、Nginxインスタンスのイメージバージョンはv41.0のみ選択できます。

Kubernetesが1.20以下のバージョンのクラスターは、Nginx Ingress コンポーネントバージョンが1.1.0ｄｄで、Nginxインスタンスのイメージバージョンはv41.0、 v49.3のみ選択できます。

Kubernetesが1.22バージョン以上のクラスターは、Nginx Ingressコンポーネントのバージョンが1.1.0のみをサポートし、 Nginxインスタンスのイメージバージョンはv1.1.3のみ選択できます。

説明

Nginxインスタンスのバージョン説明はingress-nginxドキュメントをご参照ください。クラスターのアップグレードはクラスターのアップグレード操作手順をご参照ください。Nginx Ingressコンポーネントのアップグレードはコンポーネントのアップグレード操作手順をご参照ください。

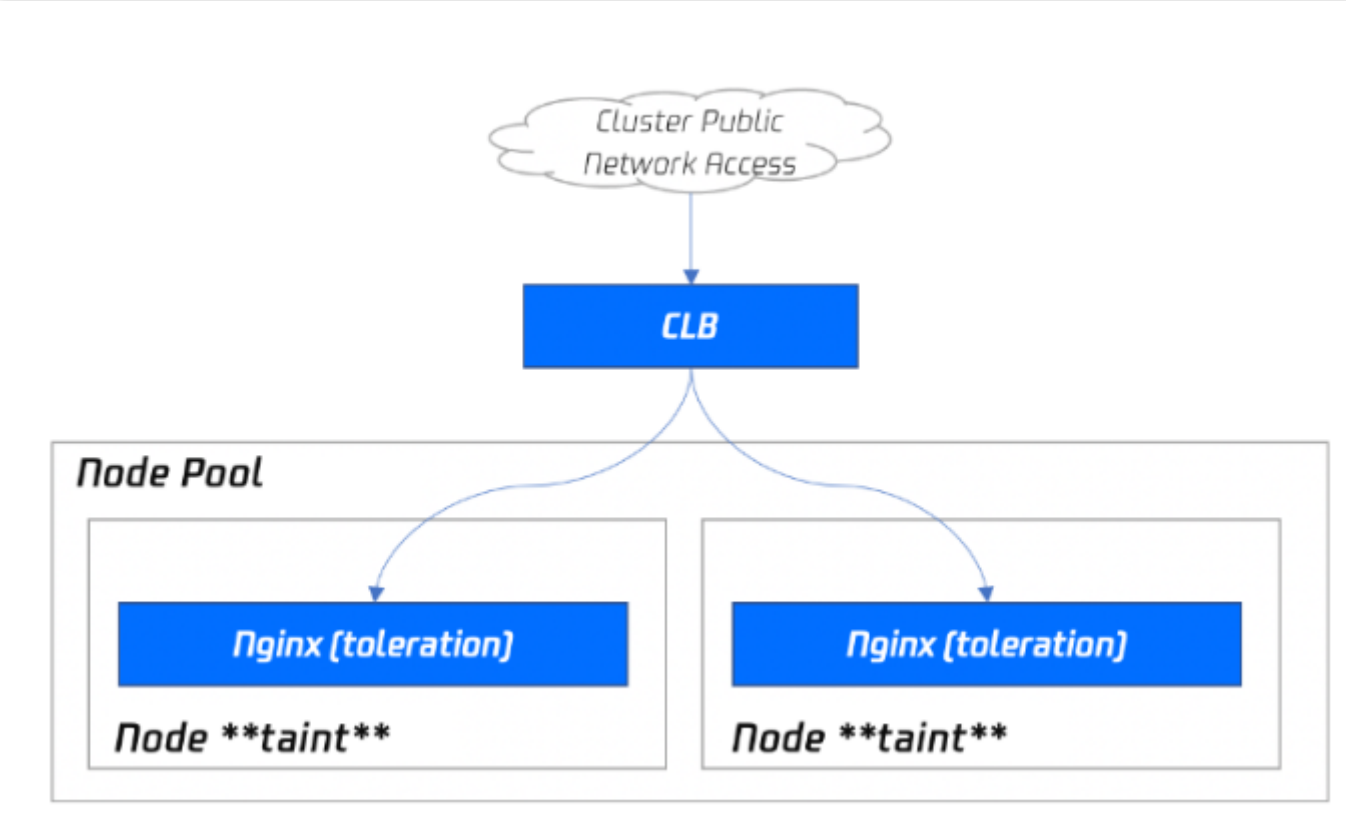
5. **OK**をクリックすると、インストールが完了します。

Nginxフロントエンドが**LB**デプロイにアクセスする方式

Nginxをクラスター内にデプロイしただけでは外部トラフィックを受信することはできず、さらにNginxのフロントエンドLBを設定する必要があります。TKEは現在製品化されたインストール機能を提供し、業務上のニーズに応じてさまざまなデプロイモードを選択することができます。

VPC-CNIモードのクラスターは**CLB**の使用により**Nginx**の**Serivce**と直接アクセスさせる（推奨）

クラスターがVPC-CNIモードのクラスターの場合、CLBを使用してNginxのSerivceに直接アクセスさせることをお勧めします。下図はノードプールでデプロイされたロードの例です。



現在の方法は性能が高く、手動でCLBをメンテナンスする必要がなく、最も理想的な方法です。この方法はクラスターがVPC-CNIをサポートしている必要があり、クラスターにVPC-CNIネットワークプラグインが設定されているか、Global Routerネットワークプラグインが設定されている場合はVPC-CNIのサポート（2種類のモードを併用）を有効化し、この方法を使用することをお勧めします。

Globalrouterモードのクラスターは通常Loadbalancerモードの Serviceを使用する

クラスターがVPC-CNIモードのネットワークをサポートしていない場合、通常のLoadbalancerモードServiceによってトラフィックにアクセスします。現在TKEのLoadBalancerタイプのServiceはデフォルトでNodePortに基づいて実現され、CLBは各ノードのNodePortをバックエンドRSとしてバインドし、トラフィックをノードのNodePortに転送します。その後ノード上でさらにiptablesまたはipvsによってリクエストをServiceに対応するバックエンドPodにルーティングします。この方法は最も簡単な方法ですが、トラフィックはNodePortを通過し、もうひとつの層によって転送されます。以下のような問題が存在する可能性があります。
転送パスが長い場合、トラフィックはNodePortに到達するとさらにk8s内部のCLB、iptablesまたはipvsを介してNginxに転送され、ネットワークの消費時間が増加します。
NodePortを通過すると必然的にSNATが発生します。トラフィックが集中しすぎるとソースポートの枯渇またはconntrack挿入の競合によってパケットロスを引き起こしやすくなり、一部のトラフィックに異常が発生します。

各ノードのNodePortも1つのロードバランサとしても機能します。CLBが大量ノードのNodePortをバインドしている場合、CLBの状態は各ノードに分散し、コンテナはグローバルロードバランシングが不均一になります。CLBはNodePortにヘルスチェックを行い、検出パケットは最終的にnginx ingressのPodに転送されます。 CLBにバインドされたノードが多く、Nginx-ingressのPodが少ない場合、検出パケットはNginx-ingressに対して大きな圧力を発生させます。

HostNetwork+LBモードを使用する

コンソールは現時点ではサポートしていません。手動でNginxワークロードのYaml設定ネットワークモードをHostNetworkに変更し、CLBバインドNginxが公開するノードポートを手動で作成します。
hostNetworkを使用する時は、ポートリスニングの競合を避けるために、Nginx-ingressのPodは同一ノードにスケジューリングすることができないことに注意する必要があります。

TKEによってNginx-ingressのデフォルトパラメータをインストールする

Nginx-ingressパラメータを設定する

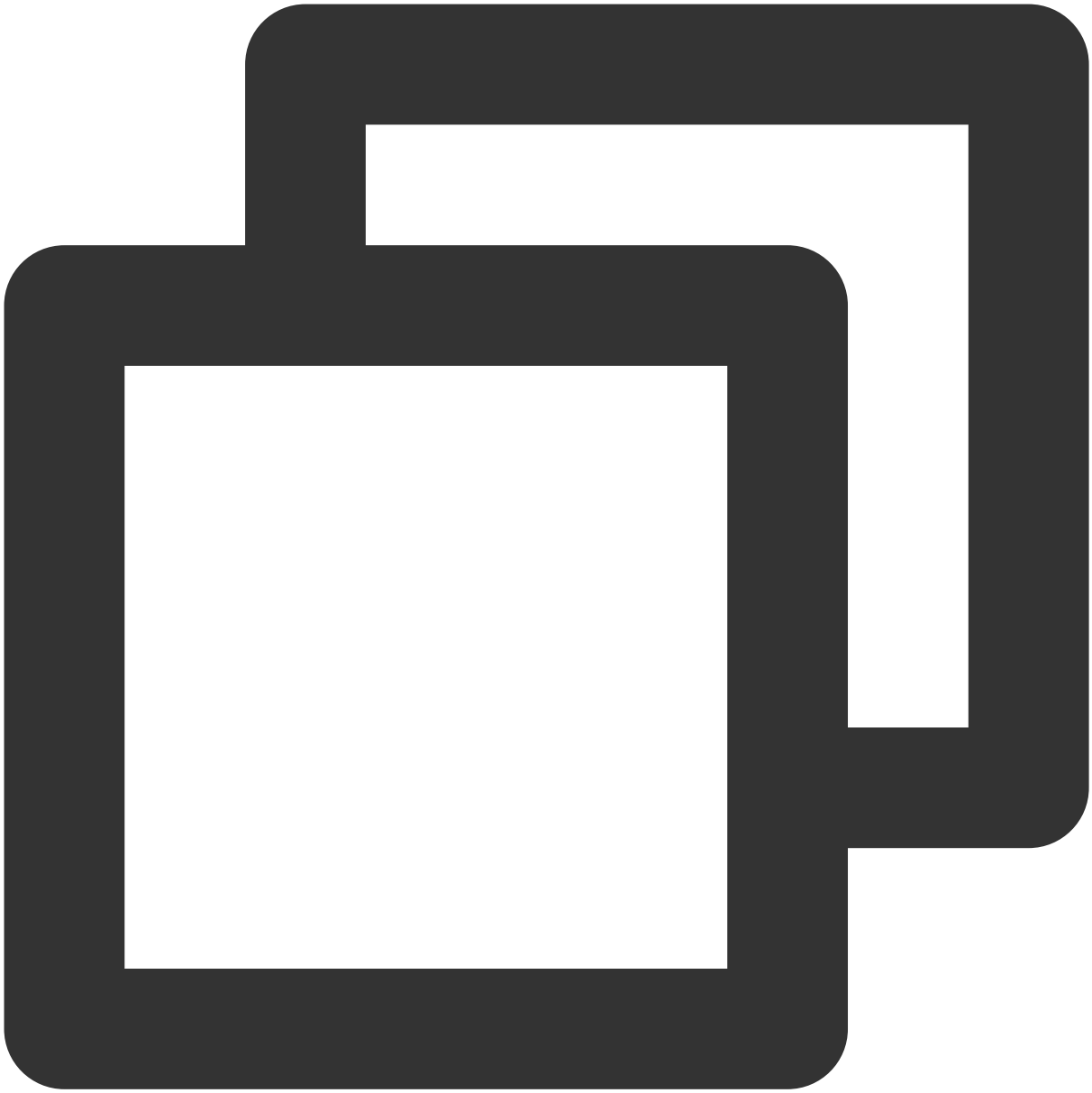
Nginx-ingressコンポーネント詳細ページで、Ningxパラメータtabから選択したNginx-ingressインスタンスにYAML編集を行います。

注意

デフォルト状態で設定パラメータはNginxを再起動させず、有効時間がわずかに遅延します。

1. TKEコンソールにログインし、左側ナビゲーションバーから**クラスター**を選択します。
2. 「クラスター管理」ページで目標のクラスターIDをクリックし、クラスター詳細ページに進みます。
3. 左側メニューバーの**コンポーネント管理**を選択し、「コンポーネントリスト」ページに進みます。
4. パラメータを設定する必要があるコンポーネント右側の**Nginx設定の更新**をクリックし、「Nginx設定」ページに進みます。
5. Nginx Ingressインスタンスを選択し、**YAMLの編集**をクリックします。
6. 「ConfigMapの更新」ページで編集し、**完了**をクリックすればパラメータの設定は完了です。

パラメータの設定例



```
apiVersion: v1
kind: ConfigMap
metadata:
  name: alpha-ingress-nginx-controller
  namespace: kube-system
data:
  access-log-path: /var/log/nginx/nginx_access.log
  error-log-path: /var/log/nginx/nginx_error.log
  log-format-upstream: $remote_addr - $remote_user [$time_iso8601] $msec "$request"
  keep-alive-requests: "10000"
  max-worker-connections: "65536"
```

```
    upstream-keepalive-connections: "200"
```

注意

`access-log-path` 、 `error-log-path` 、 `log-format-upstream` を変更しないでください。変更した場合、 CLSログ収集に影響を与える可能性があります。

業務に応じて異なるパラメータを設定する必要がある場合は、公式ドキュメントをご参照ください。

ストレージ管理
Cloud Object Storage（COS）の使用

最終更新日：：2023-05-09 16:18:00

ユースケース

Tencent Kubernetes Engine（TKE）では、PersistentVolume（PV）およびPersistentVolumeClaim（PVC）を作成し、ワークロードにデータボリュームをマウントする方法でTencent Cloud Object Storage（COS）を使用することができます。ここでは、TKEクラスターでワークロードにCOSをマウントする方法についてご説明します。

準備作業

1. COS拡張コンポーネントのインストール

説明

クラスターにCOS-CSI拡張コンポーネントがすでにインストールされている場合は、この手順をスキップしてください。

1. TKEコンソールにログインし、左側ナビゲーションバーの**クラスター**を選択します。
2. **クラスター管理**ページでターゲットのクラスターIDをクリックし、クラスター詳細ページに進みます。
3. 左側ナビゲーションバーで**コンポーネント管理**を選択し、**コンポーネント管理**ページで**新規作成**をクリックします。
4. **コンポーネントの新規作成**ページで**COS（Tencent Cloud Object Storage）**コンポーネントにチェックを入れます。
5. **完了**をクリックします。

2. アクセスキーの作成

注意

ルートアカウントキーの漏洩によるクラウド上の資産の損失を防ぐため、セキュリティ設定ポリシーを参照して、ルートアカウントによるコンソールへのログインまたはルートアカウントキーによるTencent Cloud APIへのアクセスを停止し、関連の管理権限を付与されたサブアカウント/コラボレーターを使用して関連リソースの操作を行うことをお勧めします。

ここではアクセス管理に関連する権限を付与されたサブユーザーがアクセスキーを作成または確認する場合を例にとります。サブユーザーを作成し、アクセス管理権限を実装する方法に関しては、ドキュメントサブユーザーのカスタム作成をご参照ください。

1. サブアカウントユーザーを使用してCAMコンソールにログインし、左側ナビゲーションバーで**アクセスキー** > **APIキー管理**を選択します。
2. **APIキー管理**ページで、**キーの新規作成**をクリックし、作成が完了するまで待ちます。

説明

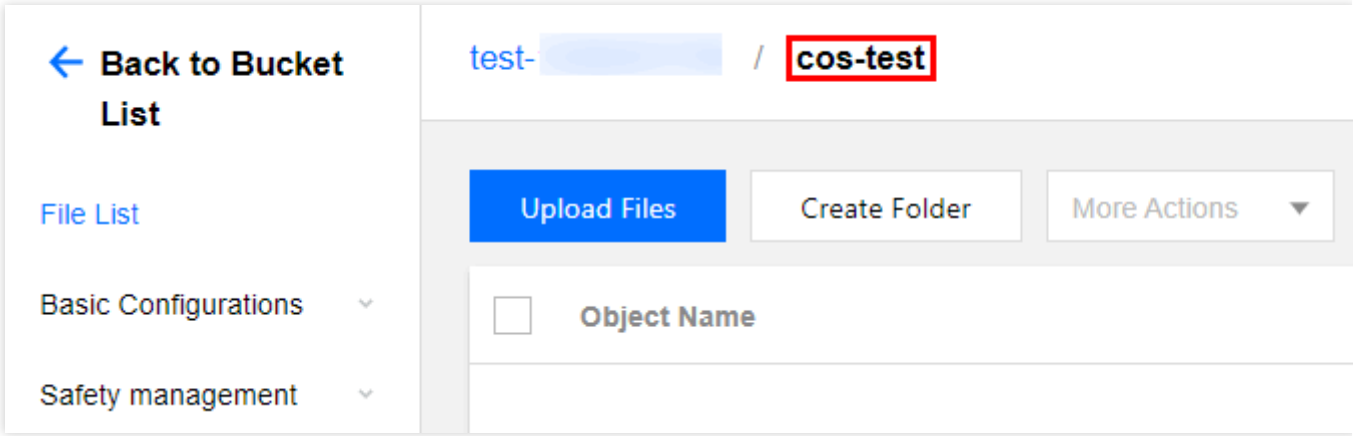
APIキーは1つのサブユーザーにつき、最大2つまで作成できます。
APIキーはTencent Cloud APIリクエストを作成するための重要なクレデンシャルです。ご自身の財産とサービスの安全性のため、キーを適切に保存し、定期的に変更してください。キーを変更した後は、古いキーを速やかに削除してください。

3. バケットの作成

COSコンソールにログインしてバケットを作成します。操作の詳細についてはバケットの作成をご参照ください。作成完了後にバケットリストで確認します。

4. バケットサブディレクトリの取得

1. **バケットリスト**ページで、作成済みのバケット名をクリックし、このバケットの詳細ページに進みます。
2. 左側ナビゲーションバーの**ファイルリスト**を選択し、ファイルリストでマウントしたいサブフォルダを選択してこのフォルダの詳細ページに進みます。ページ右上隅でサブディレクトリパス `/costest` を取得します。下の図をご覧ください。

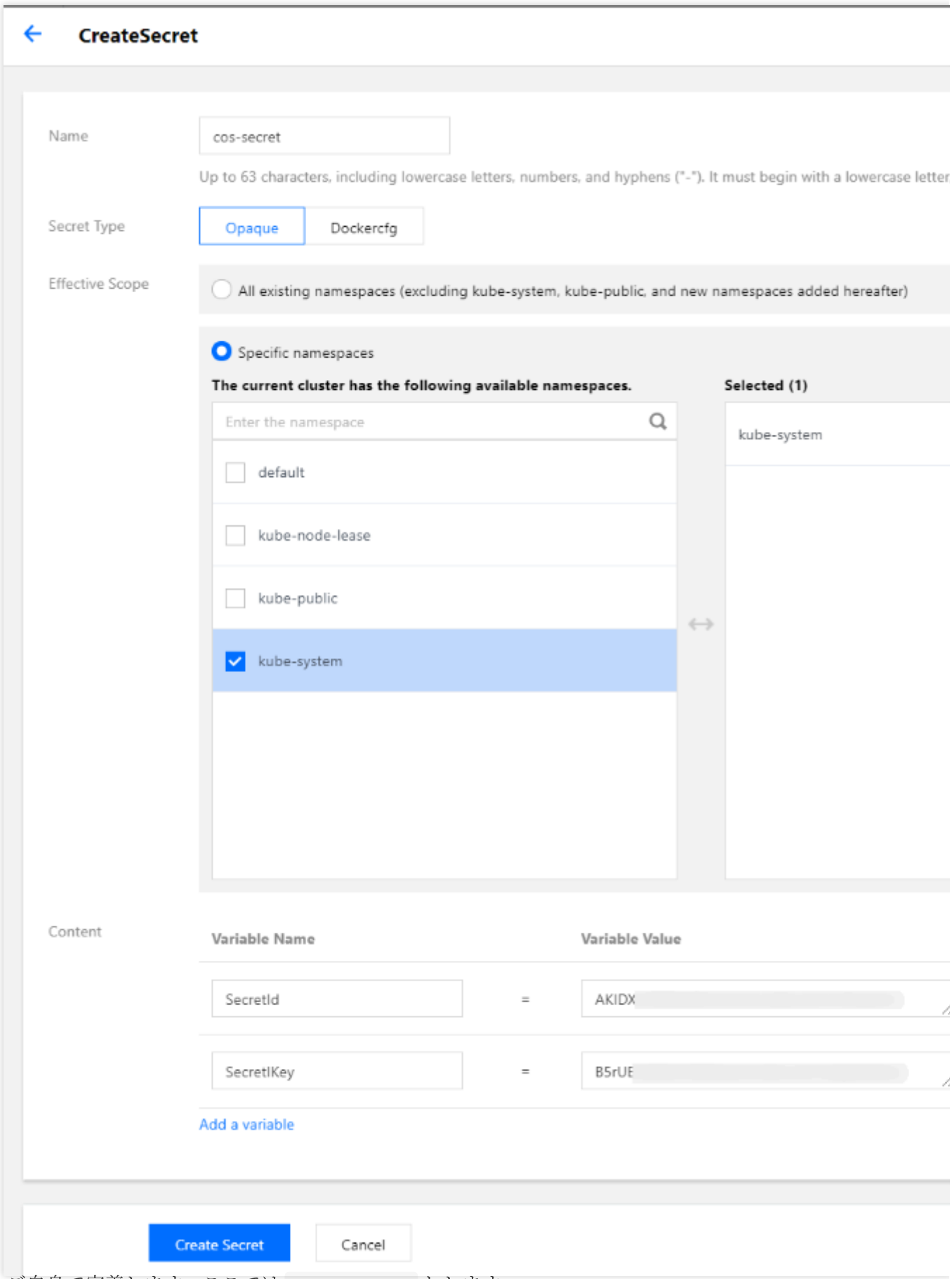


操作手順

コンソールによるCOSの使用

手順1：COSにアクセス可能なSecretを作成する

1. TKEコンソールにログインし、左側ナビゲーションバーの**クラスター**を選択します。
2. **クラスター管理**ページでターゲットのクラスターIDをクリックし、クラスター詳細ページに進みます。
3. 左側ナビゲーションバーの**設定管理** > **Secret**を選択し、Secretのページで**新規作成**をクリックします。
4. **Secretの新規作成**ページで、次の情報に従って設定を行います。下の図をご覧ください。

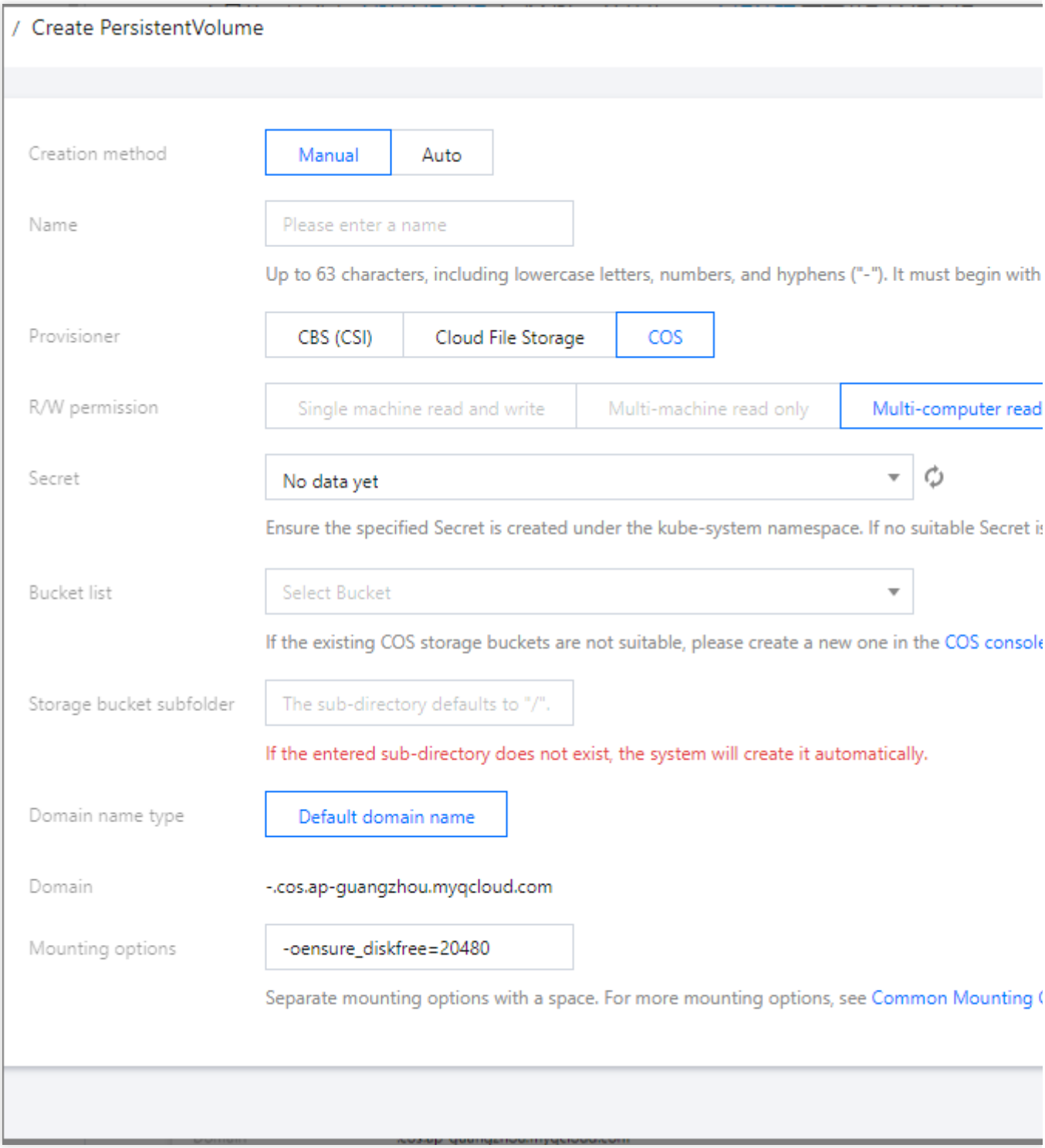


名前：ご自身で定義します。ここでは `cos-secret` とします。

Secretタイプ：**Opaque**を選択します。このタイプはキー証明書およびプロファイルの保存に適しています。ValueはBase64形式でエンコードされます。
効力の範囲：**指定のネームスペース**を選択します。Secretが `kube-system` ネームスペースに作成されていることを確認してください。
内容：ここはSecretがバケット（Bucket）にアクセスするために必要なアクセスキーの設定に使用します。変数名 `SecretId` と `SecretKey` 、およびそれらにそれぞれ対応する変数値が含まれる必要があります。アクセスキーの作成を参照して作成し、APIキー管理ページでアクセスキーを取得してください。
5. **Secretの作成**をクリックします。

ステップ2：COS-CSI動的設定をサポートするPVを作成する

注意
この手順ではバケットを使用します。現在のリージョンに使用可能なバケットがない場合は、バケットの作成を参照して作成してください。
1. ターゲットのクラスターの詳細ページで、左側メニューバーの**Storage** > **PersistentVolume**を選択し、**PersistentVolume** ページで**新規作成**をクリックします。
2. **PersistentVolumeの新規作成**ページで、次の情報を参照してPVを作成します。下の図をご覧ください。



3.

主なパラメータ情報は下記の通りです：

ソース設定：**静的作成**を選択します。

名前：ご自身で定義します。ここでは `cos-pv` とします。

Provisioner：**COS**を選択します。

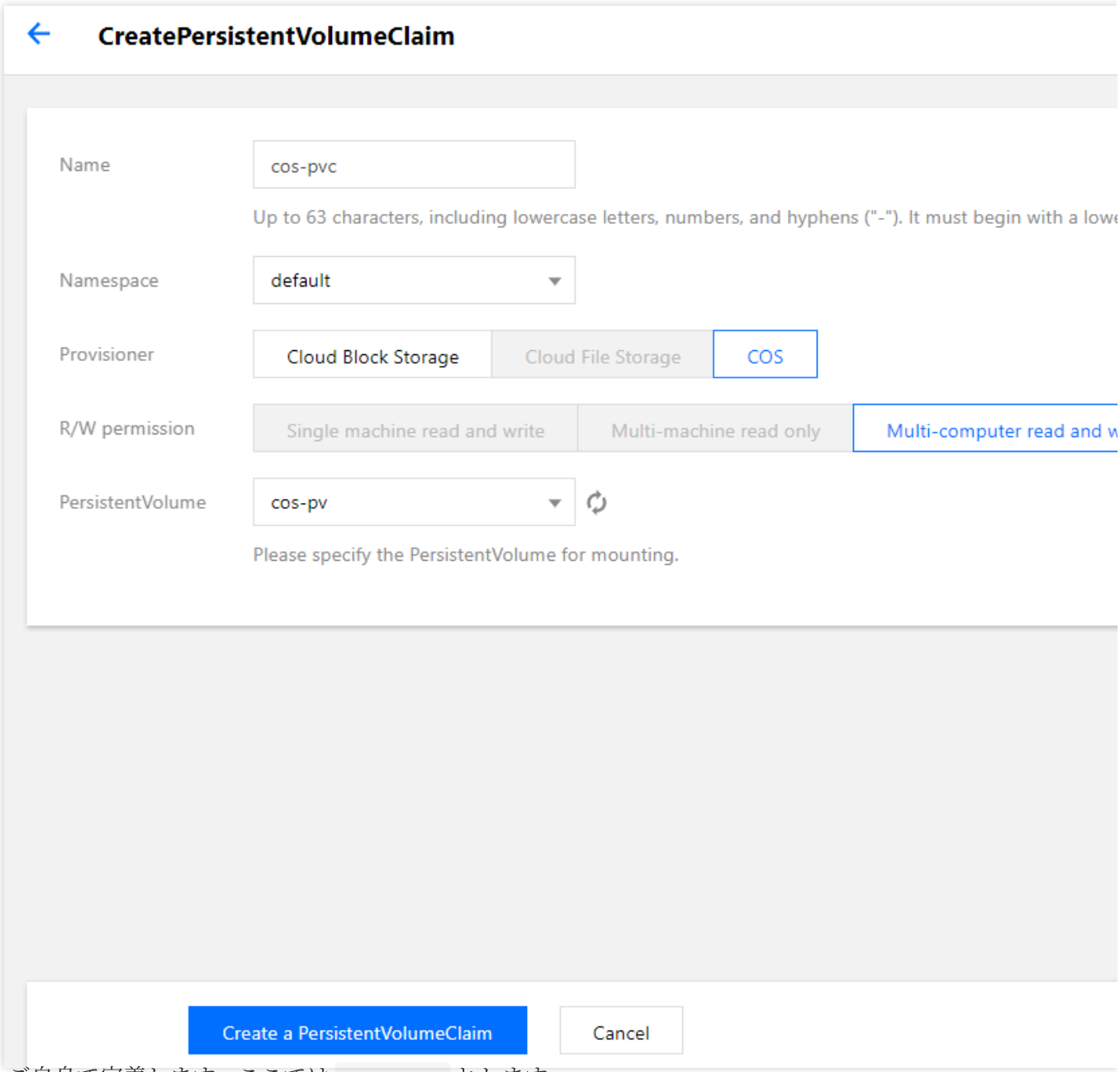
読み取り/書き込み権限：COSはマルチマシン読み取り/書き込みのみサポートしています。

説明

シングルマシン読み取り/書き込み：現在Cloud Block Storage（CBS）は同時に1台のマシンへのマウントのみをサポートしているため、単一のマシンのデータ読み込み/書き込みのみ処理できます。
マルチマシン読み取り/書き込み：Cloud File Storage（CFS）/Cloud Object Storage（COS）は複数のマシンへの同時マウントをサポートしているため、複数のマシンのデータ読み込み/書き込みを処理できます。
Secret：ステップ1 で作成したSecretを選択します。ここでは `cos-secret` とします（Secretが `kube-system` ネームスペースに作成されていることを確認してください）。
バケットリスト：COS内のオブジェクトを保存するために使用します。使用可能なバケットを必要に応じて選択します。
バケットサブディレクトリ：バケットサブディレクトリの取得で取得したバケットサブディレクトリを入力します。ここでは `/costest` とします。入力したサブディレクトリが存在しない場合は、システムによって自動的に作成されます。
ドメイン名：デフォルトドメイン名が表示されます。このドメイン名を使用してバケットにアクセスできます。
マウントオプション：COSFSツールはバケットのローカルへのマウントをサポートしており、マウント後はCOS内のオブジェクトを直接操作することができます。この項目は関連の制限条件の設定に用いられます。この例のマウントオプション `-oensure_diskfree=20480` は、キャッシュファイルが存在するディスクの空き容量が20480MB未満になった場合、COSFSが起動に失敗することを表します。
説明
各マウントオプションはスペースで区切ってください。その他のマウントオプションについては一般的なマウントオプションドキュメントをご参照ください。
4. **PersistentVolumeの作成**をクリックします。

ステップ3：PVCを作成してPVをバインドする

注意
ステータスがBoundであるPVはバインドしないでください。
1. ターゲットのクラスターの詳細ページで、左側メニューバーの**Storage** > **PersistentVolumeClaim**を選択し、**PersistentVolumeClaim**ページで**新規作成**をクリックします。
2. **PersistentVolumeClaim**の**新規作成**ページで、次の情報を参照してPVCを作成します。下の図をご覧ください。



名前：ご自身で定義します。ここでは `cos-pvc` とします。

ネームスペース： `kube-system` を選択します。

Provisioner：**COS**を選択します。

読み取り/書き込み権限：COSはマルチマシン読み取り/書き込みのみサポートしています。

PersistentVolume：ステップ2で作成したPVを選択します。ここでは `cos-pv` とします。

3. **PersistentVolumeClaimの作成**をクリックします。

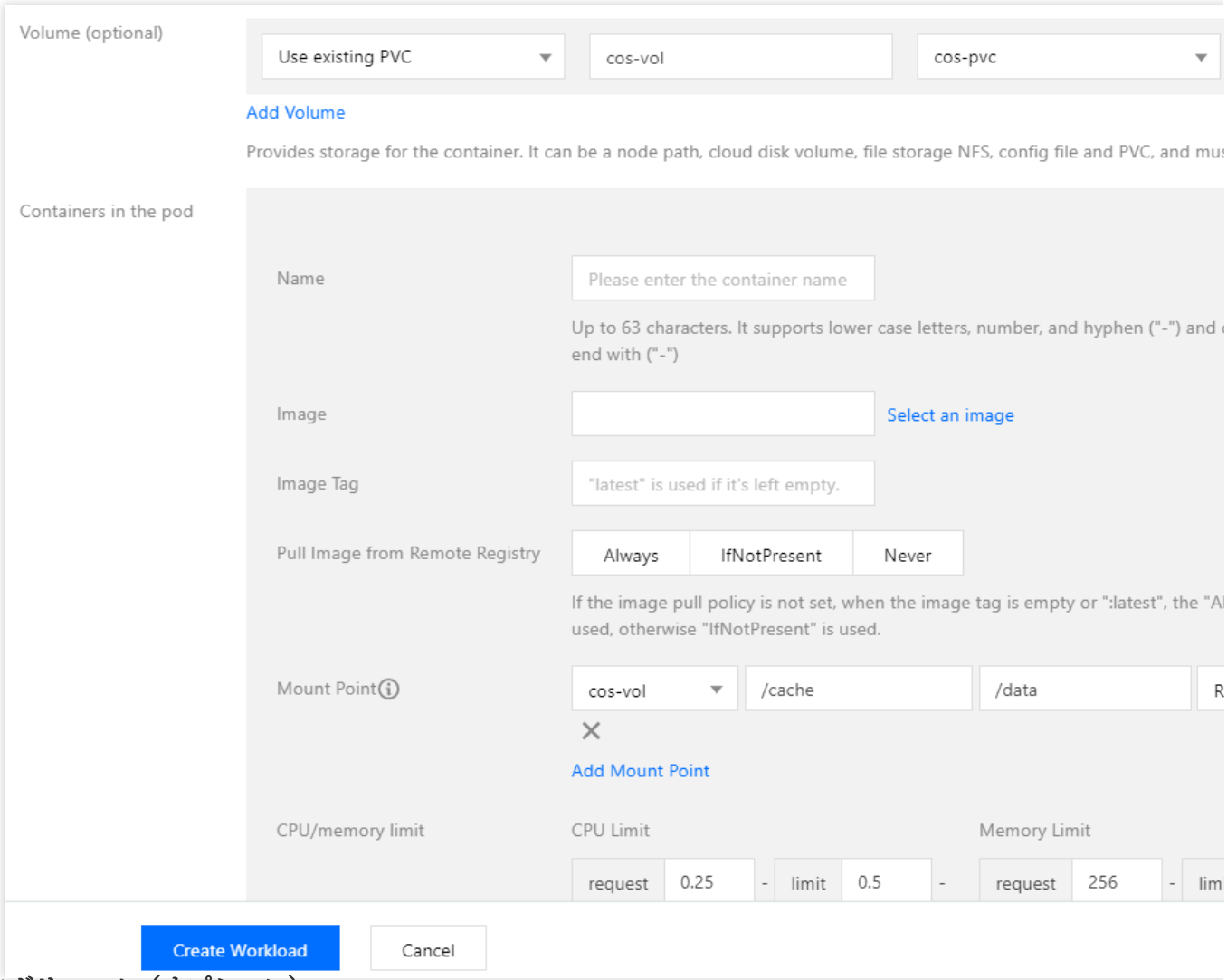
ステップ4：Podが使用するPVCを作成する

説明

この手順ではワークロードDeploymentの作成を例にとります。

1. ターゲットのクラスターの詳細ページで、左側メニューバーの**ワークロード** > **Deployment**を選択し、Deploymentページで**新規作成**をクリックします。

2. **Deploymentの新規作成**ページで、Deploymentの作成を参照して作成し、データボリュームのマウントを設定します。下の図をご覧ください。



データボリューム（オプション）：

マウント方式：**既存のPVCを使用**を選択します。

データボリューム名：ご自身で定義します。ここでは `cos-vol` とします。

PVCの選択：ステップ3で作成したPVCを選択します。ここでは `cos-pvc` を選択します。

インスタンス内コンテナ：**マウントポイントの追加**をクリックし、マウントポイントを設定します。

データボリューム：この手順で追加したデータボリューム「cos-vol」を選択します。

ターゲットパス：ターゲットパスを入力します。ここでは `/cache` とします。

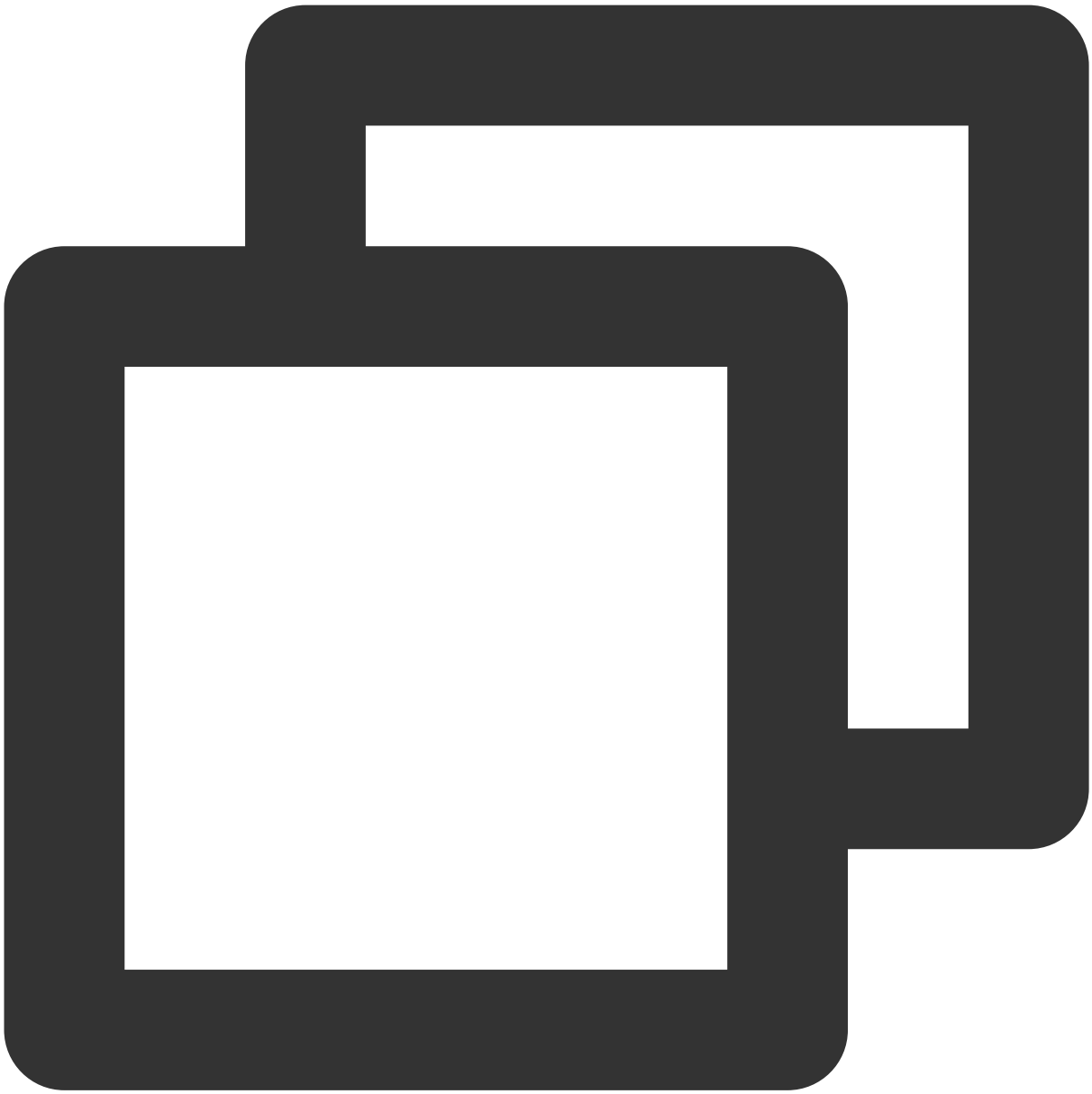
サブパスのマウント：選択したデータボリューム内のサブパスまたは単一のファイルのみをマウントします（例：`./data` または `data`）。

3. **Deploymentの作成**をクリックします。

YAMLファイルによるCOSの使用

COSにアクセス可能なSecretを作成する

COSにアクセス可能なSecretをYAMLで作成できます。テンプレートは次のとおりです。

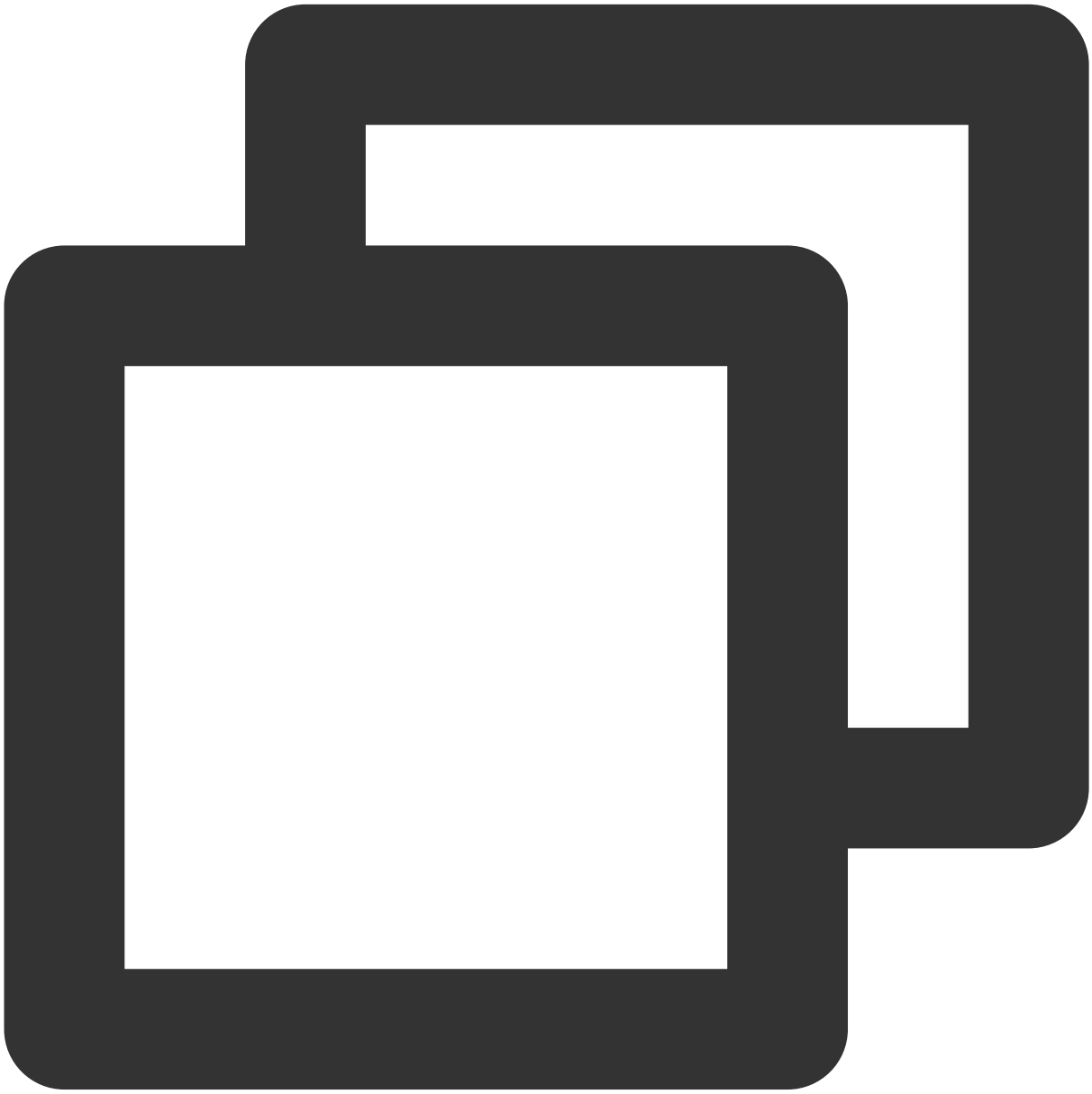


```
apiVersion: v1
kind: Secret
type: Opaque
metadata:
  name: cos-secret
  # Replaced by your secret namespace.
  namespace: kube-system
data:
  # Replaced by your temporary secret file content. You can generate a temporary se
```

```
  # Note: The value must be encoded by base64.
  SecretId: VWVEJxRk5Fb0JGbDA4M...(base64 encode)
  SecretKey: Qa3p4ZTVCMFlQek...(base64 encode)
```

COS-CSI動的設定をサポートするPVを作成する

COS-CSI動的設定をサポートするPVをYAMLで作成できます。テンプレートは次のとおりです。

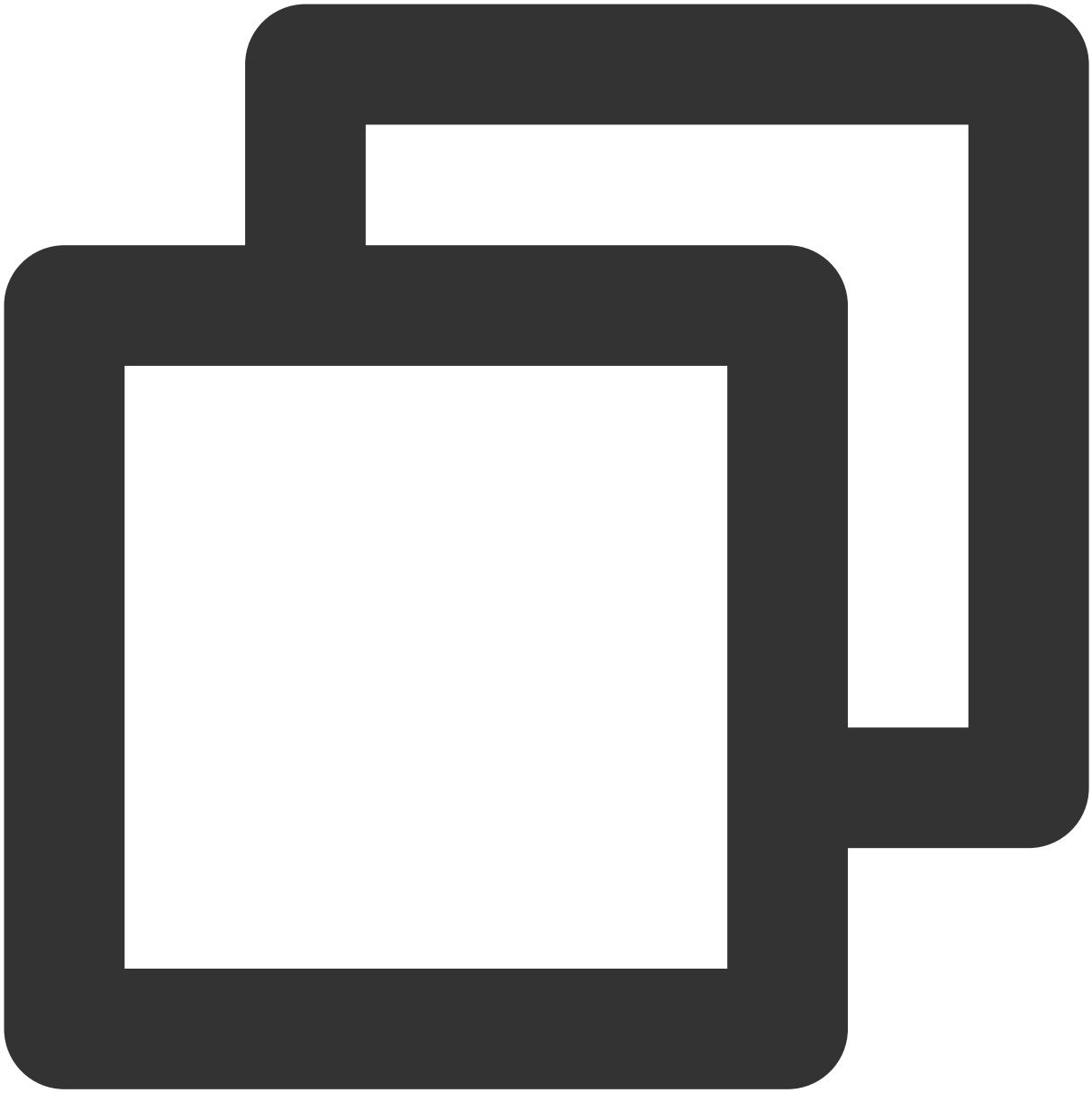


```
apiVersion: v1
kind: PersistentVolume
metadata:
```

```
  name: cos-pv
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 10Gi
  csi:
    driver: com.tencent.cloud.csi.cosfs
    nodePublishSecretRef:
      name: cos-secret
      namespace: kube-system
    volumeAttributes:
      # Replaced by the url of your region.
      url: http://cos.ap-XXX.myqcloud.com
      # Replaced by the bucket name you want to use.
      bucket: XXX-1251707795
      # You can specify sub-directory of bucket in cosfs command in here.
      path: /costest
       # You can specify any other options used by the cosfs command in here.
    # additional_args: "-oallow_other"# Specify a unique volumeHandle like bucket n
    volumeHandle: XXX
  persistentVolumeReclaimPolicy: Retain
  volumeMode: Filesystem
```

PVCを作成してPVをバインドする

上記のPVをバインドしたPVCをYAMLで作成できます。テンプレートは次のとおりです。

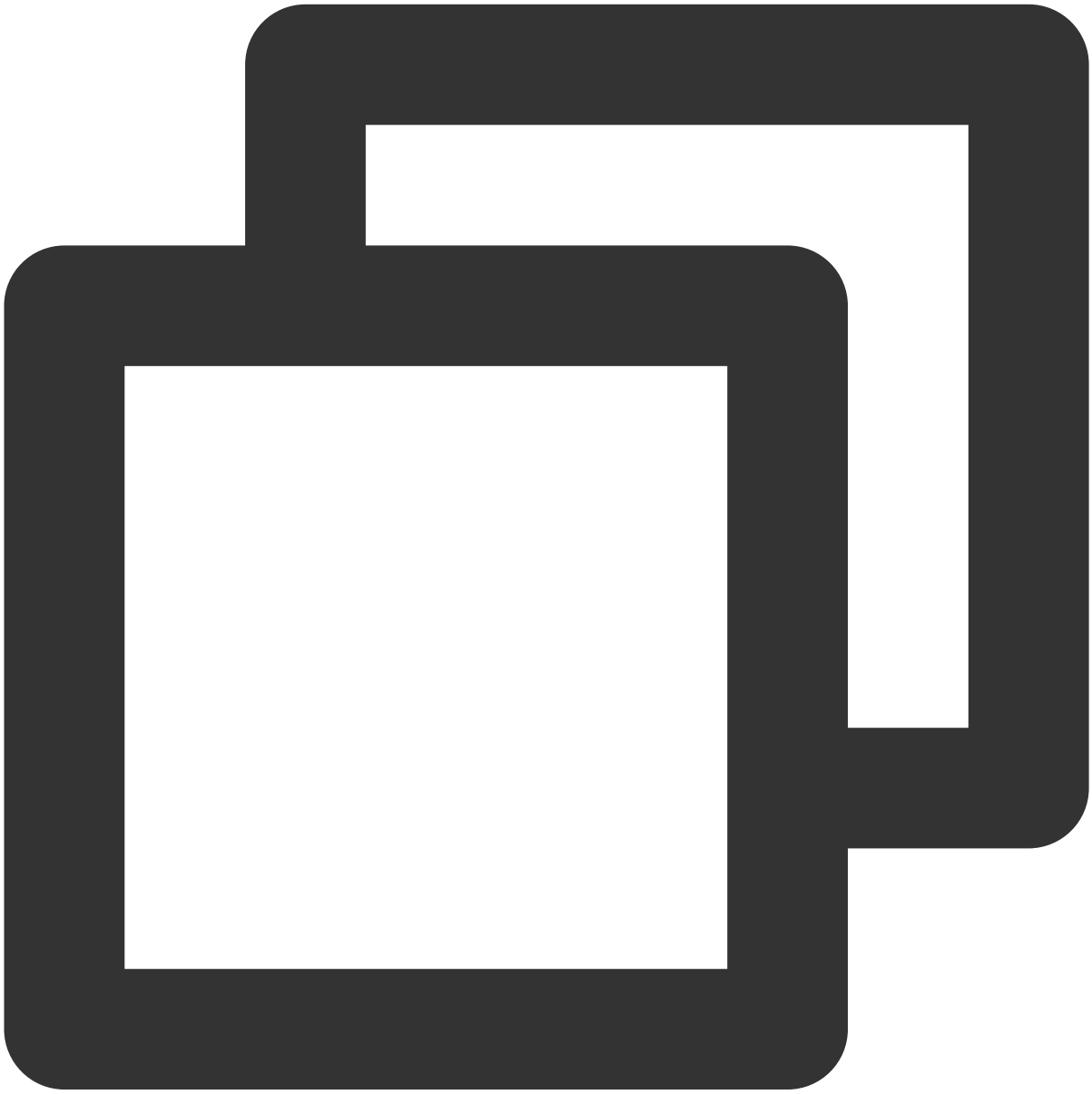


```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: cos-pvc
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  # You can specify the pv name manually or just let kubernetes to bind the pv and
```

```
  # volumeName: cos-pv
  # Currently cos only supports static provisioning, the StorageClass name should b
  storageClassName: ""
```

Podが使用するPVCを作成する

PodをYAMLで作成できます。テンプレートは次のとおりです。



```
apiVersion: v1
kind: Pod
metadata:
```

```
  name: pod-cos
spec:
  containers:
  - name: pod-cos
    command: ["tail", "-f", "/etc/hosts"]
    image: "centos:latest"
    volumeMounts:
    - mountPath: /data
      name: cos
    resources:
      requests:
        memory: "128Mi"
        cpu: "0.1"
  volumes:
  - name: cos
    persistentVolumeClaim:
      # Replaced by your pvc name.
      claimName: cos-pvc
```

関連情報

COSの使用方法に関するその他の情報については、README_COSFS.mdをご参照ください。

コンポーネント管理
DNSAutoscalerの説明

最終更新日：：2023-05-06 19:41:07

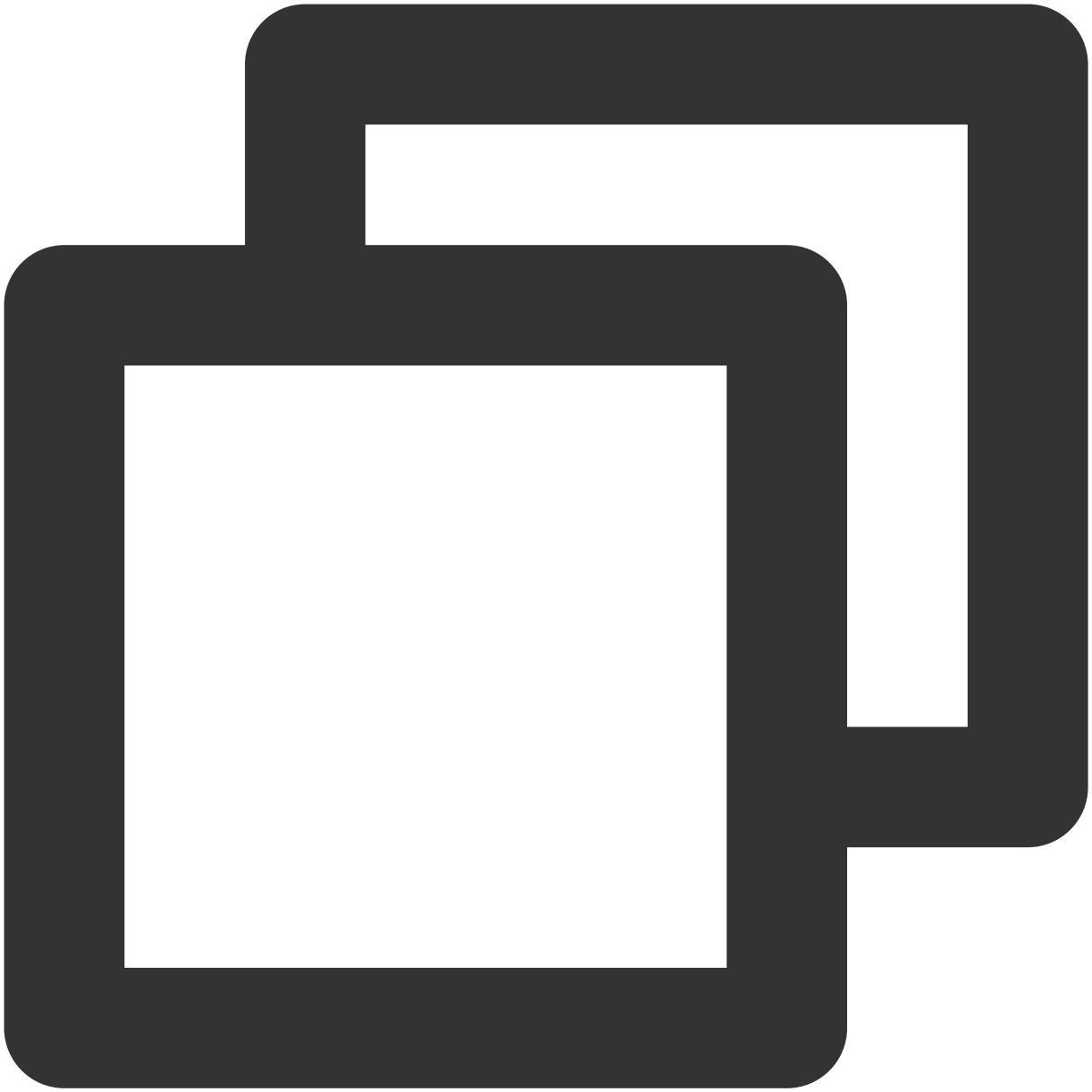
概要

コンポーネントの紹介

DNSAutoscalerはDNS自動水平スケーリングコンポーネントです。1つのdeploymentによってクラスターのノード数およびコア数を取得し、あらかじめ設定されたスケーリングポリシーに基づき、DNSのレプリカ数を自動水平スケーリングします。現在のスケーリングモードは2種類に分かれ、それぞれLinear線形モードおよびLadder段階モードです。

Linear Mode

ConfigMapの設定例は次のとおりです。



```
data:
  linear: |-
    {
      "coresPerReplica": 2,
      "nodesPerReplica": 1,
      "min": 1,
      "max": 100,
      "preventSinglePointFailure": true
    }
```

目標レプリカの計算公式：

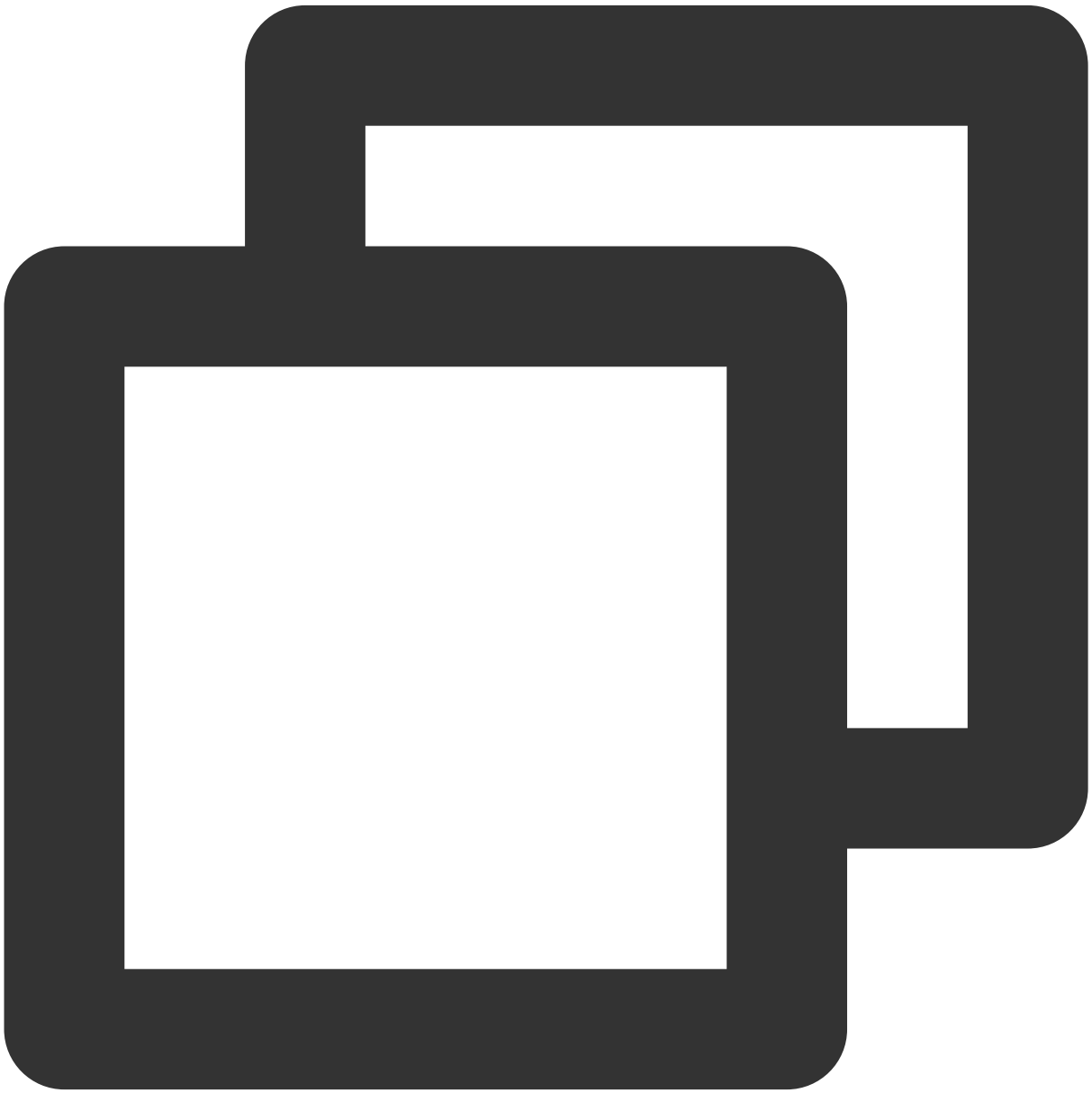
replicas = max(ceil(cores _ 1/coresPerReplica) , ceil(nodes _ 1/nodesPerReplica))

replicas = min(replicas, max)

replicas = max(replicas, min)

Ladder Mode

ConfigMapの設定例は次のとおりです。



```
data:
  ladder: |-
```

```
{
  "coresToReplicas":
  [
    [ 1, 1 ],
    [ 64, 3 ],
    [ 512, 5 ],
    [ 1024, 7 ],
    [ 2048, 10 ],
    [ 4096, 15 ]
  ],
  "nodesToReplicas":
  [
    [ 1, 1 ],
    [ 2, 2 ]
  ]
}
```

目標レプリカの計算：

100nodes/400coresのクラスター内で、上記の設定に従い、nodesToReplicasが2（100>2)、coresToReplicasが3（64<400<512）と仮定すると、両者の大きい方の値は3であり、最終replicaは3です。

クラスター内にデプロイされたKubernetesオブジェクト

kubernetsオブジェクト名	タイプ	リソースリクエスト	所属Namespace
tke-dns-autoscaler	Deployment	各ノード20mCPU，10Miのメモリ	kube-system
dns-autoscaler	ConfigMap	-	kube-system
tke-dns-autoscale	ServiceAccount	-	kube-system
tke-dns-autoscaler	ClusterRole	-	kube-system
tke-dns-autoscaler	ClusterRoleBinding	-	kube-system

制限条件

バージョン1.8以上のkubernetesクラスターのみサポートしています。

クラスター内のdns serverのワークロードはdeployment/corednsです。

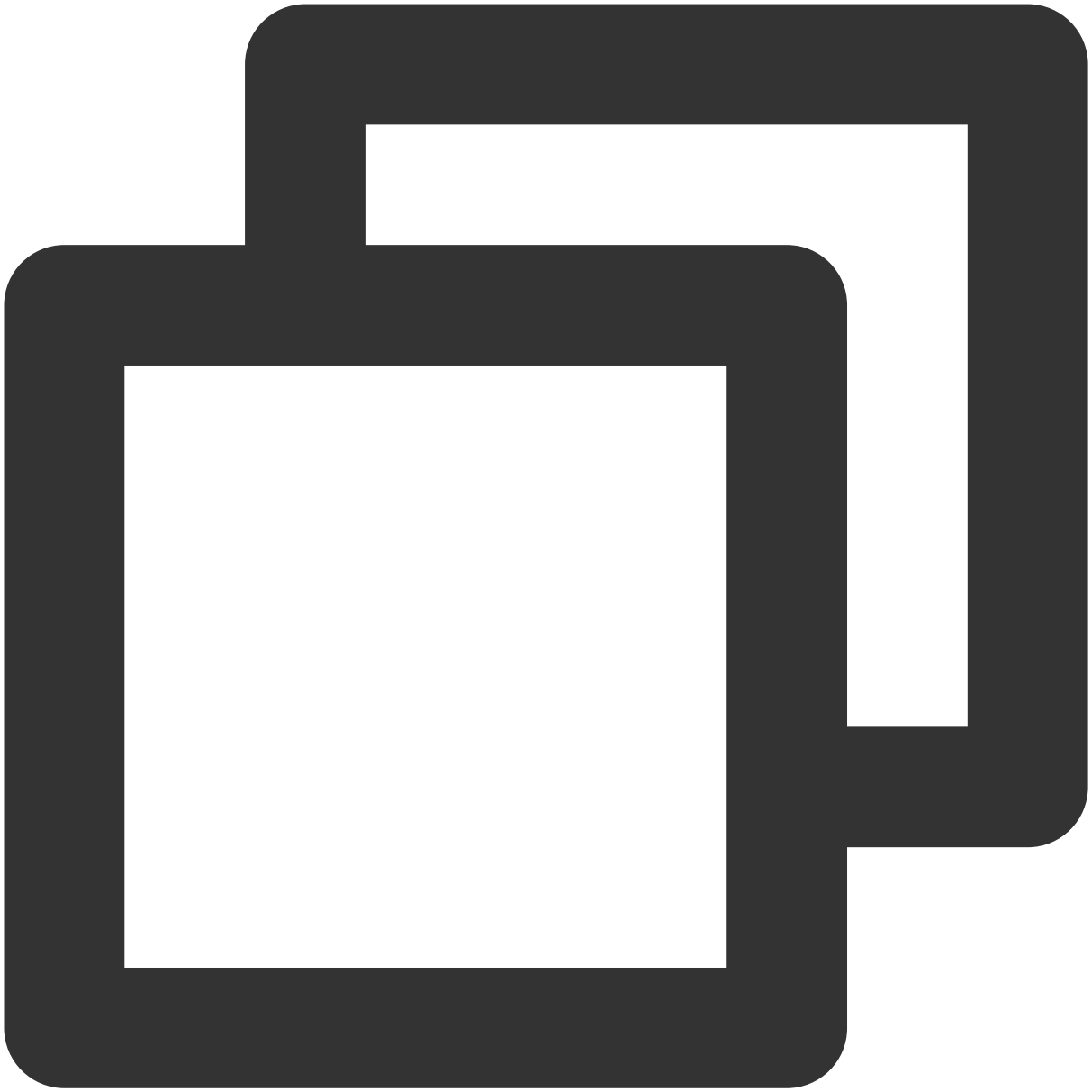
特記事項

CoreDNSの水平スケーリングによって一部のレプリカが一定時間内使用できなくなる可能性があるため、このコンポーネントをインストールする前に、関連する最適化設定を行い、DNSサービスの可用性を最大限保証することを強くお勧めします。具体的にはCoreDNSのスムーズアップグレードを設定するをご参照ください。

利用方法

1. TKEコンソールにログインし、左側ナビゲーションバーで**クラスター**を選択します
2. 「クラスター管理」ページで目標のクラスターIDをクリックし、クラスター詳細ページに進みます。
3. 左側メニューバーの**コンポーネント管理**を選択し、「コンポーネントリスト」ページに進みます。
4. 「コンポーネントリスト」ページで**新規作成**を選択し、「コンポーネントの新規作成」ページでDNSAutoscalerにチェックを入れます。

このコンポーネントのデフォルトのスケーリング設定ポリシーは次のとおりです。



```
data:
  ladder: |-
 {
   "coresToReplicas":
   [
     [ 1, 1 ],
     [ 128, 3 ],
     [ 512,4 ],
   ],
   "nodesToReplicas":
   [
```

```
      [ 1, 1 ],
      [ 2, 2 ]
    ]
  }
```

コンポーネント作成の拡張に成功すると、kube-systemネームスペース下の `configmap/tke-dns-autoscaler` を変更することによって設定を変更することができます。詳細な設定については、公式ドキュメントをご参照ください。

5. **完了**をクリックすれば、コンポーネントの作成は完了です。

DeSchedulerの説明

最終更新日：：2023-05-06 20:06:32

お知らせ

Tencent Cloudネイティブ監視TPSは、2022年5月16日よりすでにオフラインとなっています。詳しくは お知らせ をご参照ください。新しいPrometheusサービスはTMP よりご提供します。

DeScheduler以前にTPSをデータソースとし、かつ調整していない場合、 スケジューラは失効します。TMPをデータソースにする場合は、TMPがインターフェースの認証機能を追加するため、スケジューラをアップグレードしないとTMPインスタンスにバインドできません。

DeSchedulerが使用しているのが自作のPrometheusサービスの場合は、TPSオフラインでもコンポーネントには影響はありません。ですが自作のPrometheusの安定性と信頼性については、ご自身で保証する必要があります。

概要

コンポーネントの説明

DeSchedulerとは、TKEがKubernetesネイティブコミュニティ DeScheduler に基づいて実現した、1つのNodeのリアルロードについて再スケジューリングを行ったプラグインのことです。 TKEクラスターにプラグインをインストールすると、そのプラグインはKube-schedulerとの連携が有効化されます。クラスターの中の高負荷ノードをリアルタイムで監視し、低優先度Podをドレインします。TKE Dynamic Scheduler（ダイナミックスケジューラ拡張コンポーネント）とあわせて使用し、多次元でのクラスターCLBを保障されることを推奨します。このプラグインはPrometheus監視コンポーネントおよび関連するルール設定に依存します。プラグインが正しく動作するよう、インストール前に、 依存デプロイに詳しく目を通すことをお勧めします。

デプロイのクラスター内でのKubernetesオブジェクト

Kubernetesオブジェクト名	タイプ	リソースリクエスト	所属Namespace
descheduler	Deployment	インスタンスあたりCPU:200m, Memory:200Mi，合計1つのインスタンス	kube-system
descheduler	ClusterRole	-	kube-system
descheduler	ClusterRoleBinding	-	kube-system
descheduler	ServiceAccount	-	kube-system

descheduler-policy	ConfigMap	-	kube-system
probe-prometheus	ConfigMap	-	kube-system

シナリオ

DeSchedulerは再スケジューリングすることで、クラスター既存のノード上での不適切な実行形式を解決します。コミュニティバージョンDeSchedulerで提示されたポリシーは、ノードのリアルロードではなく、APIServer中のデータに基づいて実現します。そのため、ノードの監視を増やし、リアルロードに基づいた再スケジューリングの調整を行うことができます。
TKE自社開発のReduceHighLoadNodeポリシーはPrometheusおよびnode_exporterの監視データに依存しています。ノード、CPU使用率、メモリ使用率、ネットワークIO、system loadavgなどのインジケータに基づいてPodドレイン再スケジューリングを行い、ノードに極端な負荷がかかる状況を防止します。DeSchedulerのReduceHighLoadNodeおよびTKE自社開発のDynamic Schedulerは、ノードのリアルロードに基づいてスケジューリングポリシーを行い、連携して利用する必要があります。

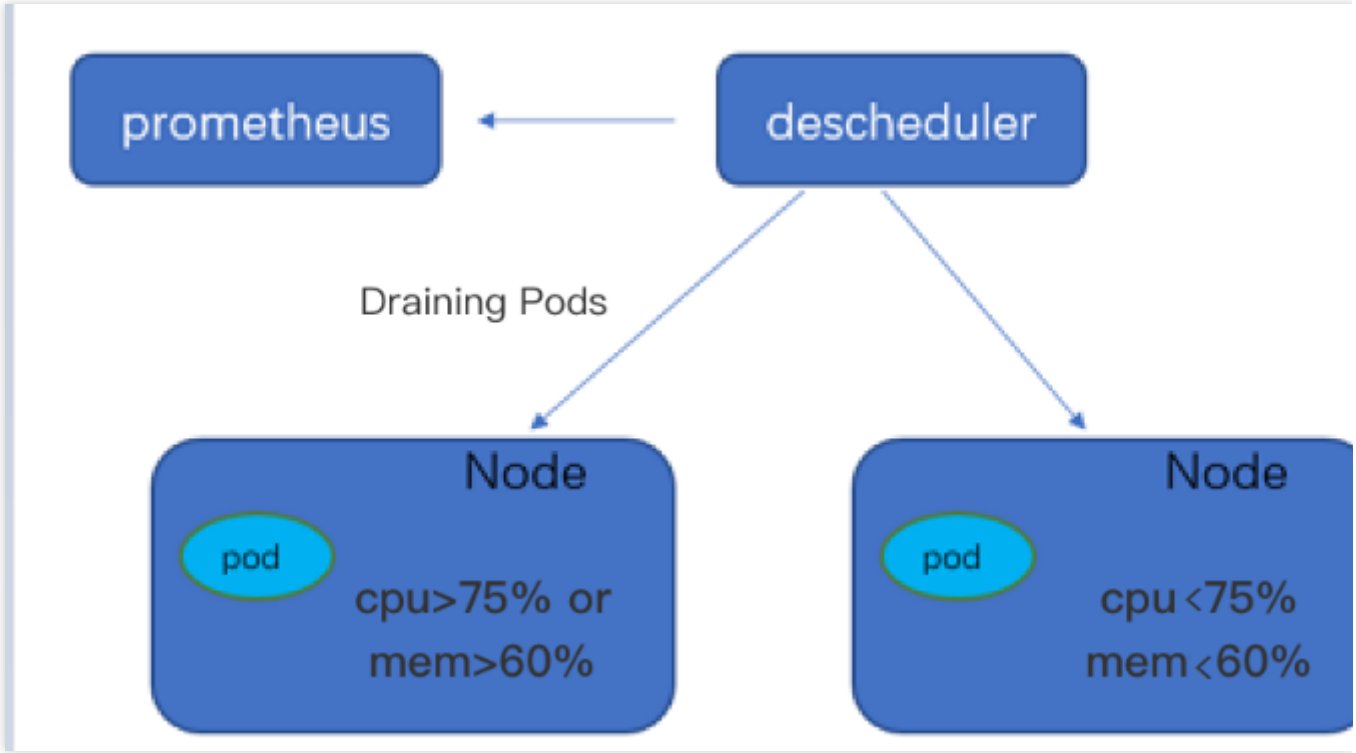
注意事項

Kubernetesバージョン ≥ v1.10.x
特定のシナリオでは、一部のPodは再スケジューリングが必要なノード上に重複してスケジューリングされ、そのためPodが繰り返しドレインされることがあります。その場合は、実際のシナリオに基づいてPodのスケジューリングが可能なノードを変更するか、Podにドレイン不可のマーカーをつけることができます。
このコンポーネントはすでにTKEの監視アラート体系に接続しています。
より良好な、コンポーネント異常および障害特定のモニタリングのため、クラスター起動イベントを永続化させることをお勧めします。DeschedulerがPodをドレインする際には、相応のイベントが発生します。reasonを「Dscheduled」タイプのイベントとして観察することで、Podが重複してドレインされているかどうかを観察します。
DeSchedulerがキーとなるPodをドレインしてしまうのを防ぐため、設計されているアルゴリズムでは、Podをドレインしないがデフォルトとなっています。ドレインしてもよいPodについて、ユーザーは表示してPod所属のworkloadを判断する必要があります。例えば、statefulset、deploymentなどのオブジェクト設定ではannotationのドレインが可能です。
ドレインの前提：ドレインされるPodに使用するリソースがない状態を防ぐため、クラスターは少なくとも5つのノードを含んでいること。4つ以上のノードの負荷が**ターゲット利用率**より低い場合のみドレインが可能です。
ドレインは高リスクな操作です。ドレイン後にスケジューリングできるノードがないといった状況がおこらないように、ノードの親和性、テイント関連の設定、また、Pod自体のノードへのリクエストの選択にご注意ください。

大量のPodをドレインすると、サービスが利用できなくなります。KubernetesネイティブはPDBオブジェクトを提供し、ドレインインターフェースが引き起こすworkloadで使用できないPodが多くなりすぎることを防ぎます。ただしユーザーはそのPDBの設定の作成が必要です。TKE自社開発のDeSchedulerコンポーネントには包括対策が追加されており、ドレインインターフェース呼び出し前に、workloadが準備しているPodの数がレプリカ数の半分を超えているかどうかを判断し、超えない場合はドレインインターフェースを呼び出しません。

コンポーネント原理

DeSchedulerは、 コミュニティバージョンDescheduler に基づいた再スケジューリングの考え方です。定期的に各ノード上の実行Podをスキャンし、ポリシー条件に合致していないものをドレインし、再スケジューリングします。コミュニティバージョンDeSchedulerではすでに一部のポリシーを提供しています。ポリシーはAPIServer中のデータに基づきます。例えば、 `LowNodeUtilization` ポリシーが依存しているのはPodのrequestとlimitデータです。このタイプのデータは、有効でバランスの良いクラスターリソースの分配、リソースフラグメントの防止が可能です。ですがコミュニティポリシーにはノードのリアルリソース占有についてのサポートが欠けています。例えば、ノードAとBが割り当てるリソースが一致している場合、Podの実際の実行状況により、CPU消費型とメモリ消費型が異なり、ピークとオフピーク時が異なると、2つのノードの負荷の差は非常に大きくなります。そこで、Tencent Cloud TKEはDeSchedulerをリリースしました。その基盤になった依頼は、ノードに対するリアルロード監視の再スケジューリングです。Prometheusを介してクラスターNodeの負荷統計情報を入手し、ユーザー設定の負荷閾値に基づいて定期的にポリシー内の検査ルールを実行し、高負荷ノード上のPodをドレインします。



コンポーネントパラメータの説明

Prometheusデータのアドレス照会

注意

コンポーネントが必要な監視データ、スケジューリングポリシーの有効化のプルを確保するため、依存デプロイ > **Prometheusファイル設定**ステップに従い、モニタリングデータ収集ルールを設定してください。

自作のPrometheusを使用する場合は、直接データを照会URL（HTTPS/HTTPS）に入力すれば完了です。

ホスティングPrometheusを使用する場合は、ホスティングインスタンスIDを選択すれば完了です。システムは自動でインスタンスが対応するデータ照会URLを解析します。

使用率の閾値およびターゲット使用率

注意

負荷閾値パラメータはすでにデフォルト値を設定していますので、追加要求がない場合は、直接採用できます。

過去5分以内に、ノードのCPU平均使用率あるいはメモリ平均使用率が設定閾値を超えると、Deschedulerはノードが高負荷ノードになったと判断してPodドレインロジックを実行し、Podの再スケジューリングによって、ノード負荷を可能な限りターゲット利用率以下まで下げます。

操作手順

依存デプロイ

DeSchedulerコンポーネントはNodeの現在と過去の一定時間のリアルロード状況に依存してスケジューリング決定を行うため、Prometheusなどの監視コンポーネントを介してシステムNodeのリアルロード情報を取得する必要があります。DeSchedulerコンポーネントを利用する前に、自作Prometheusの監視か、TKEクラウドネイティブの監視かの採用ができます。

自作Prometheusのモニタリングサービス

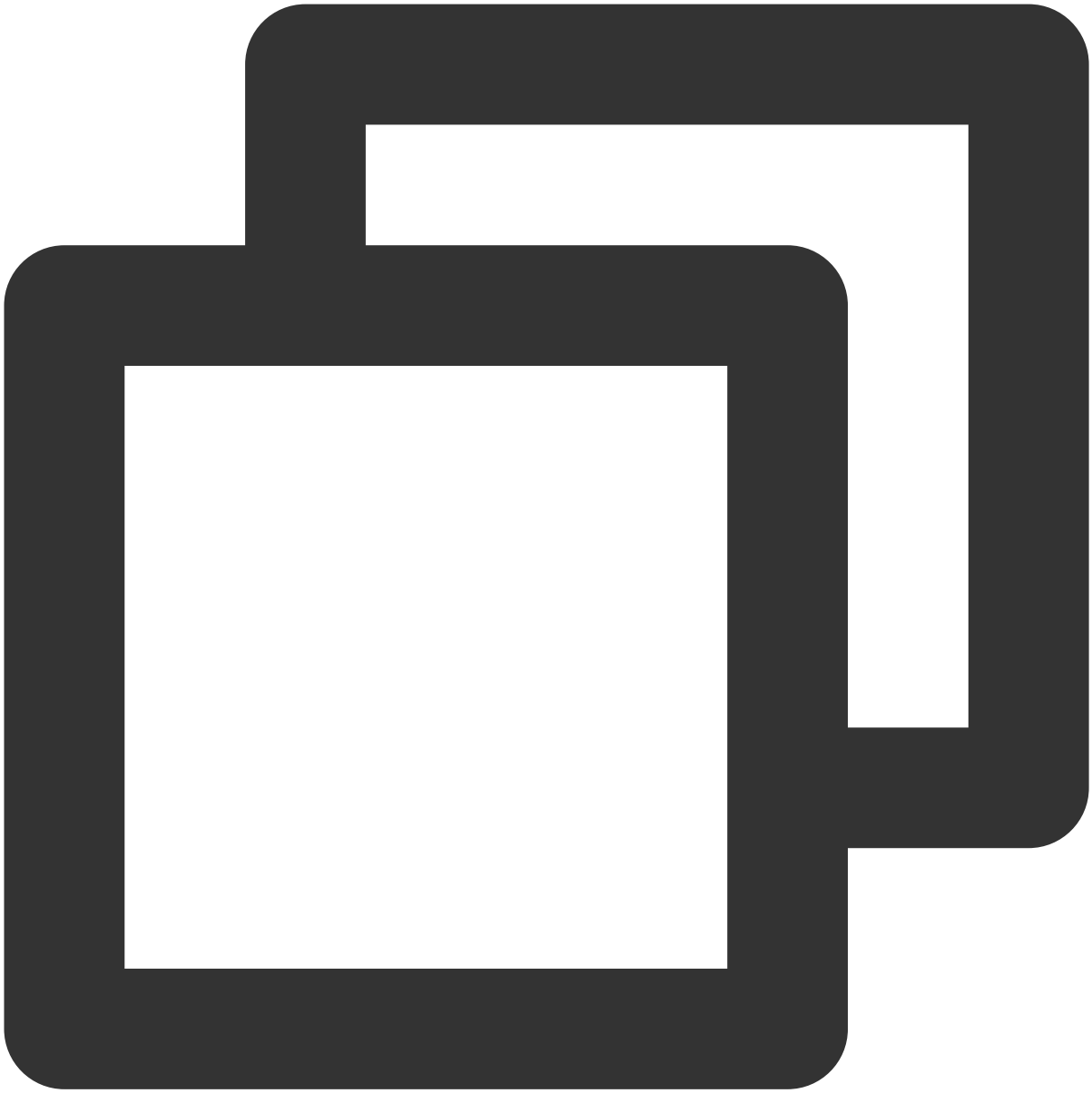
Managed Service for Prometheus

デプロイnode-exporterおよびPrometheus

node-exporterを介してNodeインジケータの監視を実現します。必要に応じてnode-exporterおよびPrometheusをデプロイできます。

集計ルールの設定

node-exporterでノードの監視データを取得した後に、Prometheusを介してオリジナルのnode-exporterの中からデータを収集し、集計を行う必要があります。 DeSchedulerで必要な `cpu_usage_avg_5m` 、 `mem_usage_avg_5m` などのインジケータを取得するためには、Prometheusのrulesの中で設定が必要です。例は次のとおりです。

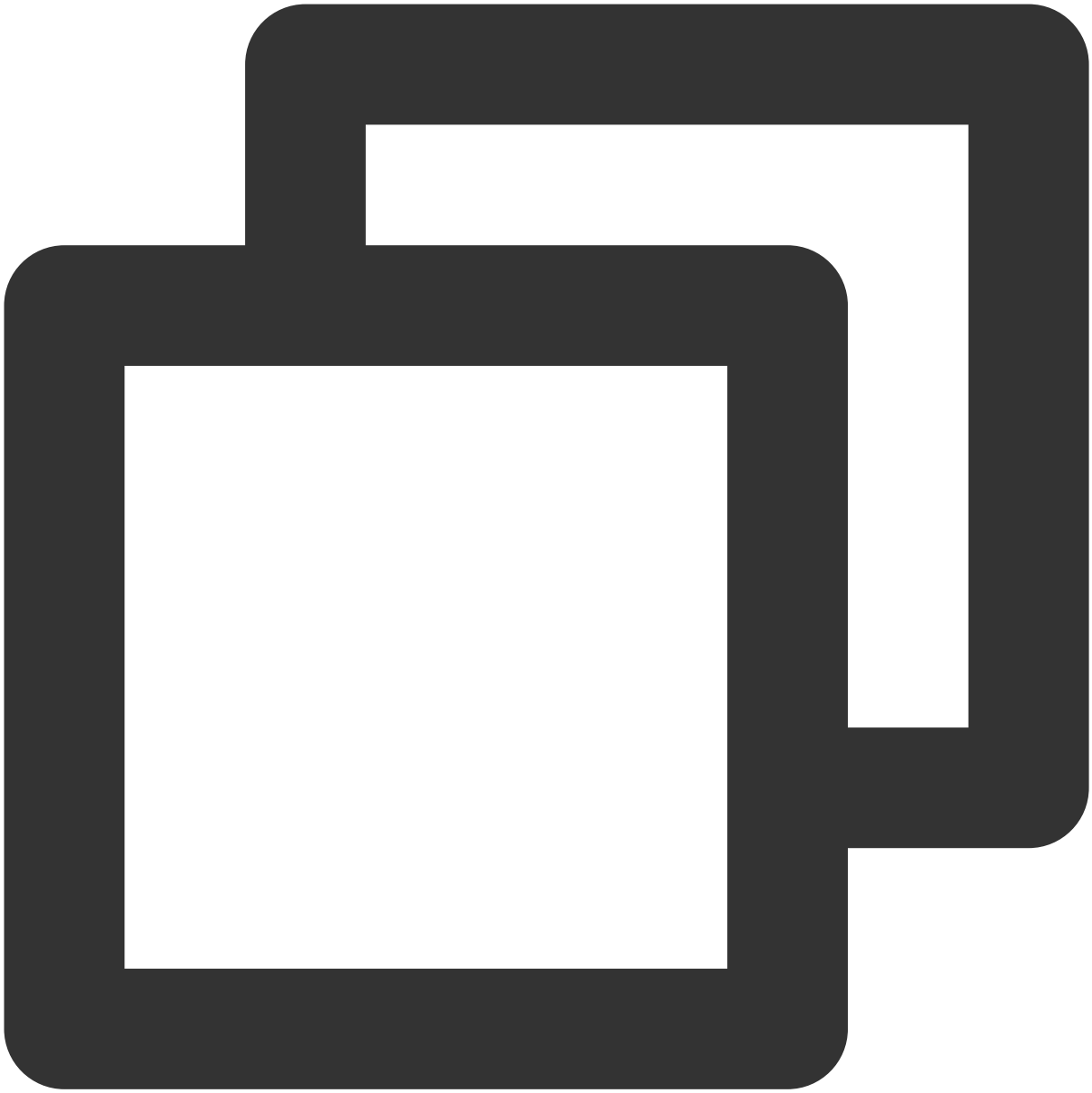


```
groups:
   - name: cpu_mem_usage_active
     interval: 30s
     rules:
     - record: mem_usage_active
       expr: 100*(1-node_memory_MemAvailable_bytes/node_memory_MemTotal_bytes)
   - name: cpu-usage-1m
     interval: 1m
     rules:
     - record: cpu_usage_avg_5m
       expr: 100 - (avg by (instance) (irate(node_cpu_seconds_total{mode="idle"}[5m
```

```
    - name: mem-usage-1m
      interval: 1m
      rules:
      - record: mem_usage_avg_5m
        expr: avg_over_time(mem_usage_active[5m])
```

注意

TKEが提供するDynamicSchedulerを利用時には、Prometheus設定でのNode監視データの集計ルールの取得が必要となります。DynamicScheduler集計ルールとDeScheduler集計ルールは一部重複していますが、完全に一致しているわけではありません。ルールを設定する際には、相互に上書きをしないでください。DynamicSchedulerとDeSchedulerを同時に利用する際には、次のルールを設定しなければなりません。

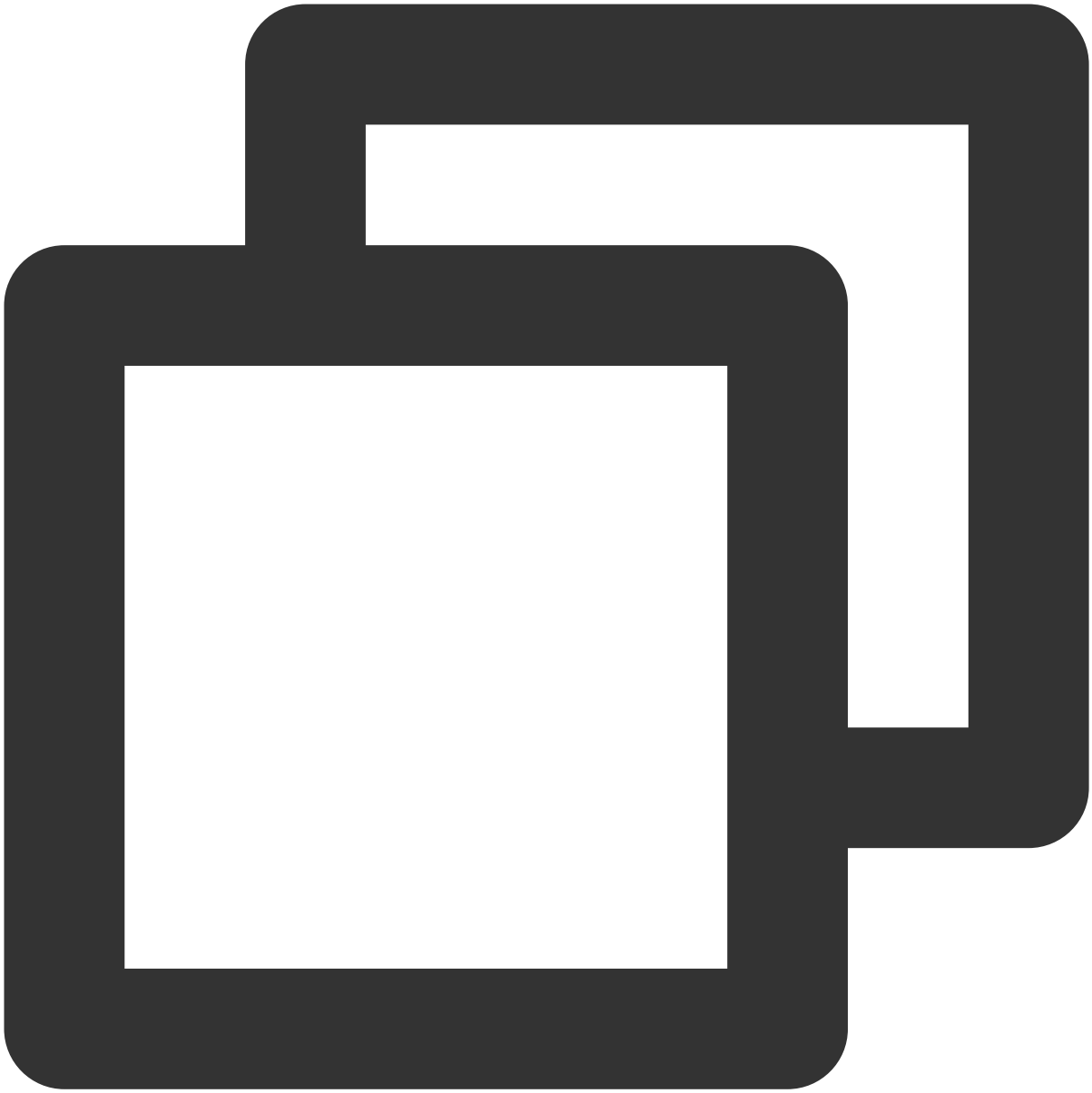


```
groups:
   - name: cpu_mem_usage_active
     interval: 30s
     rules:
     - record: mem_usage_active
       expr: 100*(1-node_memory_MemAvailable_bytes/node_memory_MemTotal_bytes)
   - name: mem-usage-1m
     interval: 1m
     rules:
     - record: mem_usage_avg_5m
       expr: avg_over_time(mem_usage_active[5m])
```

```
  - name: mem-usage-5m
    interval: 5m
    rules:
    - record: mem_usage_max_avg_1h
      expr: max_over_time(mem_usage_avg_5m[1h])
    - record: mem_usage_max_avg_1d
      expr: max_over_time(mem_usage_avg_5m[1d])
  - name: cpu-usage-1m
    interval: 1m
    rules:
    - record: cpu_usage_avg_5m
      expr: 100 - (avg by (instance) (irate(node_cpu_seconds_total{mode="idle"}[5m
  - name: cpu-usage-5m
    interval: 5m
    rules:
    - record: cpu_usage_max_avg_1h
      expr: max_over_time(cpu_usage_avg_5m[1h])
    - record: cpu_usage_max_avg_1d
      expr: max_over_time(cpu_usage_avg_5m[1d])
```

Prometheusファイルの設定

1. 上記にてDeSchedulerが必要なインジケータ計算のrulesを定義しました。rulesをPrometheusに設定するには、一般的なPrometheus設定ファイルをご参照ください。例は以下のとおりです。



```
global:
   evaluation_interval: 30s
   scrape_interval: 30s
   external_labels:
rule_files:
- /etc/prometheus/rules/*.yml # /etc/prometheus/rules/*.yml は、定義したrulesファイル
```

2. rules設定を1つのファイル（de-scheduler.yamlなど）にコピーします。ファイルは上記Prometheusコンテナの `/etc/prometheus/rules/` の下に置きます。

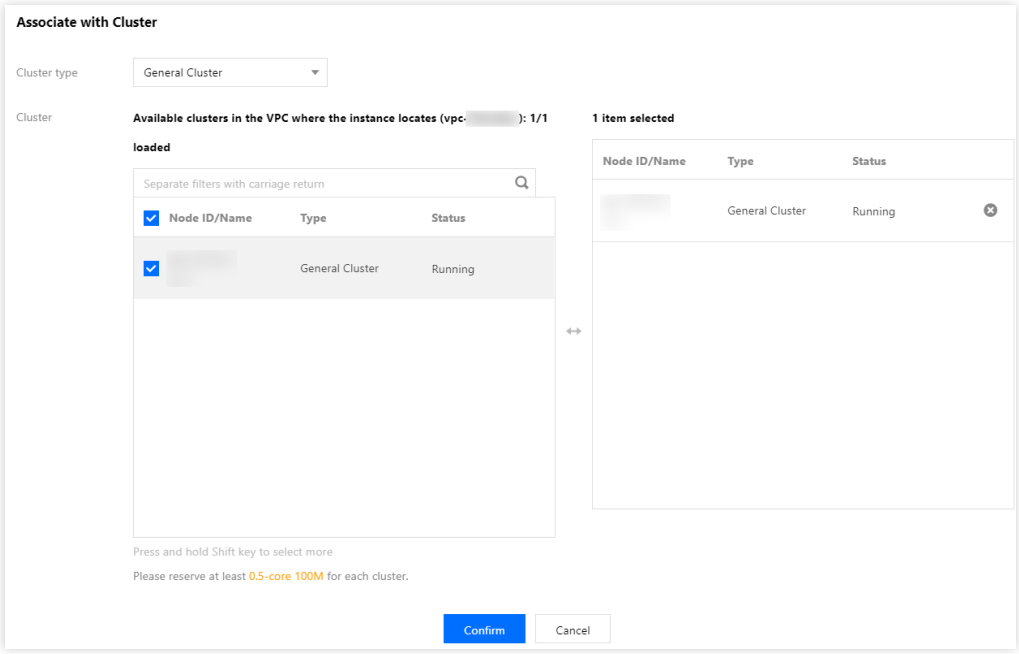
3. Prometheus serverを再ロードすると、Prometheusからダイナミックスケジューラが必要なインジケータを取得できます。

説明

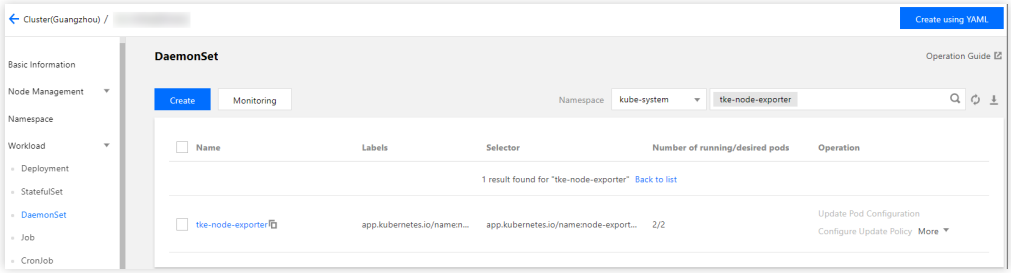
通常、上記Prometheus設定ファイルとrules設定ファイルは、どちらもconfigmapを介してストレージされてからPrometheus serverコンテナにマウントされますので、対応するconfigmapを変更すれば完了です。

1. TKEコンソールにログインし、左側メニュー欄から Prometheus監視を選択し、「Prometheus監視」ページに進みます。

2. Clusterと同じVPC下にある Prometheusインスタンスを作成し、 クラスターをバインドします。下図に示すとおりです：



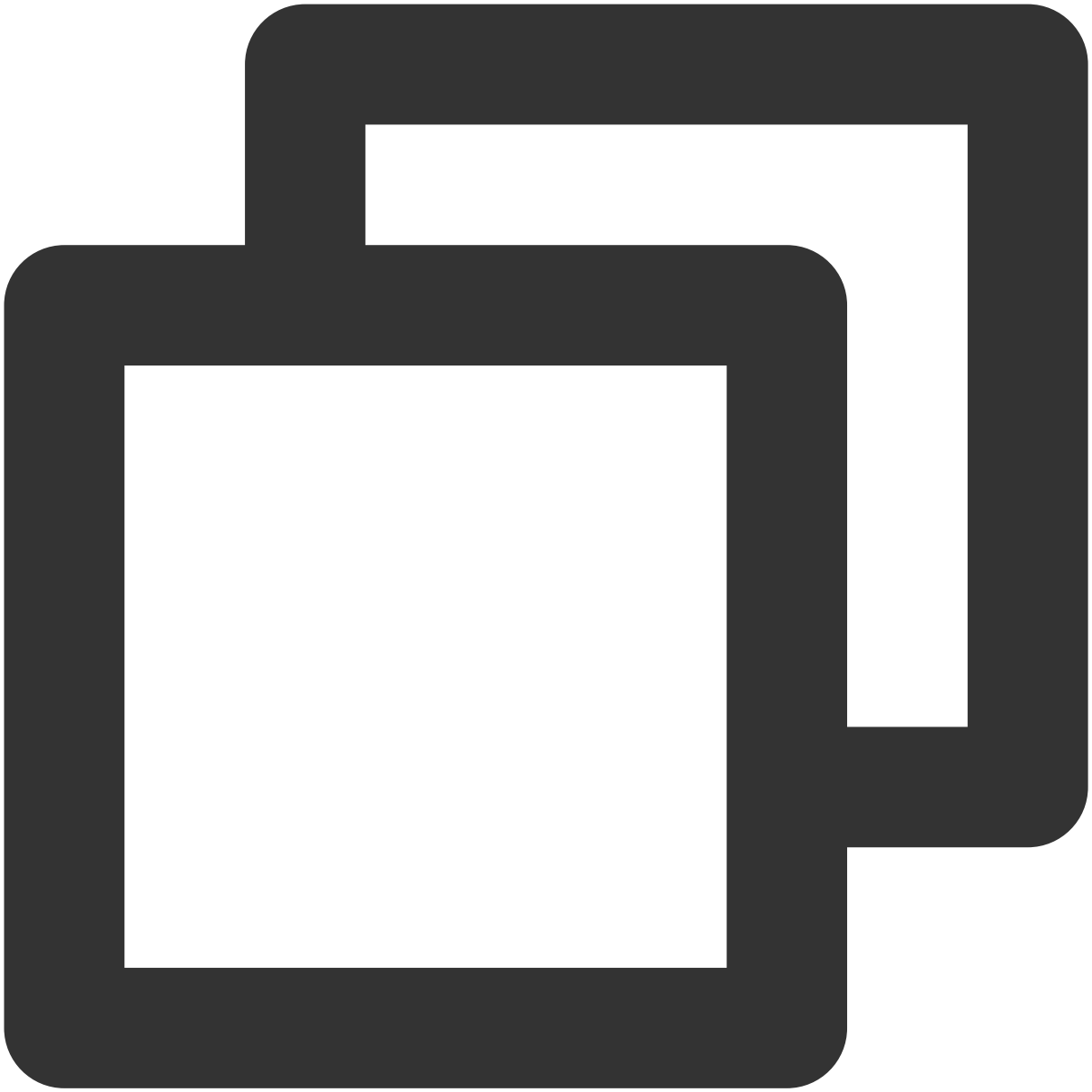
3. ネイティブホスティングクラスターとバインドすれば、ユーザークラスターにて各ノードがすべてnode-exporterをインストールしたことを確認できます。下図に示すとおりです：



4. Prometheus集計ルールを設定します。具体的なルールの内容は上述の自作Prometheusの監視サービス の「ルール設定の集計」と同じです。ルールの保存後すぐに有効化されますので、serverの再ロードは不要です。

コンポーネントのインストール

1. TKEコンソールにログインし、左側ナビゲーションバーの**クラスター**を選択します。
2. **クラスター管理**ページでターゲットのクラスターIDをクリックし、クラスター詳細ページに進みます。
3. 左側ナビゲーションバーで**コンポーネント管理**を選択し、**コンポーネントリスト**ページで**新規作成**をクリックします。
4. **コンポーネントの新規作成**ページでDecheduler（再スケジューラ）にチェックを入れて、**パラメータ設定**をクリックし、 パラメータの説明 に従って、コンポーネントに必要なパラメータを入力します。
5. **完了**をクリックすればコンポーネントの作成は完了です。インストール成功後、DeSchedulerはすぐに正常に運用されますので、 追加の設定は不要です。
6. workload（例えばstatefulset、deploymentなどのオブジェクト）をドレインする場合は、 Annotationを次のように設定できます。



```
descheduler.alpha.kubernetes.io/evictable: 'true'
```

アプリケーション管理
アプリケーション管理

最終更新日：：2023-07-06 11:04:05

このドキュメントでは、Tencent Kubernetes Engine（TKE）コンソールを使用してアプリケーションを作成、更新、ロールバック、および削除する方法について説明します。

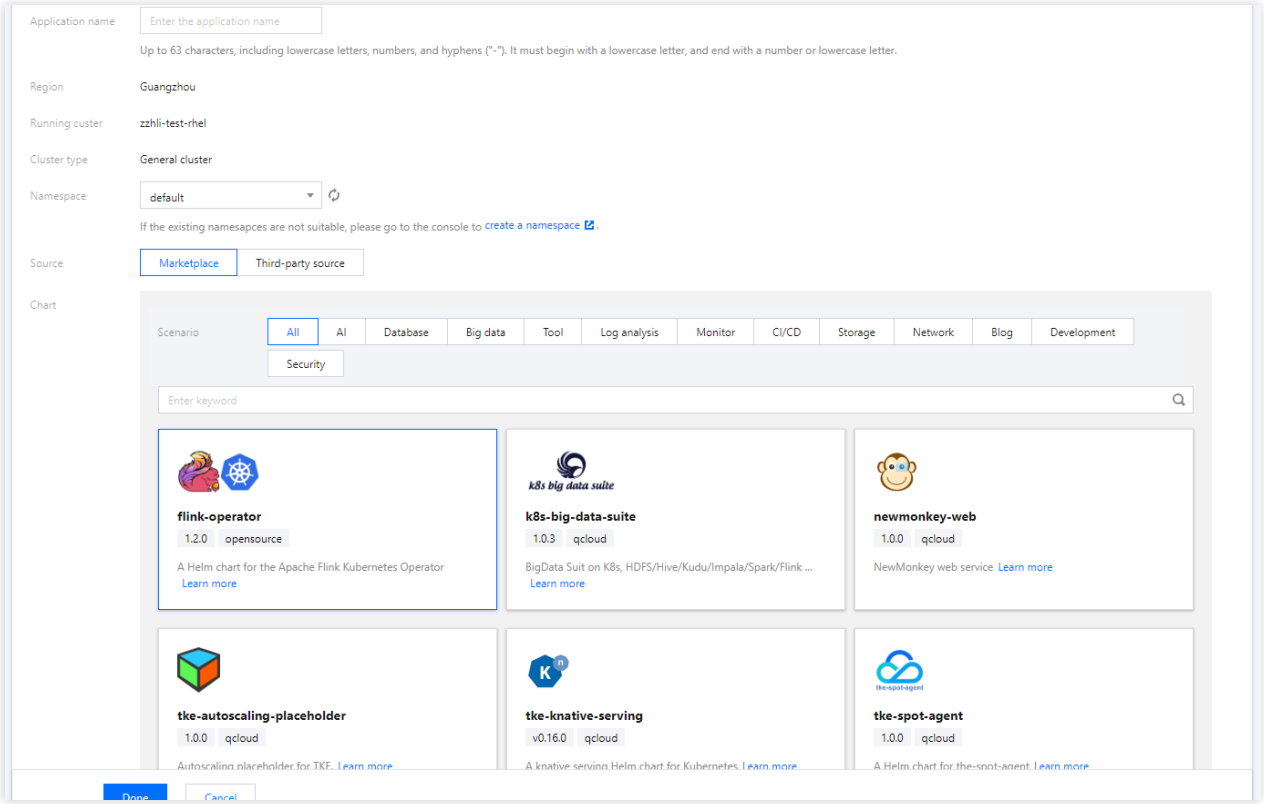
説明事項

アプリケーション管理は、Kubernetesバージョン1.8以降のクラスターのみをサポートします。

操作手順

アプリケーションの作成

1. TKEコンソールにログインし、左側ナビゲーションバーのアプリケーションを選択します。
2. **アプリケーションリスト**ページの上方で、作成するアプリケーションのクラスターおよびリージョンを選択し、**新規作成**をクリックします。
3. **アプリケーションの新規作成**ページで、以下の情報を参照してアプリケーションの基本情報を設定します。下図に示すとおりです。



主なパラメータ情報は下記の通りです：

アプリケーション名：アプリケーション名をカスタマイズします。

ソース：**アプリ市場** および**サードパーティーソース**を選択できます。説明は下表のように示されます。

ソース	設定項目
アプリ市場	クラスタータイプ、ユースケースに基づいて、チャートフィルタリングを実行します。適合するアプリケーションパッケージとChartバージョンを選択し、パラメータを編集します。
サードパーティーソース	Chartアドレス：Helm公式または自作Helm Repoリポジトリをサポートしています。必ずhttpで始まり、.tgzで終わるパラメータを設定してください。ここではhttp://139.199.162.50/test/nginx-0.1.0.tgzを例にとります。 タイプ：**パブリック**および**プライベート**の2つのタイプを提供しています。必要に応じて選択してください。 パラメータ：必要に応じてパラメータを編集します。

4. **完了**をクリックすれば、作成完了です。

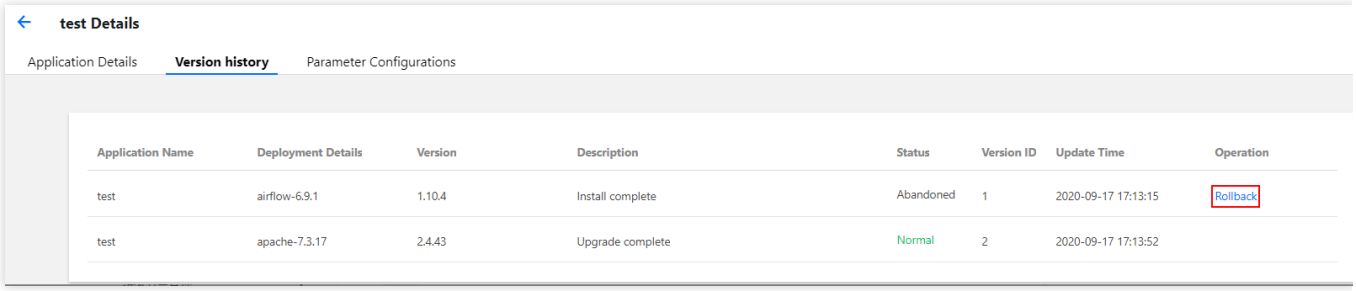
アプリケーションの更新

1. TKEコンソールに進み、左側ナビゲーションバーから**アプリケーション**を選択します。
2. **アプリケーション**ページで、更新するアプリケーションが存在する行の右側の**アプリケーションの更新**を選択します。

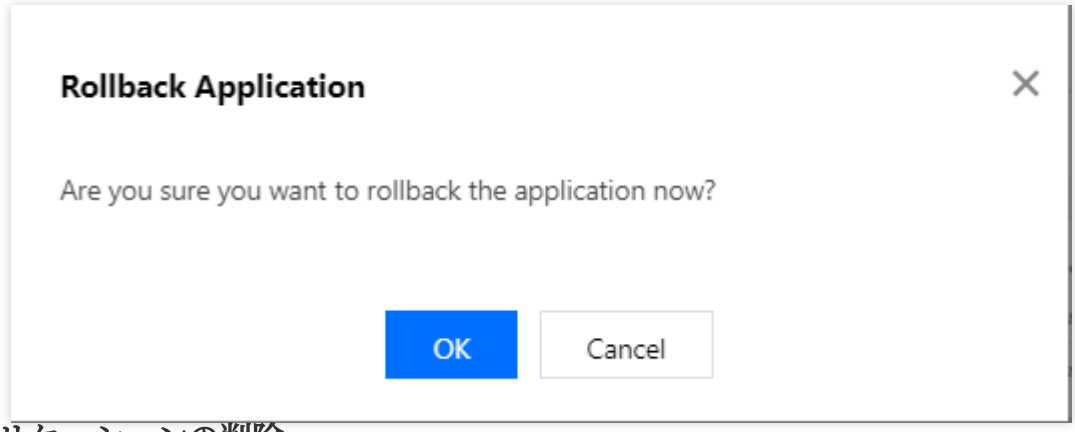
3. **アプリケーションの更新**ページで、必要に応じてキー情報の設定を行い、**完了**をクリックします。

アプリケーションのロールバック

1. TKEコンソールに進み、左側ナビゲーションバーから**アプリケーション**を選択します。
2. **アプリケーション**ページで、更新するアプリケーションを選択し、そのアプリケーションの詳細ページに進みます。
3. アプリケーション詳細ページで、**バージョン履歴**タブを選択し、ロールバックするバージョンが存在する行の右側の**ロールバック**をクリックします。下図に示すとおりです。



4. **アプリケーションのロールバック**ページで、**OK**をクリックすれば完了です。下図に示すとおりです。



アプリケーションの削除

1. アプリケーションコンソールに進み、左側ナビゲーションバーから**アプリケーション**を選択します。
2. **アプリケーション**ページで、削除するアプリケーションが存在する行の右側の**削除**を選択します。
3. **アプリケーションの削除**ページで、**OK**をクリックすれば完了です。

ネットワーク管理
コンテナネットワークの概要

最終更新日：：2023-04-26 18:43:22

コンテナネットワークとクラスターネットワークの説明

コンテナネットワークとクラスターネットワークとは、クラスターの基本属性です。クラスターネットワークとコンテナネットワークを設定することによって、クラスターのネットワーク区画をプランニングできます。

コンテナネットワークとクラスターネットワークとの関係

クラスターネットワーク：クラスター内のホストにノードネットワークアドレス範囲内のIPアドレスを割り当てるために、Virtual Private Cloud(VPC)のサブネットを選択してクラスターのノードネットワークに使用することができます。VPCの詳細については、VPCの概要をご参照ください。
コンテナネットワーク：コンテナネットワークアドレス範囲内のIPアドレスをクラスター内コンテナに割り当てます。GlobalRouterモード、VPC-CNIモード、Cilium-Overlayモードなどが含まれます。
GlobalRouterモード：3つのプライベートセグメントをコンテナネットワークとしてカスタマイズできます。選択したクラスター内のサービス数の上限に応じて、適切なサイズのCIDRセグメントがKubernetes serviceに自動的に割り当てられます。また、選択したノードあたりのPod数の上限に応じて、クラスター内の各CVMに適切なサイズのネットワークセグメントを自動的に割り当て、そのホストがPodに割り当てるIPアドレスとして使用できます。
VPC-CNIモード：コンテナ割り当て用のIPは、クラスターと同一のVPCを持つサブネットを選択します。
Cilium-Overlayモード：3つのプライベートセグメントをコンテナネットワークとしてカスタマイズできます。選択したクラスター内のサービス数の上限に応じて、適切なサイズのCIDRセグメントがKubernetes serviceに自動的に割り当てられます。また、選択したノードあたりのPod数の上限に応じて、クラスター内の各サーバーに適切なサイズのネットワークセグメントを自動的に割り当て、そのホストがPodに割り当てるIPアドレスとして使用できます。

コンテナネットワークとクラスターネットワークの制限

コンテナネットワークとクラスターネットワークセグメントは重複できません。
同一VPC内では、異なるクラスターのコンテナネットワークセグメントは重複できません。
コンテナネットワークとVPCルーティングが重複する場合は、コンテナネットワーク内の転送を優先します。

クラスターネットワークとTencent Cloudのその他リソース間での通信

クラスター内コンテナとコンテナ間での相互通信です。
クラスター内コンテナはノードと直接、相互通信します。

クラスター内コンテナは、クラウドデータベース(CDB)TencentDB、TencentDB for Redis、CDB Memcachedなどのリソースと同一VPC配下のプライベートネットワークで相互通信します。

注意：

クラスターコンテナを同一のVPC配下の他のリソースに接続する場合、セキュリティグループがコンテナネットワークセグメントを開放しているかどうかしっかり確認してください。

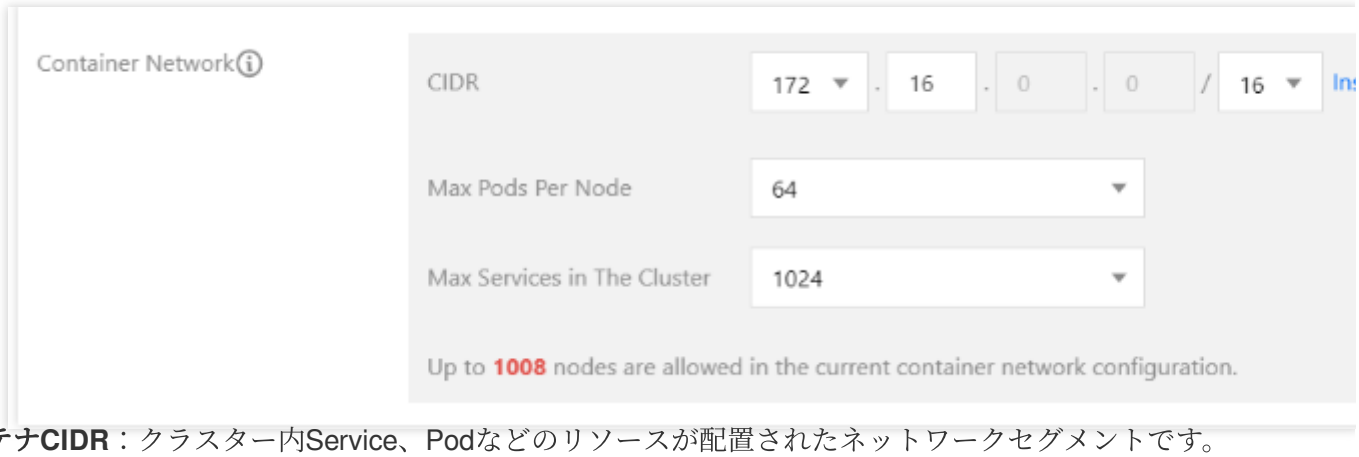
Tencent Kubernetes Engine(TKE)クラスターのip-masqコンポーネントは、コンテナがSNAT経由でクラスターネットワークとVPCネットワークにアクセスできないようにします。他のネットワークセグメントは影響を受けないため、コンテナは同一のVPC配下のその他リソース（Redisなど）にアクセスする際にコンテナセグメントを開放する必要があります。

同一リージョンクラスター間相互通信の設定ができます。

クロスリージョンにおけるクラスター間相互通信の設定ができます。

コンテナクラスターとIDC間相互通信の設定ができます。

コンテナネットワークの説明



コンテナCIDR：クラスター内Service、Podなどのリソースが配置されたネットワークセグメントです。

1ノードあたりのPod数の上限：各Nodeに割り当てられるCIDRのサイズを決定します。

説明

TKEクラスターは、デフォルトで2つのkube-dnsのPodと1つのl7-lb-controllerのPodを作成します。

1つのNode上のPodには、ネットワーク番号、ブロードキャストアドレス、ゲートウェイアドレスという3つのアドレスが割り当てられないため、Nodeの最大のPod数=podMax - 3となります。

クラスター内Service数の上限：Serviceに割り当てるCIDRのサイズを決定します。

説明

TKEクラスターは、デフォルトでkubernetes、hpa-metrics-service、kube-dnsという3つのServiceを作成すると同時に、2つのブロードキャストアドレスとネットワーク番号があります。そのため、ユーザーが使えるServices数の上限/クラスターは、ServiceMax - 5となります。

ノード：クラスター内のWorkerノードです。

説明

ノード数の計算式は、（CIDR IP 数 - クラスター内Service数の上限）/ 1ノードあたりのPod数の上限となります。

コンテナネットワークモードを選択するには、どうすればいいですか。

TKEは、ユースケースに応じてさまざまなネットワークモードを提供しています。ここでは、GlobalRouter、VPC-CNI、Cilium-Overlayという3つのネットワークモードを詳しくご説明するとともに、この3つのユースケースやメリット、使用上の制限など、さまざまな観点から比較します。ビジネスニーズに合わせてお選びいただけます。

説明

ネットワークモードを選択してクラスターを作成した場合、その後ネットワークモードを変更することはできません。

GlobalRouterモード

GlobalRouterネットワークモードでは、TKEが基盤となるプライベートネットワーク(VPC)のグローバルルーティング機能をベースとして、コンテナネットワークとVPC間の相互アクセスのためのルーティングポリシーを実装しています。詳細については、GlobalRouterモードの説明をご参照ください。

VPC-CNIモード

VPC-CNIネットワークモードは、CNIとVPC Elastic Network Interface(ENI)をベースとして実装されたTKEのコンテナネットワーク機能であり、厳しいレイテンシ要件が求められるシナリオに適しています。このネットワークモードでは、コンテナとノードは同一のネットワークプレーンに分散されています。コンテナIPは、IPAMDコンポーネントによって割り当てられたENI IPとなります。詳細については、VPC-CNIモードの説明をご参照ください。

Cilium-Overlayモード

Cilium-Overlayネットワークモードは、Cilium VXLanをベースとして実装されたTKEのコンテナネットワークプラグインで、分散型クラウドのシナリオにおいて、サードパーティノードをTKEクラスターに追加したネットワーク管理です。詳細については、Cilium-Overlayモードの説明をご参照ください。

説明

Cilium-Overlayモードのパフォーマンス低下のため、このモードは分散型クラウドにおけるサードパーティノードのシナリオにのみ対応しており、クラウド上にしかノードがないシナリオには対応していません。

ネットワークモードの選択

ここでは、ユースケース、メリット、使用上の制限といったさまざまな観点から、TKEが提供するGlobalRouter、VPC-CNI、Cilium-Overlayという3つのネットワークモードを比較します。以下の内容を参照して、最適なネットワークモードをお選びください。

観点	GlobalRouter	VPC-CNI	Cilium-Overlay
ユースケース	一般的なコンテナ事業のシナリオです。	ネットワークのレイテンシ要件が高いシナリオです。	分散型クラウドにおけるサードパーティノードのシナリオ

	オフラインコンピューティング関連事業です。	従来のアーキテクチャはコンテナプラットフォームに移行し、固定IPを持つコンテナに依存するシナリオです。	にのみ対応しています。 クラウド上にしかノードがないシナリオには対応していません。
メリット	コンテナルーティングは直接VPCを経由して、コンテナとノードと同一のネットワークプレーンに分散されます。 コンテナネットワークセグメントはフレキブルに割り当て、コンテナIPセグメントはVPCの他のセグメントを占有することはありません。使用可能なIPリソースは豊富にあります。	ENIのコンテナネットワークはVPCサブネットに属し、VPC製品の管理範囲に含めることができます。 固定IP、Cloud Load Balancer(CLB)パススルーPodなどのユーザーシナリオに対応しています。 ネットワークパフォーマンスはGlobalRouterモードより優れています。	クラウドノードとサードパーティノードは、指定されたコンテナネットワークセグメントを共有します。 コンテナネットワークセグメントはフレキブルに割り当て、コンテナIPセグメントはVPCの他のセグメントを占有することはありません。使用可能なIPリソースは豊富にあります。
使用制限	専用線、Peering ConnectionおよびCloud Connect Networkといった相互通信シナリオには、追加の設定が必要です。 固定Pod IPには対応していません。	コンテナネットワークとノードネットワークは、同一のVPCに属し、IPアドレスのリソースは限られています。 ノード内コンテナ数は、ENIとENIに割り当て可能なIPの数によって制限されます。 固定IPモードは、Podのアベイラビリティーゾーンをまたいだスケジューリングには対応していません。	Cilium VXLanトンネルパッケージングプロトコルを使用すると、最大10%のパフォーマンス損失が生じます。 Pod IPはクラスター外からは直接アクセスできません。 IDCのサードパーティノードがクラウド上の APIServerとパブリックサービスにアクセスできるように、指定されたサブネットから2つのIPを取得して、プライベートネットワークCLBを作成する必要があります。 固定Pod IPには対応していません。
追加機能を搭載しています	Kubernetesの標準機能です。	TKEは固定Pod IPに対応しています。 コンテナネットワークは、VPCコンソールから制御します。 LBは直接Podに転送し、PodはソースIPを取得できます。	Kubernetesの標準機能です。

GlobalRouterモード
GlobalRouterモードのクラスターをCloud Connect Network(CCN)に登録する

最終更新日：：2023-04-28 11:08:19

CCNの説明

Cloud Connect Network(CCN)は、クラウド上のVirtual Private Cloud(VPC)間、VPCとローカルデータセンターを相互接続するためのサービスを提供します。VPCと専用ゲートウェイインスタンスをCCNに追加することで、シングルポイントアクセスやネットワーク全体のリソース相互運用が可能になり、シンプルでインテリジェント、安全でフレキシブルなハイブリッドクラウドやグローバル相互接続ネットワークを手軽に構築することができます。

概要

既存のコンテナクラスターをCCNに登録することによって、コンテナネットワークを管理下に置くことができます。コンテナネットワークの登録が完了すると、CCNでコンテナネットワークのネットワークセグメントルーティングを有効または無効にすることができ、コンテナクラスターとCCN内のリソースとの相互運用が可能になります。

注意

コンテナクラスターをCCNに登録すると、このネットワークセグメントのインスタンスがCCNの既存ルートと競合しない場合は有効になり、競合する場合はデフォルトで無効になります。

前提条件

クラスターが配置されているVPCは、すでにCCNに存在します。CCNに関する操作については、CCNの操作概要をご参照ください。

クラスターコンテナネットワークのネットワークセグメントが、CCNネットワーク内の他のリソースネットワークセグメントと競合しているかどうかを評価します。

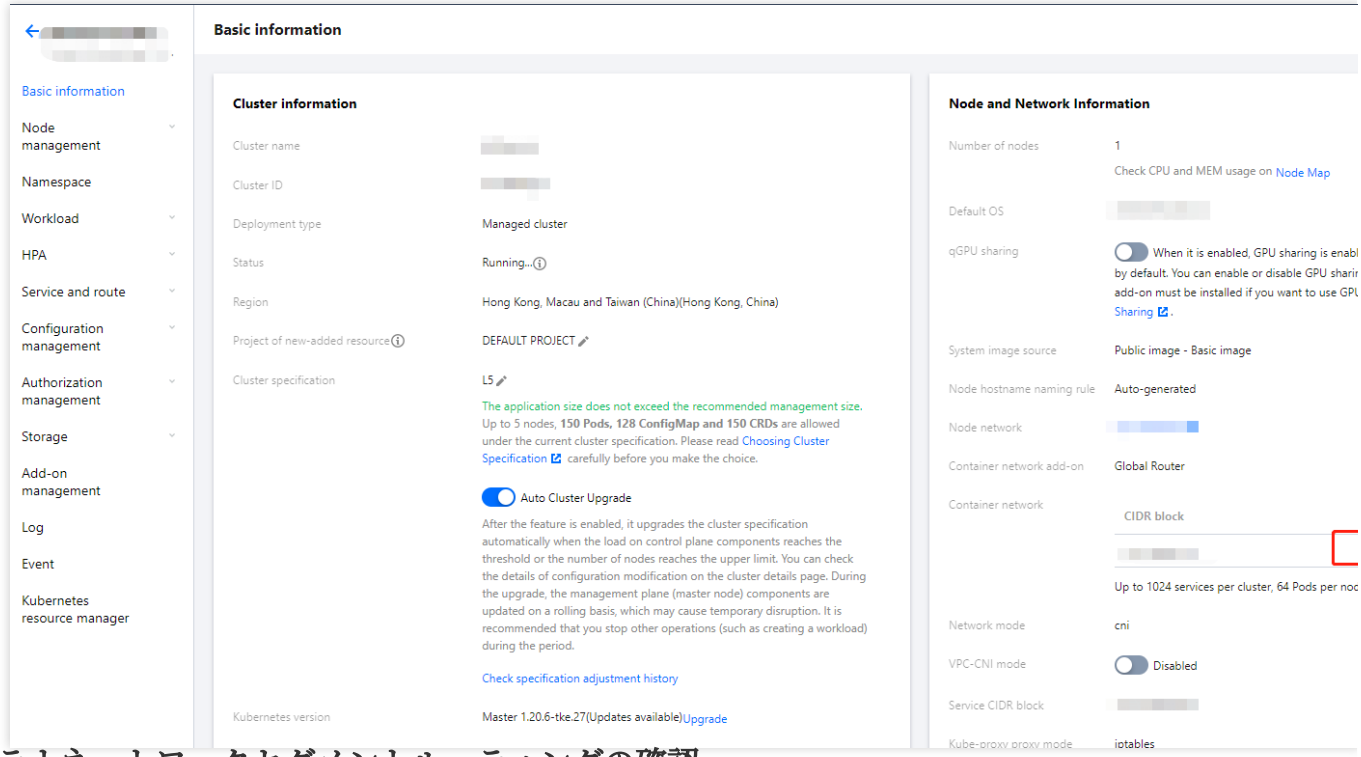
操作手順

対応するCCNへのコンテナネットワークの登録

1. TKEコンソールにログインし、左側ナビゲーションバーのクラスターをクリックしてクラスター管理ページに進みます。
2. CCNに登録する必要のあるクラスターIDを選択し、左側の*基本情報**をクリックしてクラスター基本情報ページに進みます。
3. CCNの登録スイッチをクリックして、コンテナネットワークをCCNに登録します。下図に示すとおりです。

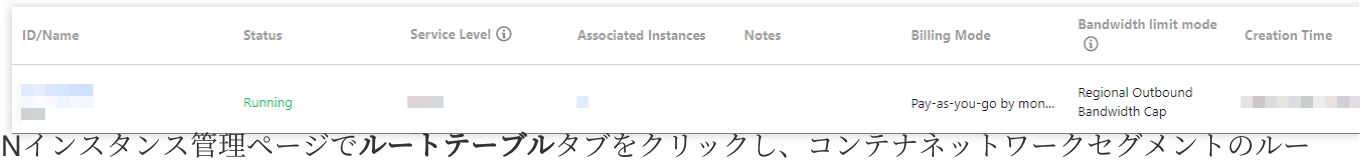
注意

このステップは、コンテナネットワークセグメントをCCNに登録するだけの手順であり、ルーティングの有効化の有無はCCN側で制御する必要があります。



コンテナネットワークセグメントルーティングの確認

1. VPCコンソールにログインし、左側ナビゲーションバーのCCNをクリックしてCCN管理ページに進みます。
2. クラスターVPCに関連付けられたCCNが配置されている行の右側にある**インスタンス管理**をクリックして、CCNインスタンス管理ページに進みます。下図に示すとおりです。



3. CCNインスタンス管理ページで**ルートテーブル**タブをクリックし、コンテナネットワークセグメントのルーティング起動状況を確認します。下図に示すとおりです。

説明

ネットワークセグメントが競合しない場合、ルーティングはデフォルトで有効になります。ネットワークセグメントが競合する場合、ルーティングはデフォルトで無効になります。

ルート競合の原則については、ルート制限をご参照ください。ルート競合を起動したい場合は、ルーティングの有効化をご参照ください。

4. コンテナクラスターとCCNの他のリソース間の相互運用性のテストを開始できます。

VPC-CNIモード
VPC-CNI パターンセキュリティグループの使用の説明

最終更新日：：2023-05-06 19:19:09

次の方式でVPC-CNIモードのために作成したElastic Network Interface（ENI）を、指定のセキュリティグループにバインドできます。

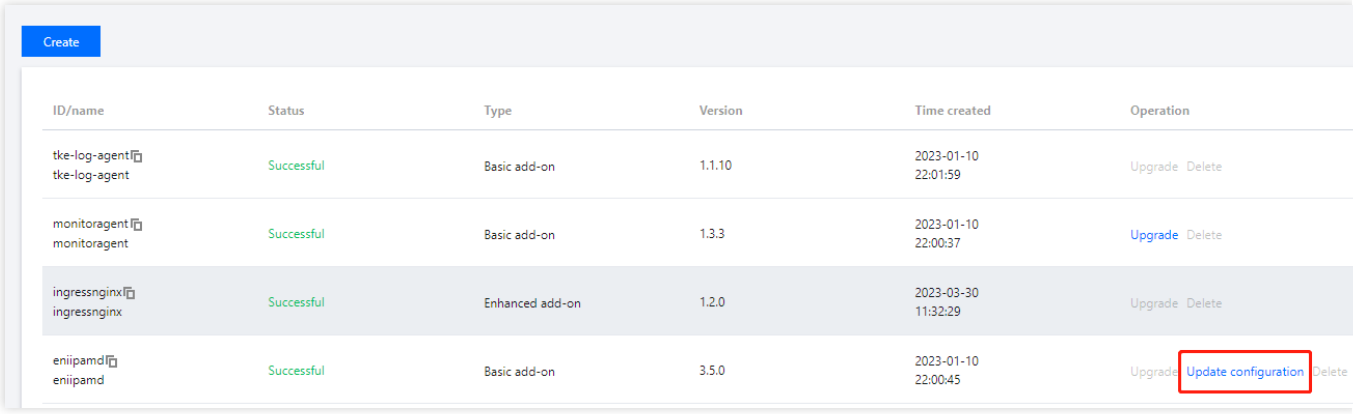
前提条件

IPAMDコンポーネントのバージョンが、v3.2.0+（イメージtag で確認できます）であること。
IPAMDコンポーネントがセキュリティグループ機能を起動すると、起動パラメータは、：`--enable-security-groups`（無効がデフォルト）となります。
現在は複数Pod共有ネットワークカードモードのみをサポートしています。

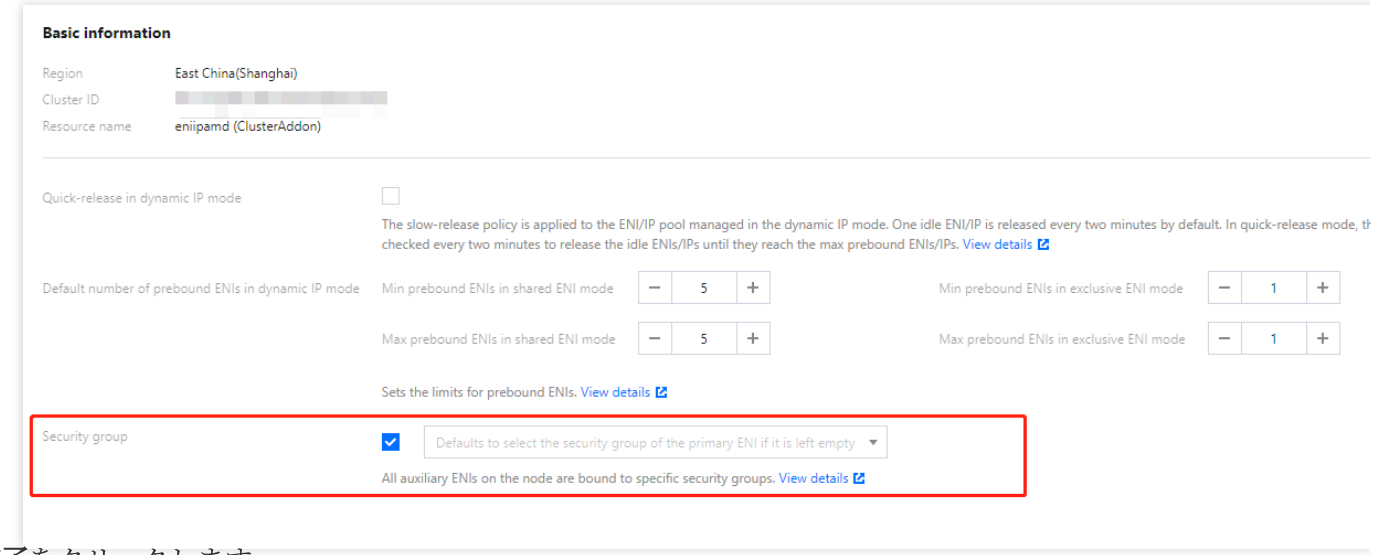
IPAMDコンポーネントでセキュリティグループ特性を起動

tke-eni-ipamdコンポーネントバージョン >= v3.5.0

1. TKEコンソールにログインし、左側ナビゲーションバーで**クラスター**をクリックします。
2. **クラスター管理**ページで、起動したいセキュリティグループのクラスターIDを選択し、クラスター詳細ページに進みます。
3. クラスター詳細ページで、左側の**コンポーネント管理**を選択し、コンポーネント管理ページで、 **eniipamd** コンポーネント右側の**設定更新**をクリックします。



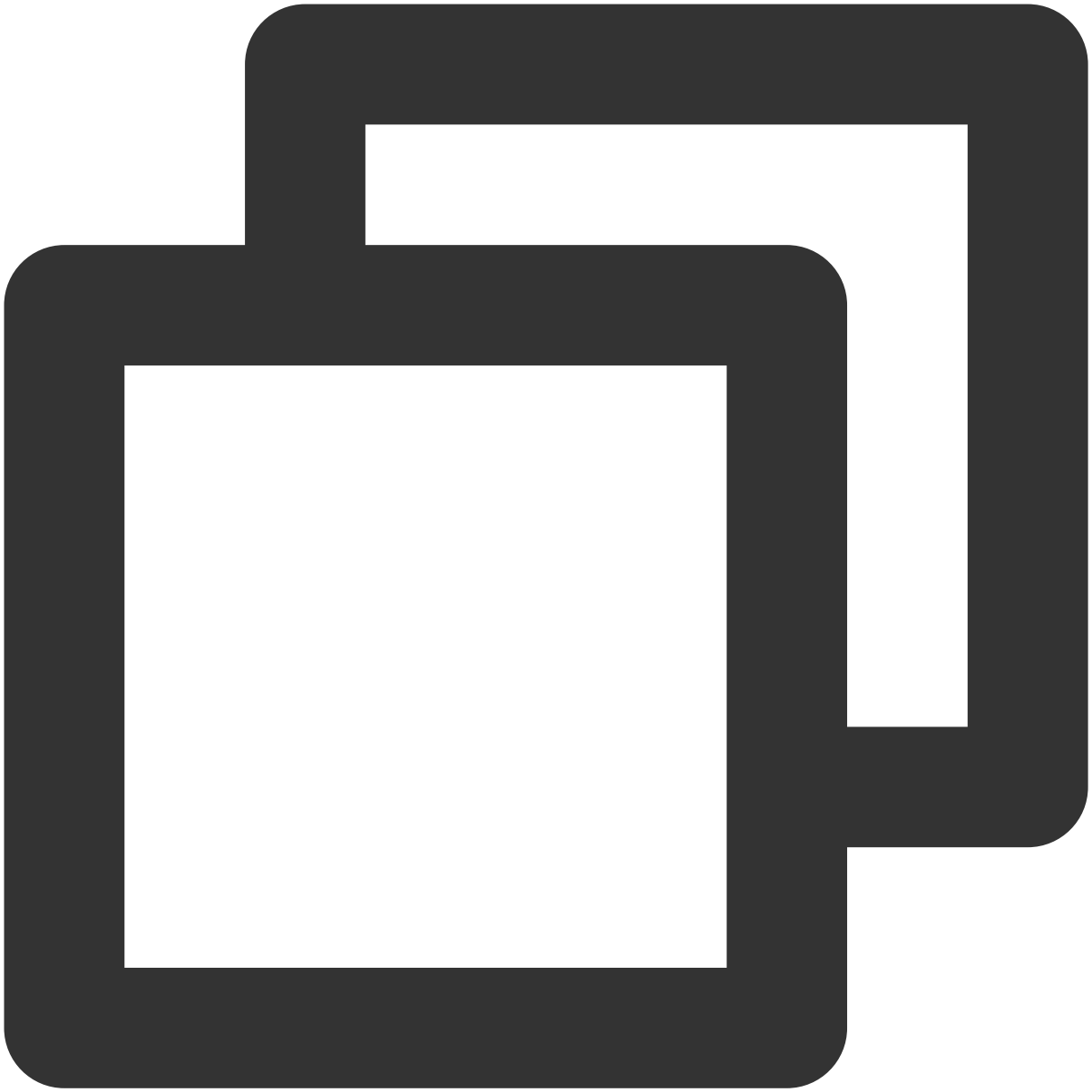
4. 設定更新ページで、**セキュリティグループ**をチェックします。マスターネットワークカードのセキュリティグループを引き継ぐ場合は、セキュリティグループは指定しません。そうでない場合、セキュリティグループの指定が必要になります。



5. **完了**をクリックします。

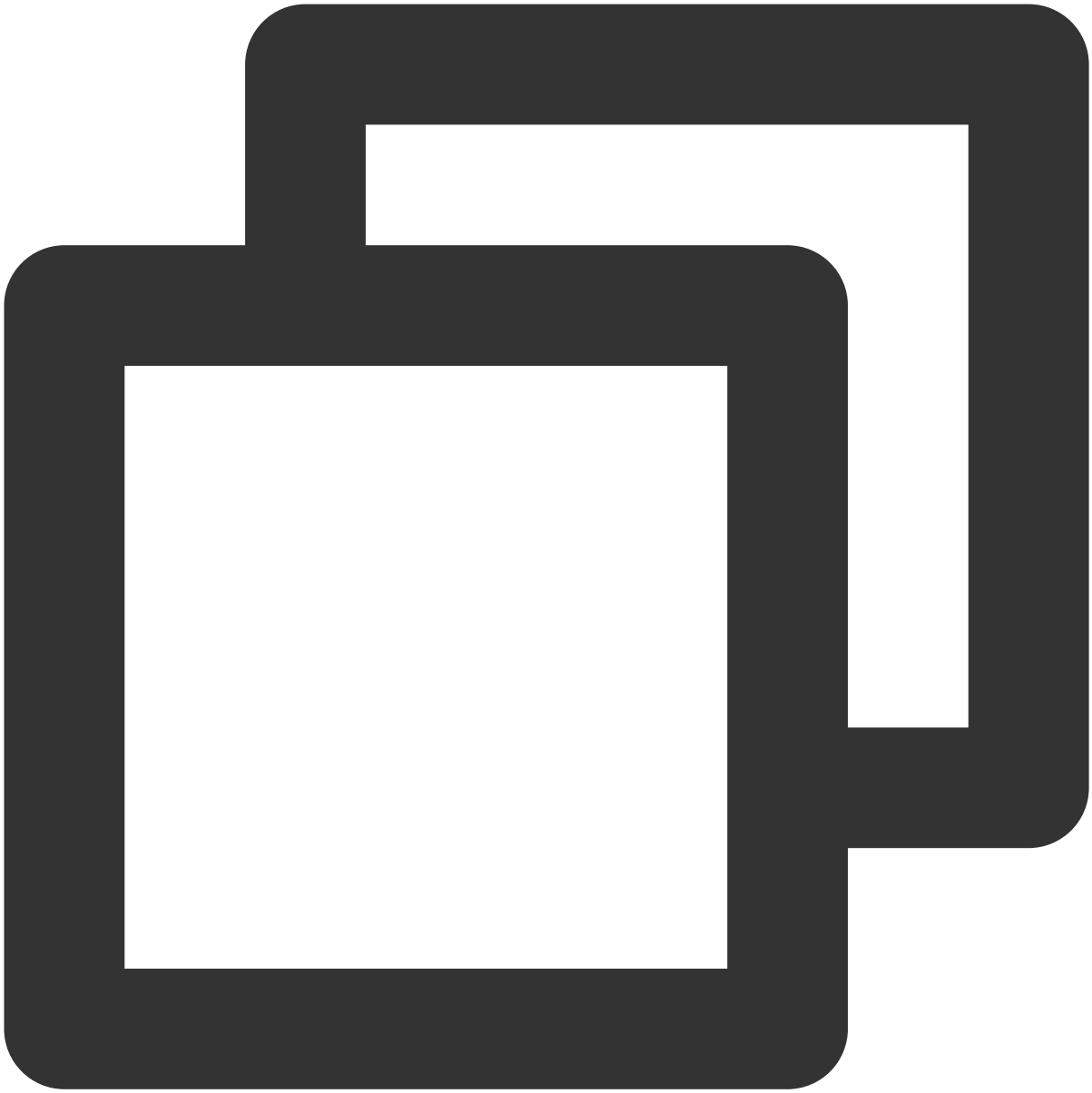
tke-eni-ipamdコンポーネントバージョン < v3.5.0あるいはコンポーネント管理の中に eniipamdコンポーネントがないこと

既存のtke-eni-ipamd deploymentを変更します。



```
kubectl edit deploy tke-eni-ipamd -n kube-system
```

次のコマンドを実行し、 `spec.template.spec.containers[0].args` の中に起動パラメータを追加します。
変更後、ipamdは自動で再起動し、有効化されます。
有効化された後、既存のノード上のセキュリティグループと関連のない補助ENIは、以下のポリシーでセキュリティグループにバインドされます。バインドされたとしても、設定したセキュリティグループとは強力に同期します。それまでに特性を起動している場合を除き、ノードセキュリティグループは設定済みとなります。ノードのENIを増分すると、すべて以下のセキュリティグループにバインドされます。

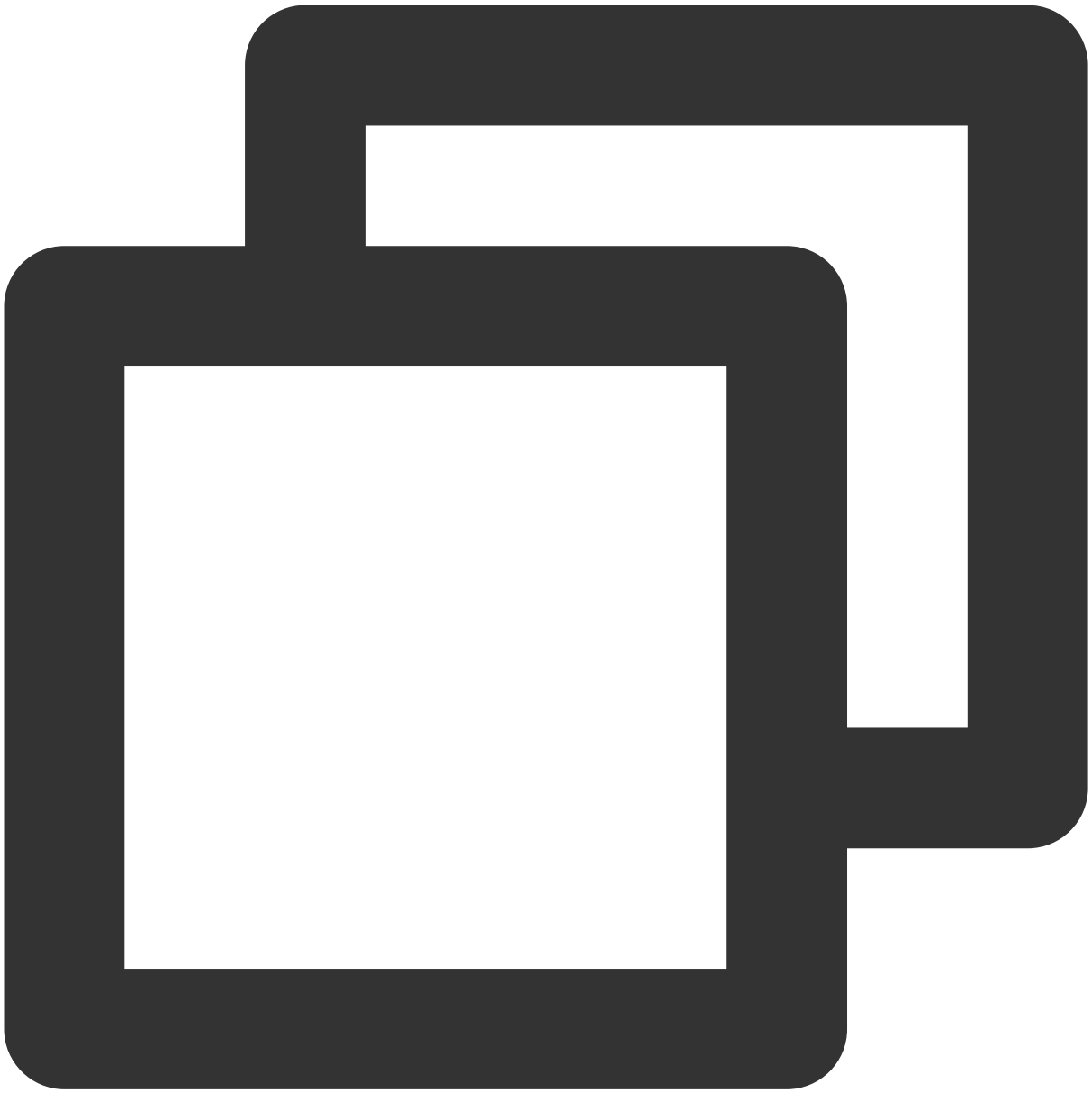


```
- --enable-security-groups
# デフォルトでご自身のネットワークカード/インスタンスのセキュリティグループを引き継ぐ場合、securit
- --security-groups=sg-xxxxxxxx,sg-xxxxxxxx
```

既存ノード同期ネットワークカードセキュリティグループの設定方法

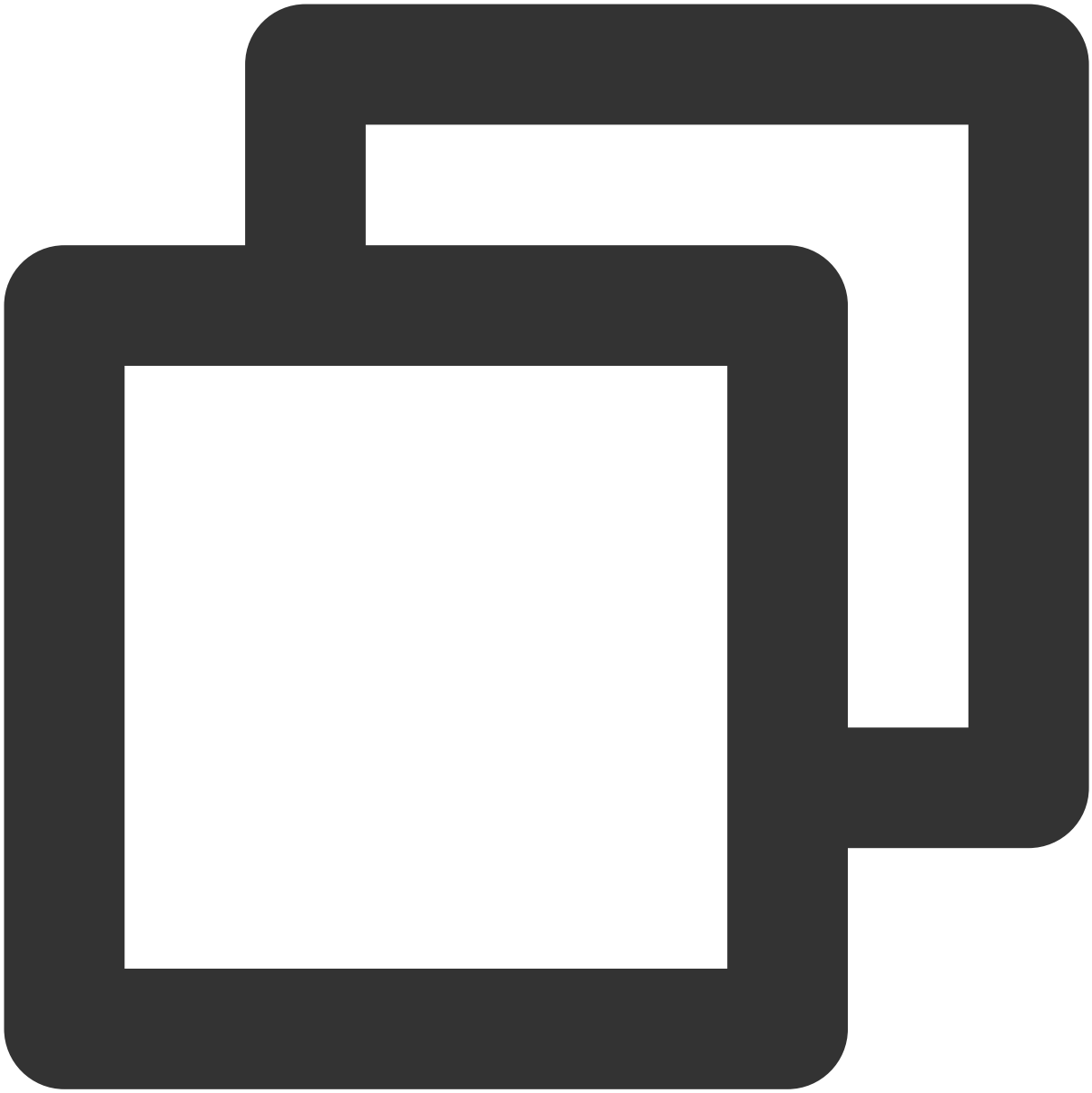
すでに設定済みのセキュリティグループの既存ノードを有効にする場合は、手動でセキュリティグループを無効化の上、再度有効化して同期させる必要があります。既存ノードの同期方法は以下のとおりです。

1. ノードにアノテーションを追加し、ノードのENIのセキュリティグループへのバインドをクリアにして無効化します。追加後は、ノードの既存ENIはすべてのセキュリティグループへのバインドを解除します。



```
kubectl annotate node <nodeName> --overwrite tke.cloud.tencent.com/disable-node-eni
```

2.（ステップ1実行完了後2-5s待ちます）再度noを設定すると、上記ポリシーが設定するセキュリティグループへのバインドを再度行うことができます。



```
kubectl annotate node <nodeName> --overwrite tke.cloud.tencent.com/disable-node-eni
```

機能ロジック

起動パラメータ `--security-groups` が未設定、あるいは値がブランクの場合、各ノードのセキュリティグループは自らのノードインスタンスがバインドしたセキュリティグループ（マスターネットワークカードがバインドしたセキュリティグループ）を継承します。特性を起動すると、ノードインスタンスセキュリティグループ

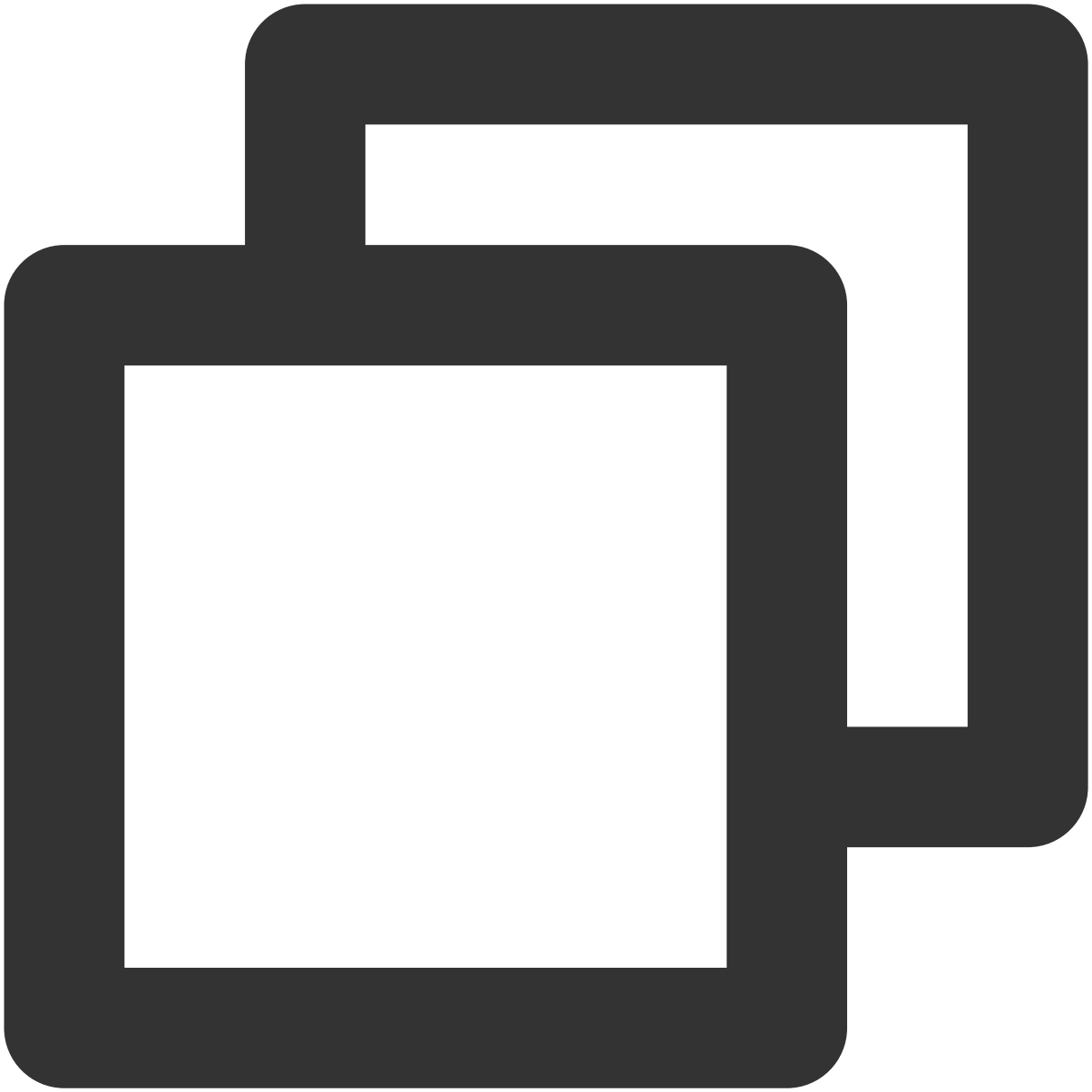
（マスターネットワークカードセキュリティグループ）に変化が現れます。補助ネットワークカードのセキュリティグループは同期認識を行わなくなるので、ノードセキュリティグループを無効化し再度有効化して同期させる必要があります。操作方法については 既存ノードとネットワークセキュリティグループの同期の設定方法をご参照ください。

特性を起動後に、 `--security-groups` を設定すると、各ノードセキュリティグループの設定は、そのセキュリティグループの集合となります。

特性を起動後に、 `--security-groups` パラメータを変更すると、増分したノードセキュリティグループの設定はグローバルパラメータと同期します。既存のノードセキュリティグループの設定は変わりません。既存のノードセキュリティグループの設定と同期させる場合は、ノードセキュリティグループを無効化し再度有効化して同期させる必要があります。操作方法については 既存ノードとネットワークカードセキュリティグループの同期の設定方法をご参照ください。

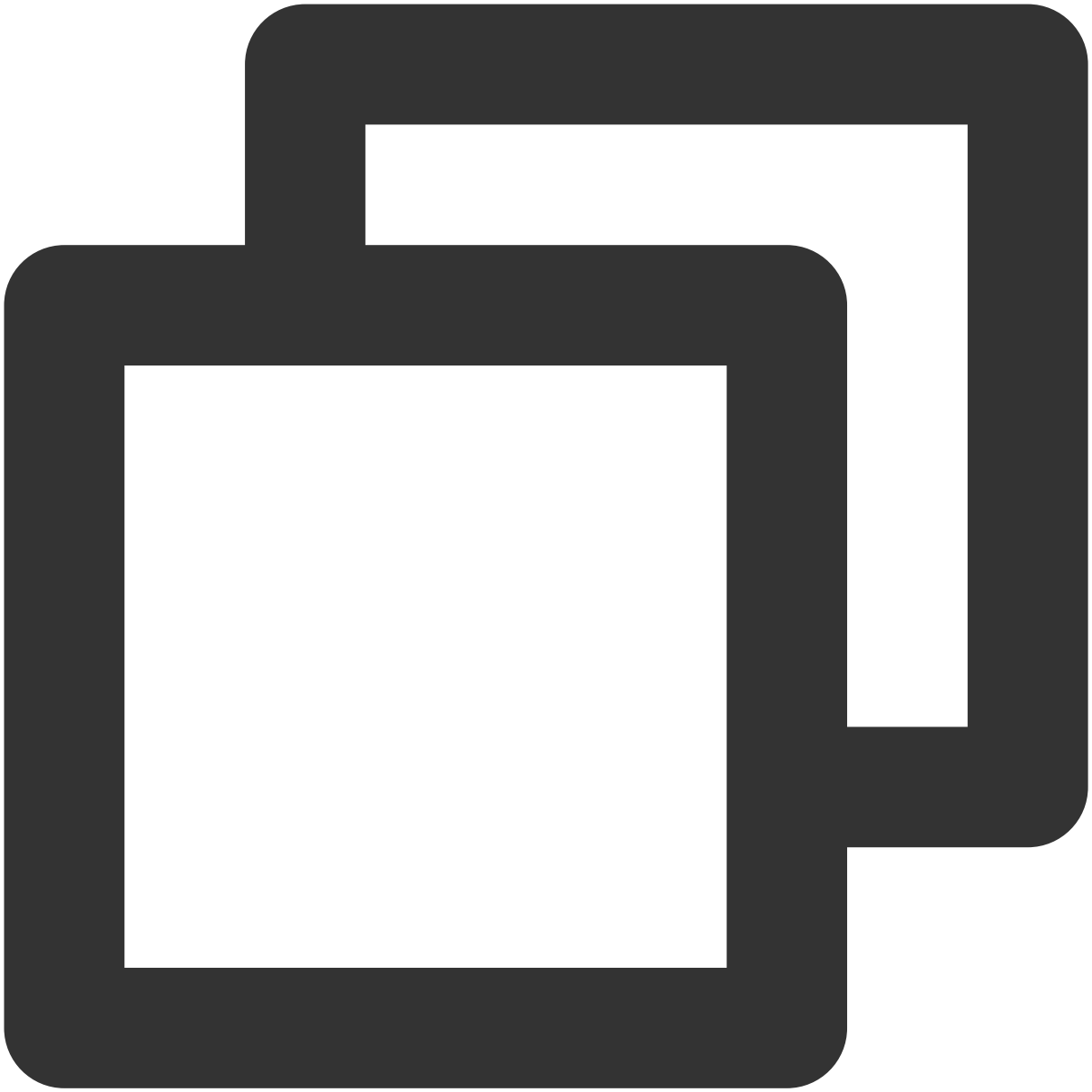
セキュリティグループが設定した優先度は、ノードセキュリティグループの設定と順序が一致しています。ご自身のネットワークカードを継承する場合は、マスターネットワークカードと一致させます。

次のコマンドを実行するとノードセキュリティグループの確認ができます。そのうち `spec.securityGroups` ドメインにはノードセキュリティグループの情報が含まれています。



```
kubectl get nec <nodeName> -oyaml
```

次のコマンドを実行するとノードセキュリティグループの変更ができます。変更後すぐに有効化されます。



```
kubectl edit nec <nodeName>
```

特性を起動すると、既存のネットワークカードがセキュリティグループにバインドされていない場合、ノードセキュリティグループにバインドされます。既存のネットワークカードのセキュリティグループはノードセキュリティグループと強力な同期が行われ、設定したノードセキュリティグループとの一致が保証されます。増分したネットワークカードはすべてノードセキュリティグループにバインドされます。

クラスターの運用保守
イベント管理
イベントストレージ

最終更新日：：2023-04-28 11:08:19

説明

ログサービスCLSはTKEによって生成されたすべての審査やイベントデータに対し、2022年6月30日まで**無料サービス**を提供します。ログセットの自動作成または既存のログセット内でのログトピックの自動作成を選択してください。

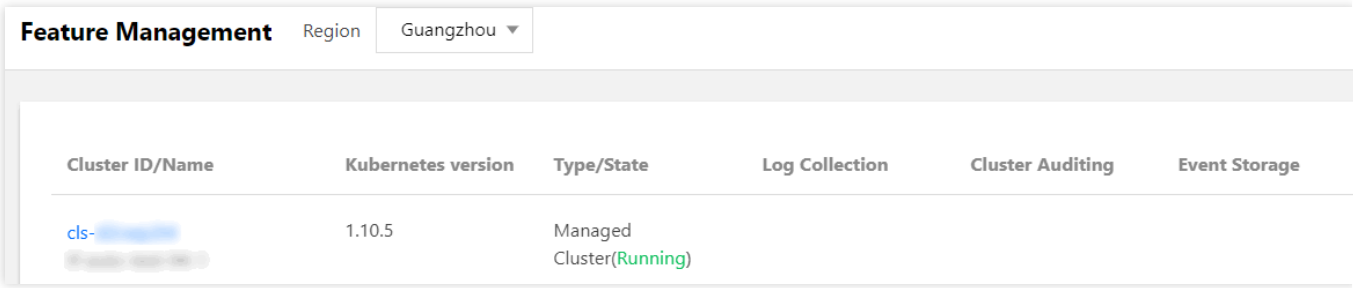
概要

Kubernetes EventsにはKubernetesクラスターの運用とさまざまなリソースのスケジューリング状況が含まれており、メンテナンス担当者がリソースの変更を日常的に観察し、問題を特定するのに役立ちます。TKEはすべてのクラスターに対してイベント永続機能の設定をサポートしており、この機能を有効にすると、クラスターイベントは設定したストレージ端末にリアルタイムでエクスポートされます。TKEはまた、Tencent Cloudが提供するPAASサービスやオープンソースソフトウェアを使用したイベントストリームの検索もサポートしています。ここでは、クラスターイベントの永続ストレージを有効にする方法についてご説明します。

操作手順

イベントストレージのオン

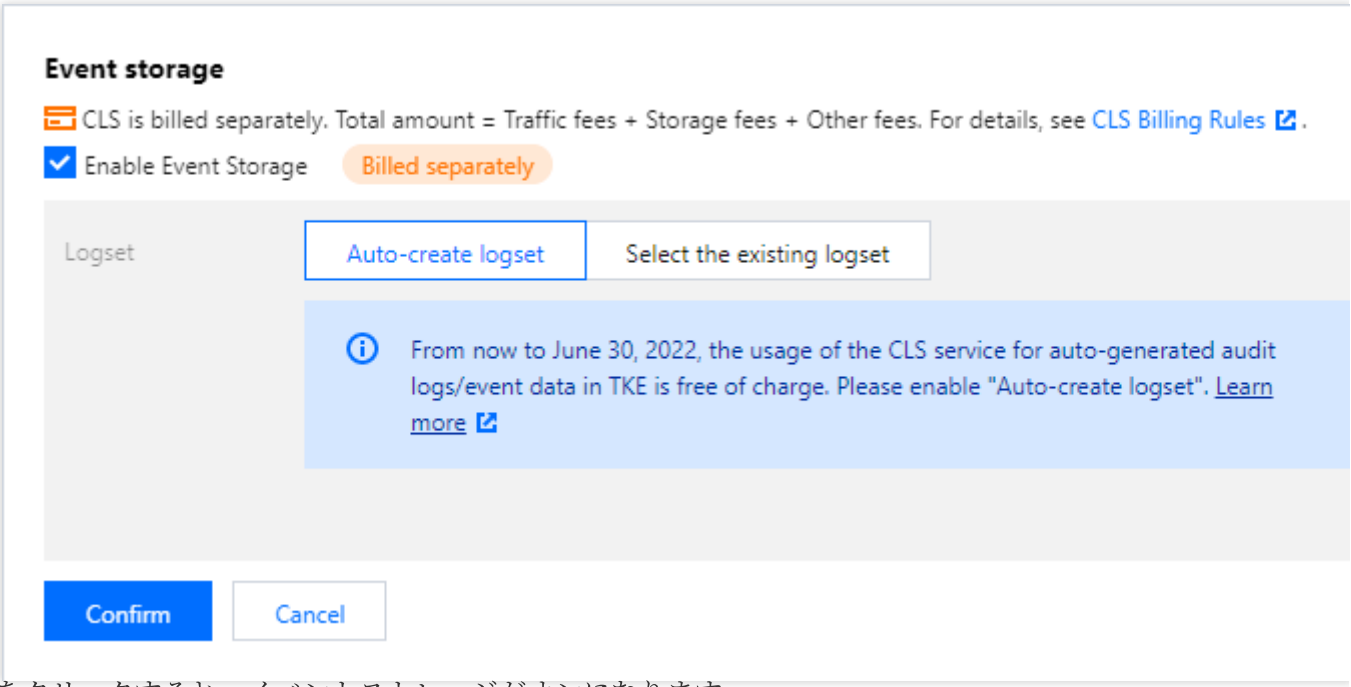
1. TKEコンソールにログインします。
2. 左側ナビゲーションバーで**運用保守機能管理**を選択します。
3. **機能管理**ページの上部でリージョンとクラスタータイプを選択し、イベントストレージをオンにしたいクラスターの右側にある**設定**をクリックします。下図に示すとおりです。



4. **機能の設定**ページで、イベントストレージの右側にある**編集**をクリックします。**イベントストレージのオン**にチェックを入れて、ログセットとログトピックを設定します。下図に示すとおりです。

注意

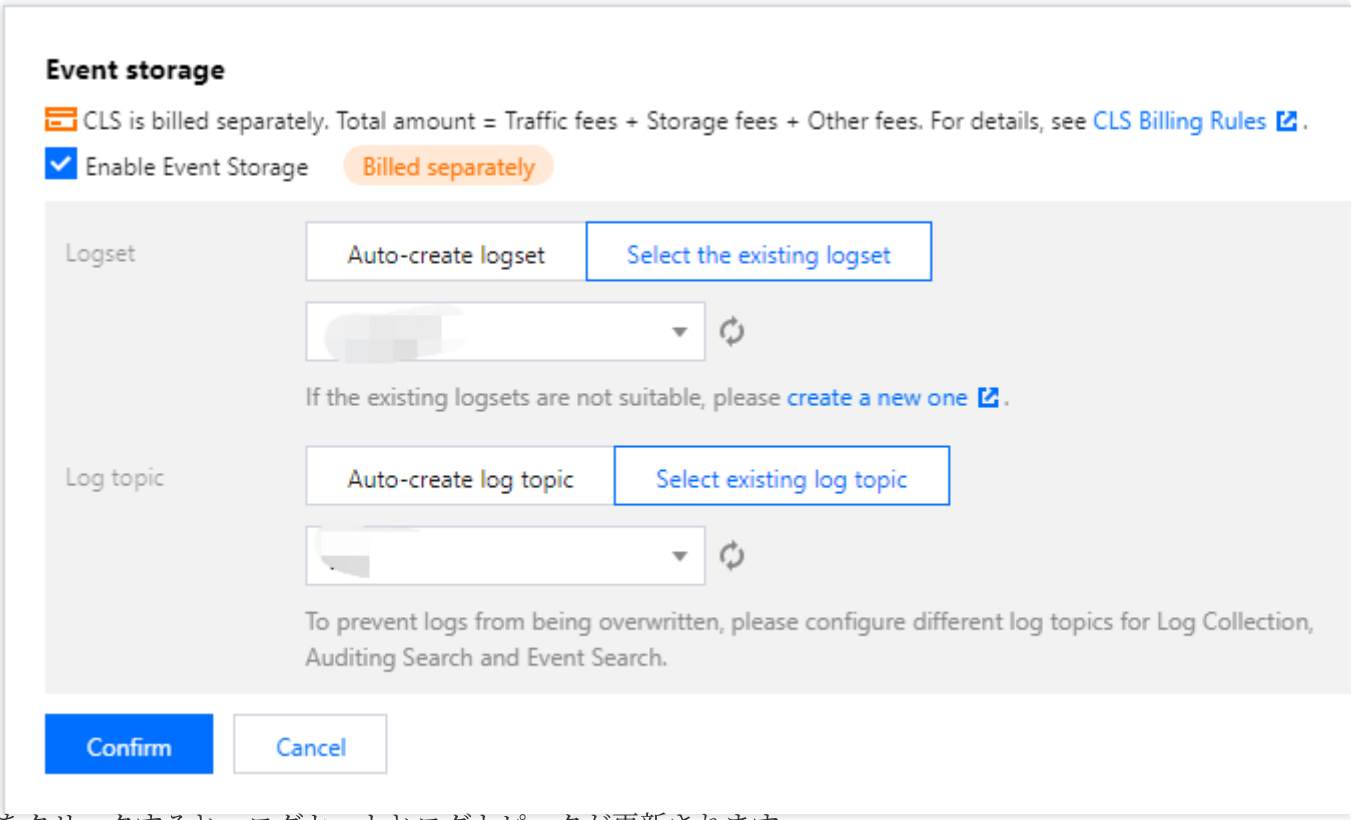
1つのログセットに存在できるログトピックは最大10個です。ログトピックを自動的に作成する場合は、そのログセットに10個以上のログトピックが存在しないようにしてください。



5. **OK**をクリックすると、イベントストレージがオンになります。

ログセットまたはログトピックの更新

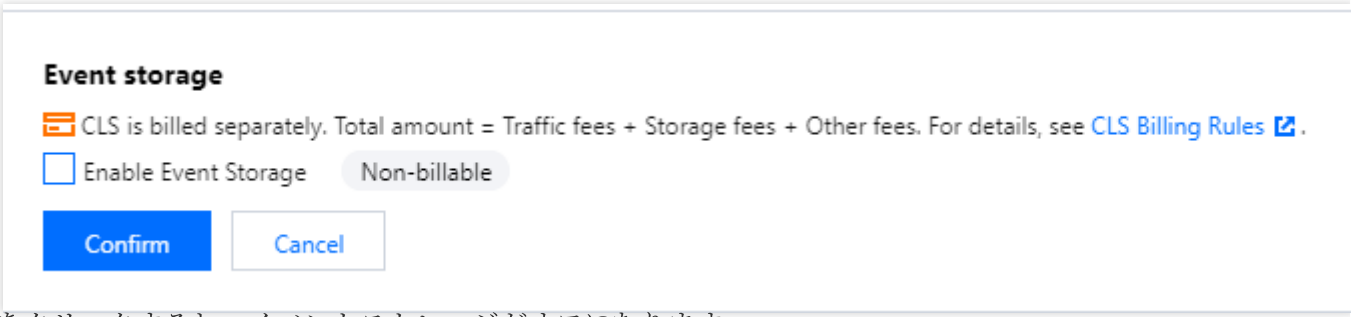
1. TKEコンソールにログインします。
2. 左側ナビゲーションバーで**運用保守機能管理**を選択します。
3. **機能管理**ページの上部でリージョンおよびクラスターのタイプを選択してクラスター右側の**設定**をクリックします。
4. **機能の設定**ページで、イベントストレージの右側にある**編集**をクリックし、ログセットとログトピックをもう一度選択します。下図に示すとおりです。



5. **OK**をクリックすると、ログセットとログトピックが更新されます。

イベントストレージのオフ

1. TKEコンソールにログインします。
2. 左側ナビゲーションバーで**運用保守機能管理**を選択します。
3. **機能管理**ページの上部でリージョンおよびクラスターのタイプを選択してクラスター右側の**設定**をクリックします。
4. **機能の設定**ページで、イベントストレージの右側にある**編集**をクリックし、**イベントストレージのオン**のチェックを外します。下図に示すとおりです。



5. **OK**をクリックすると、イベントストレージがオフになります。

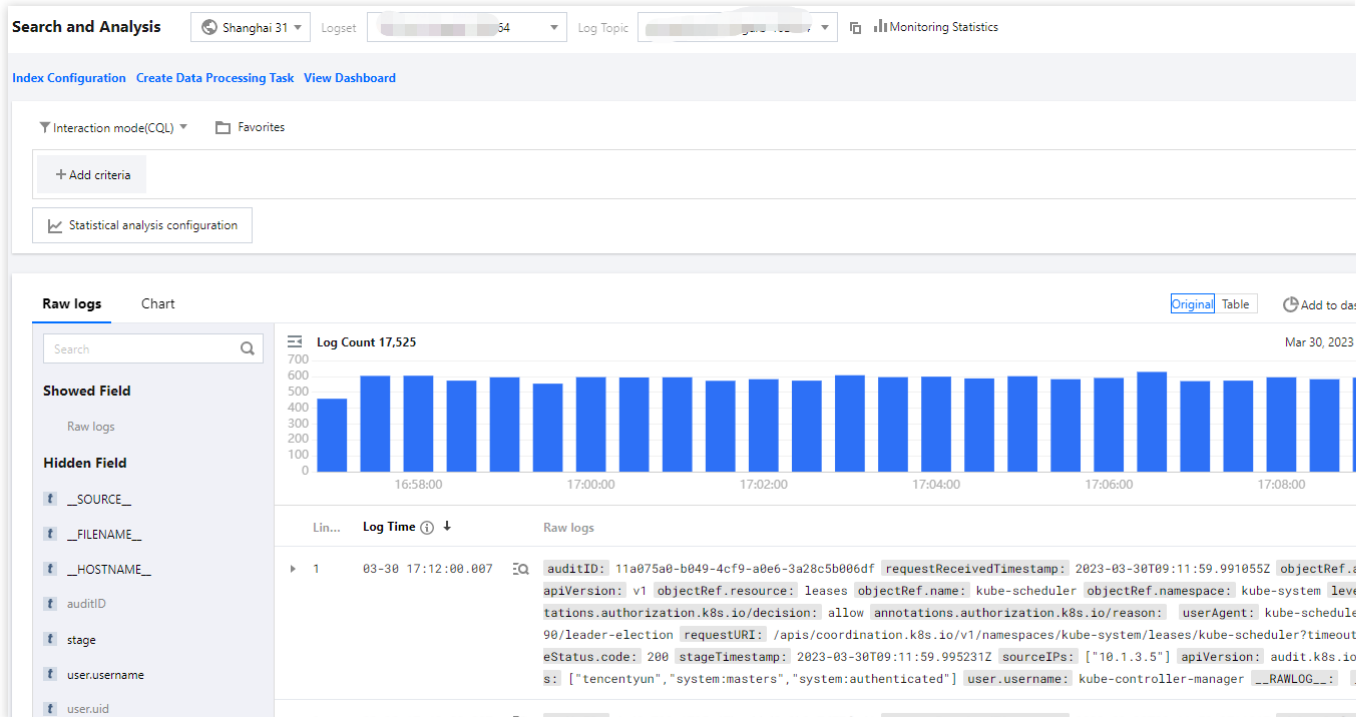
CLSコンソールでのイベントの表示

1. CLSコンソールにログインします。
2. 左側ナビゲーションバーで**検索の分析**を選択します。

3. **検索の分析**ページで、イベントストレージに設定されているログセットとログトピックを選択し、必要に応じて表示フィールドをご自身で設定し、検索の分析を行います。下図に示すとおりです。

説明：

イベントストレージをオンにすると、Topicのインデックス作成がデフォルトでオンになります。



イベントダッシュボード

最終更新日：：2023-05-06 20:09:55

概要

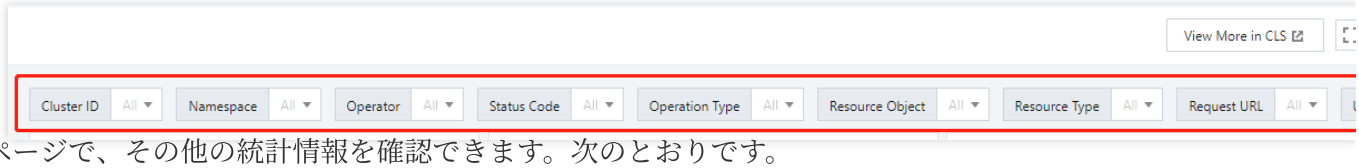
TKEはユーザーに、購入後すぐに使用可能なイベントダッシュボードを提供します。クラスターでイベントストレージ機能を有効にすると、TKEは、クラスターの各種イベントの一覧と異常イベントの集計検索分析ダッシュボードの設定を自動で行います。また、フィルタリング項目のユーザーカスタム設定をサポートするとともに、CLSのイベントのグローバル検索を内蔵することで、TKEコンソールの全面的な管理、検索、分析、問題の特定といった機能を実現しました。

機能の説明

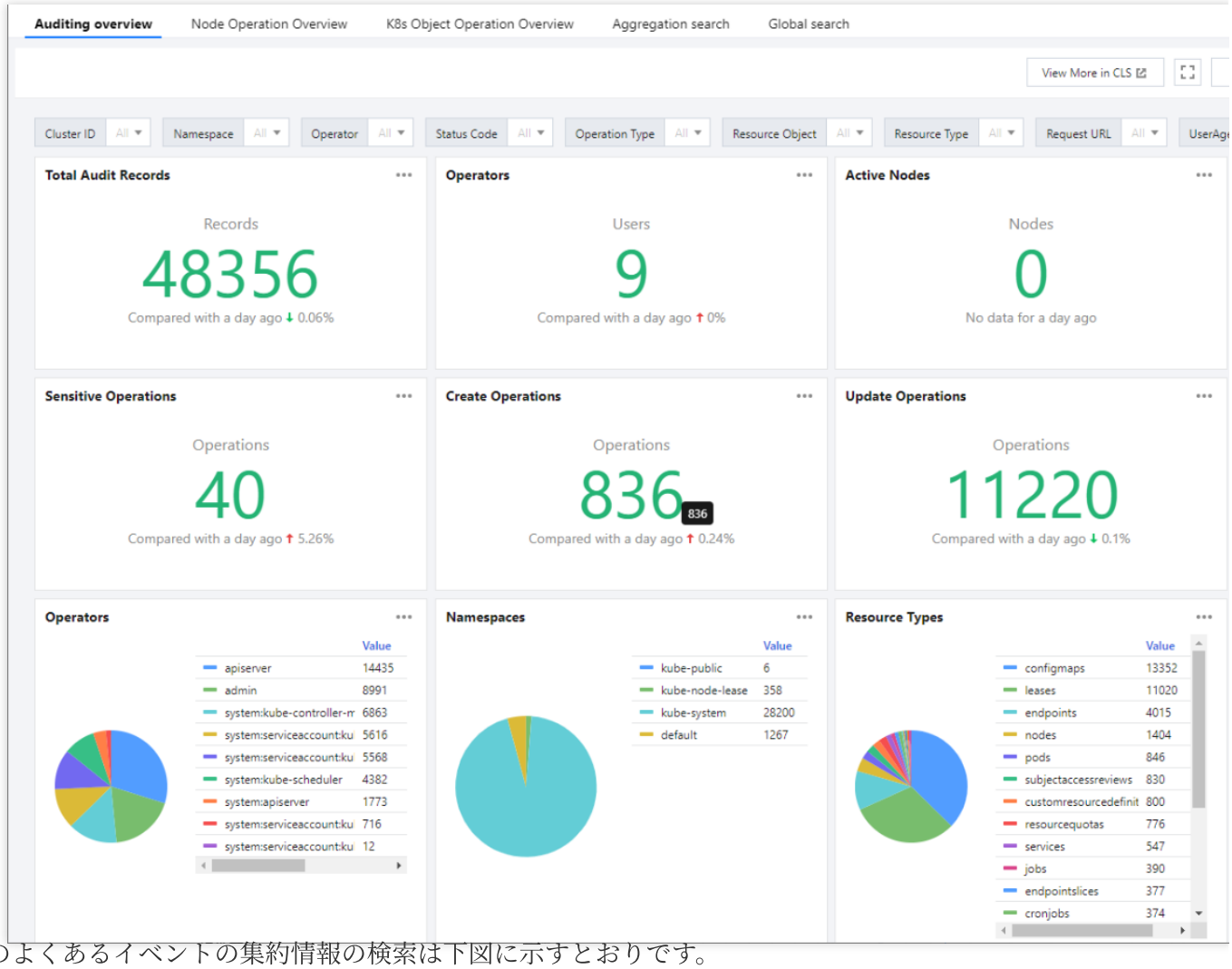
イベント検索では、「イベント一覧」、「異常イベント集計検索」、「グローバル検索」の3つのダッシュボードを設定します。次のステップで「イベント検索」ページに進み、これらの機能の使用を開始します。
1. TKEコンソールにログインします。
2. 「イベントストレージ機能」の起動について、詳しくはイベントストレージをご参照ください。
3. 左側ナビゲーションバーで、**ログ管理 > イベントログ**を選択します。
4. **イベント検索**ページの上方でリージョンおよびクラスタータイプを選択し、クラスターイベントの詳細を確認します。

イベント一覧

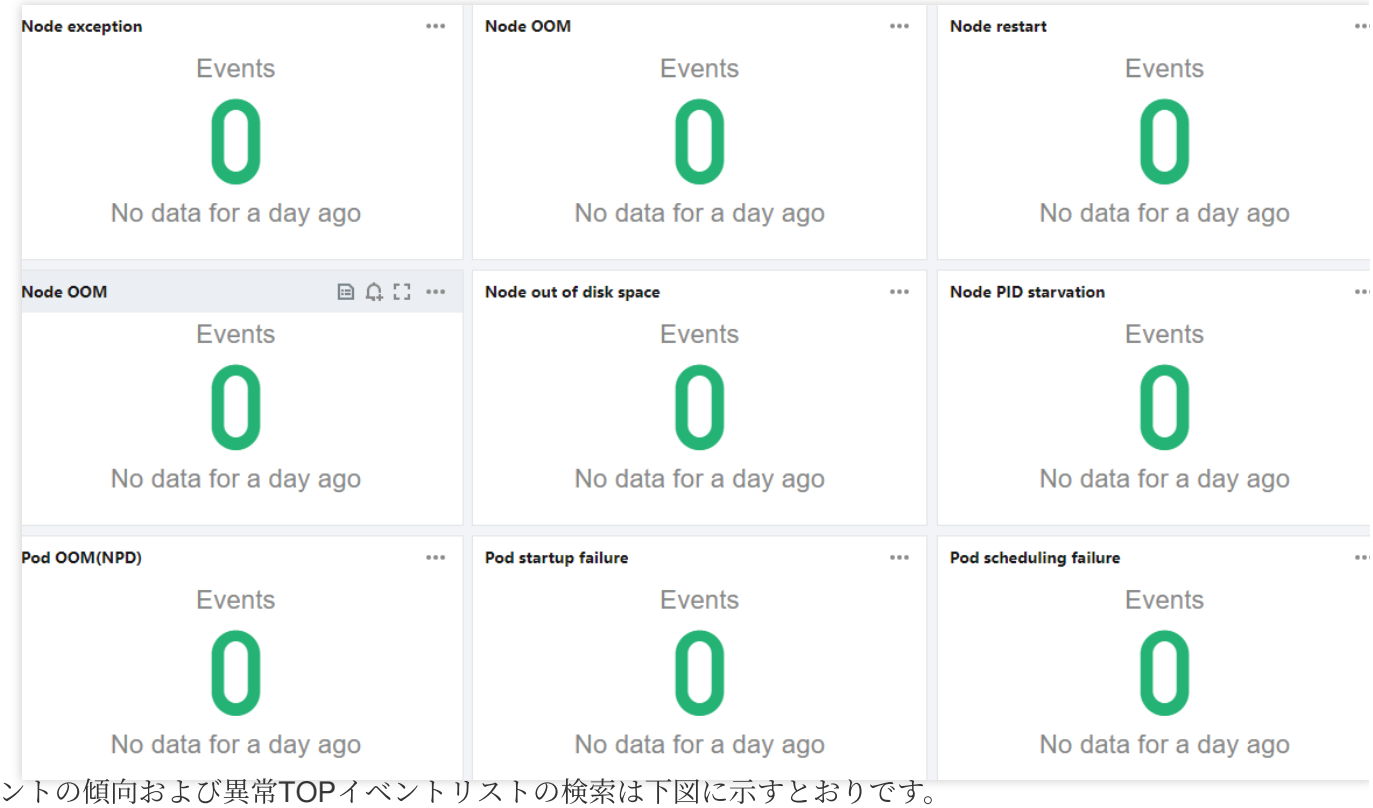
イベント一覧ページでは、クラスターID、ネームスペース、レベル、原因、リソースタイプ、リソースオブジェクトイベントソースなどの次元でイベントをフィルタリングし、コアイベントの集約統計情報を確認して、1つの周期内のデータを比較します。例えば、イベント総数、分布状況、ノード異常、Pod OOM、重要なイベントの傾向などのダッシュボードおよび異常TOPイベントリストなどです。
フィルタリング項目では、ご自身のニーズに基づいたカスタム設定が可能です。下図に示すとおりです。



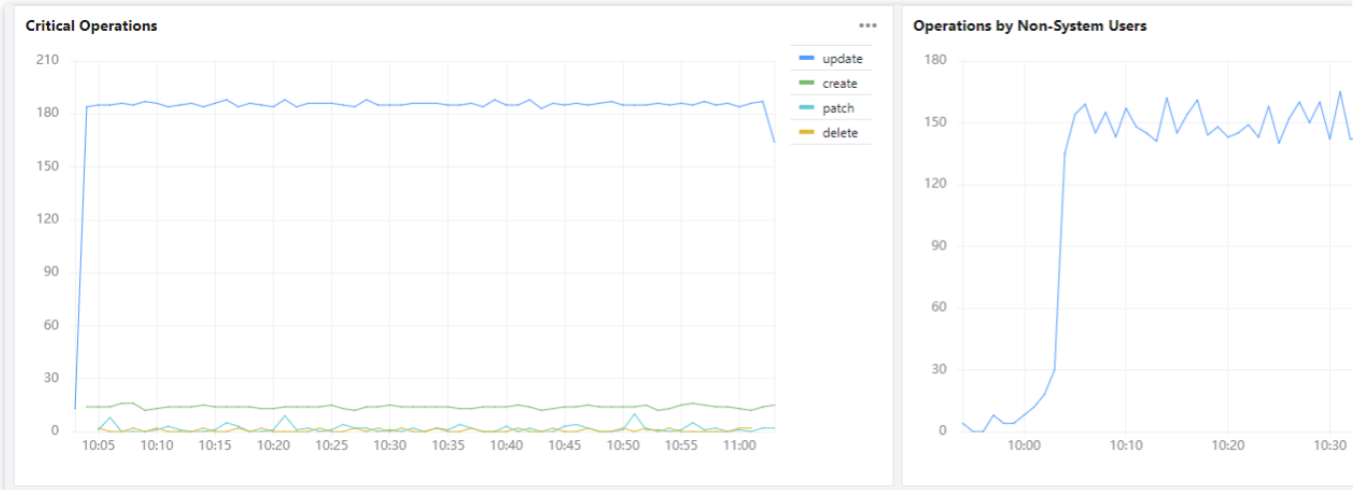
このページで、その他の統計情報を確認できます。次のとおりです。
イベント総数およびレベル分布の状況、異常イベントの原因、オブジェクトの分布状況の検索は、下図に示すとおりです。



各種のよくあるイベントの集約情報の検索は下図に示すとおりです。

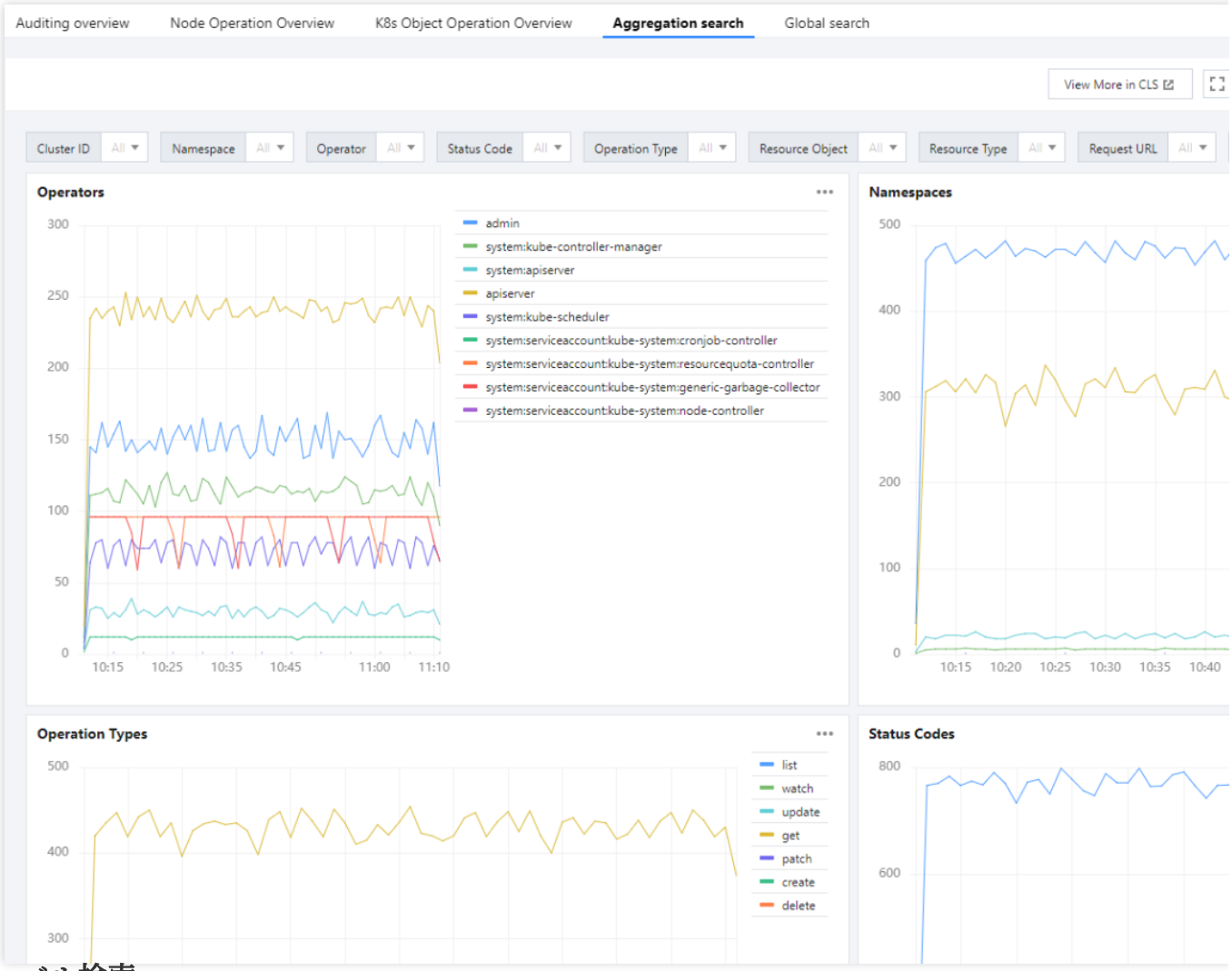


イベントの傾向および異常TOPイベントリストの検索は下図に示すとおりです。



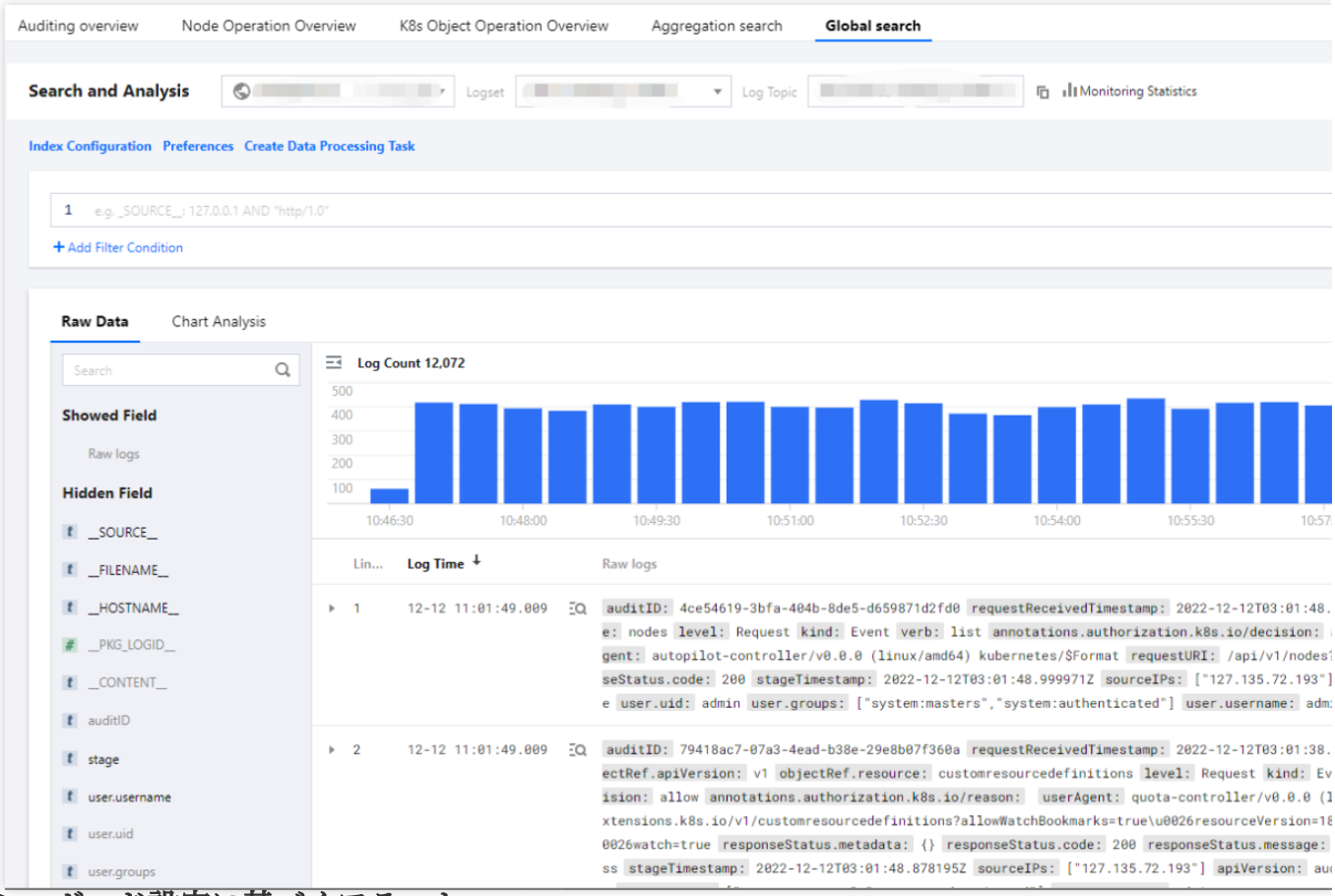
異常イベント集計の検索

異常イベント集計の検索ページにてフィルタリング条件を設定し、いずれかの時間帯の各種異常イベントのreasonおよびobject分布傾向を確認します。傾向の下方には、問題のすばやい特定に役立つ、検索可能な異常イベントリストが表示されます。下図に示すとおりです。



グローバル検索

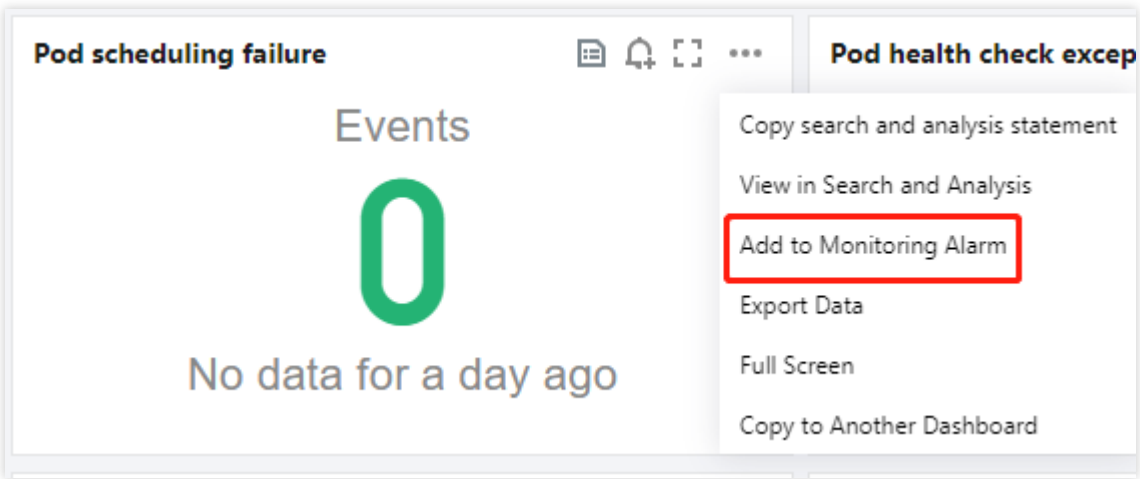
ユーザーがTKEコンソールでも、素早くすべてのイベントのグローバル検索ができるように、グローバル検索ダッシュボードにはCLSの検索分析ページが組み込まれています。下図に示すとおりです。



ダッシュボード設定に基づくアラート

上記のプリセットをしたダッシュボードでアラート設定をすると、設定条件に達した際に、アラートをトリガーすることができます。操作の詳細は次のとおりです。

1. アラートの設定が必要なダッシュボードの右側の**監視アラートを追加**をクリックします。下図に示すとおりです。



2. CLSコンソール の> **アラートポリシー** でアラートポリシーを新規作成します。詳しくは アラートポリシーの設定をご参照ください。

監視とアラーム
監視とアラームの概要

最終更新日：：2022-03-31 11:40:58

概要

優れた監視環境は、Tencent Kubernetes Engine（TKE）の高信頼性、高可用性および高性能のために重要な保証を提供します。さまざまなリソースのさまざまなディメンションの監視データを簡単に収集できるため、リソースの使用状況を簡単に把握し、障害を簡単に特定できます。
TKEは、クラスター、ノード、ワークロード、Pod、Containerの5つのレベルで監視データの収集および表示機能を提供します。
監視データの収集は、コンテナクラスター性能の通常の基準を確立するのに役立ちます。さまざまな時間とさまざまなロード条件でのコンテナクラスター性能を測定し、履歴監視データを収集することで、コンテナクラスターとサービスが実行されているときの通常の性能を明確に理解し、現在の監視データによって、サービスが異常に実行されているかどうかを速やかに判断し、問題の解決策をタイムリーに見つけます。例えば、サービスのCPU利用率、メモリ使用率およびディスクI/Oを監視できます。

監視

TKEの監視機能の使用方法については、監視データの確認をご参照ください。
現在カバーされている監視指標については、監視とアラームの指標リストをご参照ください。

アラート

TKEの異常状態をタイムリーに発見して、ビジネスの安定性と信頼性を確保するのに便利です。すべての作り出しクラスターに必要なアラームを設定することをお勧めします。アラームの設定方法については、アラームの設定をご参照ください。
現在カバーされているアラーム指標については、監視とアラームの指標リストをご参照ください。

> TKEにより提供された監視とアラーム機能は主に、Kubernetesオブジェクトのコア指標またはイベントをカバーします。クラウド監視により提供された基本的なリソース監視（Cloud Virtual Machine、Block storage、Cloud Load Balancerなど）と組み合わせて使用して、より詳細な指標カバレッジを確保します。

監視データの確認

最終更新日：：2022-04-07 11:04:08

操作ケース

Tencent Kubernetes Engine（TKE）は、デフォルトですべてのクラスターに基本的な監視機能を提供します。次の方法でTKEの監視データを確認できます。

- クラスター指標の確認
- ノード指標の確認
- ノード内のPod指標の確認
- ワークロード指標の確認
- ワークロード内のPod指標の確認
- Pod内のContainer指標の確認

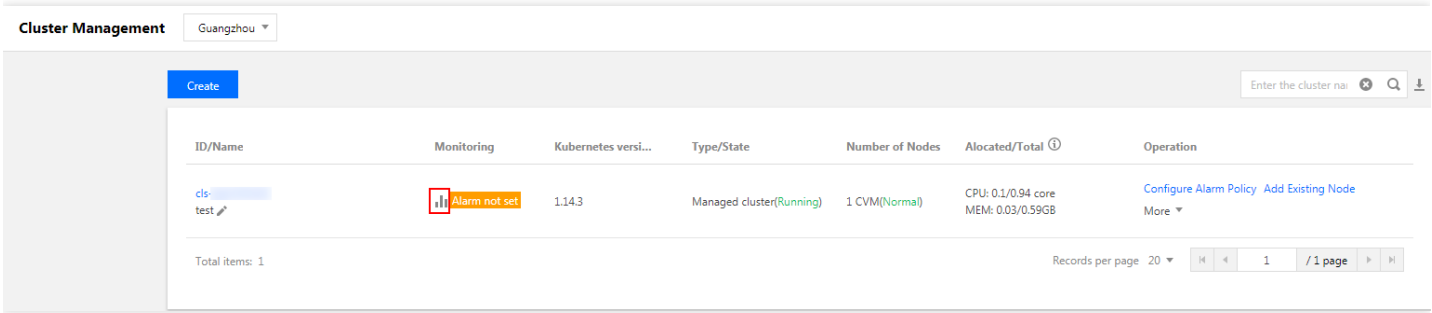
前提条件

コンソールにログインし、【 Cluster 】の管理ページへ進んでいます。

操作手順

クラスター指標の確認

監視データの確認を必要とするクラスター行では、をクリックして、クラスターによって監視される情報ページを確認できます。下図の通りです：

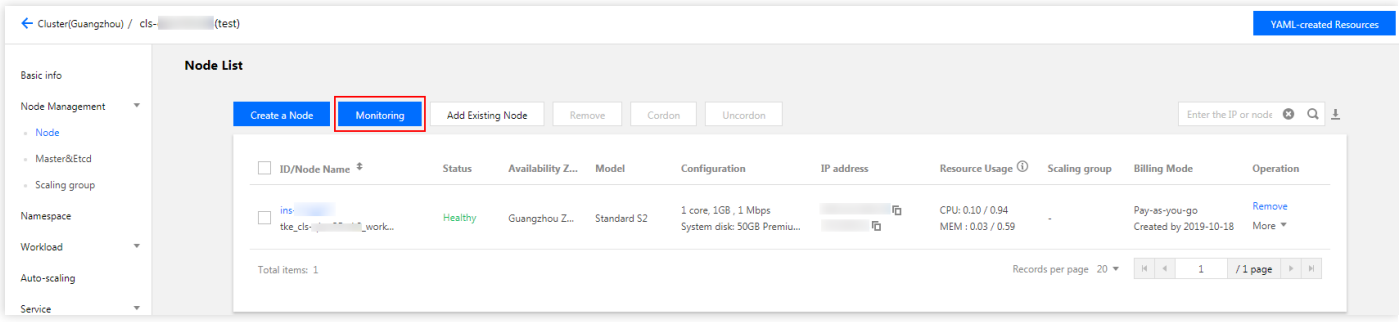


ノード指標の確認

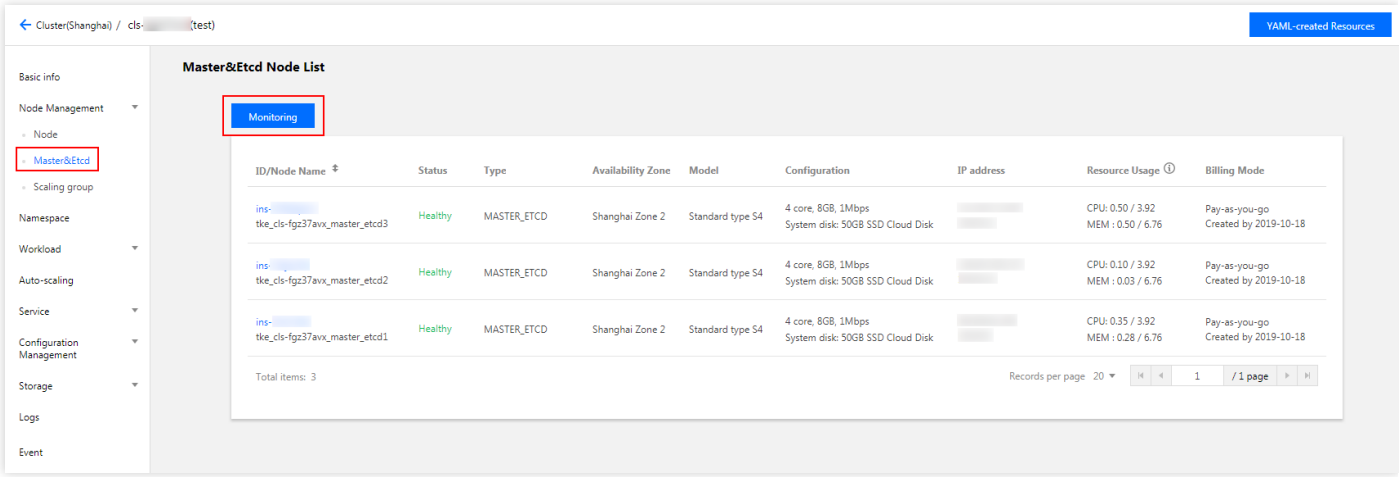
以下のアクションを使用して、ノードとMaster&Etcdノードの監視情報を確認できます。

1. 確認を必要とするクラスターID/名称を選択して、クラスターの管理ページへ進みます。
2. **Node Management **を展開して、ノードとMaster&Etcdノードの監視情報を確認できます。

- **Node>Monitoring**を選択して、** Node Monitoring**ページへ進んで監視情報を確認できます。下図の通りです：



- **Master & Etcd **> Monitoring を選択して、**Master&Etcd監視**ページへ進んで監視情報を確認できます。下図の通りです：



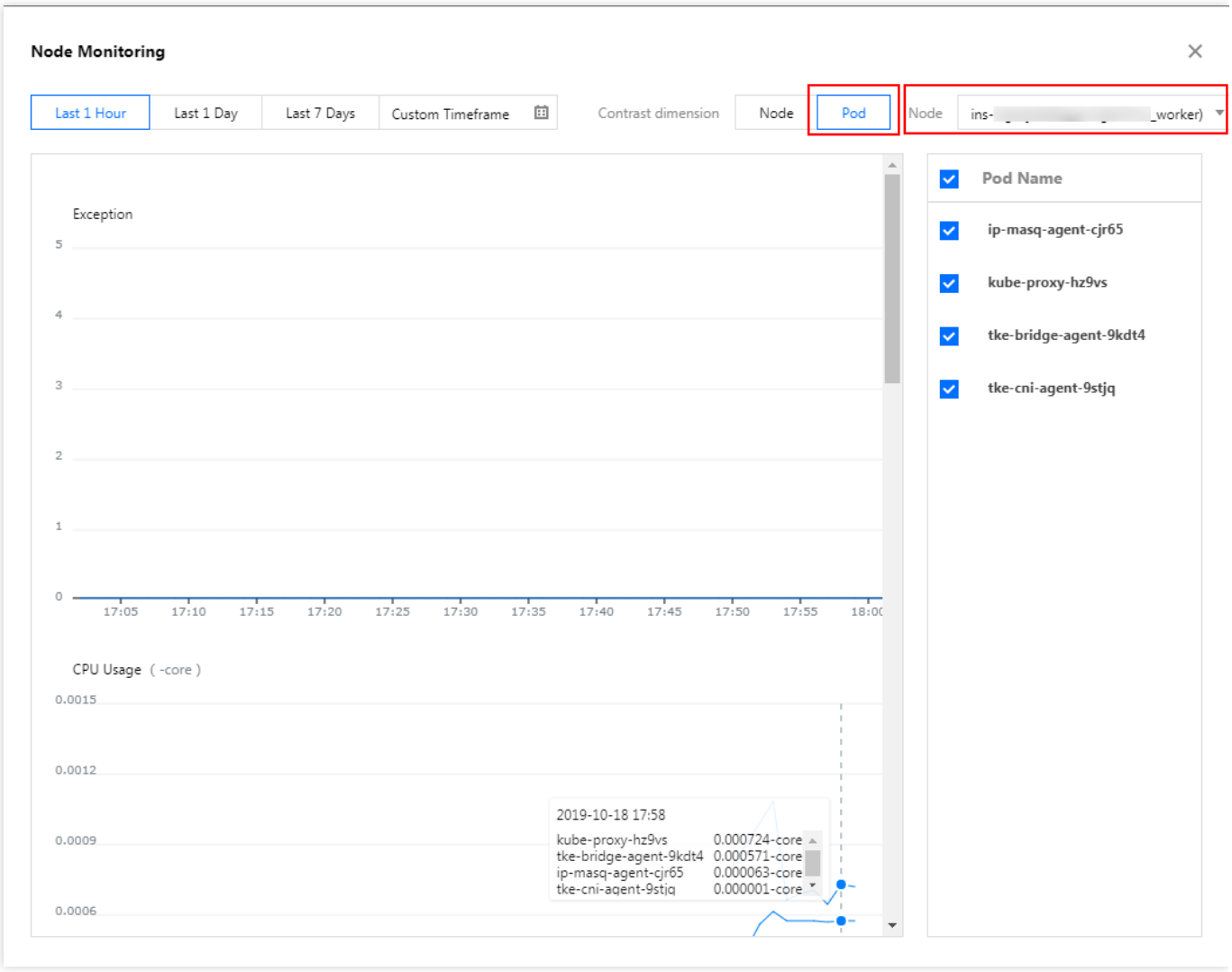
ノード内のPod指標の確認

次の2つの方法を使用して、ノード内のPod指標を確認できます。

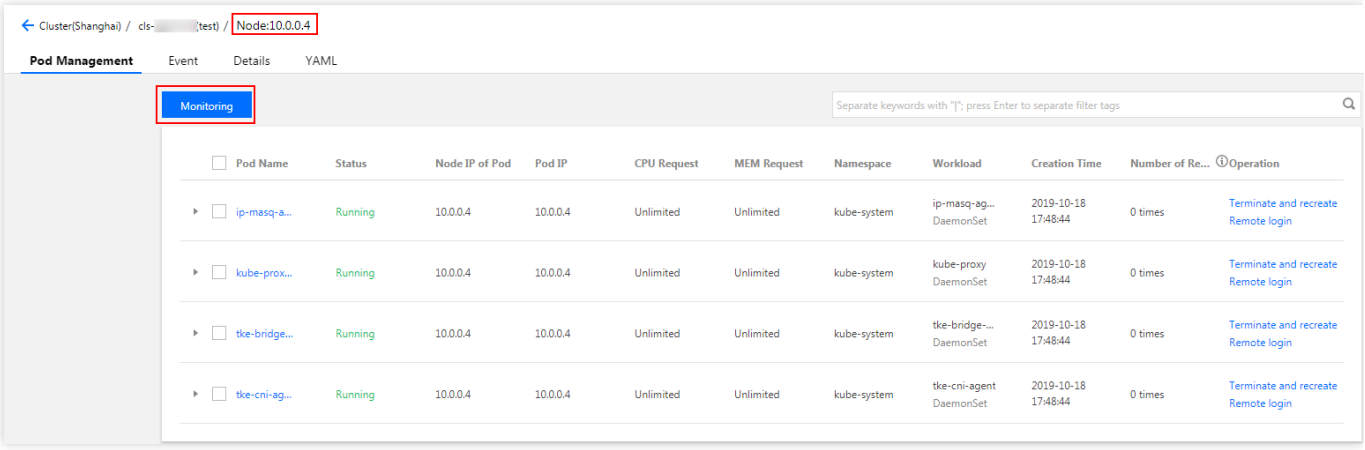
1. 確認を必要とするクラスターID/名称を選択して、クラスターの管理ページへ進みます。
2. **Node Management**>** Node**を選択して、ノードリストページへ進みます。
3. 実際のニーズに応じて、ノード内のPod指標の確認方法を選択します。

- ノードリストからPod指標を確認します。
  i. ** Monitoringをクリックして、Node Monitoring**ページへ進みます。

ii. **Pod**をクリックして、確認したいノードとして** Node**を選択すると、ノード内のPodの監視指標比較チャートを確認できます。下図の通りです：

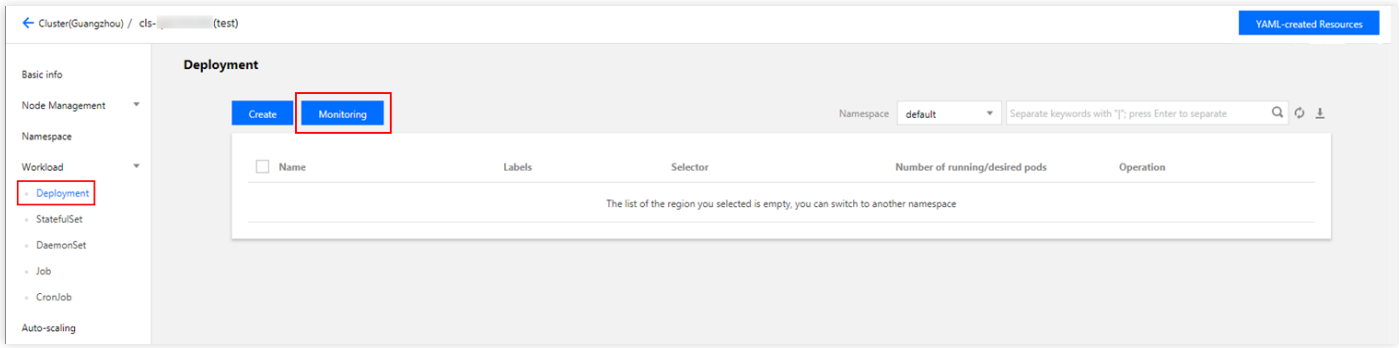


- ノードの詳細ページでPod指標を確認します。
  i. 確認を必要とするノードID/ノード名を選択して、ノードの管理ページへ進みます。
  ii. **Pod Management**タブを選択し、**Monitoring **をクリックして、ノード内のPodの監視指標比較チャートを確認できます。下図の通りです：



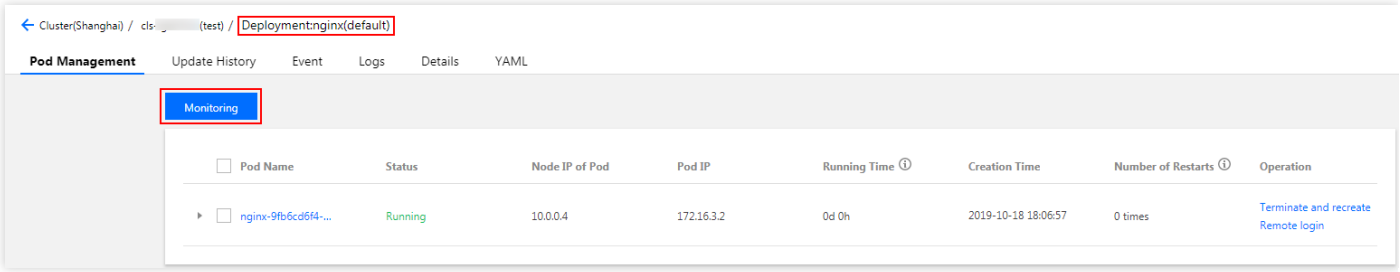
ワークロード指標の確認

1. 確認を必要とするクラスターID/名称を選択して、クラスターの管理ページへ進みます。
2. **Workload **>**Any Workload Type**を選択します。例えば、**Deployment**を選択して、Deployment管理ページへ進みます。
3. **Monitoring**をクリックして、ワークロードの監視情報を確認できます。下図の通りです：



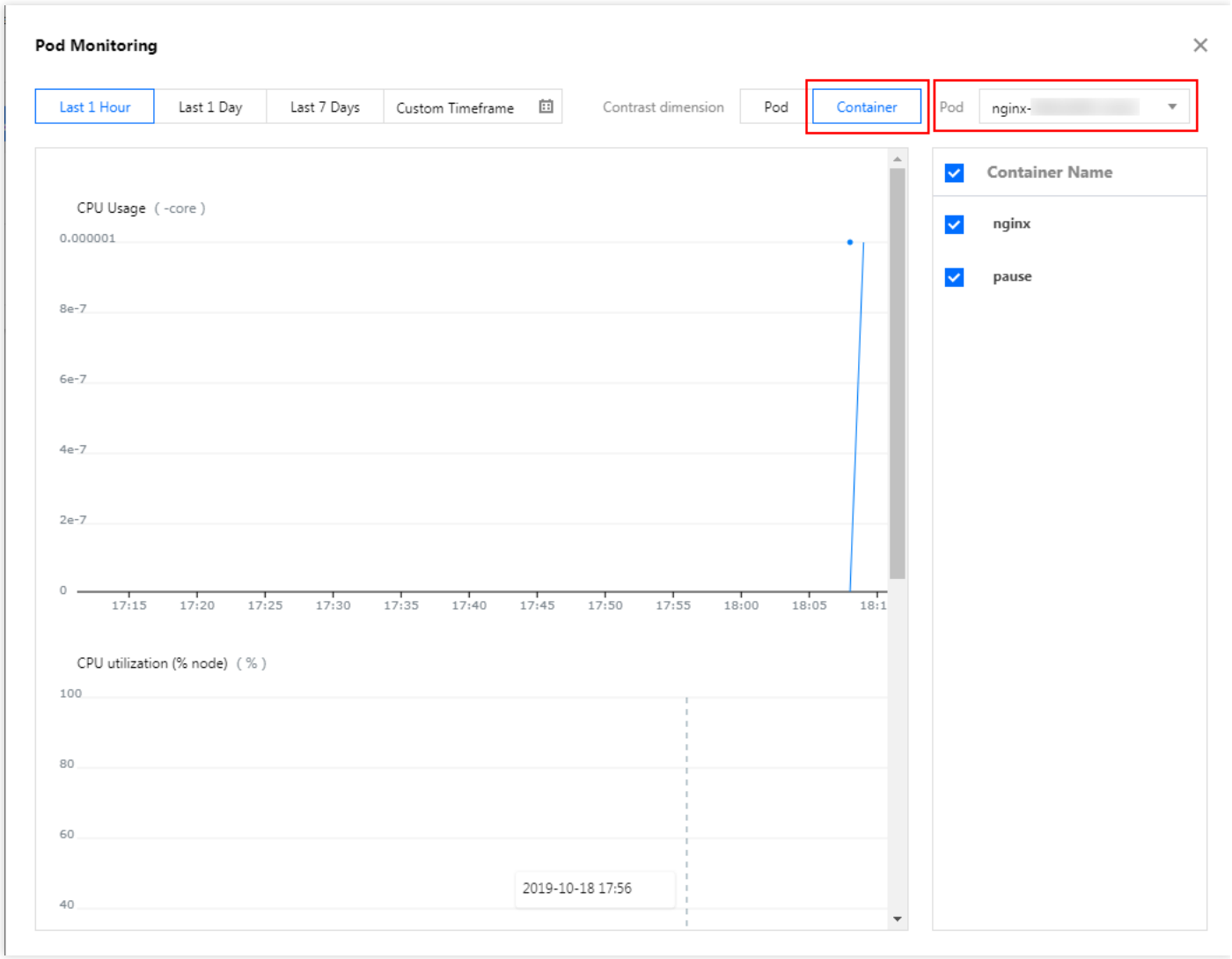
ワークロード内のPod指標の確認

1. 確認を必要とするクラスターID/名称を選択して、クラスターの管理ページへ進みます。
2. **Workload>Any Workload Type**を選択します。例えば、**Deployment**を選択して、Deployment管理ページへ進みます。
3. 確認を必要とするワークロード名称を選択して、ワークロードの管理ページへ進みます。
4. ** Pod Management**タブを選択し、**Monitoring**をクリックして、ワークロード内のすべてのPodの監視指標比較チャートを確認できます。下図の通りです：



Pod内のContainer指標の確認

1. Viewing Pod Metrics in a Workloadの手順1-手順3を参照して、ワークロードの詳細ページへ進みます。
2. **Pod Management**タブを選択し、** Monitoring **をクリックして、**Pod Monitoring**ページへ進みます。
3. **Container**をクリックして、確認したいPodとして**Pod**を選択すると、Pod内のContainerの監視指標比較チャートを確認できます。下図の通りです：



監視とアラームメトリックリスト

最終更新日：：2022-03-31 11:40:58

監視

現在、Tencent Kubernetes Engine（TKE）は、次のディメンションの監視指標を提供しています。すべての指標は、統計サイクルの**平均**です。

クラスター監視指標

監視指標	単位	説明
CPU利用率	%	クラスター全体のCPU利用率
メモリ利用率	%	クラスター全体のメモリ利用率

Master&Etcdと通常ノードの監視指標

監視指標	単位	説明
Podリスタート回数	回	ノード内のすべてのPodリスタート回数の合計
異常状態	-	ノードの状態、正常または異常
CPU利用率	%	ノードの総数に対するノード内のすべてのPodのCPU使用量の比率
メモリ利用率	%	ノードの総数に対するノード内のすべてのPodのメモリ使用量の比率
プライベートネットワークのインバウンド帯域幅	bps	ノード内のすべてのPodのプライベートネットワークインバウンド方向の帯域幅の合計
プライベートネットワークのアウトバウンド帯域幅	bps	ノード内のすべてのPodのプライベートネットワークアウトバウンド方向の帯域幅の合計
パブリックネットワークのインバウンド帯域幅	bps	ノード内のすべてのPodのパブリックネットワークインバウンド方向の帯域幅の合計
パブリックネットワークのアウトバウンド帯域幅	bps	ノード内のすべてのPodのパブリックネットワークアウトバウンド方向の帯域幅の合計
TCP接続数	個	ノードによって保持されているTCP接続数

クラスターノードの監視指標の詳細については、Cloud Virtual Machine監視をご参照ください。

クラスターノードのデータディスクの監視指標の詳細については、Cloud Block Storage監視をご参照ください。

ワークロードの監視指標

監視指標	単位	説明
Podリスタート回数	回	ワークロード内のすべてのPodリスタート回数の合計
CPU使用量	コア	ワークロード内のすべてのPodのCPU使用量
CPU利用率（クラスターに対する）	%	クラスター総数に対するワークロード内のすべてのPodのCPU使用量の比率
メモリ使用量	B	ワークロード内のすべてのPodのメモリ使用量
メモリ利用率（クラスターに対する）	%	クラスター総数に対するワークロード内のすべてのPodのメモリ使用量の比率
ネットワークのインバウンド帯域幅	bps	ワークロード内のすべてのPodのインバウンド方向の帯域幅の合計
ネットワークのアウトバウンド帯域幅	bps	ワークロード内のすべてのPodのアウトバウンド方向の帯域幅の合計
ネットワークのインバウンドトラフィック	B	ワークロード内のすべてのPodのインバウンド方向のトラフィックの合計
ネットワークのアウトバウンドトラフィック	B	ワークロード内のすべてのPodのアウトバウンド方向のトラフィックの合計
ネットワークのインバウンドパケット	個/s	ワークロード内のすべてのPodのインバウンド方向のパケットの合計
ネットワークのアウトバウンドパケット	個/s	ワークロード内のすべてのPodのアウトバウンド方向のパケットの合計

ワークロードがクラスター外でサービスを提供する場合、バインディングされたServiceのネットワーク監視指標の詳細については、Cloud Load Balancer監視をご参照ください。

Pod 監視指標

監視指標	単位	説明
異常状態	-	Podの状態、正常または異常

監視指標	単位	説明
CPU使用量	コア	PodのCPU使用量
CPU利用率（ノードに対する）	%	ノードの総数に対するPodのCPU使用量の比率
CPU利用率（Requestに対する）	%	設定されたRequest値に対するPodのCPU使用量の比率
CPU利用率（Limitに対する）	%	設定されたLimit値に対するPodのCPU使用量の比率
メモリ使用量	B	キャッシュを含むPodのメモリ使用量
メモリ使用量（Cacheを除く）	B	Pod内のすべてのContainerの実際のメモリ使用量（キャッシュを除く）
メモリ利用率（ノードに対する）	%	ノードの総数に対するPodのメモリ使用量の比率
メモリ利用率（ノードに対する、Cacheを除く）	%	ノードの総数に対するPod内のすべてのContainerの実際のメモリ使用量（キャッシュを除く）の比率
メモリ利用率（Requestに対する）	%	設定されたRequest値に対するPodのメモリ使用量の比率
メモリ利用率（Requestに対する、Cacheを除く）	%	設定されたRequest値に対するPod内のすべてのContainerの実際のメモリ使用量（キャッシュを除く）の比率
メモリ利用率（Limitに対する）	%	設定されたLimit値に対するPodのメモリ使用量の比率
メモリ利用率（Limitに対する、Cacheを除く）	%	設定されたLimit値に対するPod内のすべてのContainerの実際のメモリ使用量（キャッシュを除く）の比率
ネットワークのインバウンド帯域幅	bps	Podのインバウンド方向の帯域幅の合計
ネットワークのアウトバウンド帯域幅	bps	Podのアウトバウンド方向の帯域幅の合計
ネットワークのインバウンドトラフィック	B	Podのインバウンド方向のトラフィックの合計
ネットワークのアウトバウンドトラフィック	B	Podのアウトバウンド方向のトラフィックの合計

監視指標	単位	説明
ネットワークのインバウンドパケット	個/s	Podのインバウンド方向のパケットの合計
ネットワークのアウトバウンドパケット	個/s	Podのアウトバウンド方向のパケットの合計

Container監視指標

監視指標	単位	説明
CPU使用量	コア	ContainerのCPU使用量
CPU利用率（ノードに対する）	%	ノードの総数に対するContainerのCPU使用量の比率
CPU利用率（Requestに対する）	%	設定されたRequest値に対するContainerのCPU使用量の比率
CPU利用率（Limitに対する）	%	設定されたLimit値に対するContainerのCPU使用量の比率
メモリ使用量	B	キャッシュを含むContainerのメモリ使用量
メモリ使用量（Cacheを除く）	B	Containerの実際のメモリ使用量（キャッシュを除く）
メモリ利用率（ノードに対する）	%	ノードの総数に対するContainerのメモリ使用量の比率
メモリ利用率（ノードに対する、Cacheを除く）	%	ノードの総数に対するContainerの実際のメモリ使用量（キャッシュを除く）の比率
メモリ利用率（Requestに対する）	%	設定されたRequest値に対するContainerのメモリ使用量の比率
メモリ利用率（Requestに対する、Cacheを除く）	%	設定されたRequest値に対するContainerの実際のメモリ使用量（キャッシュを除く）の比率
メモリ利用率（Limitに対する）	%	設定されたLimit値に対するContainerのメモリ使用量の比率
メモリ利用率（Limitに対する、Cacheを除く）	%	設定されたLimit値に対するContainerの実際のメモリ使用量（キャッシュを除く）の比率
ブロックデバイスの読取帯域幅	B/s	Containerがディスクからデータを読み取るスループット
ブロックデバイスの書込帯域幅	B/s	Containerがディスクにデータを書き込むスループット
ブロックデバイスの読取IOPS	回/s	Containerがディスクからデータを読み取るIO回数
ブロックデバイスの書込IOPS	回/s	Containerがディスクにデータを書き込むIO回数

アラート

現在、TKEは、次のディメンションのアラーム指標を提供しています。すべての指標は、統計サイクルの**平均**です。

クラスターアラーム指標

監視指標	単位	説明
CPU利用率	%	クラスター全体のCPU利用率
メモリ利用率	%	クラスター全体のメモリ利用率 %
CPUアサイン率	%	クラスターのアサイン可能なCPU合計に対するクラスターのすべてのコンテナによって設定されたCPU Requestの合計の比率
メモリアサイン率	%	クラスターのアサイン可能なメモリ合計に対するクラスターのすべてのコンテナによって設定されたメモリRequestの合計の比率
Apiserver正常	-	Apiserver状態、デフォルトでFalseのときにアラームであり、独立したクラスターのみがこの指標をサポートする
Etcd正常	-	Etcd状態、デフォルトでFalseのときにアラームであり、独立したクラスターのみがこの指標をサポートする
Scheduler正常	-	Scheduler状態、デフォルトでFalseのときにアラームであり、独立したクラスターのみがこの指標をサポートする
Controll Manager正常	-	Controll Manager状態、デフォルトでFalseのときにアラームであり、独立したクラスターのみがこの指標をサポートする

ノードアラーム指標

監視指標	単位	説明
CPU利用率	%	ノードの総数に対するノード内のすべてのPodのCPU使用量の比率
メモリ利用率	%	ノードの総数に対するノード内のすべてのPodのメモリ使用量の比率
ノードでのPodリスタート回数	回	ノード内のすべてのPodリスタート回数の合計

監視指標	単位	説明
Node Ready	-	ノードの状態、デフォルトでFalseのときにアラームである

クラスターノードの指標アラームの詳細については、Cloud Virtual Machine監視およびクラウド監視によるアラームポリシーの作成。

クラスターノードのデータディスクの指標アラームの詳細については、Cloud Block Storage監視およびクラウド監視によるアラームポリシーの作成。

Podアラーム指標

監視指標	単位	説明
CPU利用率（ノードに対する）	%	ノードの総数に対するPodのCPU使用量の比率
メモリ利用率（ノードに対する）	%	ノードの総数に対するPodのメモリ使用量の比率
実際のメモリ利用率（ノードに対する）	%	ノードの総数に対するPod内のすべてのContainerの実際のメモリ使用量（キャッシュを除く）の比率
CPU利用率（Limitに対する）	%	設定されたLimit値に対するPodのCPU使用量の比率
メモリ利用率（Limitに対する）	%	設定されたLimit値に対するPodのメモリ使用量の比率
実際のメモリ利用率（Limitに対する）	%	設定されたLimit値に対するPod内のすべてのContainerの実際のメモリ使用量（キャッシュを除く）の比率
Podリスタート回数	回	Podリスタート回数
Pod Ready	-	Podの状態、デフォルトでFalseのときにアラームである
CPU使用量	コア	PodのCPU使用量
メモリ使用量	MB	キャッシュを含むPodのメモリ使用量
実際のメモリ使用量	MB	キャッシュを除くPod内のすべてのContainerの実際のメモリ使用量の合計

ログ管理
コンテナログをCLSに収集する

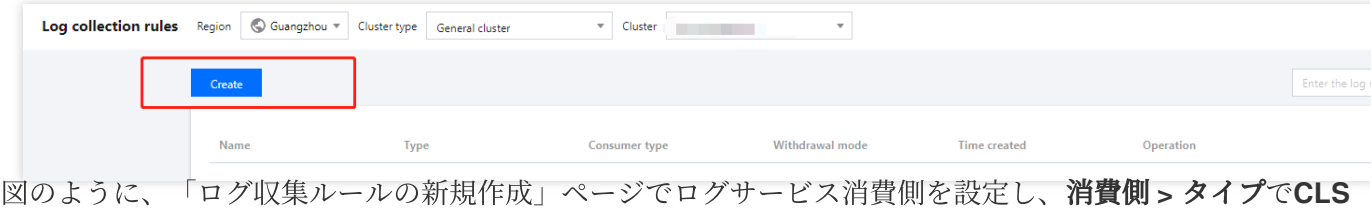
最終更新日：：2023-04-28 15:30:11

ここではTKEコンソールでログ収集ルールを設定し、Tencent Cloud Log Service（CLS）に送信する方法をご紹介します。

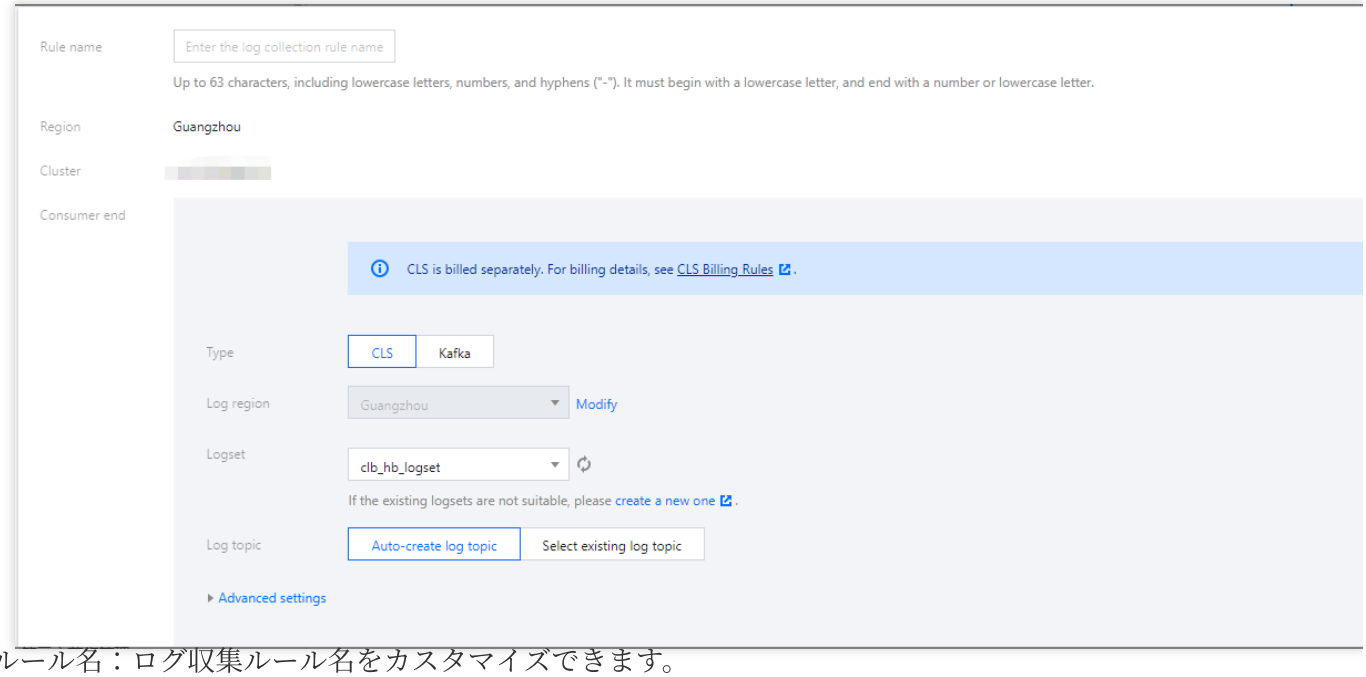
操作手順

ログ収集ルールの作成

1. TKEコンソールにログインし、左側ナビゲーションバーの**ログ管理 > ログルール**を選択します。
2. 下図のように、「ログルール」ページ上方でリージョンおよびログ収集ルールを設定する必要があるクラスターを選択し、**新規作成**をクリックします。

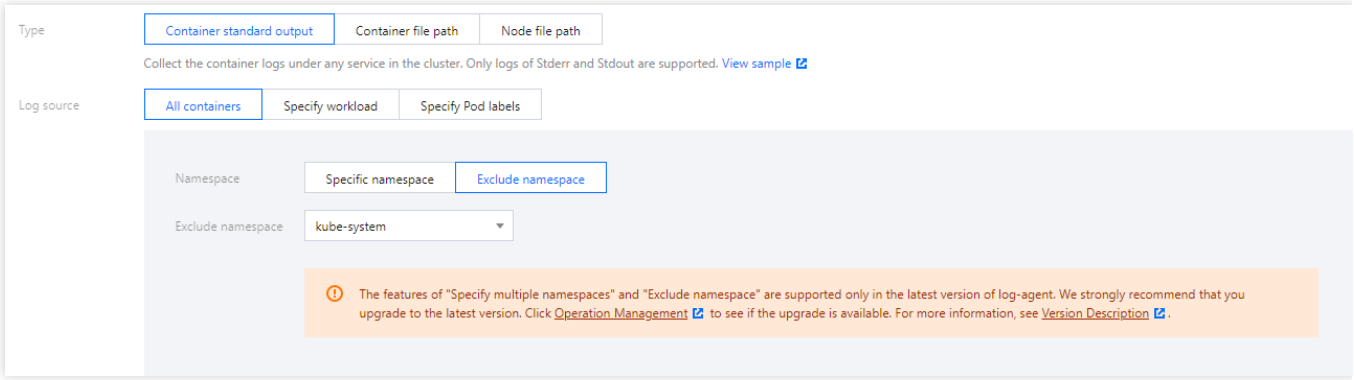


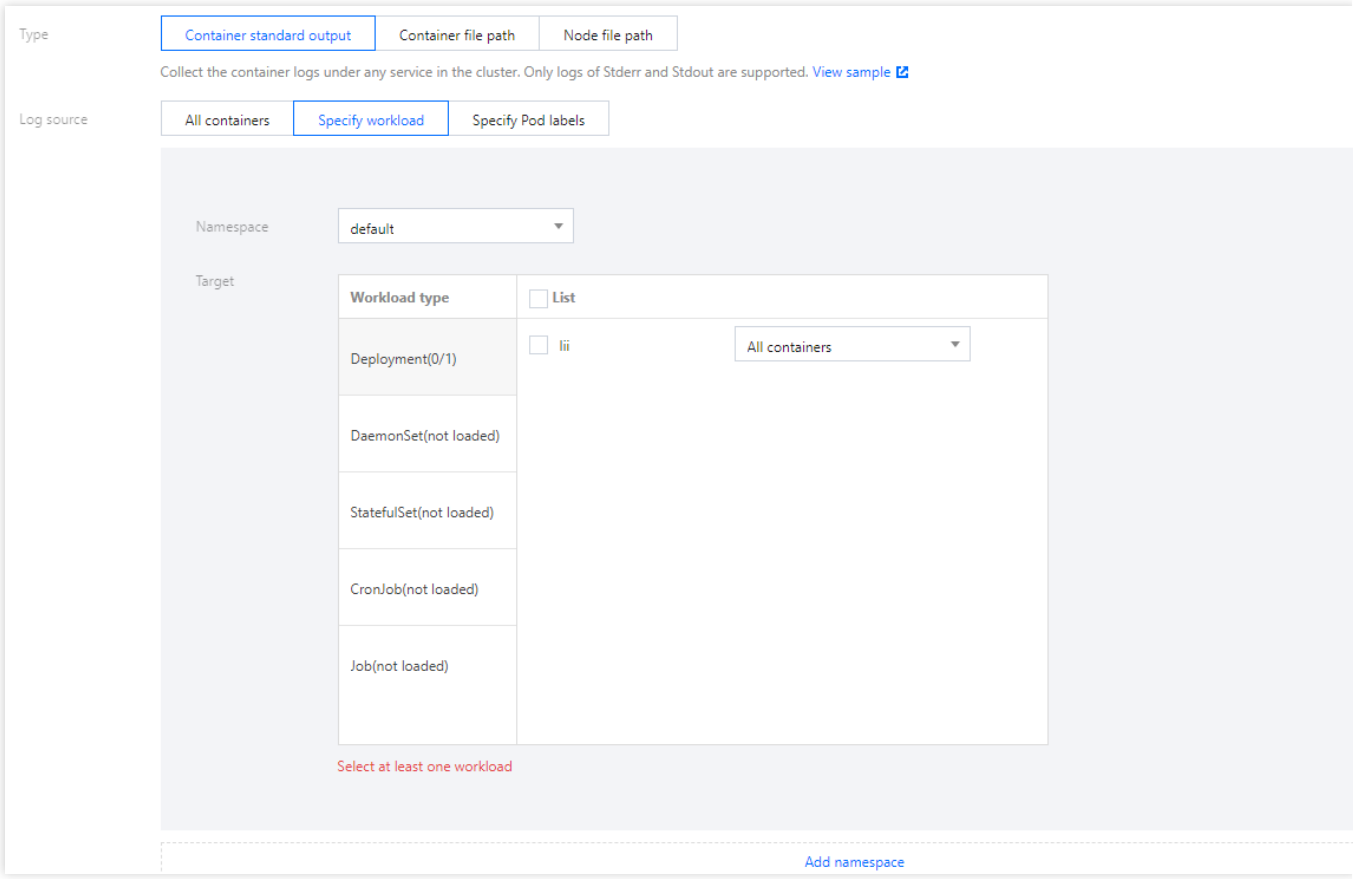
3. 下図のように、「ログ収集ルールの新規作成」ページでログサービス消費側を設定し、**消費側 > タイプ**で**CLS**を選択します。

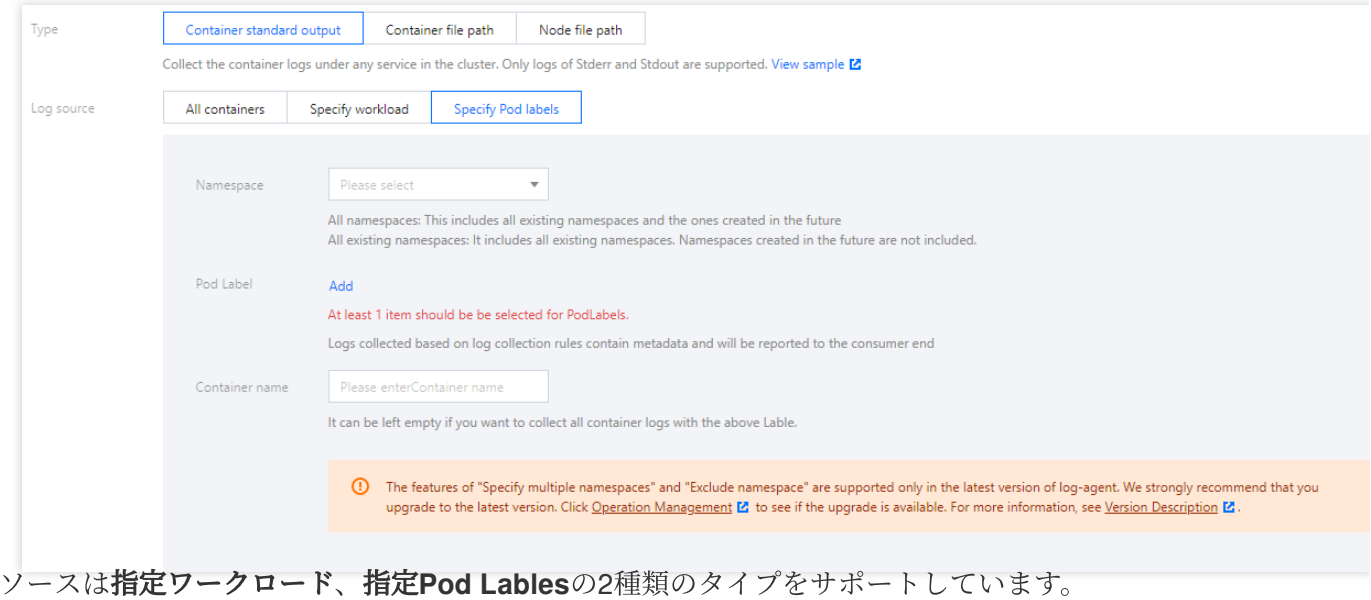


収集ルール名：ログ収集ルール名をカスタマイズできます。

ログ所在リージョン：CLSはクロスリージョンログ配信をサポートしています。「変更」をクリックすることによってログを配信する目標リージョンを選択することができます。
ログセット：ログ所在リージョンに基づいて作成済みのログセットを表示します。既存のログセットが適切でない場合は、CLSコンソールでログセットを新規作成することができます。操作の詳細については、ログセットの作成をご参照ください。
ログトピック：ログセット下で対応するログトピックを選択します。ログトピックを「自動作成」および「既存のものを選択」の2種類のモードをサポートしています。
高度な設定：
デフォルトメタデータ：CLSはデフォルトでメタデータ `pod_name` 、 `namespace` 、 `container_name` をインデックスに設定してログ検索に使用します。
カスタムメタデータ：カスタムメタデータおよびログインデックスをサポートしています。
説明
CLSは国内および海外リージョン間のクロスリージョンログ配信をサポートしていません。CLSに対してログサービスがアクティブ化されていないリージョンは近接配信のみをサポートしています。例えば深圳クラスターが収集したコンテナログは広州への配信のみをサポートし、天津クラスターが収集したコンテナログは北京への配信のみをサポートしています。詳細はコンソールによって確認することができます。
1つのログトピックは現在1つのログの設定のみサポートしています。すなわちログ、審査、イベントは同一のtopicを採用できず、上書きが発生するため、選択するログトピックがその他の収集設定に占有されていないことを確認してください。ログセット下に500個のログトピックが存在する場合、ログトピックを新規作成することはできません。
カスタムメタデータおよびメタデータベースのオープンインデックスは作成後に変更できません。CLSコンソールで変更できます。
4. 収集タイプを選択し、ログソースを設定します。現在収集タイプは**コンテナ標準出力、コンテナファイルパス**および**ノードファイルパス**をサポートしています。
コンテナ標準出力ログ
コンテナ内のファイルログログ
ノードファイルログ
下図のように、ログソースは**すべてのコンテナ、指定ワークロード、指定 Pod Lables**の3つのタイプをサポートしています。

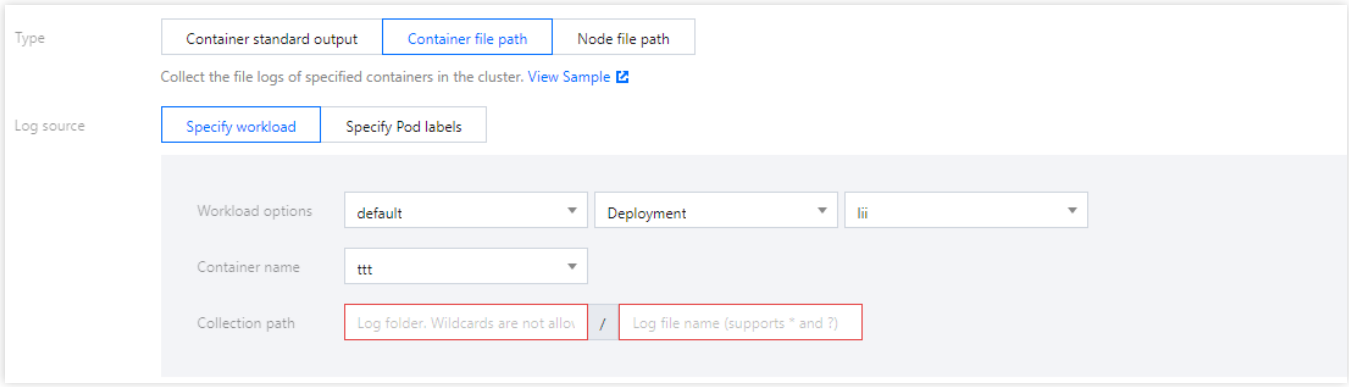


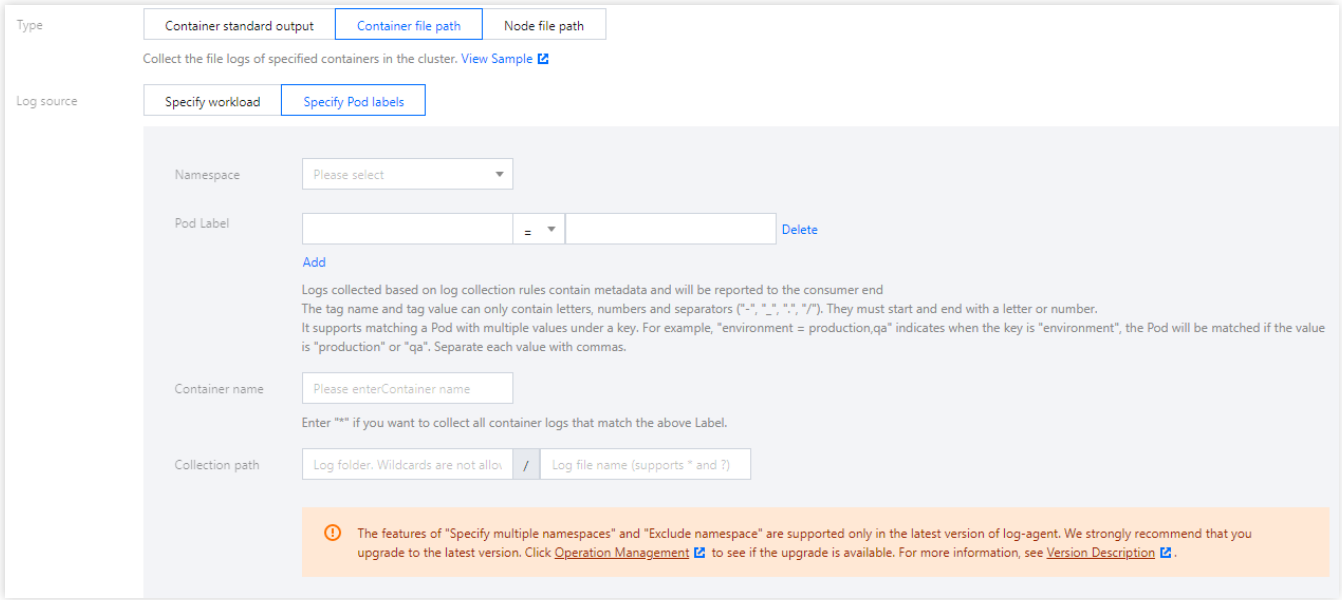




ログソースは**指定ワークロード**、**指定Pod Lables**の2種類のタイプをサポートしています。

ファイルパスの収集はファイルパスおよびワイルドカードルールをサポートしています。例えばコンテナファイルパスが `/opt/logs/*.log` の場合、収集パスを `/opt/logs` 、ファイル名を `*.log` として指定できます。下図に示すとおりです。

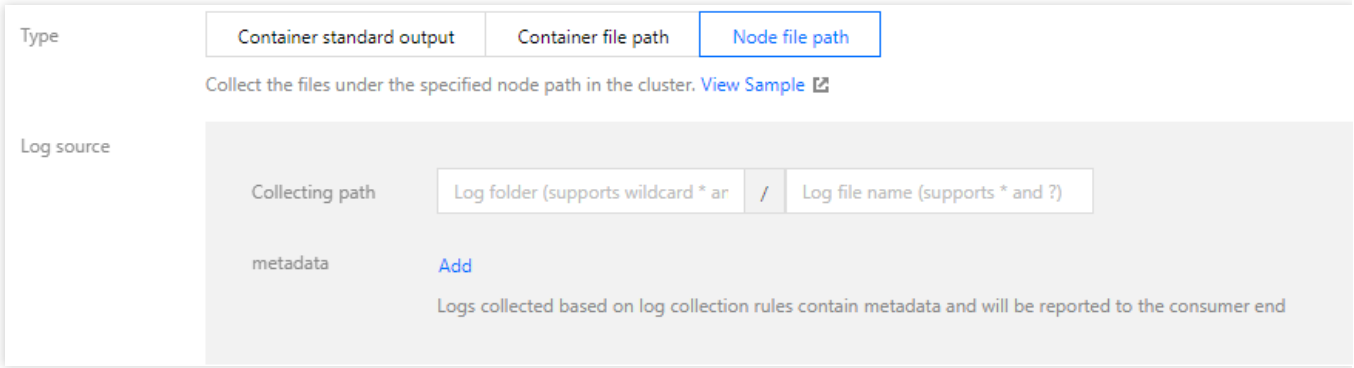




注意

「コンテナファイルパス」は**ソフトリンクにすることはできません**。そうしない場合、ソフトリンクの実際のパスがコレクターのコンテナ内に存在しなくなり、ログの収集が失敗します。

収集パスはファイルパスおよびワイルドカードルール方式の入力をサポートしています。例えばすべてのファイルパスを `/opt/logs/service1/*.log` 、 `/opt/logs/service2/*.log` の形式で収集する必要がある場合、収集パスのフォルダを `/opt/logs/service*` 、ファイル名を `*.log` として指定できます。
実際のニーズに応じてKey-Value形式のmetadataをカスタマイズして追加することができ、metadataはログレコード内に追加されます。



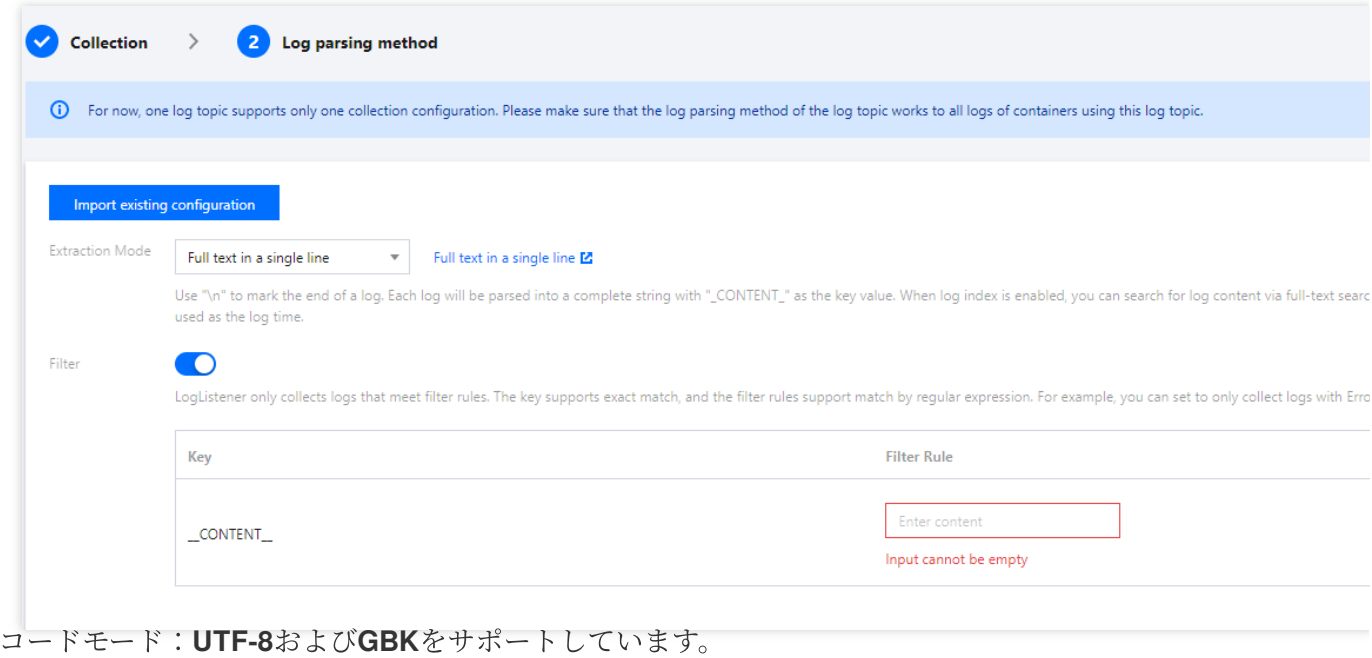
注意：
「ノードファイルパス」はソフトリンクにすることができません。そうしない場合、ソフトリンクの実際のパスがコレクターに存在しなくなり、ログの収集が失敗します。
1つのノードのログファイルは1つのログトピックのみによって収集されます。

説明
「コンテナの標準出力」および「コンテナ内ファイル「（「ノードファイルパス」、すなわちhostPathマウントは含みません）は、オリジナルのログ内容を除き、 コンテナまたはkubernetesに関連するメタデータを追加して（ログを生成するコンテナ ID）一緒にCLSに報告し、ユーザーがログを確認する時にソースの追跡またはコンテナ識別子に基づき、特徴（コンテナ名、labels）検索を行うことができるようにします。
コンテナまたはkubernetesに関連するメタデータについては下部の表をご参照ください。

フィールド名	意味
container_id	ログが属するコンテナIDです。
container_name	ログが属するコンテナ名です。
image_name	ログが属するコンテナのイメージ名IPです。
namespace	ログが属するpodのnamespaceです。
pod_uid	ログが属するpodのユーザーIDです。
pod_name	ログが属するpod名です。

pod_lable_{label name}	ログが属するpodのlabelです（例えば1つのpodに、app=nginx、env=prodという2つのlabelがある場合、アップロードされたログにはpod_label_app:nginx、pod_label_env:prodという2つのmetedataが添付されます）。

5. 収集ポリシーを設定します。**全量**または**増分**を設定できます。
全量：全量収集はログファイルの始めから収集することを指します。
増分：増分収集はファイル末尾から1Mまでの部分から収集を開始することを指します（ログファイルが1M未満の場合、全量収集と同等の効果を有します）。
6. 下図のように、**次へ**をクリックし、ログ解析方式を選択します。



エンコードモード：**UTF-8**および**GBK**をサポートしています。
抽出モード：さまざまなタイプの抽出モードをサポートしています。詳細は次のとおりです。

解析モード	説明	関連ドキュメント
フルテキストを１行で記入	1つのログは1行の内容のみを含み、改行コード\\nを1つのログの終了マーカーとし、各ログはキー値がCONTENTの完全な文字列として解析され、インデックスを有効化すると全文検索によってログ内容を検索できます。ログ時間は収集時間を基準とします。	シングルライン全文形式
フルテキストを複数行で記入	1つの完全なログが複数行に跨がることを意味し、1行目の正規表現を採用する方式でマッチングを行い、ある行のログが予め設定した正規表現にマッチングした場合、1つのログの始まりであると認識します。次の行の先頭はこのログの終了識別子として表示されます。デフォルトのキー値CONTENTも設定され、ログ時間は収集時間を基準とします。正規表現の自動生成をサポートしています。	マルチライン全文形式

シングルライン-完全な正規表現	1つの完全なログから正規表現形式で複数のkey-valueのログを抽出する解析モードを指します。まずログサンプルを入力し、次にカスタマイズされた正規表現を入力する必要があります。システムは正規表現内のキャプチャグループに基づいて対応するkey-valueを抽出します。正規表現の自動生成をサポートしています。	シングルライン-完全な正規表現形式
マルチライン-完全な正規表現	ログテキスト内の複数行にまたがる完全なログデータ（例：Javaプログラムログ）に適用され、正規表現に基づいて複数のkey-valueキー値のログ解析モードを抽出します。まずログサンプルを入力し、次にカスタマイズされた正規表現を入力する必要があります。システムは正規表現内のキャプチャグループに基づいて対応するkey-valueを抽出します。正規表現の自動生成をサポートしています。	マルチライン-完全な正規表現形式
JSON	JSON形式のログは、第1階層のkeyを対応するフィールド名として自動的に抽出します。第1階層のvalueは対応するフィールドの値とします。このような方法によりログ全体の構造化処理を行い、それぞれの完全なログは改行文字 `\\n` を終了識別子とします。	JSON形式
区切り文字	1つのログデータは指定した区切り文字に基づいてログ全体に構造化処理を行い、それぞれの完全なログは改行コード \\nを終了識別子とします。ログサービスが区切り文字形式のログ処理を行う時に、各個別のフィールドに一意のkeyを定義する必要があり、無効フィールド、すなわち収集する必要がないフィールドは空欄を埋めることができ、すべてのフィールドが空であることはサポートしていません。	区切り文字形式

フィルタ：LogListenerはフィルタルールに一致するログのみを収集します。Keyは完全一致をサポートし、フィルタリングルールは `ErrorCode = 404` のログのような正規表現のマッチングをサポートします。必要に応じてフィルタを有効化してルールを設定することができます。

説明

1つのログトピックは現在1つの収集設定のみをサポートしています。そのログトピックを選択したすべてのコンテナのログが選択するログ解析方式を受け入れることができることを確認してください。同一ログトピックで異なる収集設定を新規作成した場合、古い収集設定は上書きされます。

7. **完了**をクリックすると、CLSに配信されるコンテナログ収集ルールの作成が完了します。

ログルールの更新

1. TKEコンソールにログインし、左側ナビゲーションバーの**ログ管理 > ログルール**を選択します。
2. 下図のように、「ログルール」ページの上方でリージョンおよびログ収集ルールを更新する必要があるクラスターを選択し、右側の**収集ルールの編集**をクリックします。

3. 必要に応じて対応する設定を更新し、**完了**をクリックします。

注意

ログセットおよびログトピックは更新できません。

関連ドキュメント

コンソールを使用してログ収集を設定できるだけでなく、リソースのカスタマイズ（CustomResourceDefinitions、CRD）の方式でログ収集を設定することができます。詳細については、YAMLによってCRDを使用してログ収集を設定するをご参照ください。

CRDを使用したログ収集の設定

最終更新日：：2023-04-26 16:53:29

ユースケース

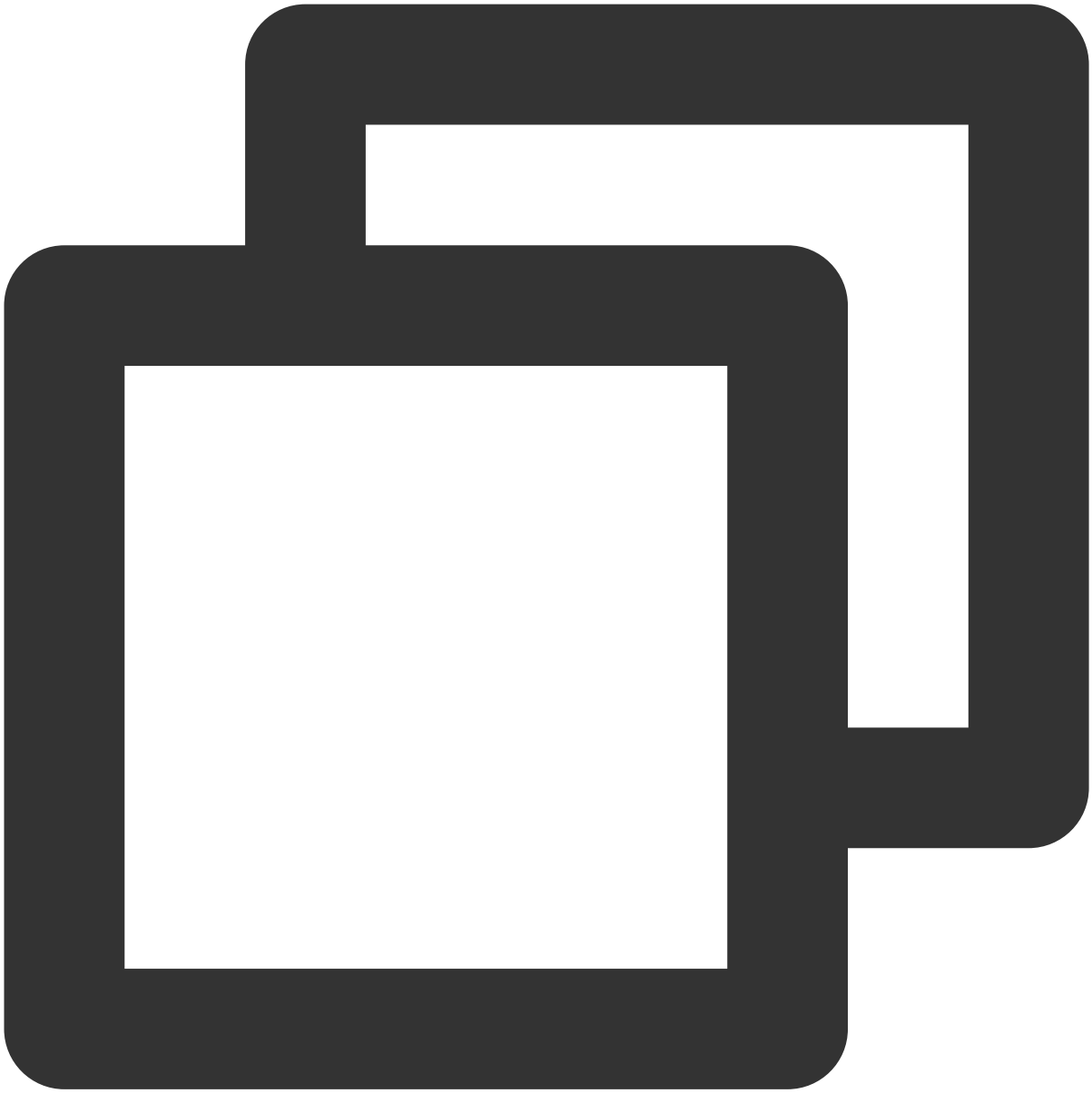
ログ収集の設定は、コンソールを使用してログ収集の設定を行うだけでなく、CustomResourceDefinitions（CRD）方式でも行うことができます。CRDはコンテナ標準出力、コンテナファイルおよびホストファイルの収集をサポートし、複数の形式でのログ収集をサポートしています。CLSへの配信およびCKafkaなどのコンシューマー側をサポートします。

前提条件

TKEコンソールの運用保守機能管理でログ収集を有効にしていること。

CRDの説明

構造一覧



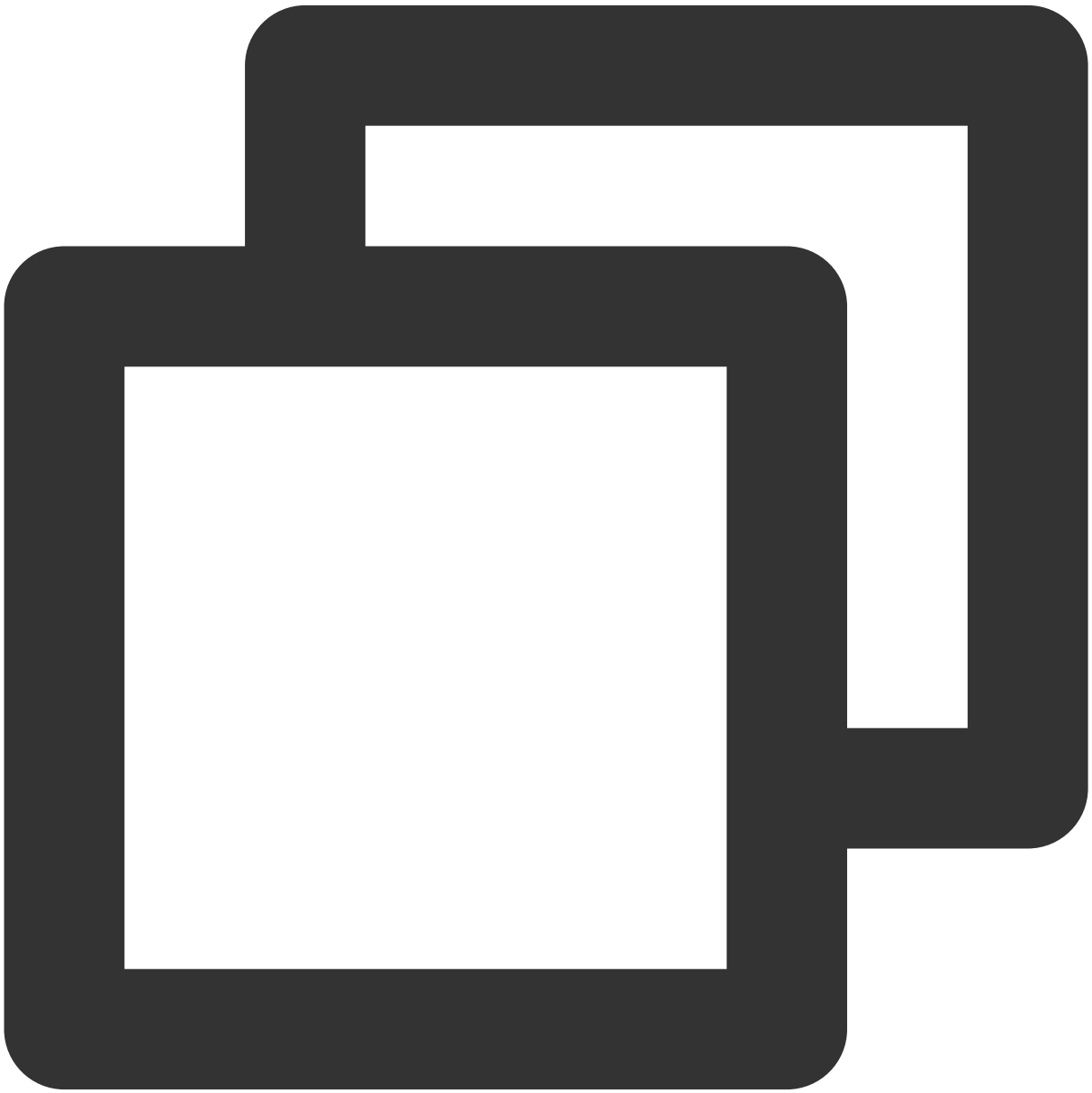
```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig                  ## デフォルト値
metadata:
  name: test                                    ## CRDリソース名。クラスター内で一意
spec:
  clsDetail:                                    ## CLSへの配信の設定
    ...
  inputDetail:                   ## 収集データソース設定
    ...
  kafkaDetail:                                  ## ckafkaまたは自作kafkaへの配信の設定
    ...
```

```
status:                                                    ## CRDリソースの状態
  status: ""
  code: ""                                                 ## インターフェース呼び出しエラー
  reason: ""                                          ## エラー理由
```

clsDetailフィールドの説明

注意

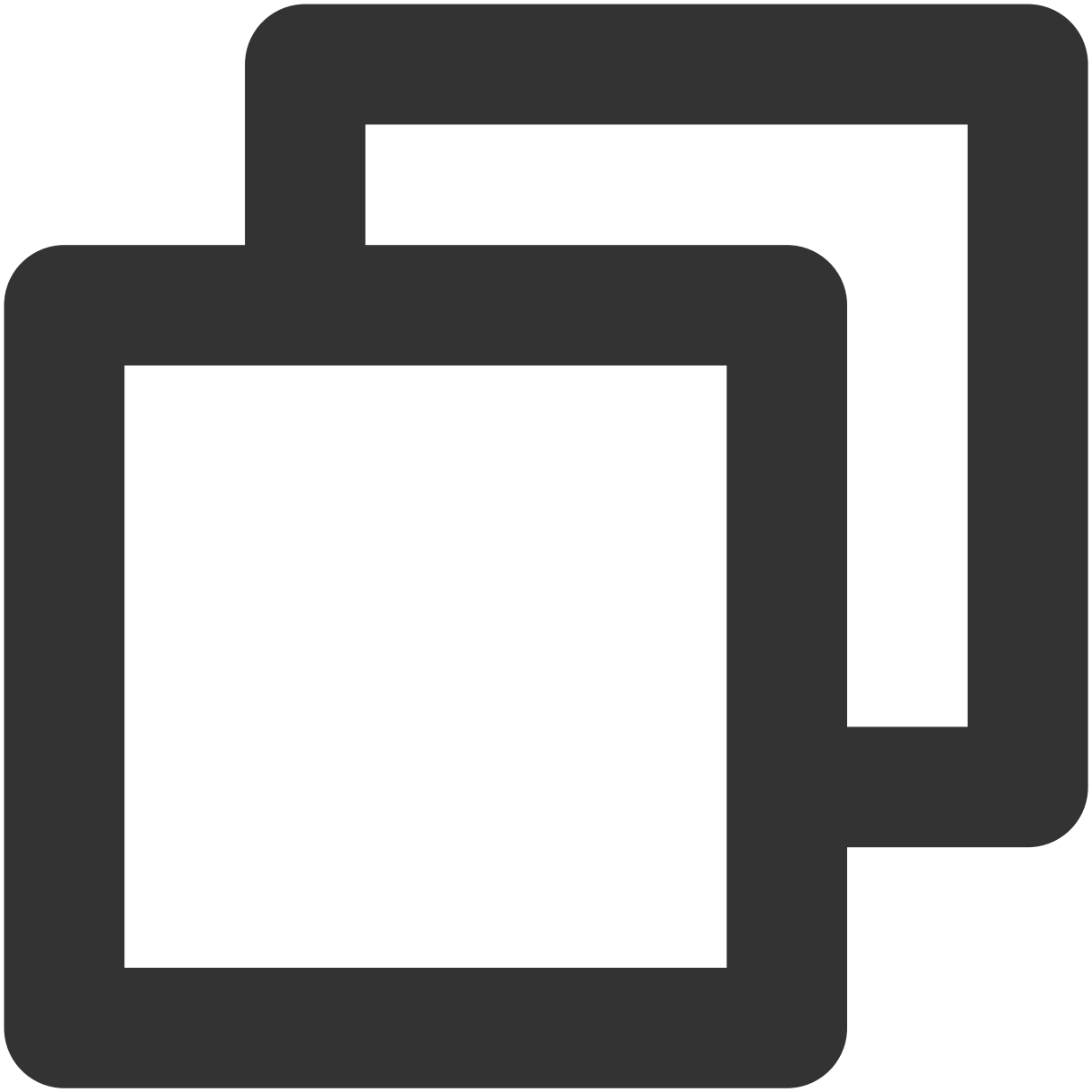
topicは指定後に変更できません。



```
  clsDetail:
```

```
## ログトピックを自動作成します。ログセットおよびトピックのnameを同時に指定する必要があります
logsetName: test                        ## CLSログセットのname。このnameのログセットがな
topicName: test                         ## CLSログトピックのname。このnameのログトピックが

# 既存のログセットとログトピックを選択します。ログセットが指定されていて、ログトピックが指定さ
logsetId: xxxxxx-xx-xx-xx-xxxxxxxx  ## CLSログセットのID。ログセットはCLS内にあらかじ
topicId: xxxxxx-xx-xx-xx-xxxxxxxx   ## CLSログトピックのID。ログトピックはCLS内にあら

logType: json_log  ## ログ収集の形式。json_logはjson形式を表し、delimiter_logは区切り
logFormat: xxx                          ## ログのフォーマット
period: 30                                              ## ライフサイクル。単位は日、可
partitionCount:                         ## Integerタイプで、ログトピックのパーティション数
tags:                                 ## タグ記述リスト。このパラメータを指定することで、タ
 - key: xxx                                                ## タグkey
 value: xxx                          ## タグvalue
autoSplit: false                                        ## booleanタイプ、自動分割を有
maxSplitPartitions:
storageType: hot                      ## ログトピックのストレージタイプ。オプション値はhot（
excludePaths:                           ## ブラックリストパスリストの収集
  - type: File                                                ##  タイプ、オプショ
      value: /xx/xx/xx/xx.log           ## typeの対応する値
indexs:                                                     ## topic作成時にカス
  - indexName:    ## キー値またはメタフィールドインデックスのフィールドを設定する必要があり
      indexType:  ## フィールドタイプ。現在はlong、text、doubleタイプをサポートしていま
      tokenizer:  ## フィールドの区切り文字。この中のそれぞれの文字が区切り文字を表します。
      sqlFlag:    ## boolean フィールドで分析機能を有効にしているかどうか
      containZH: ## boolean 中国語が含まれるかどうか
region: ap-xxx                          ## topicの所在リージョン。クロスリージョン配信に使用
userDefineRule: xxxxxx                  ## ユーザー定義の収集ルール。Json形式でシリアライズ
extractRule: {}                         ## 抽出、フィルタリングルール。 ExtractRuleを設定し
```

inputDetailフィールドの説明



```
inputDetail:
  type: container_stdout    ## ログ収集のタイプ。container_stdout（コンテナ標準出力）、co

  containerStdout:          ## コンテナ標準出力
    namespace: default      ## 収集コンテナのkubernetesネームスペース。複数のネームスペース
    excludeNamespace: nm1,nm2    ## 収集コンテナのkubernetesネームスペースを除外します。複
    nsLabelSelector: environment in (production),tier in (frontend) ## ネームスペー
    allContainers: false        ## 指定のネームスペース内のすべてのコンテナの標準出力を収集
    container: xxx              ## ログを収集するコンテナ名。空の場合は、コンテナに合致する
    excludeLabels:  ## 収集に指定のlabelを含むPodを含めません。workload、namespace、exc
      key2: value2  ## 同一のkey下にある複数のvalue値のpodのマッチングをサポートします。例
```

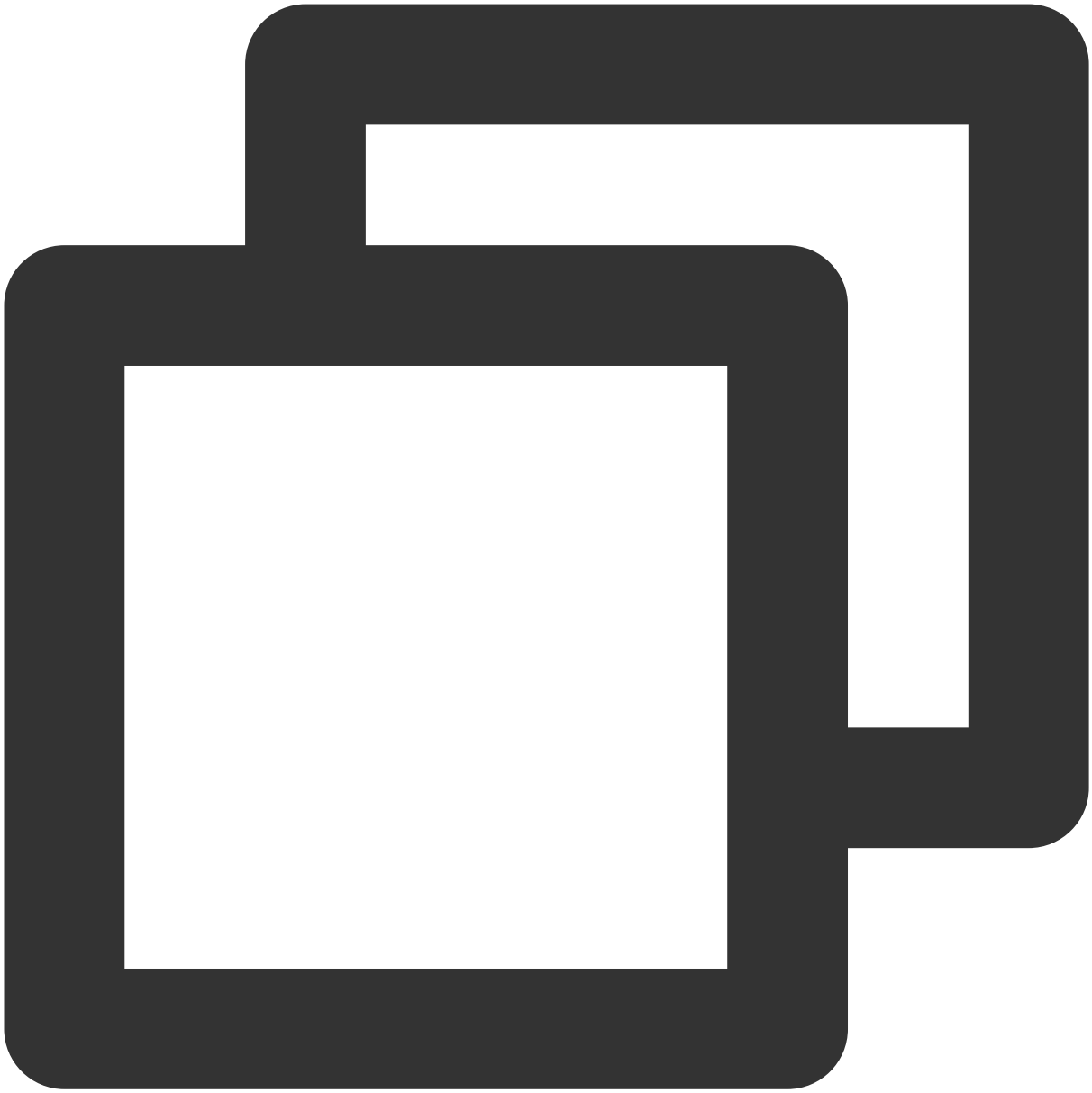
```
    includeLabels:  ## 指定のlabelを含むPodを収集します。workload、namespace、excludeN
      key: value1   ## 収集ルールで収集されたログにはmetadataが含まれ、コンシューマー側にレ

    metadataLabels:             ## 具体的にどのpod labelがメタデータとして収集されるかを指
    - label1
    customLabels:               ## ユーザー定義のmetadata
      label: l1

    workloads:
    - container: xxx    ## 収集するコンテナ名。指定しない場合はworkload Pod内のすべてのコ
      kind: deployment  ## workloadタイプ。deployment、daemonset、statefulset、job、
      name: sample-app  ## workloadの名前
      namespace: prod   ## workloadのネームスペース

  containerFile:  ## コンテナ内のファイル
    namespace: default      ## 収集コンテナのkubernetesネームスペース。1つのネームスペー
    excludeNamespace: nm1,nm2   ## 収集コンテナのkubernetesネームスペースを除外します。複
    nsLabelSelector: environment in (production),tier in (frontend) ## ネームスペー
    container: xxx          ## ログを収集するコンテナ名。 * の場合は、コンテナに合致するロ
    logPath: /var/logs      ## ログフォルダ。ワイルドカードはサポートしていません
    filePattern: app_*.log  ## ログファイル名。ワイルドカード * と ? をサポートしています。
    customLabels:   ## ユーザー定義のmetadata
      key: value
    excludeLabels:  ## 収集に指定のlabelを含むPodを含めません。workloadとは同時に指定でき
      key2: value2  ## 同一のkey下にある複数のvalue値のpodのマッチングをサポートします。例

    includeLabels:  ## 指定のlabelを含むPodを収集します。workloadとは同時に指定できません
      key: value1   ## 収集ルールで収集されたログにはmetadataが含まれ、コンシューマー側にレ
    metadataLabels:         ## 具体的にどのpod labelがメタデータとして収集されるかを指定し
    - label1                ## pod label
    workload:
      container: xxx        ## 収集するコンテナ名。指定しない場合はworkload Pod内のすべての
      name: sample-app      ## workloadの名前

  hostFile:                 ## ノードファイルパス
    filePattern: '*.log'    ## ログファイル名。ワイルドカード * と ? をサポートしています。
    logPath: /tmp/logs      ## ログフォルダ。ワイルドカードはサポートしていません
    customLabels:           ## ユーザー定義のmetadata
      label1: v1
```

extractRuleの対象の説明

名前	タイプ	入力必須項目	説明

timeKey	String	いいえ	時間フィールドのkeyの名前。time_keyとtime_formatは必ず対になります。
timeFormat	String	いいえ	時間フィールドの形式。C言語のstrftime関数の時間形式についての説明を参照してパラメータを出力します。
delimiter	String	いいえ	区切り文字タイプのログの区切り文字。log_typeがdelimiter_logの場合のみ有効です。
logRegex	String	いいえ	ログ全体のマッチングルール。log_typeがfullregex_logの場合のみ有効です。
beginningRegex	String	いいえ	行頭のマッチングルール。log_typeがmultiline_logまたはfullregex_logの場合のみ有効です。
unMatchUpload	String	いいえ	解析に失敗したログをアップロードするかどうか。trueはアップロードすること、falseはアップロードしないことを表します。
unMatchedKey	String	いいえ	失敗したログのkey。
backtracking	String	いいえ	増分収集モードでのバックトラッキングデータ量。デフォルトでは-1（全量収集）であり、0は増分を表します。
keys	Array of String	いいえ	取得した各フィールドのkeyの名前。空のkeyはこのフィールドが破棄されたことを表します。log_typeがdelimiter_logの場合のみ有効です。json_logのログはjson自体のkeyを使用します。
filterKeys	Array of String	いいえ	フィルタリングしたいログのkey。添え字によってFilterRegexに対応します。
filterRegex	Array of String	いいえ	フィルタリングしたいログのkeyに対応するregex。添え字によってFilterKeysに対応します。
isGBK	String	いいえ	Gbkエンコードかどうか。0:いいえ、1: はい。 **注意**：このフィールドは、有効な値が取得できないことを表すnullを返す場合があります。
jsonStandard	String	いいえ	標準jsonかどうか。0:いいえ、1: はい。 **注意**：このフィールドは、有効な値が取得できないことを表すnullを返す場合があります。

kafkaDetailフィールドの説明



```
kafkaDetail:
  brokers: x.x.x.x:p     ## 入力必須、brokerアドレス。通常はドメイン名:ポートであり、複数の
  topic: test                ##
  kafkaType: CKafka      ## kafkaタイプ。CKafka – ckafka、SelfBuildKafka – 自作kafka
  instanceId: xxxx       ## kafkaType = CKafkaの場合、ckafkaインスタンスidを設定します
     logType: minimalist_log ## kafkaログ解析タイプ。"minimalist_log"または""はシングル
  timestampFormat: xxx    ## タイムスタンプの形式。デフォルトではdouble
  timestampKey: xxx       ## タイムスタンプのkey値。デフォルトでは"@timestamp"
  metadata:
    formatType: default    ## metatdata形式。 "default"はデフォルト形式（eks kafkaキャ
  messageKey:              ## 1個のKeyを指定し、ログを指定のパーティションに配信することがで
```

```
        value: Field         ## 入力必須、topicID
        valueFrom:
          fieldRef:
            fieldPath: metadata.name ##  keyがFieldの場合にmetadata.name、metadata.name
```

statusフィールドの説明

status	説明
ステータスが空	初期状態
Synced	収集設定の処理に成功しました
Stale	収集設定の処理に失敗しました

CRDの例

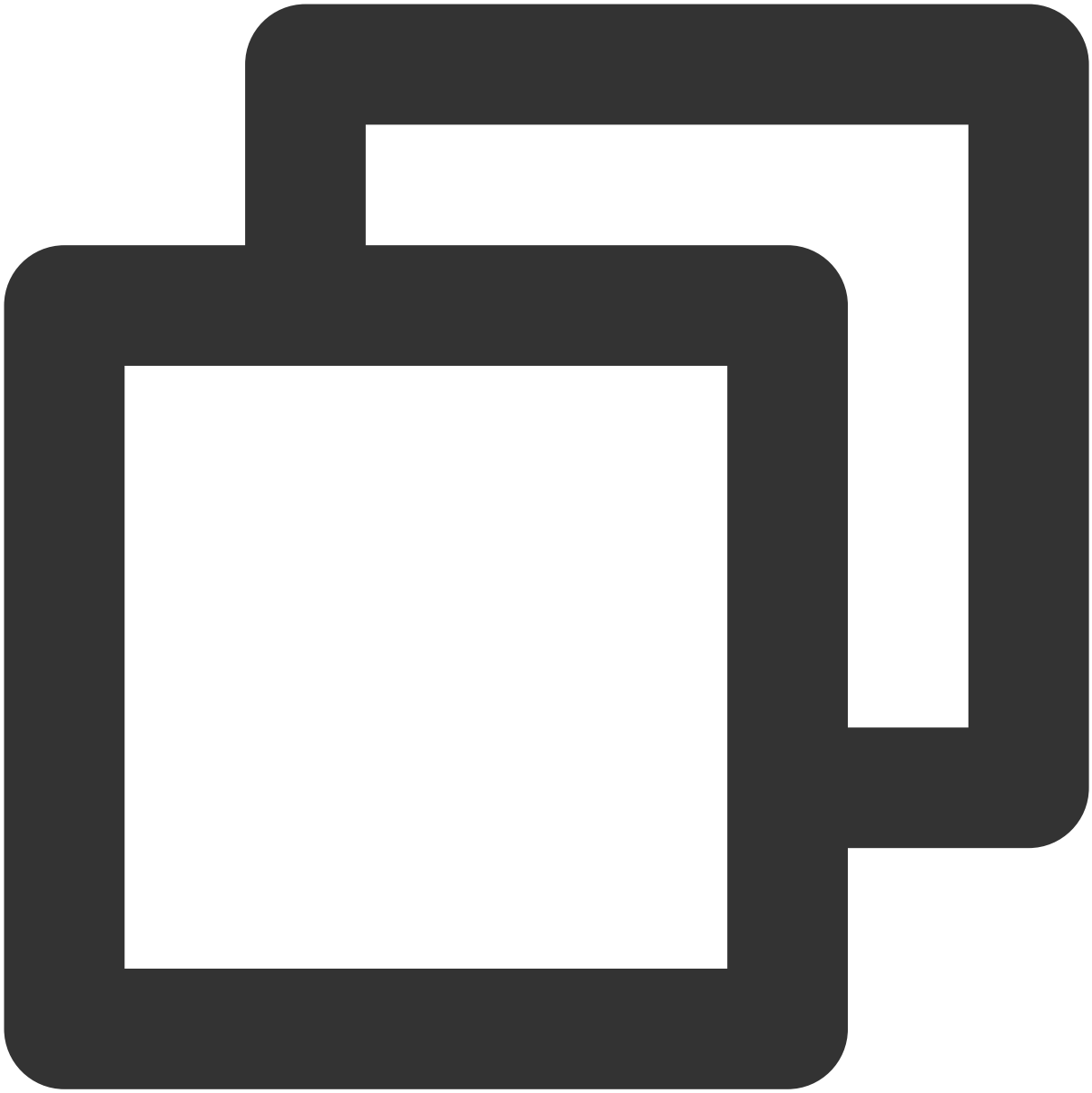
コンテナ標準出力CRDの設定の例

すべてのコンテナ

ワークロードを指定する

指定 pod Labels

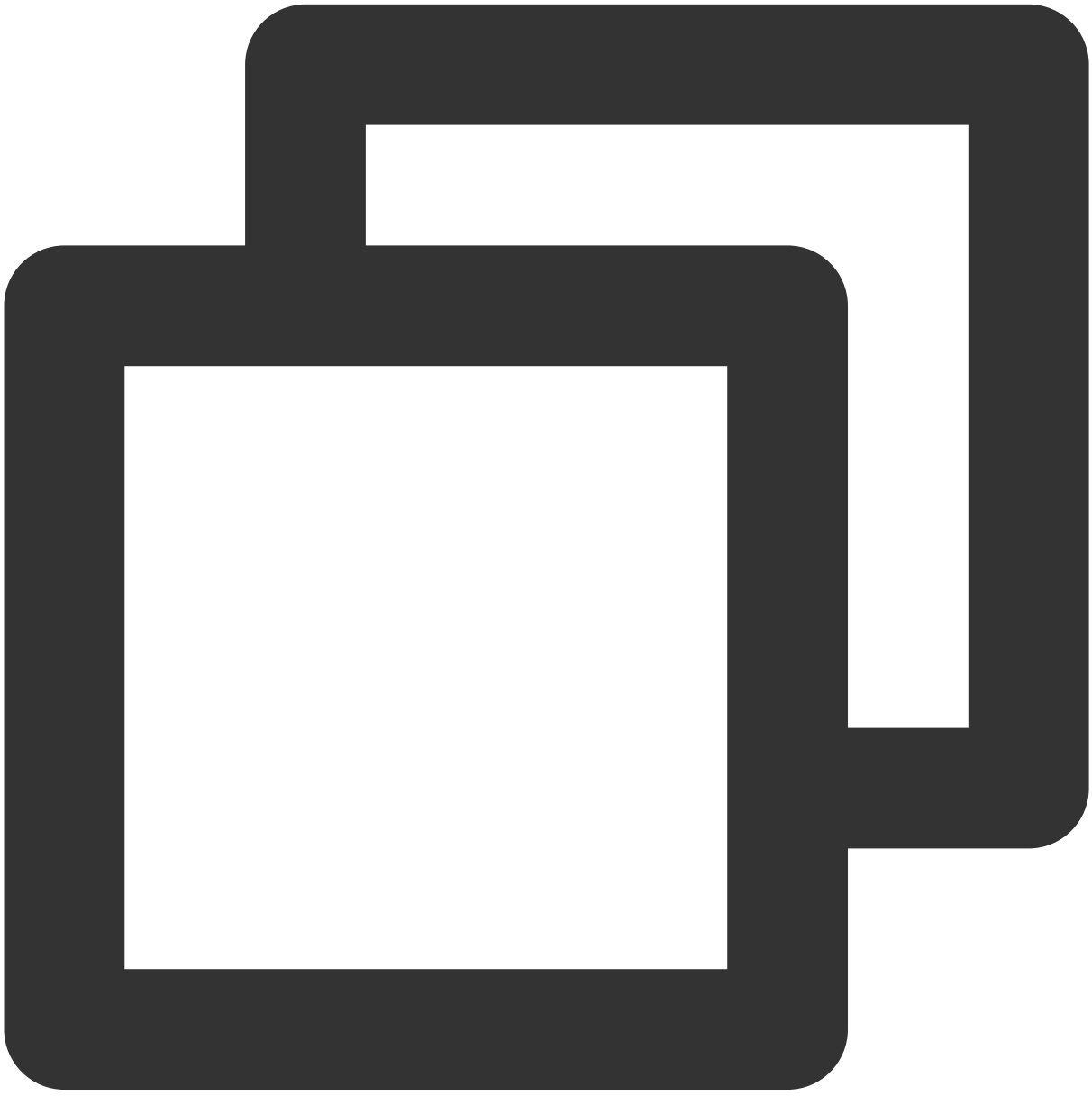
ネームスペースの指定



```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
metadata:
  name: "test"
spec:
  clsDetail:
    .......
    topicId: xxxxxx-xx-xx-xx-xxxxxxxx
  inputDetail:
    containerStdout:
      allContainers: true
```

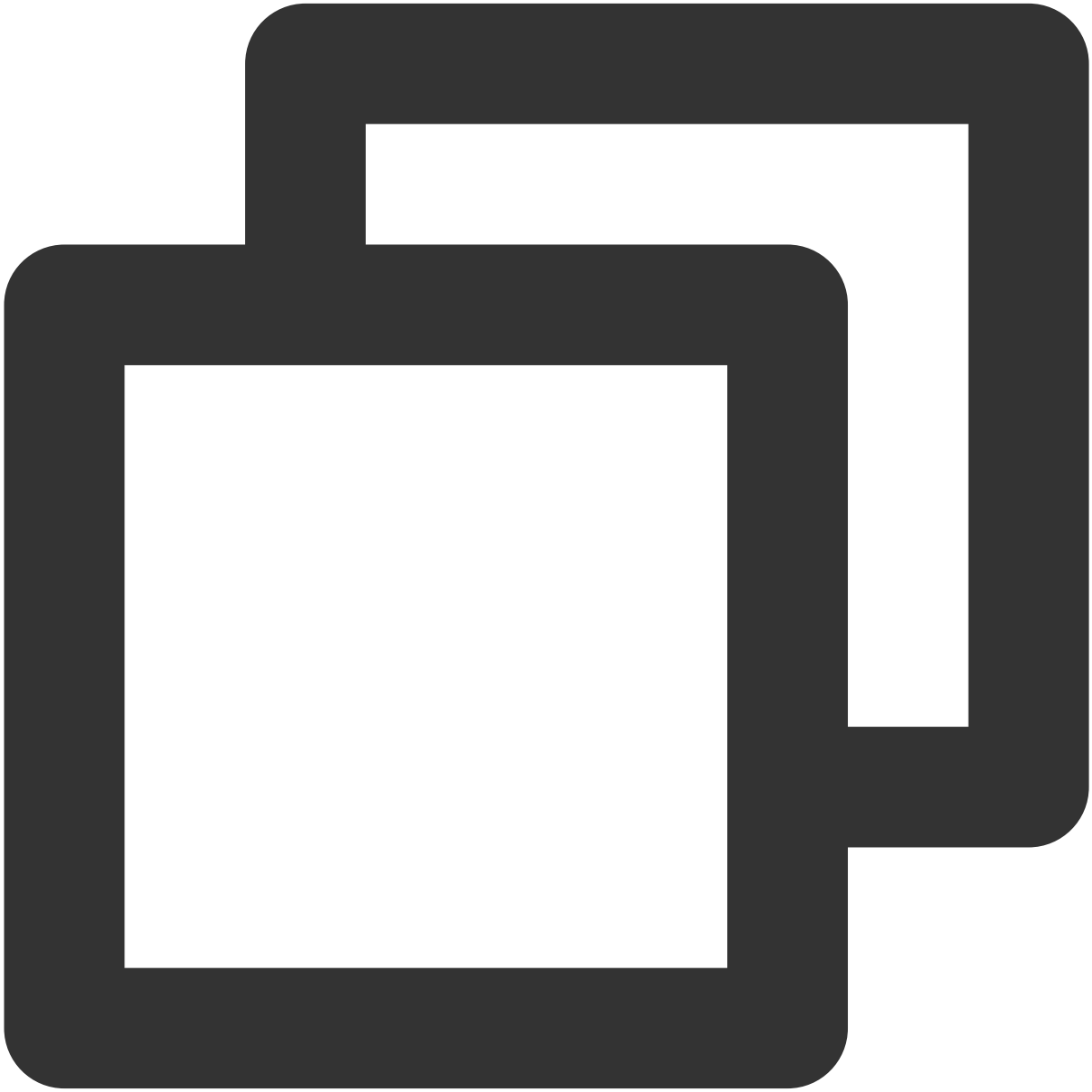
```
      namespace: default,kube-public
    type: container_stdout
```

ネームスペースの除外



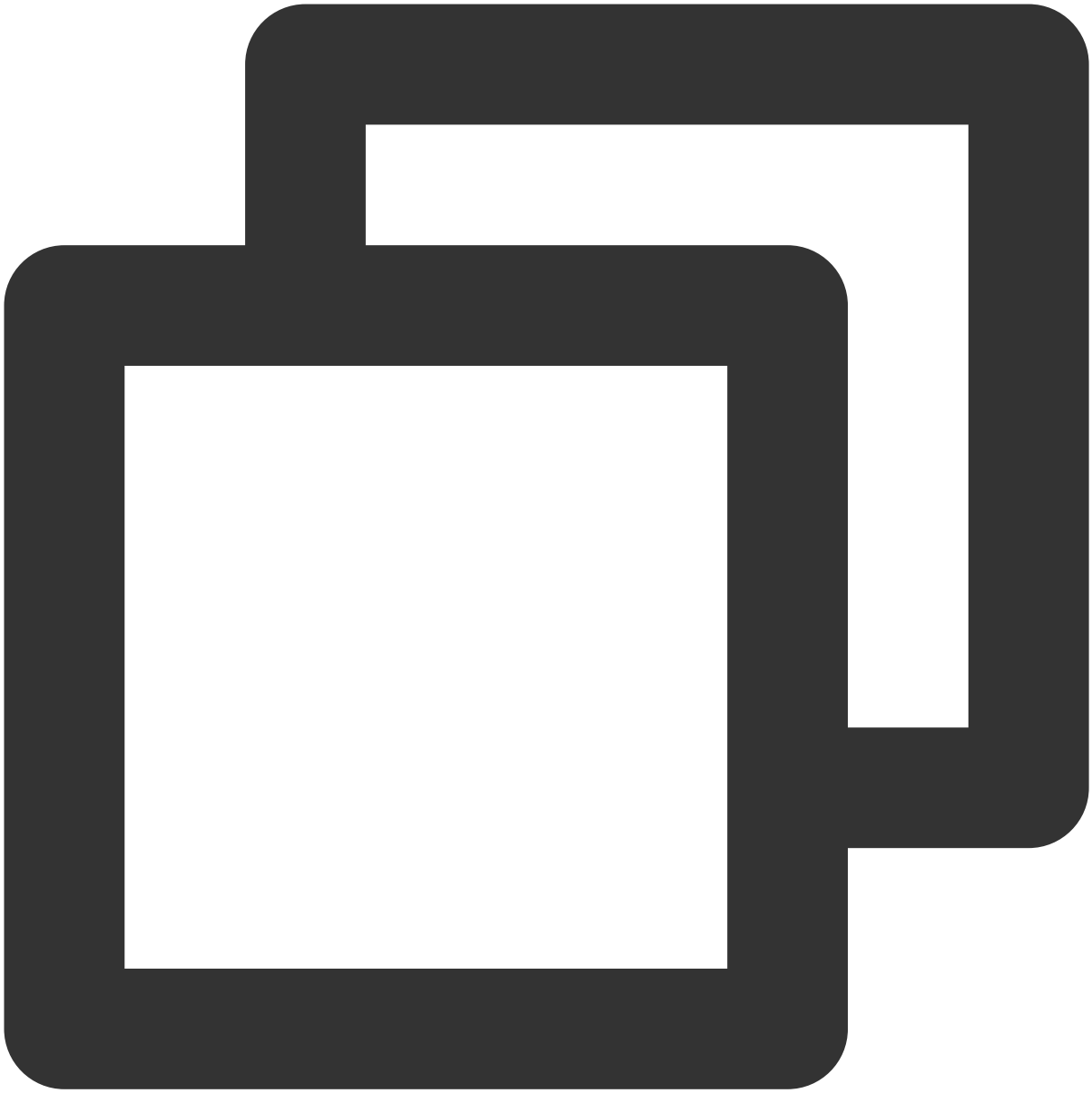
```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
metadata:
  name: "test"
spec:
  clsDetail:
```

```
    ........
    topicId: xxxxxx-xx-xx-xx-xxxxxxxx
  inputDetail:
    containerStdout:
      allContainers: true
      excludeNamespace: kube-system,kube-node-lease
    type: container_stdout
```



```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
```

```
metadata:
  name: "test"
spec:
  clsDetail:
    ......
    topicId: xxxxxx-xx-xx-xx-xxxxxxxx
  inputDetail:
    containerStdout:
      allContainers: false
      workloads:
      - container: prod
        kind: deployment
        name: sample-app
        namespace: kube-system
    type: container_stdout
```



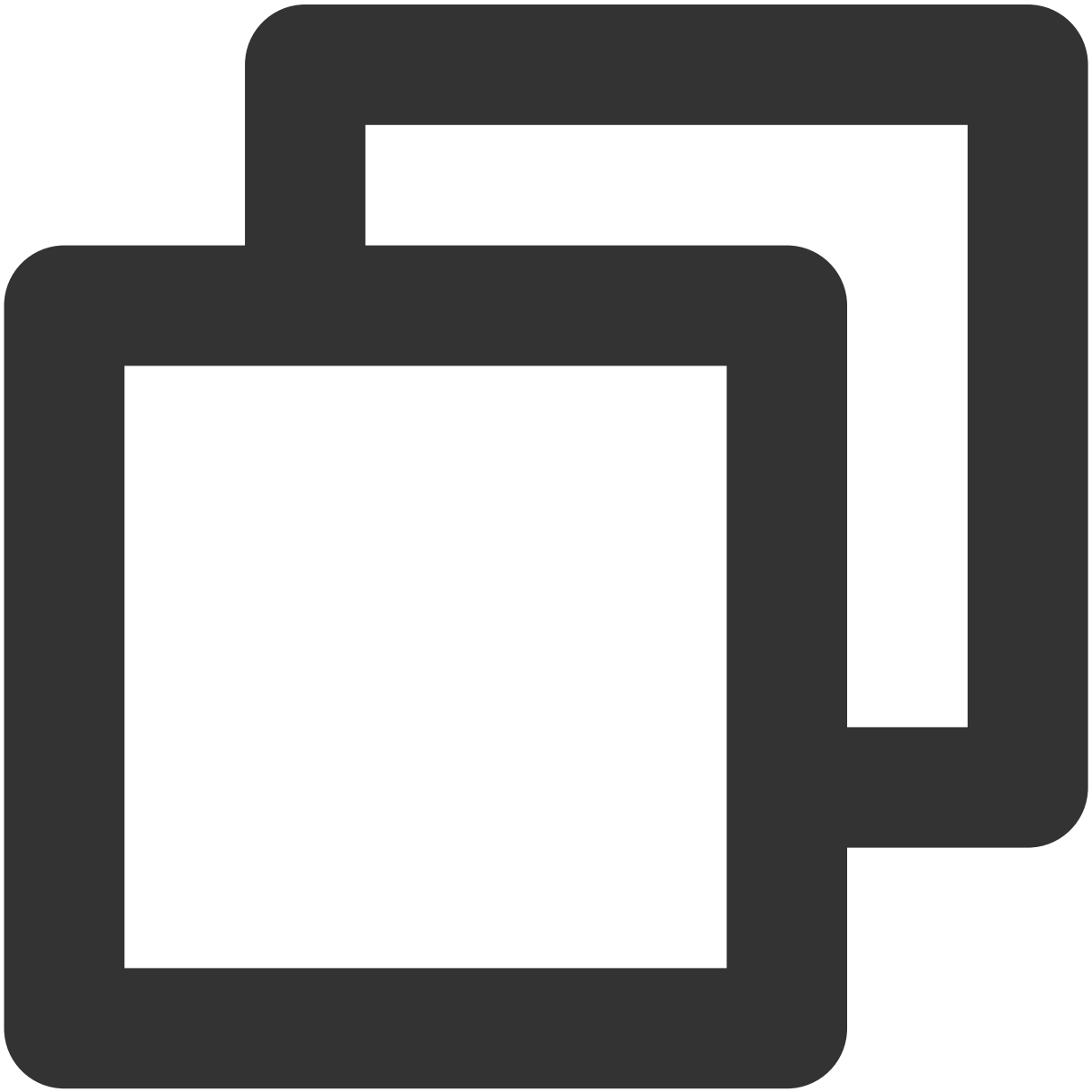
```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
metadata:
  name: test
spec:
  clsDetail:
    ......
    topicId: xxxxxx-xx-xx-xx-xxxxxxxx
  inputDetail:
    containerStdout:
      container: prod
```

```
      excludeLabels:
        key2: v2
      includeLabels:
        key1: v1
      namespace: default,kube-system
    type: container_stdout
```

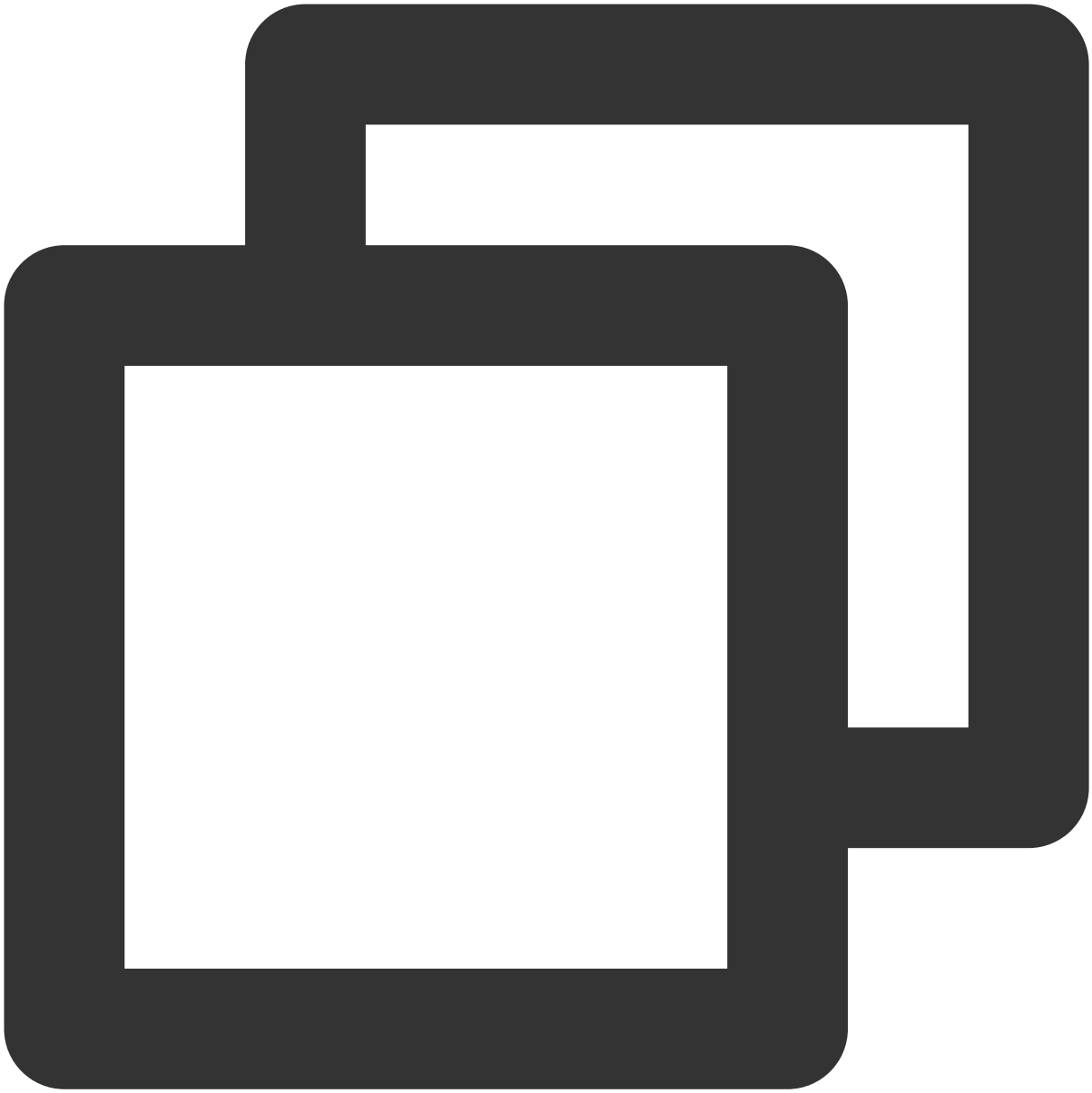
コンテナファイルパスCRDの設定の例

ワークロードを指定する

指定 pod Labels



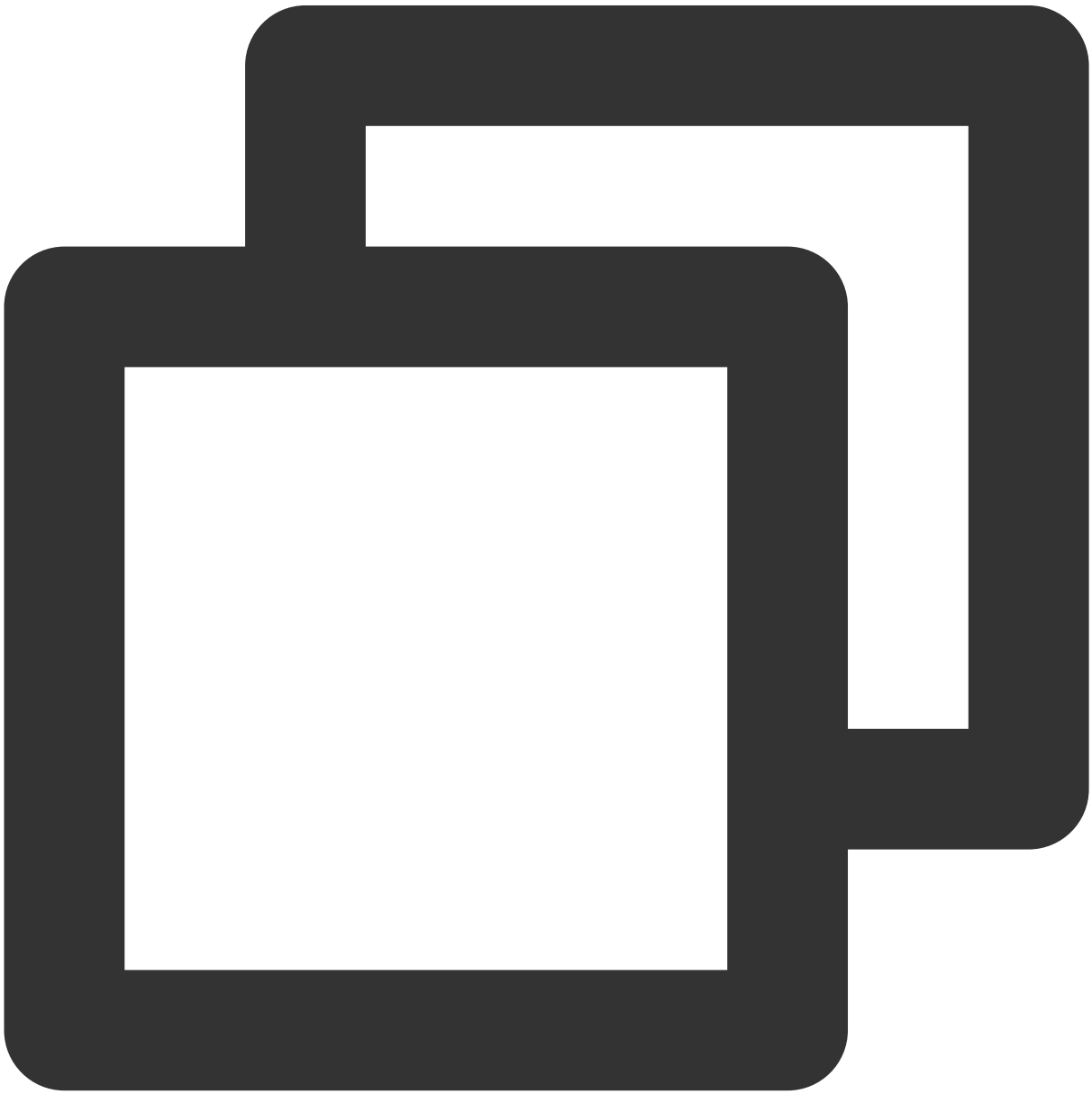
```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
metadata:
  name: test
spec:
  clsDetail:
    .......
    topicId: xxxx-xx-xx-xx-xxxx
  inputDetail:
    containerFile:
      container: prod
      filePattern: '*.log'
      logPath: /tmp/logs
      namespace: kube-system
      workload:
        kind: deployment
        name: sample-app
    type: container_file
```



```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
metadata:
  name: test
spec:
  clsDetail:
    .......
    topicId: xxxx-xx-xx-xx-xxxx
  inputDetail:
    containerFile:
      container: prod
```

```
      filePattern: '*.log'
      includeLabels:
        key1: v1
      excludeLabels:
        key2: v2
      logPath: /tmp/logs
      namespace: default,kube-public
    type: container_file
```

ノードファイルパスCRDの設定の例



```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
metadata:
  creationTimestamp: "2022-03-13T12:48:49Z"
  generation: 4
  name: test
  resourceVersion: "11729531"
  selfLink: /apis/cls.cloud.tencent.com/v1/logconfigs/test
  uid: 233f4b72-cfef-4a43-abb8-e4d033185097
spec:
  clsDetail:
    .......
    topicId: xxxx-xx-xx-xx-xxxx
  inputDetail:
    hostFile:
      customLabels:
        testmetadata: v1
      filePattern: '*.log'
      logPath: /var/logs
    type: host_file
```