云容器引擎(CCE)
22.3.5
产品介绍(for 华为云 Stack 8.1.1)

文档版本 06
发布日期 2022-09-09

华为技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址： https://e.huawei.com

目 录

1 什么是云容器引擎

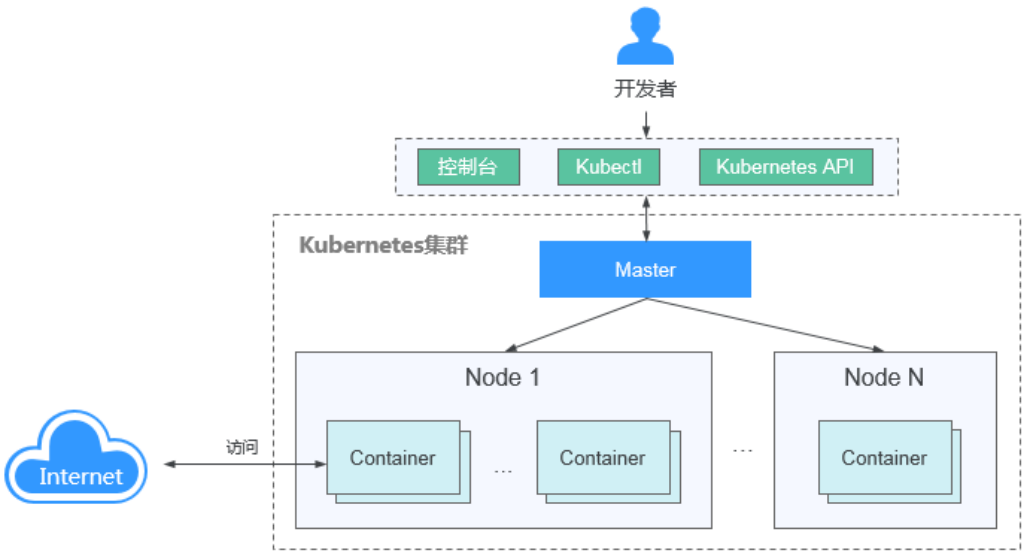
云容器引擎（Cloud Container Engine，简称CCE）提供高度可扩展的、高性能的企业级Kubernetes集群，支持运行Docker容器。借助云容器引擎，您可以在云上轻松部署、管理和扩展容器化应用程序。

云容器引擎深度整合高性能的计算（ECS）、网络（VPC/EIP/ELB）、存储（EVS/OBS/SFS）等服务，支持多可用区（Available zone，简称AZ）、多区域（Region）容灾等技术构建高可用Kubernetes集群。

全球首批Kubernetes认证服务提供商（Kubernetes Certified Service Provider，KCSP），是国内最早投入Kubernetes社区的厂商，是容器开源社区主要贡献者和容器生态领导者。也是CNCF云原生计算基金会的创始成员及白金会员，云容器引擎是全球首批通过CNCF基金会Kubernetes一致性认证的容器服务。

您可以通过CCE控制台、Kubectl命令行、Kubernetes API使用云容器引擎服务。具体请参见图1-1。

图 1-1 使用云容器引擎



产品功能

云容器引擎提供了Kubernetes集群管理、容器应用全生命周期管理、应用服务网格、Helm应用模板、插件管理、应用调度、监控与运维等容器全栈能力，为您提供一站式容器平台服务。

一站式部署和运维

使用云容器引擎，您可以一键创建Kubernetes容器集群，无需自行搭建Docker和Kubernetes集群。您可以通过云容器引擎自动化部署和一站式运维容器应用，使得应用的整个生命周期都在云容器引擎内高效完成。

支持多类型容器集群

通过云容器引擎您可以直接使用高性能的弹性云服务器、GPU加速云服务器等多种异构基础设施，您可以根据业务需要在云容器引擎中快速创建CCE集群，并通过云容器引擎对创建的集群进行统一管理。

支持多种网络访问方式

云容器引擎提供了丰富的网络访问方式，支持四层、七层负载均衡，满足不同场景下的访问诉求。

支持多种持久化存储卷

云容器引擎除支持本地磁盘存储外，还支持将工作负载数据存储在云存储上，当前支持的云存储包括：云硬盘存储卷（EVS）、文件存储卷（SFS）和对象存储卷（OBS）。

丰富的亲和/反亲和调度

云容器引擎提供工作负载和可用区、工作负载和节点以及工作负载间的亲和性/反亲和调度。您可根据业务需求设置亲和性，实现工作负载的就近部署，容器间通信就近路由，减少网络消耗；您也可以对同个工作负载的多个实例设置反亲和部署，减少宕机影响，对互相干扰的应用反亲和部署，避免干扰。

灵活的弹性伸缩策略

支持集群节点、工作负载的弹性伸缩，支持手动伸缩和自动弹性伸缩，并可以自由组合多种弹性策略以应对业务高峰期的突发流量浪涌。

深度集成Kubernetes生态工具

云容器引擎深度集成Kubernetes Helm标准模板。

模板市场基于Kubernetes Helm标准模板提供统一的资源管理与调度，高效地实现了模板的快速部署与后期管理，大幅简化了Kubernetes资源的安装管理过程。

2 产品优势

云容器引擎的优势

云容器引擎是基于业界主流的Docker和Kubernetes开源技术构建的容器服务，提供众多契合企业大规模容器集群场景的功能，在系统可靠性、高性能、开源社区兼容性等多个方面具有独特的优势，满足企业在构建容器云方面的各种需求。

简单易用

- 通过WEB界面一键创建Kubernetes集群，支持管理虚拟机节点。
- 一站式自动化部署和运维容器应用，整个生命周期都在容器服务内一站式完成。
- 通过Web界面轻松实现集群节点和工作负载的扩容和缩容，自由组合策略以应对多变的突发浪涌。
- 通过Web界面一键完成Kubernetes集群的升级。
- 深度集成Helm标准模板，真正实现开箱即用。

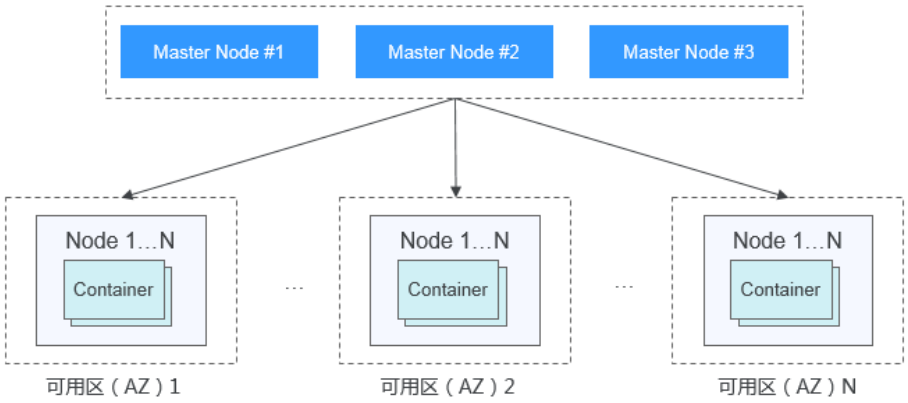
高性能

- 在计算、网络、存储、异构等方面多年的行业技术积累，提供业界领先的高性能云容器引擎，支撑您业务的高并发、大规模场景。

安全可靠

- 高可靠：集群控制面支持3 Master HA高可用，当其中某个控制节点故障时，集群依然可用，从而保障您的业务高可用。集群内节点和工作负载支持跨可用区（AZ）部署，帮助您轻松构建多活业务架构，保证业务系统在主机故障、机房中断、自然灾害等情况下可持续运行，获得生产环境的高稳定性，实现业务系统零中断。

图 2-1 集群高可用



- 高安全：私有集群，完全由用户掌控，并深度整合云账号和Kubernetes RBAC能力，支持用户在界面为子用户设置不同的RBAC权限。

开放兼容

- 云容器引擎在Docker技术的基础上，为容器化的应用提供部署运行、资源调度、服务发现和动态伸缩等一系列完整功能，提高了大规模容器集群管理的便捷性。
- 云容器引擎基于业界主流的Kubernetes实现，完全兼容Kubernetes/Docker社区原生版本，与社区最新版本保持紧密同步，完全兼容Kubernetes API和Kubectl。

云容器引擎对比自建 Kubernetes 集群

表 2-1 云容器引擎和自建 kubernetes 集群对比

对比项	自建kubernetes集群	云容器引擎
易用性	自建kubernetes集群管理基础设施通常涉及安装、操作、扩展自己的集群管理软件、配置管理系统和监控解决方案，管理复杂。每次升级集群的过程都是巨大的调整，带来繁重的运维负担。	**简化集群管理，简单易用** 借助云容器引擎，您可以一键创建和升级Kubernetes容器集群，无需自行搭建Docker和Kubernetes集群。您可以通过云容器引擎自动化部署和一站式运维容器应用，使得应用的整个生命周期都在容器服务内高效完成。 您可以通过云容器引擎轻松使用深度集成的Helm标准模板，真正实现开箱即用。 您只需启动容器集群，并指定想要运行的任务，云容器引擎帮您完成所有的集群管理工作，让您可以集中精力开发容器化的应用程序。
可扩展性	自建kubernetes集群需要根据业务流量情况和健康情况人工确定容器服务的部署，可扩展性差。	**灵活集群托管，轻松实现扩缩容** 云容器引擎可以根据资源使用情况轻松实现集群节点和工作负载的自动扩容和缩容，并可以自由组合多种弹性策略，以应对业务高峰期的突发流量浪涌。

对比项	自建kubernetes集群	云容器引擎
可靠性	自建kubernetes集群多采用单控制节点，一旦出现故障，集群和业务将不可使用。	**服务高可用** 创建集群时若“控制节点数”选项配置为3，集群将创建三个控制节点，在单个控制节点发生故障后集群可以继续使用，不影响业务功能。
高效性	自建kubernetes集群需要自行搭建镜像仓库或使用第三方镜像仓库，镜像拉取方式多采用串行传输，效率低。	**镜像快速部署，业务持续集成** 云容器引擎配合容器镜像服务提供容器自动化交付流水线，您无需编写Dockerfile与Kubernetes Manifests，基于ContainerOps流水线模板可以自定义企业级容器DevOps流程，镜像拉取方式采用并行传输，大幅提升容器交付效率。

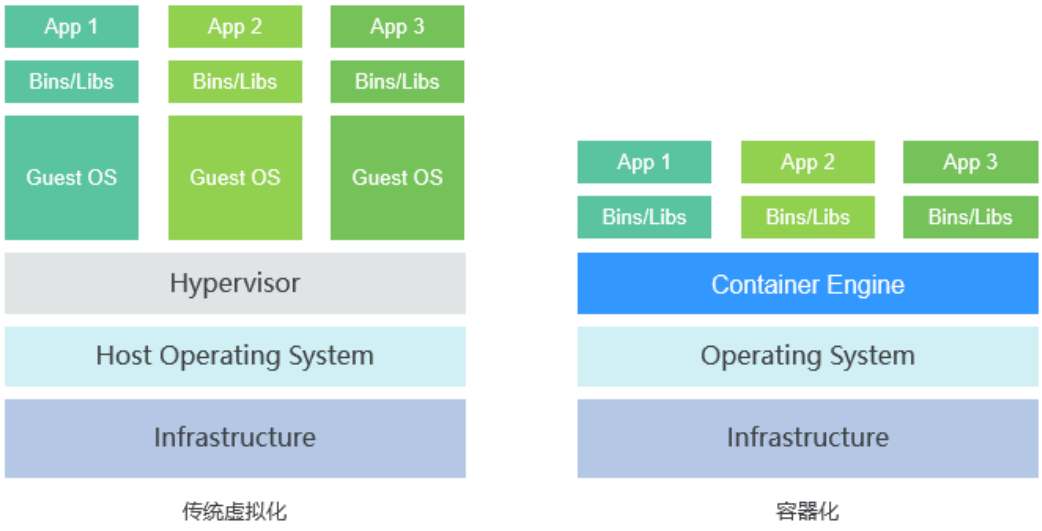
容器的优势

Docker使用Google公司推出的Go语言进行开发实现，基于Linux内核的cgroup，namespace，以及AUFS类的Union FS等技术，对进程进行封装隔离，属于操作系统层面的虚拟化技术。由于隔离的进程独立于宿主和其它的隔离的进程，因此也称其为容器。

Docker在容器的基础上，进行了进一步的封装，从文件系统、网络互联到进程隔离等，极大的简化了容器的创建和维护。

传统虚拟机技术是虚拟出一套硬件后，在其上运行一个完整操作系统，在该系统上再运行所需应用进程；而容器内的应用进程直接运行于宿主的内核，容器内没有自己的内核，而且也没有进行硬件虚拟。因此使得Docker技术比虚拟机技术更为轻便、快捷。

图 2-2 传统虚拟化和容器化方式的对比



作为一种新兴的虚拟化方式，Docker跟虚拟机相比具有众多的优势：

更高效的利用系统资源

由于容器不需要进行硬件虚拟以及运行完整操作系统等额外开销，Docker对系统资源的利用率更高。无论是应用执行速度、内存损耗或者文件存储速度，都要比传统虚拟机技术更高效。因此，相比虚拟机技术，一个相同配置的主机，往往可以运行更多数量的应用。

更快速的启动时间

传统的虚拟机技术启动应用服务往往需要数分钟，而Docker容器应用，由于直接运行于宿主内核，无需启动完整的操作系统，因此可以做到秒级、甚至毫秒级的启动时间。大大的节约了开发、测试、部署的时间。

一致的运行环境

开发过程中一个常见的问题是环境一致性问题。由于开发环境、测试环境、生产环境不一致，导致有些bug并未在开发过程中被发现。而Docker的镜像提供了除内核外完整的运行时环境，确保了应用运行环境一致性。

持续交付和部署

对开发和运维（DevOps）人员来说，最希望的就是一次创建或配置，可以在任意地方正常运行。

使用Docker可以通过定制应用镜像来实现持续集成、持续交付、部署。开发人员可以通过Dockerfile来进行镜像构建，并结合 持续集成（Continuous Integration）系统进行集成测试，而运维人员则可以直接在生产环境中快速部署该镜像，甚至结合持续部署（Continuous Delivery/Deployment） 系统进行自动部署。

而且使用Dockerfile使镜像构建透明化，不仅仅开发团队可以理解应用运行环境，也方便运维团队理解应用运行所需条件，帮助更好的生产环境中部署该镜像。

更轻松的迁移

由于Docker确保了执行环境的一致性，使得应用的迁移更加容易。Docker可以在很多平台上运行，无论是物理机、虚拟机、公有云、私有云，甚至是笔记本，其运行结果是一致的。因此用户可以很轻易的将在一个平台上运行的应用，迁移到另一个平台上，而不用担心运行环境的变化导致应用无法正常运行的情况。

更轻松的维护和扩展

Docker使用的分层存储以及镜像的技术，使得应用重复部分的复用更为容易，也使得应用的维护更新更加简单，基于基础镜像进一步扩展镜像也变得非常简单。此外，Docker团队同各个开源项目团队一起维护了一大批高质量的 官方镜像，既可以直接在生产环境使用，又可以作为基础进一步定制，大大的降低了应用服务的镜像制作成本。

表 2-2 容器对比传统虚拟机总结

特性	容器	虚拟机
启动	秒级	分钟级
硬盘使用	一般为MB	一般为GB
性能	接近原生	弱
系统支持量	单机支持上千个容器	一般几十个

3 应用场景

弹性伸缩架构

应用场景：

- 电商客户遇到促销、限时秒杀等活动期间，访问量激增，需及时、自动扩展云计算资源。
- 视频直播客户业务负载变化难以预测，需要根据CPU/内存使用率进行实时扩缩容。
- 游戏客户每天中午12点及晚上18:00-23:00间需求增长，需要定时扩容。

价值：

云容器引擎可根据用户的业务需求预设策略自动调整计算资源，使云服务器或容器数量自动随业务负载增长而增加，随业务负载降低而减少，保证业务平稳健康运行，节省成本。

优势：

- 自由灵活

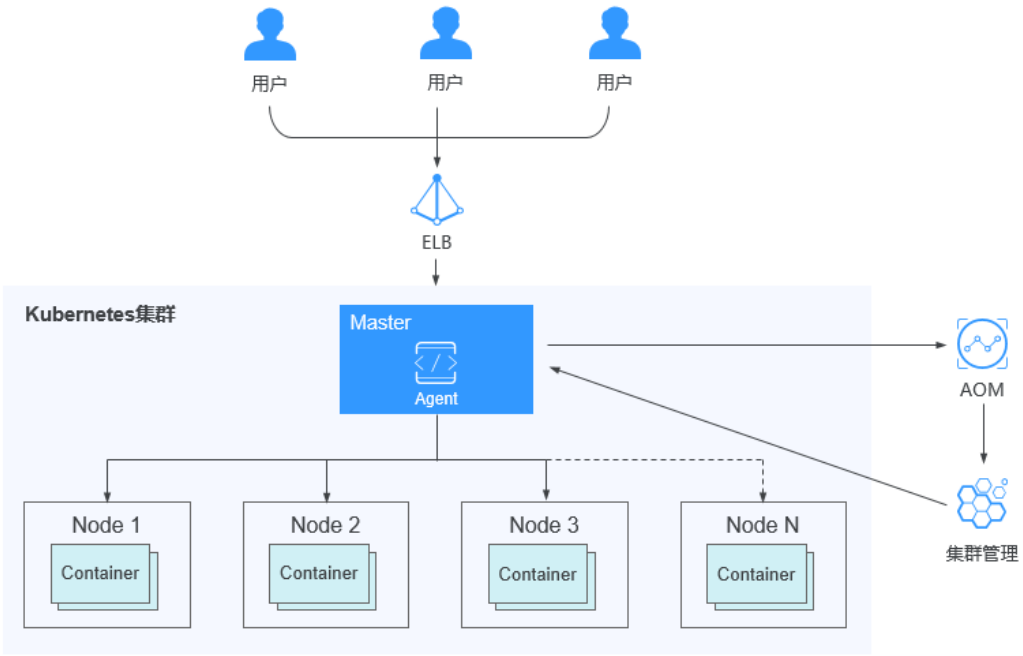
  支持多种策略配置，业务流量达到扩容指标，秒级触发容器扩容操作。
- 高可用

  自动检测伸缩组中实例运行状况，启用新实例替换不健康实例，保证业务健康可用。

建议搭配使用：

autoscaler插件（集群自动扩缩容）+应用运维管理 AOM（工作负载伸缩）

图 3-1 弹性伸缩场景



微服务治理

应用场景：

伴随着互联网技术的不断发展，各大企业的系统越来越复杂，传统的系统架构越来越不能满足业务的需求，取而代之的是微服务架构。微服务是将复杂的应用切分为若干服务，每个服务均可以独立开发、部署和伸缩；微服务和容器组合使用，可进一步简化微服务的交付，提升应用的可靠性和可伸缩性。

随着微服务的大量应用，其构成的分布式应用架构在运维、调试、和安全管理等维度变得更加复杂，在管理微服务时，往往需要在业务代码中添加微服务治理相关的代码，导致开发人员不能专注于业务开发，还需要考虑微服务治理的解决方案，并且将解决方案融合到其业务系统中。

价值：

云容器引擎深度集成应用服务网格，提供开箱即用的应用服务网格流量治理能力，用户无需修改代码，即可实现灰度发布、流量治理和流量监控能力。

优势：

- 开箱即用

  与云容器引擎无缝对接，一键开启后即可提供非侵入的智能流量治理解决方案。
- 策略化智能路由

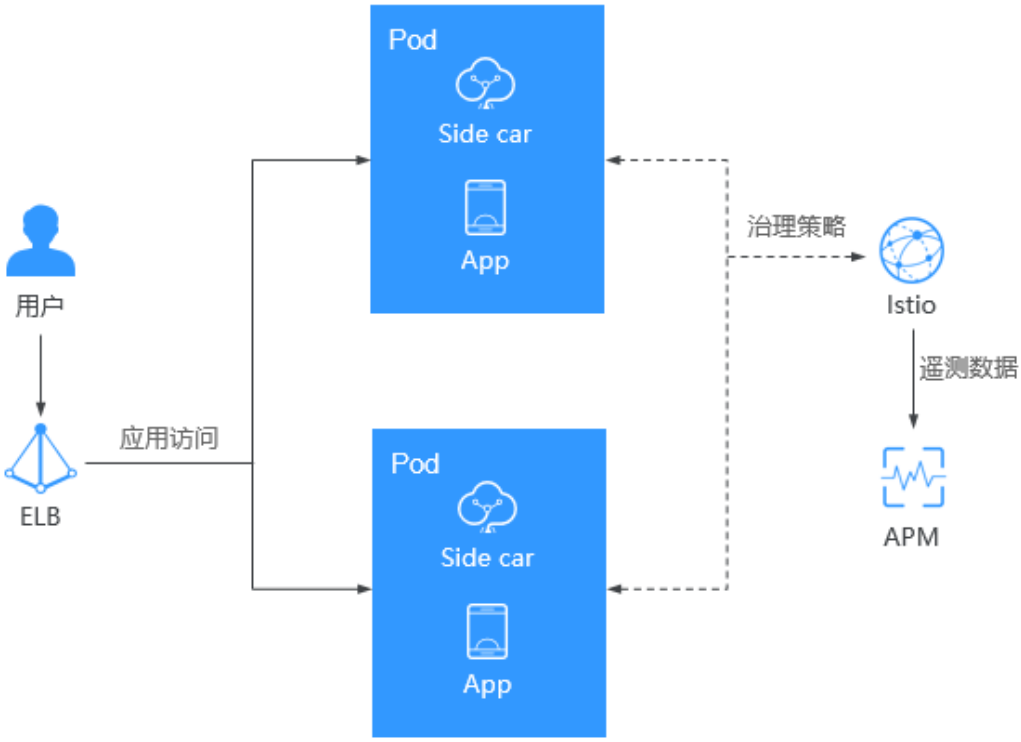
  无需修改代码，即可实现HTTP、TCP等服务连接策略和安全策略。
- 流量治理可视化

  基于无侵入的监控数据采集，深度整合APM能力，提供实时流量拓扑、调用链等服务性能监控和运行诊断，构建全景的服务运行视图，可实时、一站式观测服务流量健康和性能状态。

建议搭配使用：

弹性负载均衡 ELB + 应用性能管理 APM + 应用运维管理 AOM

图 3-2 微服务治理场景



DevOps 持续交付

应用场景:

当前IT行业发展日益快速，面对海量需求必须具备快速集成的能力。经过快速持续集成，才能保证不间断的补全用户体验，提升服务质量，为业务创新提供源源不断的动力。大量交付实践表明，不仅传统企业，甚至互联网企业都可能在持续集成方面存在研发效率低、工具落后、发布频率低等方面的问题，需要通过持续交付提高效率，降低发布风险。

价值:

云容器引擎搭配容器镜像服务提供DevOps持续交付能力，能够基于代码源自动完成代码编译、镜像构建、灰度发布、容器化部署，实现一站式容器化交付流程，并可对接已有CI/CD，完成传统应用的容器化改造和部署。

优势:

- 高效流程管理

  更优的流程交互设计，脚本编写量较传统CI/CD流水线减少80%以上，让CI/CD管理更高效。
- 灵活的集成方式

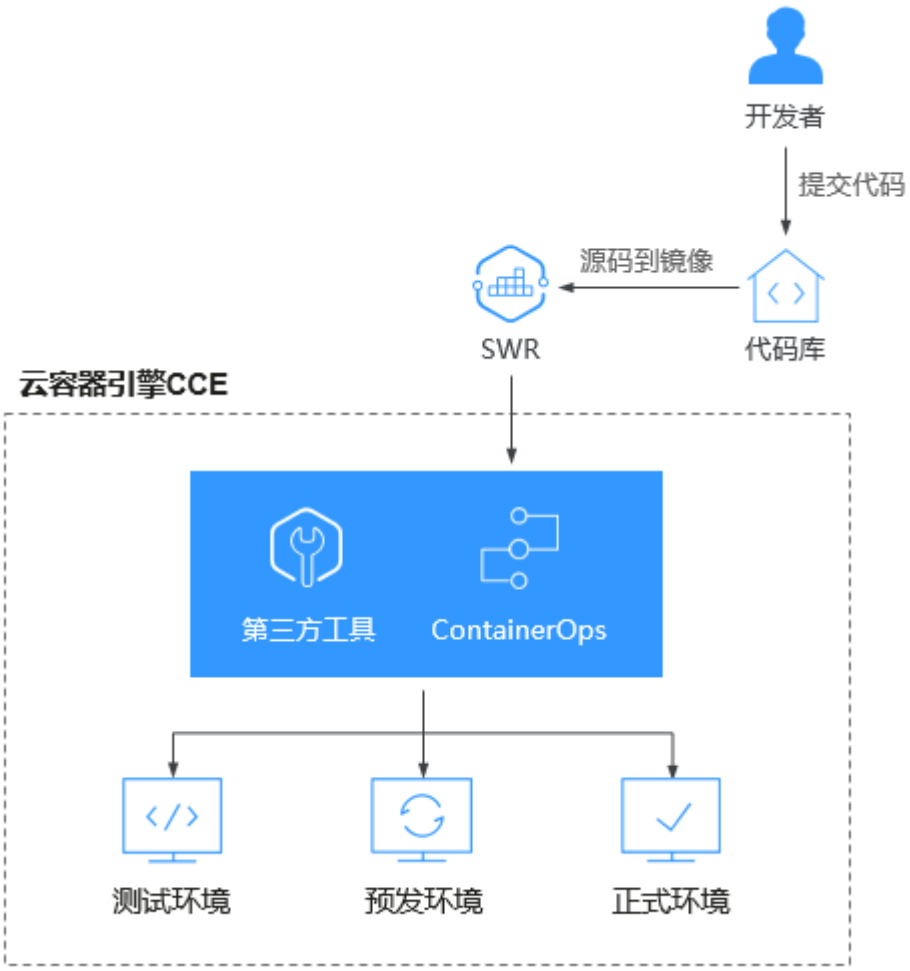
  提供丰富的接口便于与企业已有CI/CD系统进行集成，灵活适配企业的个性化诉求。
- 高性能

  全容器化架构设计，任务调度更灵活，执行效率更高。

建议搭配使用：

容器镜像服务SWR + 对象存储服务 OBS + 虚拟专用网络 VPN

图 3-3 DevOps 持续交付场景



混合云架构

应用场景：

- 多云部署、容灾备份

  为保证业务高可用，需要将业务同时部署在多个云的容器服务上，在某个云出现事故时，通过统一流量分发的机制，自动的将业务流量切换到其他云上。

- 流量分发、弹性伸缩

  大型企业客户需要将业务同时部署在不同地域的云机房中，并能自动弹性扩容和缩容，以节约成本。

- 业务上云、数据库托管

  对于金融、安全等行业用户，业务数据的敏感性要求将数据业务保留在本地的IDC中而将一般业务部署在云上，并需要进行统一管理。

- 开发与部署分离

出于IP安全的考虑，用户希望将生产环境部署在公有云上，而将开发环境部署在本地的IDC。

价值：

云容器引擎利用容器环境无关的特性，将私有云和公有云容器服务实现网络互通和统一管理，应用和数据可在云上云下无缝迁移，并可统一运维多个云端资源，从而实现资源的灵活使用以及业务容灾等目的。

优势：

- 云上容灾

  通过云容器引擎，可以将业务系统同时部署在多个云的容器服务上，统一流量分发，单云故障后能够自动将业务流量切换到其他云上，并能快速自动解决现网事故。

- 流量自动分发

  通过云容器引擎的统一流量分发机制，实现应用访问流量的地域亲和，降低业务访问时延，并需要能够将线下IDC中的业务在云上扩展，可根据业务流量峰值情况，自动弹性扩容和缩容。

- 计算与数据分离，能力共享

  通过云容器引擎，用户可以实现敏感业务数据与一般业务数据的分离，可以实现开发环境和生产环境分离，可以实现特殊计算能力与一般业务的分离，并能够实现弹性扩展和集群的统一管理，达到云上云下资源和能力的共享。

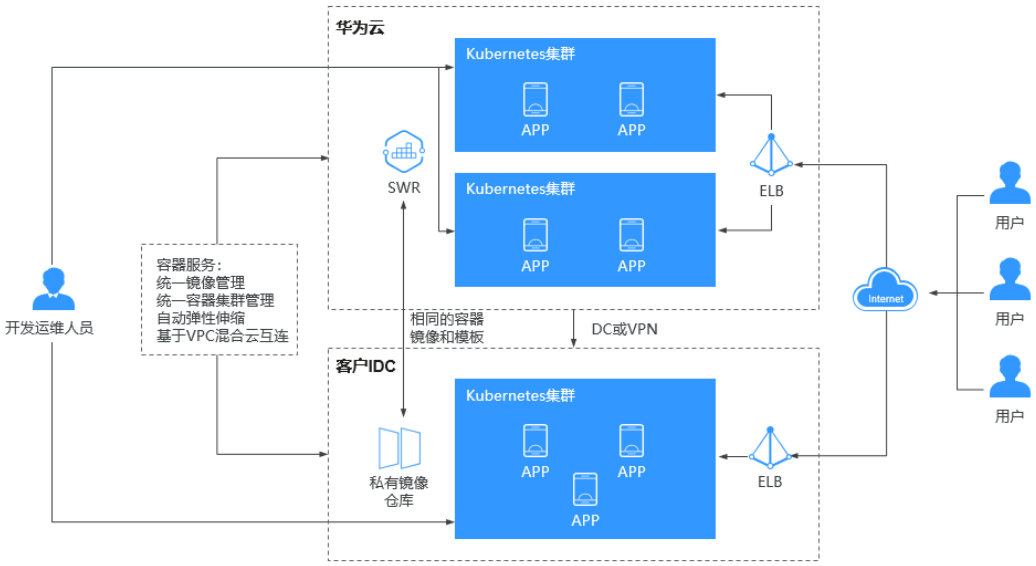
- 降低成本

  业务高峰时，利用公有云资源池快速扩容，用户不再需要根据流量峰值始终保持和维护大量资源，节约成本。

建议搭配使用：

弹性云服务器 ECS + 云专线 DC + 虚拟专用网络 VPN + 容器镜像服务 SWR

图 3-4 混合云场景



高性能 AI 计算

应用场景：

对于AI、基因测序、视频处理等行业的用户，其计算任务依赖特殊的硬件，如GPU等，用户需要将计算业务部署在云上，弹性利用云上的特殊硬件算力，而在私有云部署一般业务，避免大规模使用特殊硬件带来的成本压力。

价值：

面向AI计算的容器服务，采用高性能GPU计算实例，并支持多容器共享GPU资源，在AI计算性能上比通用方案提升3-5倍以上，并大幅降低了AI计算的成本，同时帮助数据工程师在集群上轻松部署计算应用，您无需关心复杂的部署运维，专注核心业务，快速实现从0到1快速上线。

优势：

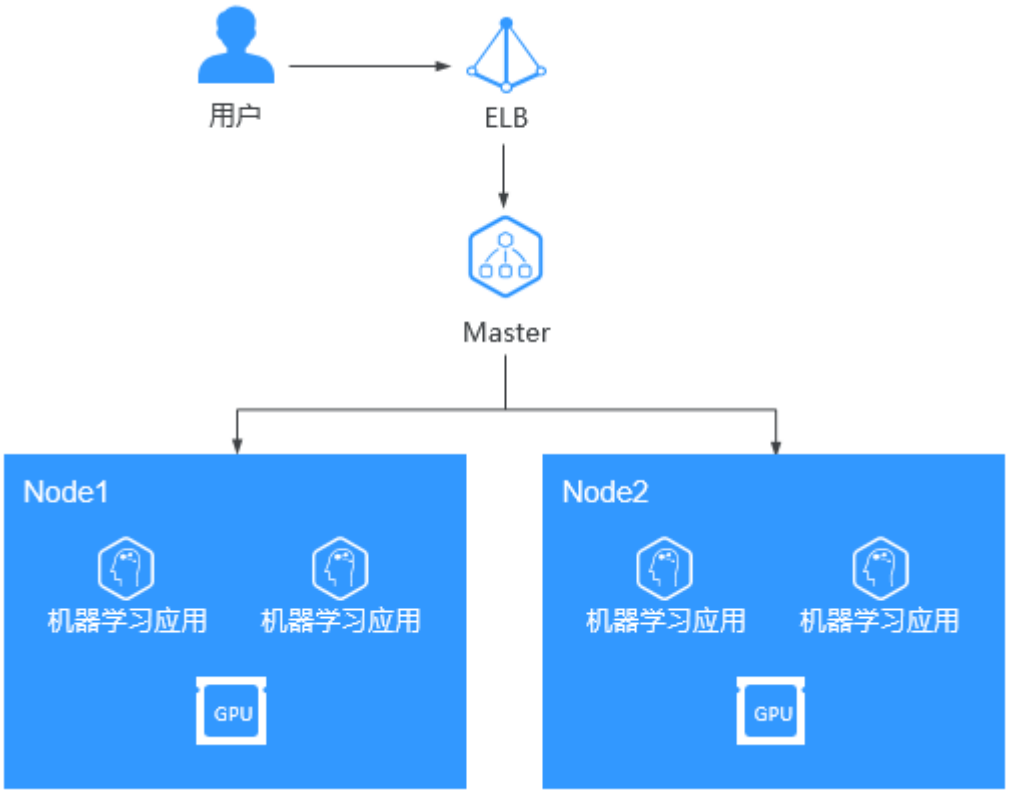
- 高效计算

  GPU资源多容器共享调度，整体计算成本大幅降低。
- 成熟应用
- 主流GPU型号全适配，并在EI产品大规模使用。

建议搭配使用：

GPU加速云服务器 + 弹性负载均衡 ELB + 对象存储服务 OBS

图 3-5 AI 计算



4 约束与限制

本文主要为您介绍云容器引擎（CCE）集群使用过程中的一些限制。

集群/节点限制

- 单资源集下可创建的集群总数限制为15个。如果配额不满足业务需求，请联系技术支持。
- 在升级场景下，CCE升级前创建的资源集下可创建的集群总数限制为5个。
- 集群一旦创建以后，不支持变更以下项：
  - 变更集群的控制节点数量。
  - 变更控制节点可用区。
  - 变更集群的网络配置，如所在的虚拟私有云VPC、子网、容器网段、服务网段、IPv6、kubeproxy代理（转发）模式。
  - 变更网络模型，例如“容器隧道网络”更换为“VPC网络”。
- 不支持应用在不同命名空间下迁移。
- 由于ECS（节点）等底层依赖产品配额及库存限制，创建集群、扩容集群或者自动弹性扩容时，可能只有部分节点创建成功。
- ECS（节点）规格要求：CPU > 2核且内存 > 4GiB。
- 海光和飞腾兼容的操作系统为欧拉2.9，欧拉2.9只支持1.15集群且VPC网络模型。
- IP V6场景的约束：仅支持容器网络为隧道模型的集群，并且service类型不支持LoadBalancer类型。

网络

- 节点访问(NodePort)的使用约束：默认为VPC内网访问，如果需要使用弹性IP通过公网访问该服务，请提前在集群的节点上绑定弹性IP。
- CCE中的负载均衡（LoadBalancer）访问类型使用弹性负载均衡 ELB提供网络访问，存在如下产品约束：
  - 自动创建的ELB实例建议不要被其他资源使用，否则会在删除时被占用，导致资源残留。
- 网络策略(NetworkPolicy)，存在如下产品约束：
  - VPC网络模型暂不支持网络策略（NetworkPolicy）。
  - 网络策略（NetworkPolicy）暂不支持设置egress路由规则。

- 网络平面(NetworkAttachmentDefinition)：
  - 仅网络模型为VPC网络（未开启IPv6）和Yangtse的集群支持创建网络平面；网络模型为容器隧道网络时列表中仅显示“default-network”，不能新增或修改。

存储卷

- 云硬盘存储卷使用约束：
  - 云硬盘不支持跨可用区挂载，且暂时不支持被多个工作负载、同一个工作负载的多个实例或多个任务使用。
  - 由于CCE集群各节点之间暂不支持共享盘的数据共享功能，多个节点挂载使用同一个云硬盘可能会出现读写冲突、数据缓存冲突等问题，所以创建无状态工作负载时，若使用了EVS云硬盘，建议工作负载只选择一个实例。
  - 创建有状态工作负载并添加云存储时，云硬盘暂不支持使用已有存储。
  - 不支持导入分区过或者具有非ext4文件系统的云硬盘。
  - 存储不支持选择企业项目，新创建的存储卷默认创建到default企业项目下。
  - 云服务器的整机快照功能会影响CCE云硬盘存储卷功能。一旦对CCE业务节点创建整机快照，此节点上工作负载使用的云硬盘存储卷无法再挂载到其他节点上。这种场景下，若工作负载迁移到其他节点上，会因为云硬盘存储卷无法挂载导致工作负载启动失败。
- 文件存储卷使用约束：
  - 存储不支持选择企业项目，新创建的存储卷默认创建到default企业项目下。
- 对象存储卷使用约束如下：
  - 存储不支持选择企业项目，新创建的存储卷默认创建到default企业项目下。

扩缩容

- 弹性伸缩功能仅支持对集群的工作节点和工作负载进行扩缩容，不能对控制节点进行扩缩容。
- 工作负载伸缩策略限制条件如下：
  - HPA策略：仅支持1.13及以上版本的集群创建。
  - CustomedHPA策略：仅支持1.15及以上版本的集群创建。
  - 每个工作负载只能创建一个策略，即如果您创建了一个HPA策略，则不能再对其创建CustomedHPA策略或其他HPA策略，您可以删除该HPA策略后再创建。

其他限制

不支持修改“VDC名称”。

服务（Service）数量

此处的服务对应kubernetes的service资源，即工作负载所添加的服务。

每个命名空间下，创建的服务数量不能超过6000个。

CCE 集群配额限制

针对每个用户，云容器引擎的集群在每个地域分配了固定配额。

限制项	普通用户限制	例外申请方式
单资源集下集群总数	15	/
单集群下节点（集群管理规模）	可选择50节点、200节点、1000节点或2000节点多种管理规模。	/
每个worker节点创建容器实例最大数	创建集群时界面可设置。 VPC网络：最大256。	没有例外

依赖底层云产品配额限制

限制大类	限制项	普通用户限制	例外申请方式
计算	实例数	1000	/
	核心数	8000核	/
	RAM容量 (MB)	16384000	/
网络	一个用户创建虚拟私有云的数量	5	/
	一个用户创建子网的数量	100	/
	一个用户拥有的安全组数量	100	/
	一个用户拥有的安全组规则数量	5000	/
	一个路由表里拥有的路由数量	100	没有例外
	一个虚拟私有云拥有路由数量	100	没有例外
	一个区域下的对等连接数量	50	没有例外
	一个用户拥有网络ACL数量	200	/
	一个用户创建二层连接网关的数量	5	/
负载均衡	弹性负载均衡	50	/
	弹性负载均衡监听器	100	/
	弹性负载均衡证书	120	/
	弹性负载均衡转发策略	500	/
	弹性负载均衡后端主机组	500	/
	弹性负载均衡后端服务器	500	/

5 基本概念

5.1 基本概念

云容器引擎（Cloud Container Engine，简称CCE）提供高度可扩展的、高性能的企业级Kubernetes集群，支持运行Docker容器。借助云容器引擎，您可以在云上轻松部署、管理和扩展容器化应用程序。

云容器引擎提供Kubernetes原生API，支持使用kubectl，且提供图形化控制台，让您能够拥有完整的端到端使用体验，使用云容器引擎前，建议您先了解相关的基本概念。

集群（Cluster）

集群指容器运行所需要的云资源组合，关联了若干云服务器节点、负载均衡等云资源。您可以理解为集群是“同一个子网中一个或多个弹性云服务器（又称：节点）”通过相关技术组合而成的计算机群体，为容器运行提供了计算资源池。

节点（Node）

每一个节点对应一台服务器（可以是虚拟机实例或者物理服务器），容器应用运行在节点上。节点上运行着Agent代理程序（kubelet），用于管理节点上运行的容器实例。集群中的节点数量可以伸缩。

节点池（NodePool）

节点池是集群中具有相同配置的一组节点，一个节点池包含一个节点或多个节点。

虚拟私有云（VPC）

虚拟私有云是通过逻辑方式进行网络隔离，提供安全、隔离的网络环境。您可以在VPC中定义与传统网络无差别的虚拟网络，同时提供弹性IP、安全组等高级网络服务。

安全组

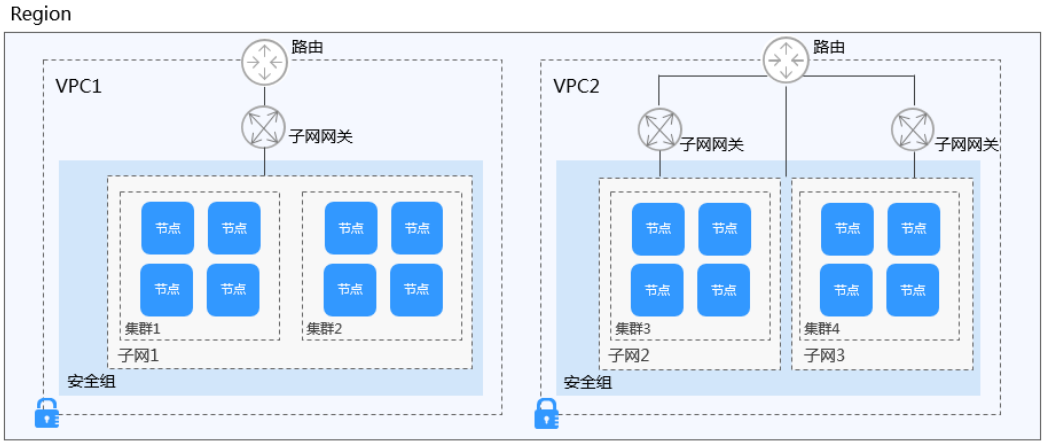
安全组是一个逻辑上的分组，为同一个VPC内具有相同安全保护需求并相互信任的弹性云服务器提供访问策略。安全组创建后，用户可以在安全组中定义各种访问规则，当弹性云服务器加入该安全组后，即受到这些访问规则的保护。

集群、虚拟私有云、安全组和节点的关系

如图5-1，同一个Region下可以有多个虚拟私有云（VPC）。虚拟私有云由一个个子网组成，子网与子网之间的网络交互通过子网网关完成，而集群就是建立在某个子网中。因此，存在以下三种场景：

- 不同集群可以创建在不同的虚拟私有云中。
- 不同集群可以创建在同一个子网中。
- 不同集群可以创建在不同的子网中。

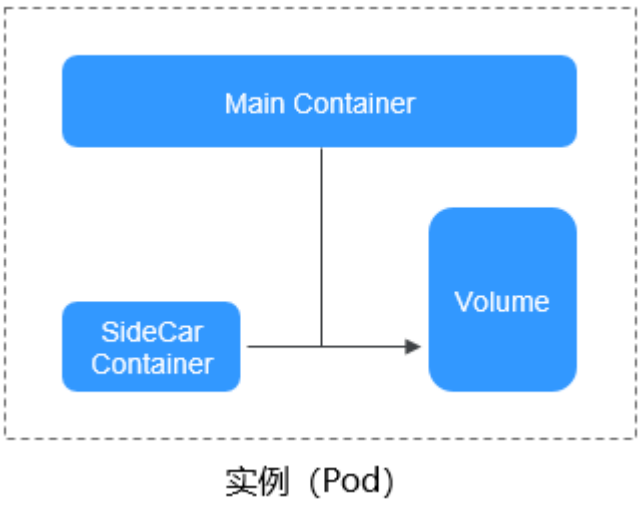
图 5-1 集群、VPC、安全组和节点的关系



实例（Pod）

实例（Pod）是 Kubernetes 部署应用或服务的最小的基本单位。一个Pod 封装多个应用容器（也可以只有一个容器）、存储资源、一个独立的网络 IP 以及管理控制容器运行方式的策略选项。

图 5-2 实例（Pod）



容器（Container）

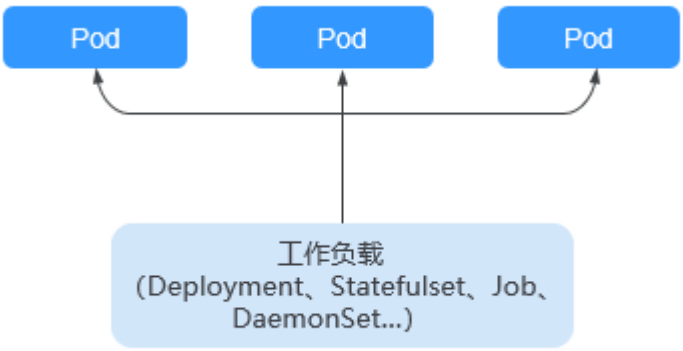
一个通过 Docker 镜像创建的运行实例，一个节点可运行多个容器。容器的实质是进程，但与直接在宿主执行的进程不同，容器进程运行于属于自己的独立的命名空间。

工作负载

工作负载即Kubernetes对一组Pod的抽象模型，用于描述业务的运行载体，包括Deployment、Statefulset、Daemonset、Job、CronJob等多种类型。

- **无状态工作负载：**即kubernetes中的“Deployment”，无状态工作负载支持弹性伸缩与滚动升级，适用于实例完全独立、功能相同的场景，如：nginx、wordpress等。
- **有状态工作负载：**即kubernetes中的“StatefulSet”，有状态工作负载支持实例有序部署和删除，支持持久化存储，适用于实例间存在互访的场景，如ETCD、mysql-HA等。
- **创建守护进程集：**即kubernetes中的“DaemonSet”，守护进程集确保全部（或者某些）节点都运行一个Pod实例，支持实例动态添加到新节点，适用于实例在每个节点上都需要运行的场景，如ceph、fluentd、Prometheus Node Exporter等。
- **普通任务：**即kubernetes中的“Job”，普通任务是一次性运行的短任务，部署完成后即可执行。使用场景为在创建工作负载前，执行普通任务，将镜像上传至镜像仓库。
- **定时任务：**即kubernetes中的“CronJob”，定时任务是按照指定时间周期运行的短任务。使用场景为在某个固定时间点，为所有运行中的节点做时间同步。

图 5-3 工作负载与 Pod 的关系



编排模板

编排模板包含了一组容器服务的定义和其相互关联，可以用于多容器应用的部署和管理。

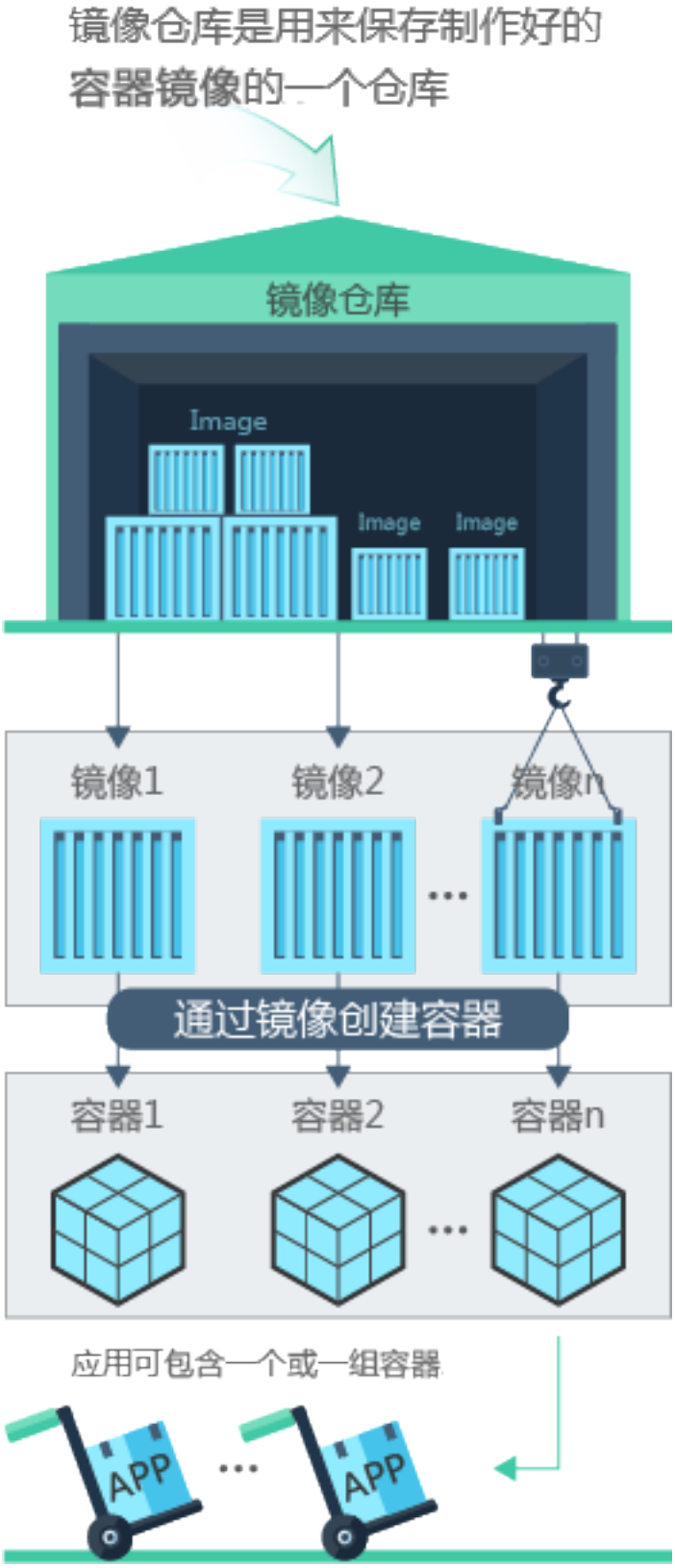
镜像（Image）

Docker镜像是一个模板，是容器应用打包的标准格式，用于创建Docker容器。或者说，Docker镜像是一个特殊的文件系统，除了提供容器运行时所需的程序、库、资源、配置等文件外，还包含了一些为运行时准备的配置参数（如匿名卷、环境变量、用户等）。镜像不包含任何动态数据，其内容在构建之后也不会被改变。在部署容器化应用时可以指定镜像，镜像可以来自于容器镜像服务或者用户的私有Registry。例如

一个Docker镜像可以包含一个完整的Ubuntu操作系统环境，里面仅安装了用户需要的应用程序及其依赖文件。

镜像（Image）和容器（Container）的关系，就像是面向对象程序设计中的类和实例一样，镜像是静态的定义，容器是镜像运行时的实体。容器可以被创建、启动、停止、删除、暂停等。

图 5-4 镜像、容器、工作负载的关系



命名空间（Namespace）

命名空间是对一组资源和对象的抽象整合。在同一个集群内可创建不同的命名空间，不同命名空间中的数据彼此隔离。使得它们既可以共享同一个集群的服务，也能够互不干扰。例如：

- 可以将开发环境、测试环境的业务分别放在不同的命名空间。
- 常见的pods, services, replication controllers和deployments等都是属于某一个namespace的（默认是default），而node, persistentVolumes等则不属于任何namespace。

服务（Service）

Service是将运行在一组 Pods 上的应用程序公开为网络服务的抽象方法。

使用Kubernetes，您无需修改应用程序即可使用不熟悉的服务发现机制。 Kubernetes为Pods提供自己的IP地址和一组Pod的单个DNS名称，并且可以在它们之间进行负载平衡。

Kubernetes允许指定一个需要的类型的 Service，类型 的取值以及行为如下：

- ClusterIP：集群内访问。通过集群的内部 IP 暴露服务，选择该值，服务只能够在集群内部可以访问，这也是默认的 ServiceType。
- NodePort：节点访问。通过每个 Node 上的 IP 和静态端口（NodePort）暴露服务。NodePort 服务会路由到 ClusterIP 服务，这个 ClusterIP 服务会自动创建。通过请求 <NodeIP>:<NodePort>，可以从集群的外部访问一个 NodePort 服务。
- LoadBalancer：负载均衡。使用云提供商的负载均衡器，可以向外部暴露服务。外部的负载均衡器可以路由到 NodePort 服务和 ClusterIP 服务。

七层负载均衡（Ingress）

Ingress是为进入集群的请求提供路由规则的集合，可以给service提供集群外部访问的URL、负载均衡、SSL终止、HTTP路由等。

网络策略（NetworkPolicy）

NetworkPolicy提供了基于策略的网络控制，用于隔离应用并减少攻击面。它使用标签选择器模拟传统的分段网络，并通过策略控制它们之间的流量以及来自外部的流量。

配置项（Configmap）

ConfigMap用于保存配置数据的键值对，可以用来保存单个属性，也可以用来保存配置文件。ConfigMap跟secret很类似，但它可以更方便地处理不包含敏感信息的字符串。

密钥（Secret）

Secret解决了密码、token、密钥等敏感数据的配置问题，而不需要把这些敏感数据暴露到镜像或者Pod Spec中。Secret可以以Volume或者环境变量的方式使用。

标签（Label）

标签其实就一对 key/value，被关联到对象上，比如Pod。标签的使用我们倾向于能够标示对象的特殊特点，并且对用户而言是有意义的，但是标签对内核系统是没有直接意义的。

选择器（LabelSelector）

Label selector是Kubernetes核心的分组机制，通过label selector客户端/用户能够识别一组有共同特征或属性的资源对象。

注解（Annotation）

Annotation与Label类似，也使用key/value键值对的形式进行定义。

Label具有严格的命名规则，它定义的是Kubernetes对象的元数据（Metadata），并且用于Label Selector。

Annotation则是用户任意定义的“附加”信息，以便于外部工具进行查找。

存储卷（PersistentVolume）

PersistentVolume（PV）是集群之中的一块网络存储。跟 Node 一样，也是集群的资源。

存储声明（PersistentVolumeClaim）

PV 是存储资源，而 PersistentVolumeClaim (PVC) 是对 PV 的请求。PVC 跟 Pod 类似：Pod 消费 Node 资源，而 PVC 消费 PV 资源；Pod 能够请求 CPU 和内存资源，而 PVC 请求特定大小和访问模式的数据卷。

弹性伸缩（HPA）

Horizontal Pod Autoscaling，简称HPA，是Kubernetes中实现POD水平自动伸缩的功能。Kubernetes集群可以通过Replication Controller的scale机制完成服务的扩容或缩容，实现具有伸缩性的服务。

亲和性与反亲和性

在应用没有容器化之前，原先一个虚机上会装多个组件，进程间会有通信。但在做容器化拆分的时候，往往直接按进程拆分容器，比如业务进程一个容器，监控日志处理或者本地数据放在另一个容器，并且有独立的生命周期。这时如果他们分布在网络中两个较远的点，请求经过多次转发，性能会很差。

- 亲和性：可以实现就近部署，增强网络能力实现通信上的就近路由，减少网络的损耗。如：应用A与应用B两个应用频繁交互，所以有必要利用亲和性让两个应用的尽可能的靠近，甚至在一个节点上，以减少因网络通信而带来的性能损耗。
- 反亲和性：主要是出于高可靠性考虑，尽量分散实例，某个节点故障的时候，对应用的影响只是 N 分之一或者只是一个实例。如：当应用采用多副本部署时，有必要采用反亲和性让各个应用实例打散分布在各个节点上，以提高HA。

节点亲和性（NodeAffinity）

通过选择标签的方式，可以限制pod被调度到特定的节点上。

节点反亲和性（NodeAntiAffinity）

通过选择标签的方式，可以限制pod不被调度到特定的节点上。

工作负载亲和性（PodAffinity）

指定工作负载部署在相同节点。用户可根据业务需求进行工作负载的就近部署，容器间通信就近路由，减少网络消耗。

工作负载反亲和性（PodAntiAffinity）

指定工作负载部署在不同节点。同个工作负载的多个实例反亲和部署，减少宕机影响；互相干扰的应用反亲和部署，避免干扰。

资源配额（Resource Quota）

资源配额（Resource Quotas）是用来限制用户资源用量的一种机制。

资源限制（Limit Range）

默认情况下，K8S中所有容器都没有任何CPU和内存限制。LimitRange(简称limits)用来给Namespace增加一个资源限制，包括最小、最大和默认资源。在pod创建时，强制执行使用limits的参数分配资源。

环境变量

环境变量是指容器运行环境中设定的一个变量，您可以在创建容器模板时设定不超过30个的环境变量。环境变量可以在工作负载部署后修改，为工作负载提供了极大的灵活性。

在CCE中设置环境变量与Dockerfile中的“ENV”效果相同。

应用服务网格（Istio）

Istio是一个提供连接、保护、控制以及观测功能的开放平台。

云容器引擎深度集成了应用服务网格，提供非侵入式的微服务治理解决方案，支持完整的生命周期管理和流量治理能力，兼容Kubernetes和Istio生态。一键开启应用服务网格后即可提供非侵入的智能流量治理解决方案，其功能包括负载均衡、熔断、限流等多种治理能力。应用服务网格内置金丝雀、蓝绿等多种灰度发布流程，提供一站式自动化的发布管理。基于无侵入的监控数据采集，深度整合应用性能管理（APM）能力，提供实时流量拓扑、调用链等服务性能监控和运行诊断，构建全景的服务运行视图。

5.2 CCE 与原生 Kubernetes 名词对照

Kubernetes，简称K8S，是开源的容器集群管理系统，可以实现容器集群的自动化部署、自动扩缩容、维护等功能。它既是一款容器编排工具，也是全新的基于容器技术的分布式架构领先方案。在Docker技术的基础上，为容器化的应用提供部署运行、资源调度、服务发现和动态伸缩等功能，提高了大规模容器集群管理的便捷性。

本文主要为您介绍云容器引擎CCE与原生Kubernetes名词对照情况和简单解释。

表 5-1 CCE 与原生 Kubernetes 名词对照

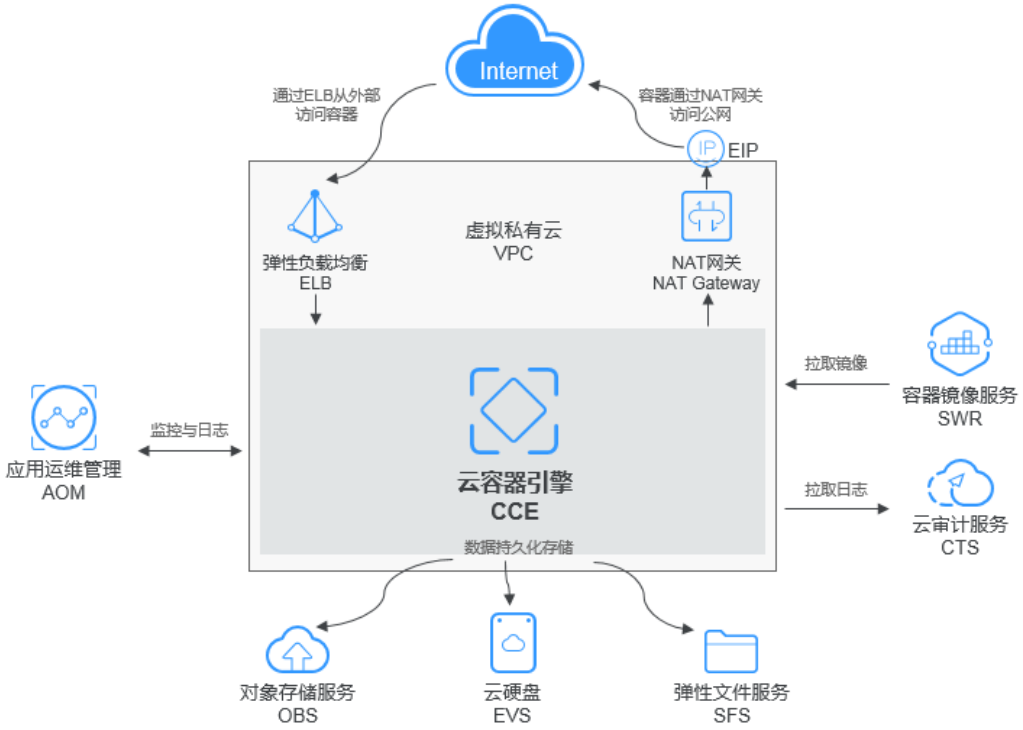
云容器引擎CCE	原生Kubernetes
集群	Cluster
节点	Node
节点池	NodePool
容器	Container
镜像	Image
命名空间	Namespace
无状态工作负载	Deployment
有状态工作负载	StatefulSet
守护进程集	DaemonSet
普通任务	Job
定时任务	CronJob
实例（容器组）	Pod
服务（Service）	Service
虚拟集群IP	Cluster IP
节点端口	NodePort
负载均衡	LoadBalancer
七层负载均衡	Ingress
网络策略	NetworkPolicy
模板	Template
配置项	ConfigMap
密钥	Secret
标签	Label
选择器	LabelSelector
注解	Annotation
存储卷	PersistentVolume
存储声明	PersistentVolumeClaim
弹性伸缩	HPA
节点亲和性	NodeAffinity
节点反亲和性	NodeAntiAffinity

云容器引擎CCE	原生Kubernetes
工作负载亲和性	PodAffinity
工作负载反亲和性	PodAntiAffinity
触发器	Webhook
终端节点	Endpoint
资源配额	Resource Quota
资源限制	Limit Range

6 与其它云服务的关系

云容器引擎需要与其他云服务协同工作，云容器引擎需要获取如下云服务资源的权限。

图 6-1 云容器引擎与其他服务的关系



弹性云服务器 ECS

弹性云服务器是由CPU、内存、镜像、云硬盘组成的一种可随时获取、弹性可扩展的计算服务器，同时它结合虚拟私有云、虚拟防火墙、数据多副本保存等能力，为您打造一个高效、可靠、安全的计算环境，确保您的服务持久稳定运行。

在云容器引擎中具有多个云硬盘的一台弹性云服务器就是一个节点，您可以在创建节点时指定弹性云服务器的规格。

虚拟私有云 VPC

虚拟私有云（Virtual Private Cloud，VPC）是用户在云上申请的隔离的、私密的虚拟网络环境。用户可以自由配置VPC内的IP地址段、子网、安全组等子服务，也可以申请弹性带宽和弹性IP搭建业务系统。

虚拟私有云为弹性云服务器提供一个逻辑上完全隔离的虚拟网络环境。您可以完全掌控自己的虚拟网络，包括申请弹性带宽/IP、创建子网、配置DHCP、设置安全组等。此外您也可以通过专线/VPN等连接方式将VPC与传统数据中心互联互通，灵活整合资源。

弹性负载均衡 ELB

弹性负载均衡（ Elastic Load Balance，ELB）将访问流量自动分发到多台云服务器，扩展应用系统对外的服务能力，实现更高水平的应用容错。

云容器引擎支持将创建的应用对接到弹性负载均衡，从而提高应用系统对外的服务能力，提高应用程序容错能力。

NAT 网关

NAT网关能够为VPC内的容器实例提供网络地址转换（Network Address Translation）服务，SNAT功能通过绑定弹性公网IP，实现私有IP向公有IP的转换，可实现VPC内的容器实例共享弹性公网IP访问Internet。

容器镜像服务 SWR

容器镜像服务（Software Repository for Container，SWR）是一种支持容器镜像全生命周期管理的服务， 提供简单易用、安全可靠的镜像管理功能，帮助用户快速部署容器化服务。

容器镜像服务提供的镜像仓库是用于存储、管理docker容器镜像的场所，可以让使用人员轻松存储、管理、部署docker容器镜像。

云硬盘 EVS

可以将云硬盘挂载到云服务器，并可以随时扩容云硬盘容量。

在云容器引擎中一个节点就是具有多个云硬盘的一台弹性云服务器，您可以在创建节点时指定云硬盘的大小。

对象存储服务 OBS

对象存储服务是一个基于对象的海量存储服务，为客户提供海量、安全、高可靠、低成本的数据存储能力，包括：创建、修改、删除桶，上传、下载、删除对象等。

云容器引擎支持创建OBS对象存储卷并挂载到容器的某一路径下。

弹性文件服务 SFS

弹性文件服务（Scalable File Service）为用户提供托管的共享文件存储，符合标准文件协议（NFS），能够弹性伸缩至PB规模，具备可扩展的性能，为海量数据、高带宽型应用提供有力支持。

您可以使用弹性文件服务作为容器的持久化存储，在创建任务负载的时候挂载到容器上。

应用运维管理 AOM

应用运维管理AOM为您提供一站式立体运维平台，实时监控应用、资源运行状态，通过数十种指标、告警与日志关联分析，快速锁定问题根源，保障业务顺畅运行。

云容器引擎对接了AOM，AOM会采集容器日志存储中的“.log”等格式日志文件，转储到AOM中，方便您查看和检索；并且云容器引擎基于AOM进行资源监控，为您提供弹性伸缩能力。

