Container Service

Product Introduction

Product Introduction

Alibaba Cloud Container Service is a high-performance and scalable container management service which enables you to run distributed Docker-containerized applications on a managed cluster of Alibaba Cloud ECS instances. Being a fully-managed service, Alibaba Cloud Container Service helps you to focus on your applications rather than on managing container infrastructure. It can also be integrated with Server Load Balancer, VPC, and other cloud services, allowing you to manage container provisioning from the console or through simple APIs.

Features

Effortless cluster management

- Allows you to create or delete a cluster in a region based on your needs.
- Allows you to select either classic network or a specific VPC environment.

Multiple server hosting modes

- Allows you to dynamically scale ECS instances in a specified cluster.
- Allows you to register an existing ECS instance to a specified cluster.

One-stop container lifecycle management

- **Network:** Supports intercommunication between containers across different hosts with domain names defined by container name or hostname.
- **Storage:** Supports volume management. OSSFS is supported.
- **Log:** Supports automatic log collection.
- **Monitoring:** Facilitates monitoring of Docker containers and VM.
- **Scheduling:** Supports cross-zone scheduling of highly available nodes and automatic rescheduling of failed nodes.
- **Routing:** Supports forwarding of Layer-4 and Layer-7 requests and their binding to backend containers.
- **Sub-account:** Provides authorization management for clusters.

Docker compatibility

- Compatible with standard Docker API.

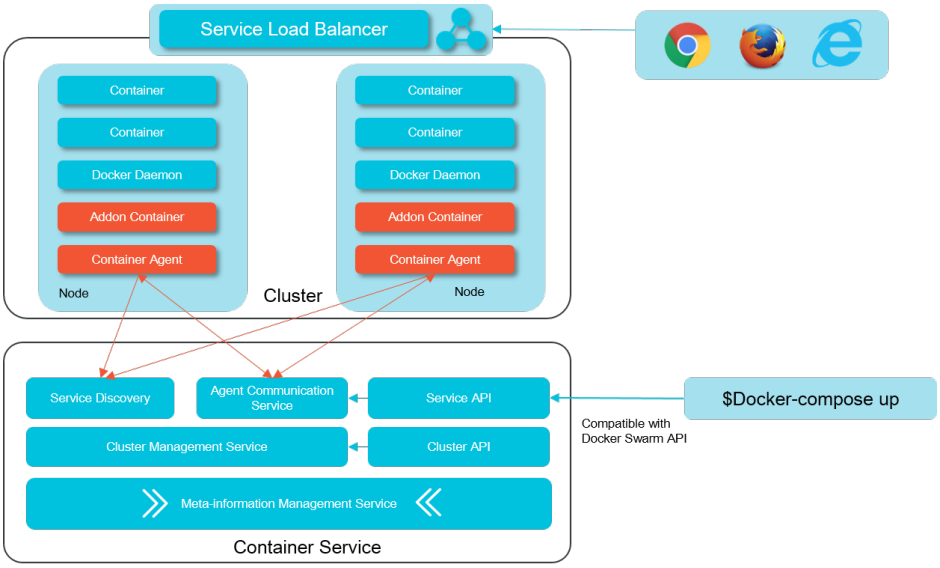
- Compatible with Docker Swarm 1.2.6.
- Compatible with Docker Engine CE 17.03.1.
- Supports Docker Compose V1/V2/V3.

Easy-to-integrate

- Closely integrates with VPC or existing software delivery process.
- Enhances Docker Compose templates with labels.
- Integrates with the Server Load Balancer service to ensure load balancing of containers.

Flexible scheduling policies

- Simplifies application delivery from development to production.
- Supports service affinity and automatic scaling.
- Supports high availability and auto-recovery across zones.
- Offers open APIs for cluster and application management, which can be easily integrated with CI/CD systems on either on-premises or off-premises.
- Supports multiple application delivery models, including rolling update and blue-green release, to ensure that the application remains up-to-date without any risks.



The basic architecture of Container Service is as shown in the preceding figure, and is described as follows.

- **Cluster management service:** Docker clusters are managed and scheduled.
- **Service discovery:** Storage of metadata (including Docker status) is supported.
- **Agent communication service:** Communication between each host and cluster management service is supported.
- **Cluster API:** Unified OpenAPIs of Alibaba Cloud are provided.
- **Service API:** APIs compatible with Docker Swarm are provided.

Ease to use

- Supports one-click creation of container clusters.
- Enables orchestration of applications using Docker Compose templates.
- Facilitates graphical user interfaces and open-sourced APIs.

Secure and controllable

- Offers configurable access to ECS servers running containers.
- Allows you to customize security groups and VPC subnet rules.
- Launches containers on your ECS instances and ensures a high level of isolation.

Protocol compatible

- Compatible with standard Docker APIs and overall Docker ecosystem.
- Supports seamless migration of applications to dockerized cloud platforms.
- Supports the Docker Compose template.
- Interoperates with APIs for third-party scheduled task delivery and system integration.
- Supports multiple hybrid cloud scenarios.

Efficient and reliable

- Able to start massive containers in seconds.
- Supports different workloads including web, mobile, HPC, event-driven and more.
- Supports automatic recovery and scaling of containers.
- Distributes container groups across multiple zones.

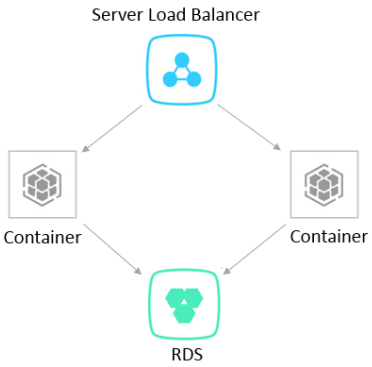
High traffic websites

For websites which have a high volume of traffic, or experience sudden spikes for a very short time, Alibaba Cloud Container Service provides automatic scaling of Docker applications. Used in integration with Server Load Balancer, the Container Service can efficiently manage traffic peaks and maintain a consistent user experience.

Solution architecture:

ECS instances + Server Load Balancer instance to the container cluster + ApsaraDB for data storage

You can use WordPress or other container images to deploy a web application in one single click.



Creation of a continuously integrated system

When a new application is launched in the market, a lot of steps are involved in deploying the application from your codebase to the production site, resulting in high overhead costs.

In such scenarios, Alibaba Cloud Container Service implements a CI/CD pipeline which reduces cost as well as time to market. This helps companies to accelerate application development, automate the deployment steps, and offer a more stable product.

Steps to implement CI/CD pipeline:

1. Create a Docker repository of the automatic build in Alibaba Cloud Container Registry service, and associate the source control management systems to GitHub or Alibaba Cloud Code.
2. Once code is committed, the Container Registry service is triggered and builds the Docker image automatically.
3. After creating an image, call back the Open API of the Container Service to update the container application.

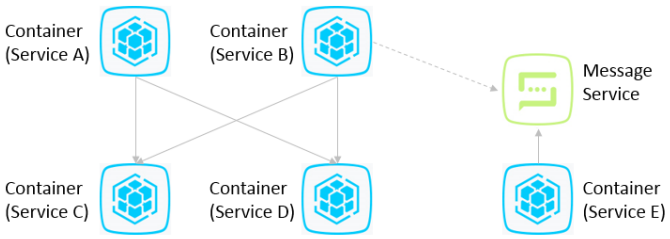
Microservice architecture

Monolithic applications are extremely complex, and can be difficult to maintain, upgrade, and update with new features. In order to update a small feature, you generally need to redeploy the entire application. Implementing microservice architecture resolves this issue by creating a single application as a suite of small, independent services that run in their own processes and are developed and deployed independently.

Alibaba Cloud Container Service packages such microservices and deploys them with Docker compose templates.

Solution architecture:

1. Split a monolithic application into several microservices, and package microservices with Docker images.
2. Use the Docker Compose template to describe the configurations and dependencies of services.
3. Deploy the application with the selected Compose template.



Cluster

A cluster is a collection of cloud resources that are required to run containers. It associates with several server nodes, Server Load Balancer instances, VPCs, and other cloud resources.

Node

A node is a server (either a VM instance or a physical server) that is installed with a Docker Engine and is used to deploy and manage clusters. The Agent program of Container Service is installed in a node and registered to a cluster. The quantity of nodes in a cluster is scalable.

Container

A container is an instance created using a Docker image. A single node can run multiple containers.

Image

A Docker image is a standard packaging format of a container application. You can specify an image to deploy container applications. The image may be from the Docker Hub, Alibaba Cloud Container Hub, or your private Registry. An image ID is uniquely identified by the URI of the image repository and the image tag name (the latest tag name is used by default).

Application template

An application template contains definitions of a group of container services and the interconnection relationships between these container services, and can be used to deploy and manage multiple container applications. The Container Service is compatible with Docker Compose and can be scaled.

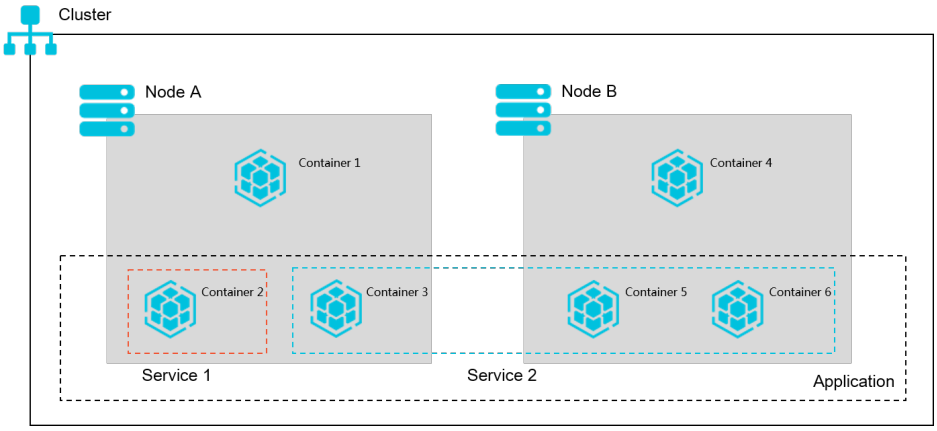
Application

An application can be created from an image or an orchestration template. Each application can contain one or more container services.

Service

A service is a group of containers running identical images with identical configurations. It is used as a scalable micro-service.

Associations



Reference

For more container related glossaries, refer to Docker glossary.