Menu

Кластеры

Обзор

Обзор

Создание кластера On-Premise

Создание кластера On-Premise

Как сделать

Добавление внешнего адреса для встроенного реестра

Выбор контейнерного рантайма

Обновление учетных данных публичного репозитория

Обзор

Кластер — это базовый набор ресурсов для запуска контейнеризованных приложений, включающий узлы, балансировщики нагрузки, хранилища и другие критически важные компоненты. Он является необходимым условием для успешного запуска контейнеризованных приложений на платформе. При первоначальной установке платформы создаётся стандартный Kubernetes кластер, известный как `global` кластер. В дальнейшем в `global` кластер можно интегрировать несколько кластеров для их единого управления.

Содержание

Тип кластера

On-Premises кластер

On-Premises кластер — это Kubernetes кластеры, создаваемые непосредственно платформой. Пользователи предоставляют виртуальные или физические машины, а платформа устанавливает и настраивает Kubernetes кластеры на этих машинах. Такой подход подходит для предприятий с уже имеющимися аппаратными ресурсами, позволяя полностью использовать инфраструктуру.

Managed кластер

Managed кластер — это Kubernetes кластеры, предоставляемые облачными провайдерами, которые интегрируются в платформу для единого управления. Поддерживаемые методы интеграции включают:

Метод	Описание	Сценарий использования	Ключевые характеристики
Import	Интеграция существующих Kubernetes кластеров	Существующие кластеры с прямым сетевым доступом	• Информация о кластере передаётся в `global` кластер • `global` кластер должен иметь сетевой доступ к кластеру
Register	Интеграция кластеров с жёсткими требованиями безопасности	Кластеры с высокими требованиями к безопасности	• На целевом кластере устанавливаются специальные плагины • Обратный прокси создаёт защищённый туннель

Метод	Описание	Сценарий использования	Ключевые характеристики
			• Обеспечивается безопасность кластера при возможности управления
Proxy Create	Создание кластеров через облачных провайдеров	Использование публичных Kubernetes сервисов облака	• Требуются учётные данные облачного провайдера • Платформа создаёт Kubernetes кластеры с использованием предоставленных учётных данных

Поддержка мультиоблаков и гибридных облаков

Эти подходы к управлению кластерами удовлетворяют потребности предприятий в сценариях мультиоблаков и гибридных облаков, поддерживая контейнерную трансформацию на разных этапах:

- Существующее оборудование: создание кластеров, предоставляемых платформой
- Существующие кластеры: импорт или регистрация в платформе
- Эластичные потребности: быстрое создание кластеров в публичном облаке

Особенности реализации и ограничения

Совместимость версий

- Поддерживаемые версии Kubernetes: 1.28, 1.29, 1.30, 1.31
- Как On-Premises, так и Managed кластеры должны обеспечивать совместимость версий
- Несовпадение версий может привести к ограничениям функционала или проблемам совместимости

Сетевые и требования безопасности

- Обеспечить сетевое соединение между `global` и целевыми кластерами
- Реализовать соответствующие политики безопасности сети и файрвола
- Безопасно управлять учётными данными доступа и механизмами аутентификации

Рекомендации по управлению кластерами

1. Предварительная оценка

- Провести тщательный анализ инфраструктуры и рабочих нагрузок
- Определить конкретные требования для каждого кластера
- Разработать комплексную стратегию миграции и интеграции

2. Безопасность и соответствие требованиям

- Внедрить управление доступом на основе ролей (RBAC)
- Использовать сетевые политики для ограничения коммуникаций между кластерами
- Регулярно проводить аудит и обновление конфигураций безопасности
- Обеспечить соответствие отраслевым стандартам и нормативам

3. Мониторинг и наблюдаемость

- Настроить централизованный сбор логов и мониторинг
- Внедрить проактивные механизмы оповещений

- Использовать инструменты наблюдаемости, предоставляемые платформой
- Отслеживать производительность кластера, использование ресурсов и состояние

4. Резервное копирование и аварийное восстановление

- Установить регулярные процедуры резервного копирования
- Создать и протестировать планы аварийного восстановления
- Реализовать стратегии резервного копирования для нескольких кластеров
- Обеспечить минимальное время простоя и потерю данных

5. Непрерывная оптимизация

- Регулярно пересматривать конфигурации кластеров
- Оптимизировать распределение ресурсов
- Обновлять до последних поддерживаемых версий Kubernetes
- Использовать возможности платформы для автоматических обновлений и масштабирования

Создание кластера On-Premise

Содержание

Предварительные требования

Требования к узлам

1. Если вы скачали пакет установки для одной архитектуры с Download Installation Package, убедитесь, что архитектура ваших узлов совпадает с архитектурой пакета. В противном случае узлы не запустятся из-за отсутствия образов, специфичных для архитектуры.

2. Проверьте, что операционная система и ядро узлов поддерживаются. Подробности смотрите в разделе Supported OS and Kernels.

3. Выполните проверки доступности узлов. Для конкретных пунктов проверки обратитесь к разделу Node Preprocessing > Node Checks.

4. Если IP-адреса узлов недоступны напрямую по SSH, предоставьте для узлов SOCKS5 прокси. Кластер `global` будет обращаться к узлам через этот прокси-сервис.

Балансировка нагрузки

Для производственных сред необходим балансировщик нагрузки для узлов управляющей плоскости кластера, чтобы обеспечить высокую доступность.
Вы можете использовать собственный аппаратный балансировщик нагрузки или включить `Self-built VIP`, который обеспечивает программную балансировку нагрузки с помощью haproxy + keepalived.
Рекомендуется использовать аппаратный балансировщик нагрузки, поскольку:

- **Лучшая производительность**: Аппаратная балансировка работает эффективнее программной.
- **Меньшая сложность**: Если вы не знакомы с keepalived, неправильная настройка может привести к недоступности кластера, что вызовет длительный поиск и устранение неисправностей и серьезно повлияет на надежность кластера.

При использовании собственного аппаратного балансировщика нагрузки в качестве параметра `IP Address / Domain` можно указать VIP балансировщика. Если у вас есть доменное имя, разрешающееся в VIP балансировщика, можно использовать это доменное имя.
Примечание:

- Балансировщик нагрузки должен корректно перенаправлять трафик на порты `6443`, `11780` и `11781` на всех узлах управляющей плоскости кластера.
- Если в кластере только один узел управляющей плоскости и вы используете IP этого узла как `IP Address / Domain`, то впоследствии масштабировать кластер с одного узла до высокодоступного многозвенного нельзя. Поэтому рекомендуется использовать балансировщик нагрузки даже для однозвенных кластеров.

При включении `Self-built VIP` необходимо подготовить:

1. Доступный VRID

2. Хост-сеть, поддерживающую протокол VRRP

3. Все узлы управляющей плоскости и VIP должны находиться в одной подсети, при этом VIP должен отличаться от IP-адресов узлов.

Подключение кластера `global` и рабочих кластеров

Платформа требует взаимного доступа между кластером `global` и рабочими кластерами. Если они находятся в разных сетях, необходимо:

1. Обеспечить `External Access` для рабочего кластера, чтобы кластер `global` мог получить к нему доступ. Требования к сети: `global` должен иметь доступ к портам `6443`, `11780` и `11781` на всех узлах управляющей плоскости.

2. Добавить дополнительный адрес в `global`, доступный рабочему кластеру. При создании рабочего кластера добавьте этот адрес в аннотации кластера с ключом `cpaas.io/platform-url` и значением, равным публичному адресу доступа к `global`.

Реестр образов

Образы кластера поддерживают варианты: встроенный в платформу, приватный репозиторий и публичный репозиторий.

- **Platform Built-in**: Использует реестр образов, предоставляемый кластером `global`. Если кластер не может получить доступ к `global`, смотрите Add External Address for Built-in Registry.
- **Private Repository**: Использует ваш собственный реестр образов. Для подробностей по загрузке необходимых образов в ваш реестр обратитесь в техническую поддержку.
- **Public Repository**: Использует публичный реестр образов платформы. Перед использованием выполните Updating Public Repository Credentials.

Сетевое взаимодействие контейнеров

Если вы планируете использовать Kube-OVN Underlay для вашего кластера, обратитесь к разделу Preparing Kube-OVN Underlay Physical Network.

Процедура создания

1. Перейдите в представление **Platform Management** и в левом навигационном меню выберите **Clusters/Clusters**.
2. Нажмите **Create Cluster**.
3. Настройте следующие разделы согласно инструкциям ниже: Basic Info, Container Network, Node Settings и Extended Parameters.

Basic Info

Параметр	Описание
Kubernetes Version	Все доступные версии тщательно протестированы на стабильность и совместимость. **Рекомендация:** Выбирайте последнюю версию для оптимальных возможностей и поддержки.
Container Runtime	По умолчанию используется containerd. Если вы предпочитаете Docker в качестве runtime, обратитесь к разделу Choosing a Container Runtime.

Cluster Network Protocol	Поддерживаются три режима: IPv4 single stack, IPv6 single stack, IPv4/IPv6 dual stack. **Примечание:** При выборе dual stack убедитесь, что на всех узлах корректно настроены IPv6-адреса; после установки изменить сетевой протокол нельзя.
Cluster Endpoint	`IP Address / Domain`: Введите заранее подготовленное доменное имя или VIP, если домена нет. `Self-Built VIP`: По умолчанию отключено. Включайте только если не предоставлен LoadBalancer. При включении установщик автоматически развернет `keepalived` для поддержки программной балансировки нагрузки. `External Access`: Введите внешний адрес доступа, подготовленный для кластера, если он находится в другой сетевой среде, нежели `global`.

Container Network

Kube-OVN

Корпоративная система оркестрации сетей контейнеров Cloud Native Kubernetes, разработанная Alauda. Она переносит зрелые сетевые возможности из области OpenStack в Kubernetes, поддерживает управление сетями между облаками, традиционную сетевую архитектуру и инфраструктурное межсоединение, сценарии развертывания на периферийных кластерах, значительно повышая безопасность, эффективность управления и производительность сетей контейнеров Kubernetes.

Параметр	Описание
Subnet	Также известен как Cluster CIDR, представляет **подсеть по умолчанию**. После создания кластера можно добавлять дополнительные подсети.
Transmit Mode	**Overlay**: Виртуальная сеть, абстрагированная над инфраструктурой, не потребляющая физические сетевые ресурсы. При создании Overlay подсети по умолчанию все Overlay подсети кластера используют одинаковую конфигурацию cluster NIC и node NIC. **Underlay**: Этот режим передачи опирается на физические сетевые устройства. Он может напрямую выделять физические сетевые адреса для Pod, обеспечивая лучшую производительность и связь с физической сетью. Узлы в Underlay подсети должны иметь несколько NIC, а NIC, используемый для мостовой сети, должен использоваться исключительно для Underlay и не должен нести другой трафик, например SSH. При создании Underlay подсети по умолчанию cluster NIC фактически является NIC по умолчанию для мостовой сети, а node NIC — это конфигурация NIC узла в мостовой сети. • **Default Gateway**: Адрес шлюза физической сети, который является шлюзом для сегмента Cluster CIDR (должен находиться в диапазоне адресов Cluster CIDR). • **VLAN ID**: Идентификатор виртуальной локальной сети (номер VLAN), например, `0`. • **Reserved IPs**: Укажите зарезервированные IP-адреса, которые не будут автоматически выделяться, например, IP-адреса в подсети, уже используемые другими устройствами.
Service CIDR	Диапазон IP-адресов, используемый Kubernetes Service типа ClusterIP. Не должен пересекаться с диапазоном подсети по умолчанию.

Join CIDR	В режиме передачи Overlay это диапазон IP-адресов для связи между узлами и pod. Не должен пересекаться с подсетью по умолчанию или Service CIDR.

Calico

Calico — решение сетевого уровня 3, обеспечивающее безопасные сетевые соединения для контейнеров.

Параметр	Описание
Default Subnet	Также известен как Cluster CIDR, представляет **подсеть по умолчанию**. После создания кластера можно добавлять дополнительные подсети.
Service CIDR	Диапазон IP-адресов, используемый Kubernetes Service типа ClusterIP. Не должен пересекаться с диапазоном подсети по умолчанию.

Flannel

Flannel обеспечивает плоскую сетевую среду для всех контейнеров в кластере, предоставляя контейнерам, созданным на разных узлах, уникальный виртуальный IP-адрес по всему кластеру. Подсеть pod равномерно делится между узлами кластера согласно маске, и pod на каждом узле получают IP-адреса из сегмента, выделенного этому узлу. Это повышает эффективность связи между контейнерами без необходимости учитывать вопросы трансляции IP.

Параметр	Описание
Cluster CIDR	Диапазон IP-адресов, используемый pod, создаваемыми при запуске кластера. Поддерживается настройка максимального

	количества IP-адресов, которые могут быть выделены pod на каждом узле в текущей контейнерной сети. **Примечание**: Платформа автоматически рассчитает максимальное количество узлов, которое может вместить кластер, исходя из этой настройки, и отобразит это в подсказке под полем ввода. **Важно**: После создания кластера сетевую конфигурацию изменить нельзя, поэтому планируйте сеть внимательно.
Service CIDR	Диапазон IP-адресов, используемый Kubernetes Service типа ClusterIP. Не должен пересекаться с диапазоном подсети контейнеров.

Custom

Если необходимо установить другие сетевые плагины, выберите режим **Custom**. Вы сможете вручную установить сетевые плагины после успешного создания кластера.

Параметр	Описание
Cluster CIDR	Диапазон IP-адресов, используемый pod, создаваемыми при запуске кластера.
Service CIDR	Диапазон IP-адресов, используемый Kubernetes Service типа ClusterIP. Не должен пересекаться с диапазоном подсети контейнеров.

Node Settings

Параметр	Описание
Network Interface Card	Имя сетевого интерфейса хоста, используемого сетевым плагином кластера. **Примечание**: • При выборе режима передачи Underlay для подсети по умолчанию Kube-OVN необходимо указать имя сетевого интерфейса, который будет использоваться как NIC по умолчанию для мостовой сети. - По умолчанию мониторинг трафика сетевых интерфейсов платформы распознает трафик на интерфейсах с именами, начинающимися на `eth.|en.|wl.|ww.`. Если вы используете интерфейсы с другими именами, после добавления кластера обратитесь к разделу Collect Network Data from Custom-Named Network Interfaces для изменения соответствующих ресурсов и обеспечения корректного мониторинга трафика.
Node Name	Можно выбрать использование IP-адреса узла или имени хоста в качестве имени узла на платформе. **Примечание**: При выборе имени хоста убедитесь, что имена хостов узлов, добавляемых в кластер, уникальны.
Nodes	**Добавьте узлы** в кластер или **Восстановите из черновика** временно сохранённую информацию об узлах. Подробные описания параметров добавления узлов приведены ниже.
Monitoring Type	Поддерживаются **Prometheus** и **VictoriaMetrics**. При выборе **VictoriaMetrics** необходимо указать **Deploy Type**:

	- Deploy VictoriaMetrics: Разворачивает все компоненты, включая **VMStorage**, **VMAlert**, **VMAgent** и др. **- Deploy VictoriaMetrics Agent**: Разворачивает только компонент сбора логов **VMAgent**. При таком способе необходимо связать с уже развернутым экземпляром VictoriaMetrics на другом кластере платформы для предоставления мониторинга.
Monitoring Nodes	Выберите узлы для развертывания компонентов мониторинга кластера. Поддерживается выбор вычислительных узлов и узлов управляющей плоскости, разрешающих развертывание приложений. Чтобы не влиять на производительность кластера, рекомендуется отдавать предпочтение вычислительным узлам. После успешного создания кластера компоненты мониторинга с типом хранения **Local Volume** будут развернуты на выбранных узлах.

Параметры добавления узлов

Параметр	Описание
Type	**Control Plane Node**: Отвечает за запуск компонентов kube-apiserver, kube-scheduler, kube-controller-manager, etcd, контейнерной сети и некоторых компонентов управления платформой в кластере. При включенной опции **Application Deployable** узлы управляющей плоскости могут использоваться как вычислительные. **Worker Node**: Отвечает за размещение бизнес-подов, работающих в кластере.

IPv4 Address	IPv4-адрес узла. Для кластеров, созданных в режиме внутренней сети, укажите **приватный IP** узла.
IPv6 Address	Действительно при включенном dual stack IPv4/IPv6. IPv6-адрес узла.
Application Deployable	Действительно при типе узла **Control Plane Node**. Разрешить ли развертывание бизнес-приложений на этом узле управляющей плоскости, то есть планировать бизнес-поды на этот узел.
Display Name	Отображаемое имя узла.
SSH Connection IP	IP-адрес, по которому можно подключиться к узлу через SSH. Если вы можете войти на узел с помощью `ssh <username>@<IPv4-адрес узла>`, этот параметр не обязателен; в противном случае укажите публичный IP или внешний NAT IP узла, чтобы кластер `global` и прокси могли подключаться к узлу по этому IP.
Network Interface Card	Укажите имя сетевого интерфейса, используемого узлом. Приоритет эффективности конфигурации сетевого интерфейса следующий (слева направо, по убыванию): **Kube-OVN Underlay**: Node NIC > Cluster NIC **Kube-OVN Overlay**: Node NIC > Cluster NIC > NIC, соответствующий default route узла

	Calico: Cluster NIC > NIC, соответствующий default route узла **Flannel**: Cluster NIC > NIC, соответствующий default route узла
Associated Bridge Network	**Примечание**: При создании кластера конфигурация мостовой сети не поддерживается; этот параметр доступен только при **добавлении узлов** в кластер, в котором уже созданы Underlay подсети. Выберите существующую Add Bridge Network. Если не хотите использовать NIC по умолчанию мостовой сети, можно отдельно настроить node NIC.
SSH Port	Номер порта SSH, например, `22`.
SSH Username	Имя пользователя SSH с правами root, например, `root`.
Proxy	Использовать ли прокси для доступа к SSH-порту узла. Если кластер `global` не может напрямую подключиться к добавляемому узлу по SSH (например, кластер `global` и рабочий кластер находятся в разных подсетях; IP узла — внутренний, недоступный для `global`), включите этот переключатель и настройте параметры прокси. После настройки прокси доступ к узлу и развертывание будут осуществляться через прокси. **Примечание**: В настоящее время поддерживается только SOCKS5 прокси.

	Access URL: Адрес прокси-сервера, например, `192.168.1.1:1080`. **Username**: Имя пользователя для доступа к прокси-серверу. **Password**: Пароль для доступа к прокси-серверу.
SSH Authentication	Метод аутентификации и соответствующая информация для входа на добавляемый узел. Варианты: **Password**: Требуется имя пользователя с правами root и соответствующий **SSH пароль**. **Key**: Требуется **приватный ключ** с правами root и **пароль к приватному ключу**.
Save Draft	Сохраняет текущие настройки в диалоге как черновик и закрывает окно **Add Node**. Не покидая страницу **Create Cluster**, можно выбрать **Restore from draft** для открытия диалога **Add Node** и восстановления сохранённых данных. **Примечание**: Восстанавливаются данные самого последнего сохранённого черновика.

Extended Parameters

Примечание:

- Помимо обязательных настроек, не рекомендуется задавать расширенные параметры, так как неправильные настройки могут сделать кластер недоступным, а

после создания изменить их нельзя.

- Если введённый **Key** совпадает с ключом параметра по умолчанию, он переопределит стандартную конфигурацию.

Процедура

1. Нажмите **Extended Parameters** для раскрытия области настройки расширенных параметров. При необходимости можно задать следующие параметры:

Параметр	Описание
Docker Parameters	`dockerExtraArgs` — дополнительные параметры конфигурации Docker, которые будут записаны в `/etc/sysconfig/docker`. Рекомендуется не изменять. Для настройки Docker через файл `daemon.json` параметры должны быть заданы в формате ключ-значение.
Kubelet Parameters	`kubeletExtraArgs` — дополнительные параметры конфигурации Kubelet. **Примечание**: При вводе параметра **Node IP Count** в разделе **Container Network** автоматически создаётся параметр **Kubelet** с ключом `max-pods` и значением **Node IP Count**, задающий максимальное количество pod, которые могут работать на любом узле кластера. Этот параметр не отображается в интерфейсе. Если добавить новый параметр `max-pods: максимальное количество pod` в разделе **Kubelet Parameters**, он переопределит значение по умолчанию. Допускается любое положительное целое число, но рекомендуется использовать значение по умолчанию (Node IP Count) или не превышать `256`.

Controller Manager Parameters	`controllerManagerExtraArgs` — дополнительные параметры конфигурации Controller Manager.
Scheduler Parameters	`schedulerExtraArgs` — дополнительные параметры конфигурации Scheduler.
APIServer Parameters	`apiServerExtraArgs` — дополнительные параметры конфигурации APIServer.
APIServer URL	`publicAlternativeNames` — адреса доступа к APIServer, указанные в сертификате. Можно вводить только IP или доменные имена, максимум 253 символа.
Cluster Annotations	Аннотации кластера — метаданные в виде ключ-значение, отмечающие характеристики кластера для получения соответствующей информации компонентами платформы или бизнес-компонентами.

1. Нажмите **Create**. Вы вернётесь на страницу списка кластеров, где статус кластера будет **Creating**.

Действия после создания

Просмотр прогресса создания

На странице списка кластеров можно просмотреть список созданных кластеров. Для кластеров со статусом **Creating** доступен просмотр прогресса выполнения.

Процедура

1. Нажмите маленькую иконку **View Execution Progress** справа от статуса кластера.

2. В появившемся диалоговом окне можно просмотреть прогресс выполнения кластера (status.conditions).

   Совет: Если какой-либо тип находится в процессе или в состоянии ошибки с причиной, наведите курсор на причину (отображается синим текстом), чтобы увидеть подробную информацию о причине (status.conditions.reason).

Ассоциация с проектами

После создания кластера вы можете добавить его в проекты в представлении управления проектами.

Menu

Как сделать

Добавление внешнего адреса для встроенного реестра

Содержание

Overview

Когда кластер `global` использует реестр `Platform Built-in`, кластеры рабочих нагрузок обычно также используют этот реестр для загрузки образов. Реестр обслуживает не только компоненты внутри кластера `global`, но и должен быть доступен узлам кластеров рабочих нагрузок.

В некоторых сценариях узлы кластера рабочих нагрузок не могут напрямую получить доступ к адресу реестра кластера `global` — например, когда кластер `global` находится в частном дата-центре, а кластеры рабочих нагрузок — в публичных облаках или на периферии.

В этом руководстве объясняется, как настроить внешний доступный адрес для реестра платформы по умолчанию, чтобы кластеры рабочих нагрузок могли загружать образы.

Prerequisites

Перед началом подготовьте следующее:

- Доменное имя, доступное для узлов кластера рабочих нагрузок
- IP-адрес, на который указывает доменное имя
- Действительный SSL-сертификат для доменного имени

WARNING

- Доменное имя должно отличаться от адреса доступа к платформе
- Убедитесь, что IP-адрес домена может перенаправлять трафик на все узлы управляющей плоскости кластера `global`

Procedure

Настройка сертификата и правил маршрутизации для реестра платформы

1. Скопируйте действительный сертификат домена на любой узел управляющей плоскости кластера `global`
2. Создайте TLS-секрет, содержащий сертификат домена:

```
kubectl create secret tls registry-address.tls --cert=<certificate-filename> --key=
<key-filename> -n kube-system
```

Пример:

```
kubectl create secret tls registry-address.tls --cert=custom.crt --key=custom.key -n
kube-system
```

Примечание: После создания сертификата следите за сроком действия секрета **registry-address.tls** в пространстве имён **kube-system** кластера `global`. Замените

сертификат до его истечения.

3. Создайте ingress-правила на любом узле управляющей плоскости кластера `global`:

```
REGISTRY_DOMAIN_NAME=<www.registry.com> # Замените на ваше доступное доменное имя
cat << EOF | kubectl create -f -
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    nginx.ingress.kubernetes.io/backend-protocol: HTTPS
  name: registry-address
  namespace: kube-system
  labels:
    service_name: registry
spec:
  rules:
    - host: $REGISTRY_DOMAIN_NAME
      http:
        paths:
          - backend:
              service:
                name: registry
                port:
                  number: 443
            path: /v2/
            pathType: ImplementationSpecific
          - backend:
              service:
                name: registry
                port:
                  number: 443
            path: /v2/_catalog
            pathType: ImplementationSpecific
          - backend:
              service:
                name: registry
                port:
                  number: 443
            path: /v2/.+/tags/list
            pathType: ImplementationSpecific
          - backend:
              service:
                name: registry
                port:
                  number: 443
            path: /v2/.+/manifests/[A-Za-z0-9_+.-:]+
```

```
            pathType: ImplementationSpecific
          - backend:
              service:
                name: registry
                port:
                  number: 443
            path: /v2/.+/blobls/[A-Za-z0-9-:]+
            pathType: ImplementationSpecific
          - backend:
              service:
                name: registry
                port:
                  number: 443
            path: /v2/.+/blobls/uploads/[A-Za-z0-9-:]+
            pathType: ImplementationSpecific
          - backend:
              service:
                name: registry
                port:
                  number: 443
            path: /auth/token
            pathType: ImplementationSpecific
  tls:
    - secretName: registry-address.tls
      hosts:
        - $REGISTRY_DOMAIN_NAME
EOF
```

Ответ, похожий на `... created`, означает успешное создание ingress.

4. Проверьте, существует ли ресурс Service для реестра:

```
kubectl -n kube-system get svc | grep registry
```

Если Service отсутствует, создайте его:

```
cat << EOF | kubectl create -f -
apiVersion: v1
kind: Service
metadata:
  labels:
    name: registry
    service_name: registry
  name: registry
  namespace: kube-system
spec:
  ports:
    - protocol: TCP
      port: 443
      targetPort: 60080
  selector:
    component: registry
  type: ClusterIP
EOF
```

5. Проверьте конфигурацию, загрузив образ из реестра по доменному имени:

```
crictl pull <registry-domain-name>/automation/qaimages:helloworld
```

Или

```
docker pull <registry-domain-name>/automation/qaimages:helloworld
```

Выбор контейнерного рантайма

Содержание

Overview

Container Runtime — это ключевой компонент Kubernetes, отвечающий за управление жизненным циклом образов и контейнеров.

При создании кластеров через платформу вы можете выбрать в качестве компонента рантайма либо Containerd, либо Docker.

Примечание: Kubernetes версии 1.24 и выше официально больше не поддерживает Docker runtime. Официально рекомендуемый рантайм — Containerd. Если вам всё же необходимо использовать Docker runtime, сначала нужно включить `cri-docker` в feature gate, после чего при создании кластера можно будет выбрать Docker в качестве компонента рантайма. Подробнее о работе с feature gates смотрите в разделе Feature Gate Configuration.

Быстрый гид по выбору

Выберите Containerd	Выберите Docker
• Более короткая цепочка вызовов • Меньше компонентов • Более стабильный • Меньше потребляет ресурсов узла	• Поддержка docker-in-docker • Возможность использовать команды `docker build/push/save/load` на узлах • Возможность вызова Docker API • Поддержка docker compose или docker swarm

Отличия между Docker и Containerd

Общие команды

Containerd	Docker	Описание
crictl ps	docker ps	Просмотр запущенных контейнеров
crictl inspect	docker inspect	Просмотр деталей контейнера
crictl logs	docker logs	Просмотр логов контейнера
crictl exec	docker exec	Выполнение команд в контейнере
crictl attach	docker attach	Подключение к контейнеру
crictl stats	docker stats	Отображение использования ресурсов контейнером
crictl create	docker create	Создание контейнера
crictl start	docker start	Запуск контейнера

Containerd	Docker	Описание
crictl stop	docker stop	Остановка контейнера
crictl rm	docker rm	Удаление контейнера
crictl images	docker images	Просмотр списка образов
crictl pull	docker pull	Загрузка образа
None	docker push	Отправка образа
crictl rmi	docker rmi	Удаление образа
crictl pods	None	Просмотр списка подов
crictl inspectp	None	Просмотр деталей пода
crictl runp	None	Запуск пода
crictl stopp	docker images	Просмотр образов
ctr images ls	None	Остановка пода
crictl stopp	docker load/save	Импорт/экспорт образов
ctr images import/export	None	Остановка пода
ctr images pull/push	docker pull/push	Загрузка/отправка образов
ctr images tag	docker tag	Тегирование образов

Отличия в цепочке вызовов

- Docker в качестве контейнерного рантайма Kubernetes имеет следующую цепочку вызовов:

  kubelet > cri-dockerd > dockerd > containerd > runC

- Containerd в качестве контейнерного рантайма Kubernetes имеет следующую цепочку вызовов:

  kubelet > cri plugin (в процессе containerd) > containerd > runC

Итог: Несмотря на то, что dockerd добавляет такие возможности, как swarm cluster, docker build и Docker API, он может приводить к ошибкам и добавляет дополнительный уровень в цепочку вызовов. Containerd имеет более короткую цепочку вызовов, меньше компонентов, большую стабильность и меньше потребляет ресурсов узла.

Сравнение логов и параметров

Сравнение	Docker	Containerd
Путь хранения	При использовании Docker в качестве контейнерного рантайма Kubernetes логи контейнеров хранятся Docker в каталогах вида `/var/lib/docker/containers/$CONTAINERID`. Kubelet создаёт символические ссылки в `/var/log/pods` и `/var/log/containers`, указывающие на файлы логов в этом каталоге.	При использовании Containerd в качестве контейнерного рантайма Kubernetes логи контейн хранятся Kubelet в катал `/var/log/pods/$CONTAINER_` при этом в каталоге `/var/log/containers` создаются символическ ссылки на файлы логов.
Параметры конфигурации	Задаются в конфигурационном файле Docker: `"log-driver": "json-file", "log-opts": {"max-size": "100m","max-file": "5"}`	*Метод 1*: Задаются параметрами kubelet: `--container-log-max-files` `--container-log-max-size="100Mi"` *Метод 2*: Задаются в KubeletConfiguration: `"containerLogMaxSize": "100Mi", "containerLogMaxFiles": 5`
Сохранение логов	Монтировать диск данных в "data-root" (по умолчанию `/var/lib/docker`).	Создать символическую ссылку `/var/log/pods`,

Сравнение	Docker	Containerd
контейнеров на диске данных		указывающую на катало точке монтирования дис данных.

Сравнение CNI-сети

Сравнение	Docker	Containerd
Кто вызывает CNI	cri-dockerd	cri-plugin, встроенный в Containerd (начиная с containerd 1.1)
Как настраивать CNI	Параметры cri-dockerd: `--cni-conf-dir`, `--cni-bin-dir`, `--cni-cache-dir`	Конфигурационный файл Containerd (toml): `[plugins.cri.cni]` `bin_dir = "/opt/cni/bin"` `conf_dir = "/etc/cni/net.d"`

Обновление учетных данных публичного репозитория

Содержание

Overview

`Public Repository` — это сервис реестра образов, предоставляемый платформой и доступный в публичном интернете. Если вы хотите, чтобы ваши кластеры использовали `Public Repository` в качестве реестра образов, необходимо обновить встроенные облачные учетные данные `public-registry-credential`. Это гарантирует, что ваша платформа имеет разрешение на загрузку образов из публичного реестра.

Procedure

1. Войдите в **Customer Portal** и скачайте файл аутентификации вашей организации из раздела **Enterprise Management**, расположенного в выпадающем меню **User Information** в правом верхнем углу.
2. Перейдите в **Clusters** > **Cloud Credential** в левой навигационной панели консоли **Platform Management**.

3. Найдите облачные учетные данные с именем `public-registry-credential` и выберите **Update** в выпадающем меню справа.

4. В разделе **Upload Public Repository Address** загрузите файл аутентификации, который вы скачали из **Customer Portal**.

5. Нажмите **Update**, чтобы применить изменения.