

首云容器产品Kubernetes操作指南

简介

- 1.产品简介
- 2.使用须知

集群管理

- 1.简介
- 2.使用须知
- 3.操作说明

节点管理

- 1.简介
- 2.使用须知
- 3.操作说明

存储管理

- 1.简介
- 2.操作说明

应用管理

- 1.简介
- 2.前提条件
- 3.操作说明

网络管理

- 1.简介
- 2.操作说明

监控管理

- 1.简介
- 2.操作说明
 - (1) 开启监控
 - (2) 概览页监控查看以及资源介绍
 - (3) 查看详细监控

命名空间管理

- 1.简介

- 2.操作说明

弹性伸缩

1. 工作原理

- 2.创建自动伸缩

案例--如何创建一个Nginx

1. 创建无状态应用nginx

2. 配置容器

3. 高级配置

4. 创建成功

案例--如何创建一个WordPress

方法一：nodePort + Haproxy 外网访问 WordPress 配置方法

1. 创建 StorageClass、Namespace 和 pvc 资源

2. 部署MySQL容器组

3. 部署WordPress容器组

4. 配置 Haproxy 负载均衡

5. 访问 WordPress

方法二：Ingress 外网访问 WordPress 配置方法

1. 创建 Namespace 和 PVC 资源

2. 部署MySQL容器组

3. 部署WordPress容器组

4. 创建 Service

5. 创建 Ingress

6. Haproxy 策略配置上述 Ingress 与服务映射的 80 端口

7. 访问 Wordpress

简介

1.产品简介

容器服务 Kubernetes 版（CCK），提供高性能可伸缩的容器应用管理能力，支持Kubernetes社区原生应用和工具。简化集群的搭建和扩容等运维类工作，整合首云虚拟化（裸金属）、存储、网络和安全能力，打造云端最佳的容器化应用运行环境。

2.使用须知

目前开放节点：无锡A，东京A，香港A，新加坡A，达拉斯A，法兰克福A。

注：根据客户需求可以一天内在新节点部署好容器服务。

集群管理

1.简介

首云支持

集群管理操作，包括集群创建、删除和控制台访问集群

2.使用须知

需通过首云集群管理页面进行上述操作

3.操作说明

- 创建集群
 - 进入集群页面 -> 右上角点击创建集群



集群ID	集群名称	VDC	节点个数	创建时间	集群状态	操作
84d9e300-5f54-11ea-8336-0242ac110451	zbh-0306	zhangbaohong-test	11	2020-03-06 10:45:20	正常	管理节点 集群扩容 控制台 更多
982689be-61f1-11ea-bf07-0242ac1104a3	test-wrwh-1	模板传输测试-达拉斯	5	2020-03-09 18:34:46	正常	管理节点 集群扩容 控制台 更多
5ce554ca-6368-11ea-9a4b-0242ac11050d	test-wrwh-2	模板传输测试-达拉斯	10	2020-03-11 15:17:27	正常	管理节点 集群扩容 控制台 更多
853de7a4-67fd-11ea-a494-0242ac11005f	TEST	广州A	4	2020-03-17 11:15:15	正常	管理节点 集群扩容 控制台 更多
1ce56738-68ba-11ea-8c5d-0242ac11006d	test318	模板传输测试-无锡	6	2020-03-18 10:13:53	正常	管理节点 集群扩容 控制台 更多
8cb802ba-69c1-11ea-97ce-0242ac11009a	zbh-0319	zhangbaohong-test	4	2020-03-19 17:11:00	正常	管理节点 集群扩容 控制台 更多
f7b0a08a-6cce-11ea-9ab7-0242ac1100bd	zbh-0323	zhangbaohong-test	4	2020-03-23 14:24:36	正常	管理节点 集群扩容 控制台 更多

- 设置集群名称 -> 选择虚拟数据中心 -> 选择集群私网ip网段 -> 选择计费方式-> 设置master节点 -> 设置worker节点 -> 选择HA配置 -> 选择集群公网ip -> 设置集群ssh登录密码 -> 确认无误后点击确认

1 集群名称

注：长度为1-26个字符，只能包含数字、字母和“-”，且首尾只能是字母或数字！

2 虚拟数据中心

全部 hfeng-k8s-dev-guangzhou

hfeng-k8s-dev-guangzhou 中国大陆-广州-可用区A	hostName测试 亚太地区-新加坡-可用区A	Singapore-hys 亚太地区-新加坡-可用区A
test-容器 亚太地区-香港-可用区A	zhangbaohong-test 中国大陆-无锡-可用区A	广州A 中国大陆-广州-可用区A

3 集群网络

私网1 10.241.85.0/16-10.241.85.255/16 可用数量254个
建议选择B类网段

4 计费方式

按需计费

5 Master配置

计算类型与规格

8核8G 8核16G 8核32G 8核64G 16核16G 16核32G 16核64G 16核128G

云硬盘类型	容量 (GB)	最高IOPS	操作
性能型	60	600	

本地盘 Master还可以添加14块硬盘

添加云硬盘

购买数量 3台

6 Worker配置

计算类型与规格

8核8G 8核16G 8核32G 8核64G 16核16G 16核32G 16核64G 16核128G

云硬盘类型	容量 (GB)	最高IOPS	操作
性能型	60	600	

本地盘 Worker还可以添加14块硬盘

添加云硬盘

购买数量 1台

7 HA配置 (集群入网)

实例规格 1核2G 2核4G 4核8G 8核16G 16核32G

最大连接数：50000，超出50000之后，会出现丢失情况。

103.228.160.64/29 可用IP数量为：2个

HA网络 外网IP数量最少为3个才能购买HaProxy
选取地址段后，在对应的网段内自动分配haproxy的虚拟IP，主节点IP地址，备节点IP地址

8 公网配置 (集群出网)

103.228.160.64/29 可用IP数量为：2个
作为整个集群访问公网的出口，比如拉取镜像等操作。

9 master、worker 登陆密码

(master节点和worker节点的登陆名为：ccp)

请输入密码

8-30个字符，且同时包含三项(大写字母、小写字母、数字、特殊符号)

请再次输入密码

当前配置信息

虚拟数据中心: hfeng-k8s-dev-guangzhou
 集群网络: 私网1
 计费方式: 按需计费
 集群服务费: ¥ 2.74/每天

Master配置
 云服务器规格: 8核8G ¥ 22.67/每天
 性能型: 60GB ¥ 0.00/每天
 购买数量: 3

Worker配置
 云服务器规格: 8核8G ¥ 22.67/每天
 性能型: 60GB ¥ 0.00/每天
 购买数量: 1

HA配置
 实例规格: 1核2G ¥ 4.68/每天
 网络: 公网1

10 当前配置总价: ¥ 98.10/每天

确认

- 进入集群页面 -> 查看创建的集群，状态为正常代表创建成功

集群

请选择虚拟数据中心

请输入集群ID或者名称查询

集群ID	集群名称	VDC	节点个数	创建时间	集群状态	操作
84d9e300-5f54-11ea-8336-0242ac110451	zbh-0306	zhangbaohong-test	11	2020-03-06 10:45:20	正常	管理节点 集群扩容 控制台 更多
982689be-61f1-11ea-bf07-0242ac1104a3	test-wrwh-1	模板传输测试-达拉斯	5	2020-03-09 18:34:46	正常	管理节点 集群扩容 控制台 更多
5ce554ca-6368-11ea-9a4b-0242ac11050d	test-wrwh-2	模板传输测试-达拉斯	10	2020-03-11 15:17:27	正常	管理节点 集群扩容 控制台 更多
853de7a4-67fd-11ea-a494-0242ac11005f	TEST	广州A	4	2020-03-17 11:15:15	正常	管理节点 集群扩容 控制台 更多
1ce56738-68be-11ea-8c9d-0242ac11006d	test318	模板传输测试-无锡	6	2020-03-18 10:13:53	正常	管理节点 集群扩容 控制台 更多
8cb802ba-69c1-11ea-97ce-0242ac11009a	zbh-0319	zhangbaohong-test	4	2020-03-19 17:11:00	正常	管理节点 集群扩容 控制台 更多
f7b0a08a-6cce-11ea-9ab7-0242ac1100bd	zbh-0323	zhangbaohong-test	4	2020-03-23 14:24:36	正常	管理节点 集群扩容 控制台 更多
2363ace4-6cd5-11ea-9d3f-0242ac1100bd	zbh-0323-2	zhangbaohong-test	5	2020-03-23 15:08:47	正常	管理节点 集群扩容 控制台 更多

访问集群

- 进入集群页面 -> 选择集群，点击管理节点

集群

请选择虚拟数据中心

请输入集群ID或者名称查询

集群ID	集群名称	VDC	节点个数	创建时间	集群状态	操作
84d9e300-5f54-11ea-8336-0242ac110451	zbh-0306	zhangbaohong-test	11	2020-03-06 10:45:20	正常	2 管理节点 集群扩容 控制台 更多
982689be-61f1-11ea-bf07-0242ac1104a3	test-wrwh-1	模板传输测试-达拉斯	5	2020-03-09 18:34:46	正常	管理节点 集群扩容 控制台 更多
5ce554ca-6368-11ea-9a4b-0242ac11050d	test-wrwh-2	模板传输测试-达拉斯	10	2020-03-11 15:17:27	正常	管理节点 集群扩容 控制台 更多
853de7a4-67fd-11ea-a494-0242ac11005f	TEST	广州A	4	2020-03-17 11:15:15	正常	管理节点 集群扩容 控制台 更多
1ce56738-68be-11ea-8c9d-0242ac11006d	test318	模板传输测试-无锡	6	2020-03-18 10:13:53	正常	管理节点 集群扩容 控制台 更多
8cb802ba-69c1-11ea-97ce-0242ac11009a	zbh-0319	zhangbaohong-test	4	2020-03-19 17:11:00	正常	管理节点 集群扩容 控制台 更多

- 选择要进入的节点 -> 点击控制台

zbh-0306 / 节点列表

创建节点

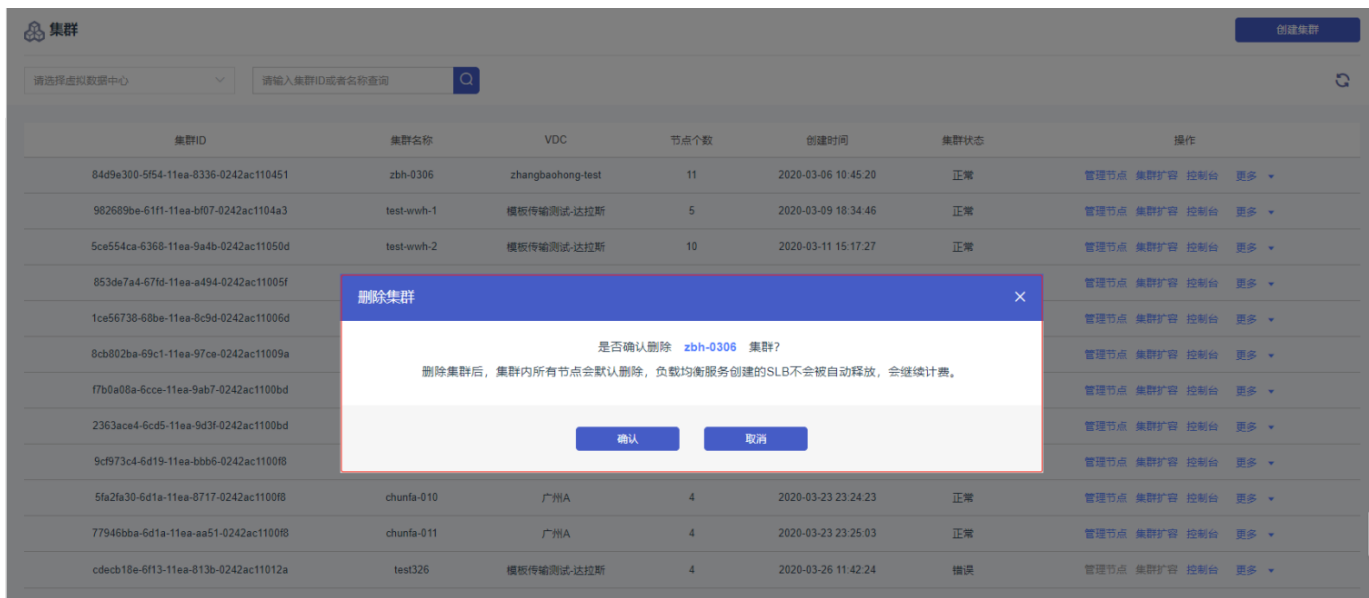
主机名称	节点资源请求量/使用量 (%)	配置	网络	计费方式	创建时间	节点类型	状态	k8s状态	操作
zbh-0306-master001	CPU: 22/2 内存: 17/33	8核8G 高性能型	10.240.89.46	按需计费	2020-03-06 10:45:20	master 不可调度	正常	Ready	2 控制台 移除 设置为可调度 数据盘
zbh-0306-master002	CPU: 18/2 内存: 13/29	8核8G 高性能型	10.240.89.48	按需计费	2020-03-06 10:45:20	master 不可调度	正常	Ready	控制台 移除 设置为可调度 数据盘
zbh-0306-master003	CPU: 16/2 内存: 12/34	8核8G 高性能型	10.240.89.49	按需计费	2020-03-06 10:45:20	master 不可调度	正常	Ready	控制台 移除 设置为可调度 数据盘
zbh-0306-master004	CPU: 20/2 内存: 20/50	8核8G 高性能型	10.240.89.54	按需计费	2020-03-06 11:31:38	master 不可调度	正常	Ready	控制台 移除 设置为可调度 数据盘

删除集群

- 进入集群页面 -> 选择要删除集群，点击更多 -> 点击删除



- 仔细阅读提示后，点击确认执行删除



节点管理

1.简介

首云支持

集群节点的手动管理，包括节点的增加、删除和设置节点是否可调度操作

2.使用须知

需通过首云集群管理页面进行上述操作

3.操作说明

- 添加master节点
 - 进入集群页面 -> 选择需要操作的集群 -> 点击集群扩容

集群

请选择目标数据中心 请输入集群ID或者名称查询

集群ID	集群名称	VDC	节点个数	创建时间	集群状态	操作
84d9e300-5f54-11ea-8336-0242ac110451	zbh-0306	zhangbaohong-test	8	2020-03-06 10:45:20	正常	管理节点 集群扩容 仪表盘 更多
982689be-61f1-11ea-bf07-0242ac1104a3	test-wvh-1	模板传输测试-达拉斯	5	2020-03-09 18:34:46	正常	管理节点 集群扩容 仪表盘 更多
5ce554ca-6368-11ea-9a4b-0242ac11050d	test-wvh-2	模板传输测试-达拉斯	10	2020-03-11 15:17:27	正常	管理节点 集群扩容 仪表盘 更多
853de7a4-67fd-11ea-a494-0242ac11005f	TEST	广州A	4	2020-03-17 11:15:15	正常	管理节点 集群扩容 仪表盘 更多
8f2e4aa4-67ff-11ea-bc83-0242ac11005f	chunfa-003	广州A	9	2020-03-17 11:29:50	正常	管理节点 集群扩容 仪表盘 更多
1ce56738-68be-11ea-8c9d-0242ac11006d	test318	模板传输测试-无槽	6	2020-03-18 10:13:53	正常	管理节点 集群扩容 仪表盘 更多
077dfb4a-68d6-11ea-aa28-0242ac11006d	test3181	模板传输测试-无槽	4	2020-03-18 13:05:05	正常	管理节点 集群扩容 仪表盘 更多
8cb802ba-69c1-11ea-97ca-0242ac11009a	zbh-0319	zhangbaohong-test	4	2020-03-19 17:11:00	正常	管理节点 集群扩容 仪表盘 更多

- 核对集群ID -> 选择增加节点类型为master -> 选择计算类型与规格 -> 添加云盘（可不选） -> 设置添加数量 -> 输入登录用户密码 -> 核对无误后，点击确定进行添加

集群 1 zbh-0306

节点类型 2 Worker Master

计费方式 按需计费

计算类型与规格 3 8核8G 8核16G 8核32G 8核64G 16核16G 16核32G 16核64G 16核128G

云硬盘类型	容量 (GB)	最高IOPS	操作
性能型	60	600	

本地盘 添加云硬盘 4

购买数量 5 2 台 master节点必须保证单数台，已有3台必须偶数台扩容。

密码(ccp用户) 6 为保证容器集群的稳定及安全，您可以通过ccp用户操作集群。
 请输入密码 8-30个字符，且同时包含三项（大写字母、小写字母、数字、特殊符号）
 请再次输入密码

当前配置信息
 计算类型与规格
 云服务器规格: 8核8G
 ￥22.67/每天
 本地盘:
 计费方式: 按需计费
 购买数量: 2

当前配置总价 ￥45.34/每天
 确认

- 进入节点查看页面 -> 选择对应集群，新添加节点状态由创建中 -> 正常代表添加成功

主机名称	节点资源请求量/使用量	配置	网络	计费方式	创建时间	节点类型	状态	操作
zbh-0306-master001	["cpu": 7, "memory": 32]	8核8G 高性能型	10.240.89.46	按需计费	2020-03-06 10:45:20	master	正常	控制台 移除 设置不可调度 数据盘
zbh-0306-master002	["cpu": 3, "memory": 29]	8核8G 高性能型	10.240.89.48	按需计费	2020-03-06 10:45:20	master	正常	控制台 移除 设置不可调度 数据盘
zbh-0306-master003	["cpu": 4, "memory": 33]	8核8G 高性能型	10.240.89.49	按需计费	2020-03-06 10:45:20	master	正常	控制台 移除 设置不可调度 数据盘
zbh-0306-master004	["cpu": 3, "memory": 46]	8核8G 高性能型	10.240.89.54	按需计费	2020-03-06 11:31:38	master	正常	控制台 移除 设置不可调度 数据盘
zbh-0306-master005	["cpu": 3, "memory": 46]	8核8G 高性能型	10.240.89.55	按需计费	2020-03-06 11:31:38	master	正常	控制台 移除 设置不可调度 数据盘
zbh-0306-master006	["cpu": 84, "memory": 23]	8核8G 高性能型	10.240.89.56	按需计费	2020-03-22 10:54:39	master	正常	控制台 移除 设置不可调度 数据盘
zbh-0306-master007	["cpu": 104, "memory": 23]	8核8G 高性能型	10.240.89.57	按需计费	2020-03-22 10:54:39	master	正常	控制台 移除 设置不可调度 数据盘
zbh-0306-worker001	["cpu": 6, "memory": 53]	8核8G 高性能型	10.240.89.50	按需计费	2020-03-06 10:45:20	worker	正常	控制台 移除 设置不可调度 数据盘
zbh-0306-worker002	["cpu": 1, "memory": 35]	8核8G 高性能型	10.240.89.52	按需计费	2020-03-06 11:24:03	worker	正常	控制台 移除 设置不可调度 数据盘

• 添加worker节点

- 进入集群页面 -> 选择需要操作的集群 -> 点击集群扩容

集群ID	集群名称	VDC	节点个数	创建时间	集群状态	操作
84d9e300-5f54-11ea-8336-0242ac110451	zbh-0306	zhangbaohong-test	8	2020-03-06 10:45:20	正常	管理节点 集群扩容 仪表盘 更多
982689be-51f1-11ea-bf07-0242ac1104a3	test-vwh-1	模板传输测试-达拉斯	5	2020-03-09 18:34:46	正常	管理节点 集群扩容 仪表盘 更多
5ce554ca-6368-11ea-9a4b-0242ac11050d	test-vwh-2	模板传输测试-达拉斯	10	2020-03-11 15:17:27	正常	管理节点 集群扩容 仪表盘 更多
853de7a4-67fd-11ea-a494-0242ac11005f	TEST	广州IA	4	2020-03-17 11:15:15	正常	管理节点 集群扩容 仪表盘 更多
8f2e4aa4-67ff-11ea-bc83-0242ac11005f	chunfa-003	广州IA	9	2020-03-17 11:29:50	正常	管理节点 集群扩容 仪表盘 更多
1ce56738-68be-11ea-8c9d-0242ac11006d	test318	模板传输测试-无锡	6	2020-03-18 10:13:53	正常	管理节点 集群扩容 仪表盘 更多
077dfb4a-68d6-11ea-a28-0242ac11006d	test3181	模板传输测试-无锡	4	2020-03-18 13:05:05	正常	管理节点 集群扩容 仪表盘 更多
8cb802ba-69c1-11ea-97ce-0242ac11009a	zbh-0319	zhangbaohong-test	4	2020-03-19 17:11:00	正常	管理节点 集群扩容 仪表盘 更多

- 核对集群ID -> 选择增加节点类型为worker -> 选择计算类型与规格 -> 添加云盘（可不选） -> 设置添加数量 -> 输入登录用户密码 -> 核对无误后，点击确定进行添加

集群 1

节点类型 2

计费方式

计算类型与规格 3

云硬盘类型	容量 (GB)	最高IOPS	操作
性能型	60	600	

本地盘 4 Worker还可以添加14块硬盘

购买数量 5 台

密码(ccp用户) 6 为保证容器集群的稳定及安全, 您可以通过ccp用户操作集群。
 8-30 个字符, 且同时包含三项 (大写字母、小写字母、数字、特殊符号)

当前配置信息

计算类型与规格
云服务器规格: 8核8G
¥22.67/每天

本地盘:
计费方式: 按需计费
购买数量: 1

当前配置总价 ¥22.67/每天
7

。进入节点查看页面 -> 选择对应集群, 新添加节点状态由创建中 -> 正常代表添加成功

/ 节点列表

主机名称	节点资源请求量/使用量	配置	网络	计费方式	创建时间	节点类型	状态	操作
zbh-0306-master001	["cpu": 9, "memory": 32]	8核8G 高性能型	10.240.89.46	按需计费	2020-03-06 10:45:20	master	正常	控制台 移除 设置不可调度 数据盘
zbh-0306-master002	["cpu": 7, "memory": 29]	8核8G 高性能型	10.240.89.48	按需计费	2020-03-06 10:45:20	master	正常	控制台 移除 设置不可调度 数据盘
zbh-0306-master003	["cpu": 8, "memory": 32]	8核8G 高性能型	10.240.89.49	按需计费	2020-03-06 10:45:20	master	正常	控制台 移除 设置不可调度 数据盘
zbh-0306-master004	["cpu": 7, "memory": 46]	8核8G 高性能型	10.240.89.54	按需计费	2020-03-06 11:31:38	master	正常	控制台 移除 设置不可调度 数据盘
zbh-0306-master005	["cpu": 8, "memory": 46]	8核8G 高性能型	10.240.89.55	按需计费	2020-03-06 11:31:38	master	正常	控制台 移除 设置不可调度 数据盘
zbh-0306-worker001	["cpu": 10, "memory": 51]	8核8G 高性能型	10.240.89.50	按需计费	2020-03-06 10:45:20	worker	正常	控制台 移除 设置不可调度 数据盘
zbh-0306-worker002	["cpu": 6, "memory": 35]	8核8G 高性能型	10.240.89.52	按需计费	2020-03-06 11:24:03	worker	正常	控制台 移除 设置不可调度 数据盘
zbh-0306-worker003	["cpu": 6, "memory": 34]	8核8G 高性能型	10.240.89.53	按需计费	2020-03-06 11:24:03	worker	正常	控制台 移除 设置不可调度 数据盘
3 zbh-0306-worker004	["cpu": 98, "memory": 13]	8核8G 高性能型	10.240.89.42	按需计费	2020-03-22 10:37:57	worker	正常	控制台 移除 设置不可调度 数据盘

• 删除worker节点 (master节点不可移除)

。进入节点查看页面 -> 选择对应集群, 选择要删除的worker节点 -> 点击删除

主机名称	节点资源请求量/使用量	配置	网络	计费方式	创建时间	节点类型	状态	操作
zbh-0306-master001	{"cpu": 3, "memory": 32}	8核 8G 高性能型	10.240.89.46	按需计费	2020-03-06 10:45:20	master	正常	控制台 移除 设置不可调度 数据盘
zbh-0306-master002	{"cpu": 2, "memory": 29}	8核 8G 高性能型	10.240.89.48	按需计费	2020-03-06 10:45:20	master	正常	控制台 移除 设置不可调度 数据盘
zbh-0306-master003	{"cpu": 3, "memory": 33}	8核 8G 高性能型	10.240.89.49	按需计费	2020-03-06 10:45:20	master	正常	控制台 移除 设置不可调度 数据盘
zbh-0306-master004	{"cpu": 3, "memory": 46}	8核 8G 高性能型	10.240.89.54	按需计费	2020-03-06 11:31:38	master	正常	控制台 移除 设置不可调度 数据盘
zbh-0306-master005	{"cpu": 2, "memory": 47}	8核 8G 高性能型	10.240.89.55	按需计费	2020-03-06 11:31:38	master	正常	控制台 移除 设置不可调度 数据盘
zbh-0306-master006	{"cpu": 2, "memory": 22}	8核 8G 高性能型	10.240.89.56	按需计费	2020-03-22 10:54:39	master	正常	控制台 移除 设置不可调度 数据盘
zbh-0306-master007	{"cpu": 2, "memory": 22}	8核 8G 高性能型	10.240.89.57	按需计费	2020-03-22 10:54:39	master	正常	控制台 移除 设置不可调度 数据盘
zbh-0306-worker001	{"cpu": 5, "memory": 53}	8核 8G 高性能型	10.240.89.50	按需计费	2020-03-06 10:45:20	worker	正常	控制台 3 移除 设置不可调度 数据盘
zbh-0306-worker002	{"cpu": 0, "memory": 35}	8核 8G 高性能型	10.240.89.52	按需计费	2020-03-06 11:24:03	worker	正常	控制台 移除 设置不可调度 数据盘

。点击删除后，仔细阅读提示后，没问题点击确认执行删除操作

主机名称	节点资源请求量/使用量	配置	网络	计费方式	创建时间	节点类型	状态	操作
zbh-0306-master001	{"cpu": 7, "memory": 32}	8核 8G 高性能型	10.240.89.46	按需计费	2020-03-06 10:45:20	master	正常	控制台 移除 设置不可调度 数据盘
zbh-0306-master002	{"cpu": 3, "memory": 29}	8核 8G 高性能型	10.240.89.48	按需计费	2020-03-06 10:45:20	master	正常	控制台 移除 设置不可调度 数据盘
zbh-0306-master003	{"cpu": 4, "memory": 33}	8核 8G 高性能型	10.240.89.49	按需计费	2020-03-06 10:45:20	master	正常	控制台 移除 设置不可调度 数据盘
zbh-0306-master004	{"cpu": 3, "memory": 46}	8核 8G 高性能型	10.240.89.54	按需计费	2020-03-06 11:31:38	master	正常	控制台 移除 设置不可调度 数据盘
zbh-0306-master005	{"cpu": 3, "memory": 46}	8核 8G 高性能型	10.240.89.55	按需计费	2020-03-06 11:31:38	master	正常	控制台 移除 设置不可调度 数据盘
zbh-0306-master006	{"cpu": 84, "memory": 23}	8核 8G 高性能型	10.240.89.56	按需计费	2020-03-22 10:54:39	master	正常	控制台 移除 设置不可调度 数据盘
zbh-0306-master007	{"cpu": 104, "memory": 23}	8核 8G 高性能型	10.240.89.57	按需计费	2020-03-22 10:54:39	master	正常	控制台 移除 设置不可调度 数据盘
zbh-0306-worker001	{"cpu": 6, "memory": 53}	8核 8G 高性能型	10.240.89.50	按需计费	2020-03-06 10:45:20	worker	正常	控制台 移除 设置不可调度 数据盘
zbh-0306-worker002	{"cpu": 1, "memory": 35}	8核 8G 高性能型	10.240.89.52	按需计费	2020-03-06 11:24:03	worker	正常	控制台 移除 设置不可调度 数据盘
zbh-0306-worker003	{"cpu": 1, "memory": 34}	8核 8G 高性能型	10.240.89.53	按需计费	2020-03-06 11:24:03	worker	正常	控制台 移除 设置不可调度 数据盘
zbh-0306-worker004	{"cpu": 1, "memory": 14}	8核 8G 高性能型	10.240.89.42	按需计费	2020-03-22 10:37:57	worker	正常	控制台 移除 设置不可调度 数据盘

。进入节点查看页面 -> 选择对应集群，删除节点状态由删除中 -> 节点消失代表删除成功



zbh-0306 / 节点列表

主机名称	节点资源请求量/使用量	配置	网络	计费方式	创建时间	节点类型	状态	操作
zbh-0306-master001	{"cpu": 3, "memory": 32}	8核8G 高性能型	10.240.89.46	按需计费	2020-03-06 10:45:20	master	正常	控制台 移除 设置为不可调度 数据盘
zbh-0306-master002	{"cpu": 2, "memory": 29}	8核8G 高性能型	10.240.89.48	按需计费	2020-03-06 10:45:20	master	正常	控制台 移除 设置为不可调度 数据盘
zbh-0306-master003	{"cpu": 3, "memory": 33}	8核8G 高性能型	10.240.89.49	按需计费	2020-03-06 10:45:20	master	正常	控制台 移除 设置为不可调度 数据盘
zbh-0306-master004	{"cpu": 3, "memory": 47}	8核8G 高性能型	10.240.89.54	按需计费	2020-03-06 11:31:38	master	正常	控制台 移除 设置为不可调度 数据盘
zbh-0306-master005	{"cpu": 2, "memory": 47}	8核8G 高性能型	10.240.89.55	按需计费	2020-03-06 11:31:38	master	正常	控制台 移除 设置为不可调度 数据盘
zbh-0306-master006	{"cpu": 2, "memory": 22}	8核8G 高性能型	10.240.89.56	按需计费	2020-03-22 10:54:39	master	正常	控制台 移除 设置为不可调度 数据盘
zbh-0306-master007	{"cpu": 2, "memory": 22}	8核8G 高性能型	10.240.89.57	按需计费	2020-03-22 10:54:39	master	正常	控制台 移除 设置为不可调度 数据盘
zbh-0306-worker001	{"cpu": 5, "memory": 52}	8核8G 高性能型	10.240.89.50	按需计费	2020-03-06 10:45:20	worker	正常	控制台 移除 设置为不可调度 数据盘
zbh-0306-worker002	{"cpu": 0, "memory": 35}	8核8G 高性能型	10.240.89.52	按需计费	2020-03-06 11:24:03	worker	正常	控制台 移除 设置为不可调度 数据盘
zbh-0306-worker003	{"cpu": 0, "memory": 34}	8核8G 高性能型	10.240.89.53	按需计费	2020-03-06 11:24:03	worker	正常	控制台 移除 设置为不可调度 数据盘
zbh-0306-worker004	{"cpu": 0, "memory": 14}	8核8G 高性能型	10.240.89.42	按需计费	2020-03-22 10:37:57	worker	删除中	控制台 移除 设置为不可调度 数据盘

• 设置节点调度

- 进入节点查看页面 -> 选择对应集群 -> 选择要设置的节点 -> 设置是否可调度



zbh-0306 / 节点列表

主机名称	节点资源请求量/使用量 (%)	配置	网络	计费方式	创建时间	节点类型	状态	k8s状态	操作
zbh-0306-master001	CPU: 22/2 内存: 17/33	8核8G 高性能型	10.240.89.46	按需计费	2020-03-06 10:45:20	master 不可调度	正常	Ready	控制台 移除 设置为可调度 数据盘
zbh-0306-master002	CPU: 18/2 内存: 13/29	8核8G 高性能型	10.240.89.48	按需计费	2020-03-06 10:45:20	master 不可调度	正常	Ready	控制台 移除 设置为可调度 数据盘
zbh-0306-master003	CPU: 16/2 内存: 12/34	8核8G 高性能型	10.240.89.49	按需计费	2020-03-06 10:45:20	master 不可调度	正常	Ready	控制台 移除 设置为可调度 数据盘
zbh-0306-master004	CPU: 20/2 内存: 20/50	8核8G 高性能型	10.240.89.54	按需计费	2020-03-06 11:31:38	master 不可调度	正常	Ready	控制台 移除 设置为可调度 数据盘
zbh-0306-master005	CPU: 20/2 内存: 20/51	8核8G 高性能型	10.240.89.55	按需计费	2020-03-06 11:31:38	master 不可调度	正常	Ready	控制台 移除 设置为可调度 数据盘
zbh-0306-master006	CPU: 19/1 内存: 19/25	8核8G 高性能型	10.240.89.56	按需计费	2020-03-22 10:54:39	master 不可调度	正常	Ready	控制台 移除 设置为可调度 数据盘
zbh-0306-master007	CPU: 19/1 内存: 19/25	8核8G 高性能型	10.240.89.57	按需计费	2020-03-22 10:54:39	master 不可调度	正常	Ready	控制台 移除 设置为可调度 数据盘
zbh-0306-worker002	CPU: 13/1 内存: 18/17	8核8G 高性能型	10.240.89.42	按需计费	2020-03-24 16:07:19	worker 不可调度	正常	Ready	控制台 移除 设置为可调度 数据盘
zbh-0306-worker003	CPU: 15/1 内存: 21/17	8核8G 高性能型	10.240.89.52	按需计费	2020-03-24 16:07:19	worker 可调度	正常	Ready	控制台 移除 3 设置为不可调度 数据盘
zbh-0306-worker004	CPU: 15/2 内存: 25/29	8核8G 高性能型	10.240.89.53	按需计费	2020-03-24 16:07:19	worker 可调度	正常	Ready	控制台 移除 设置为不可调度 数据盘

- 仔细阅读提示，没问题后点击确认

zbh-0306 / 节点列表 创建节点

主机名称	节点资源请求量/使用量	配置	网络	计费方式	创建时间	节点类型	状态	操作
zbh-0306-master001	{"cpu": 3, "memory": 32}	8核8G 高性能型	10.240.89.46	按需计费	2020-03-06 10:45:20	master	正常	控制台 移除 设置为不可调度 数据盘
zbh-0306-master002	{"cpu": 2, "memory": 29}	8核8G 高性能型	10.240.89.48	按需计费	2020-03-06 10:45:20	master	正常	控制台 移除 设置为不可调度 数据盘
zbh-0306-master003	{"cpu": 3, "memory": 33}	8核8G 高性能型	10.240.89.49	按需计费	2020-03-06 10:45:20	master	正常	控制台 移除 设置为不可调度 数据盘
zbh-0306-master004	{"cpu": 3, "memory": 46}							控制台 移除 设置为不可调度 数据盘
zbh-0306-master005	{"cpu": 2, "memory": 47}							控制台 移除 设置为不可调度 数据盘
zbh-0306-master006	{"cpu": 2, "memory": 22}							控制台 移除 设置为不可调度 数据盘
zbh-0306-master007	{"cpu": 2, "memory": 22}	8核8G 高性能型	10.240.89.57	按需计费	2020-03-22 10:54:39	master	正常	控制台 移除 设置为不可调度 数据盘
zbh-0306-worker001	{"cpu": 5, "memory": 53}	8核8G 高性能型	10.240.89.50	按需计费	2020-03-06 10:45:20	worker	正常	控制台 移除 设置为不可调度 数据盘

节点调度

确认将 **zbh-0306-master007** 主机设置为不可调度吗? 您在后续进行应用部署时, Pod 不会再调度到该节点。

确认
取消

- 进入节点查看页面 -> 选择对应集群 -> 查看设置是否成功

zbh-0306 / 节点列表 创建节点

主机名称	节点资源请求量/使用量 (%)	配置	网络	计费方式	创建时间	节点类型	状态	k8s状态	操作
zbh-0306-master001	CPU: 22/2 内存: 17/33	8核8G 高性能型	10.240.89.46	按需计费	2020-03-06 10:45:20	master 不可调度	正常	Ready	控制台 移除 设置为可调度 数据盘
zbh-0306-master002	CPU: 18/2 内存: 13/30	8核8G 高性能型	10.240.89.48	按需计费	2020-03-06 10:45:20	master 不可调度	正常	Ready	控制台 移除 设置为可调度 数据盘
zbh-0306-master003	CPU: 16/2 内存: 12/34	8核8G 高性能型	10.240.89.49	按需计费	2020-03-06 10:45:20	master 不可调度	正常	Ready	控制台 移除 设置为可调度 数据盘
zbh-0306-master004	CPU: 20/2 内存: 20/50	8核8G 高性能型	10.240.89.54	按需计费	2020-03-06 11:31:38	master 不可调度	正常	Ready	控制台 移除 设置为可调度 数据盘
zbh-0306-master005	CPU: 20/2 内存: 20/51	8核8G 高性能型	10.240.89.55	按需计费	2020-03-06 11:31:38	master 不可调度	正常	Ready	控制台 移除 设置为可调度 数据盘
zbh-0306-master006	CPU: 19/1 内存: 19/25	8核8G 高性能型	10.240.89.56	按需计费	2020-03-22 10:54:39	master 不可调度	正常	Ready	控制台 移除 设置为可调度 数据盘
zbh-0306-master007	CPU: 19/1 内存: 19/25	8核8G 高性能型	10.240.89.57	按需计费	2020-03-22 10:54:39	master 不可调度	正常	Ready	控制台 移除 设置为可调度 数据盘
zbh-0306-worker002	CPU: 13/1 内存: 18/17	8核8G 高性能型	10.240.89.42	按需计费	2020-03-24 16:07:19	worker 可调度	正常	Ready	控制台 移除 设置为不可调度 数据盘
zbh-0306-worker003	CPU: 15/1 内存: 21/17	8核8G 高性能型	10.240.89.52	按需计费	2020-03-24 16:07:19	worker 不可调度	正常	Ready	控制台 移除 设置为可调度 数据盘
zbh-0306-worker004	CPU: 15/2 内存: 25/29	8核8G 高性能型	10.240.89.53	按需计费	2020-03-24 16:07:19	worker 可调度	正常	Ready	控制台 移除 设置为不可调度 数据盘

存储管理

1. 简介

首云提供了NAS文件存储服务, 并且针对Kubernetes集群提供了自研的存储驱动的支持

2. 操作说明

- 创建文件存储 NAS

- 选择文件存储NAS后, 选择右上的新建按钮可以创建一块新的NAS存储

- 在新页面中，依次选择可用区，输入名称，选择类型和容量，最后单击确定
- 在该页面，可以看到所配置的存储的价格
- 可以用去的选择要和容器集群所在区域一致，不同区域的集群和存储间不能挂载和访问

创建NAS存储

可用区 1

名称 2

注: 长度为1-30个字符, 只能包含数字、字母、和“-”, 且首尾只能是字母或数字!

类型 3

容量 4 500 4000G 注: 扩容时仅支持扩容到2000G

IOPS 3000

当前配置信息

可用区: 中国大陆-广州-可用区A

名称:

性能型: 500G

5 当前配置总价: ¥ 5.83/天

• 创建挂载点

- 新创建的NAS盘，需要挂载后才能使用
- 在挂载窗口中，请选择需要挂载的集群，此处的集群仅显示和该NAS所在区域相同的容器集群
- 挂载成功后，会显示私网IP
- 挂载过程可能用时较长，3-5分钟

NAS存储ID	名称	区域	集群	类型	最高IOPS	容量	使用量	状态	创建时间	私网IP	操作
d8fe337e-7a0b-11ea-bc3a-0242ac110257	test1002	中国大陆-广州-可用区A	未挂载	性能型	3000	500G	--	正常	2020-04-09 10:43:09	--	挂载 存储卷管理 扩容 删除
d1135286-7991-11ea-b87a-0242ac110256	test-1	中国大陆-广州-可用区A	zbh-testtt	性能型	3000	500G	0.01%	已挂载	2020-04-08 20:09:37	10.240.73.64	卸载 存储卷管理 扩容 删除
873732ca-794d-11ea-becf-0242ac11023c	404nas1	中国大陆-广州-可用区A	chunfa-101	性能型	3000	500G	0.01%	已挂载	2020-04-01 19:09:16	44.150	卸载 存储卷管理 扩容 删除
80ef2d5c-794b-11ea-a485-0242ac11023c	404nas									73.186	卸载 存储卷管理 扩容 删除
37e27a30-78db-11ea-8a20-82b0d54620aa	zdns									105.34	卸载 存储卷管理 扩容 删除
340d569c-7899-11ea-a06c-82b0d54620aa	zbh-rancher									73.101	卸载 存储卷管理 扩容 删除
3a09fe7c-7409-11ea-910e-d2a8eefc26d	chunfa-101-nas01									10.240.73.254	卸载 存储卷管理 扩容 删除
64755ee8-7406-11ea-b973-0242ac110207	test-nfs-3									亚太地区-新加坡-可用区A	add-to-rancher-no-deletion

1、新创建的NAS盘，状态为正常，私网IP为空

2、新创建的盘需要挂载后才能使用。
点击挂载后，在弹出的对话框中，选择要使用的集群，只有与NAS盘同区域的集群才能被挂载。

3、当挂载完成后，状态变为已挂载，并且出现私网IP，此时NAS盘可以被集群正常使用（此过程可能需要耗时几分钟）

• 挂载NAS盘到本地机器

- 对于已经挂载好的NAS盘，该集群内的任何一台计算资源（master和worker）均可访问
- ssh登录任意一台集群
- 创建一个新的目录：`mkdir ~/nas`
- 挂载NAS盘到本地：

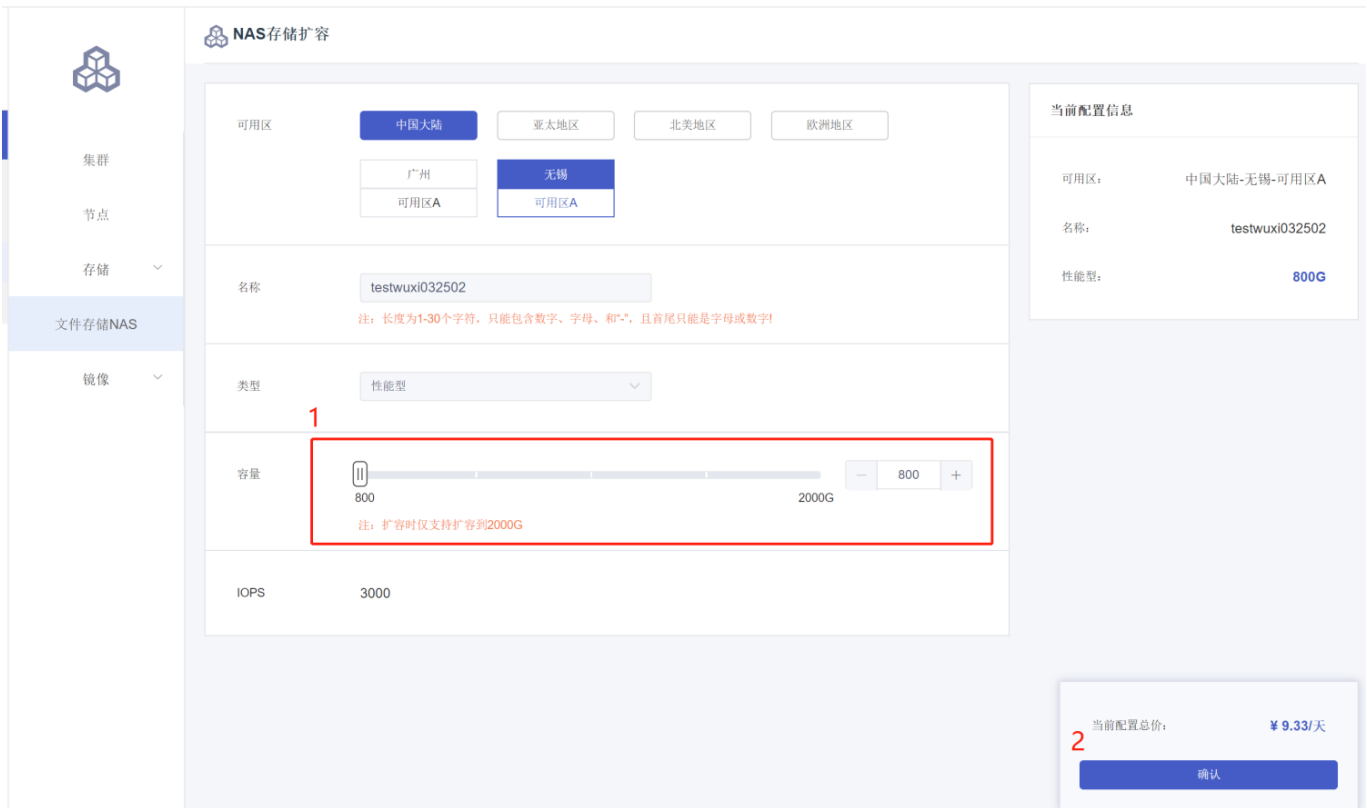
```
sudo mount -v -t nfs -o "vers=4,noresvport" <NAS盘挂载点的私网
```

```
IP>:/nfsshare ~/nas
```

- 挂载后，/nas 目录即为NAS盘的所有内容，此时可以做任意操作
- 使用后，执行下述命令即可卸载本次挂载：`sudo umount /nas`

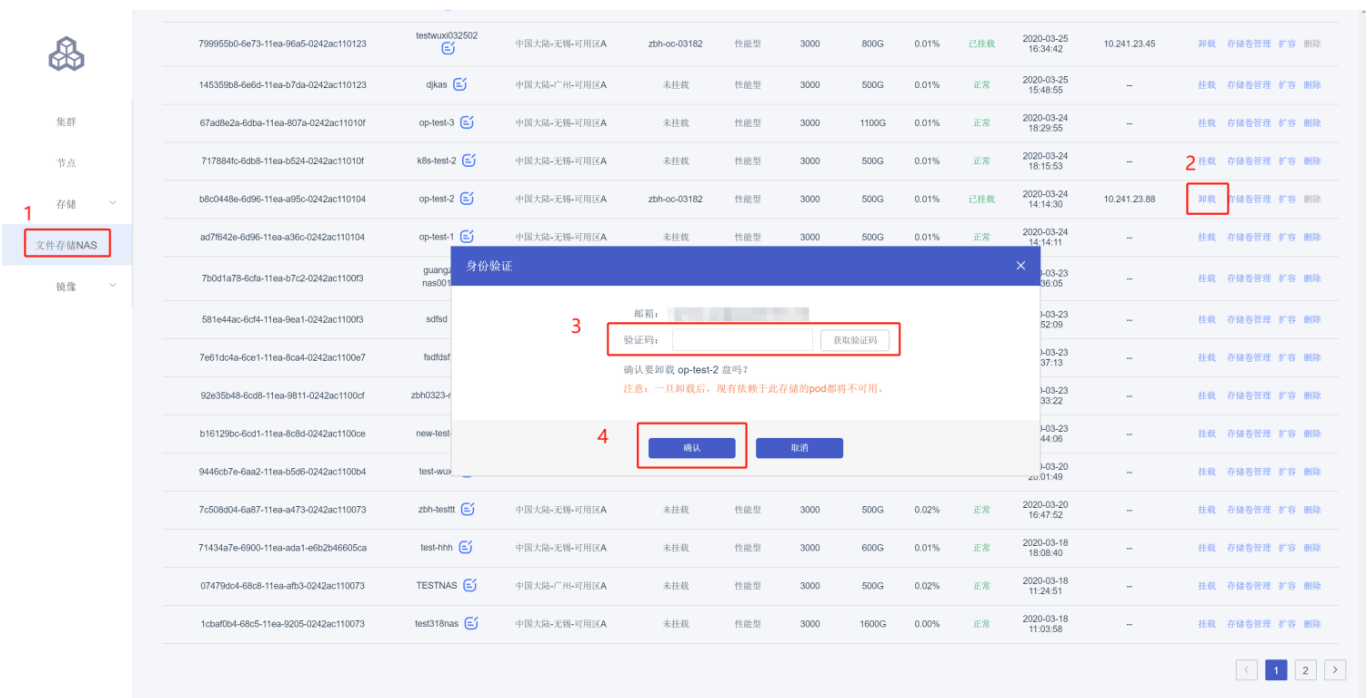
• NAS盘扩容

- 在文件存储NAS页面点击扩容，即可对该盘进行在线扩容
- 在新页面中，选择容量并确定即可完成扩容
- 目前只支持扩容操作，不可缩容



• 卸载NAS盘

- 对于已挂载的NAS盘，可以从集群中卸载，卸载后的NAS盘可以挂在到同一区域下的其他集群使用
- NAS盘一旦卸载，会导致该集群内所有依赖该存储的POD的PV皆不可用，请谨慎操作
- 卸载需要邮箱验证



• 删除NAS盘

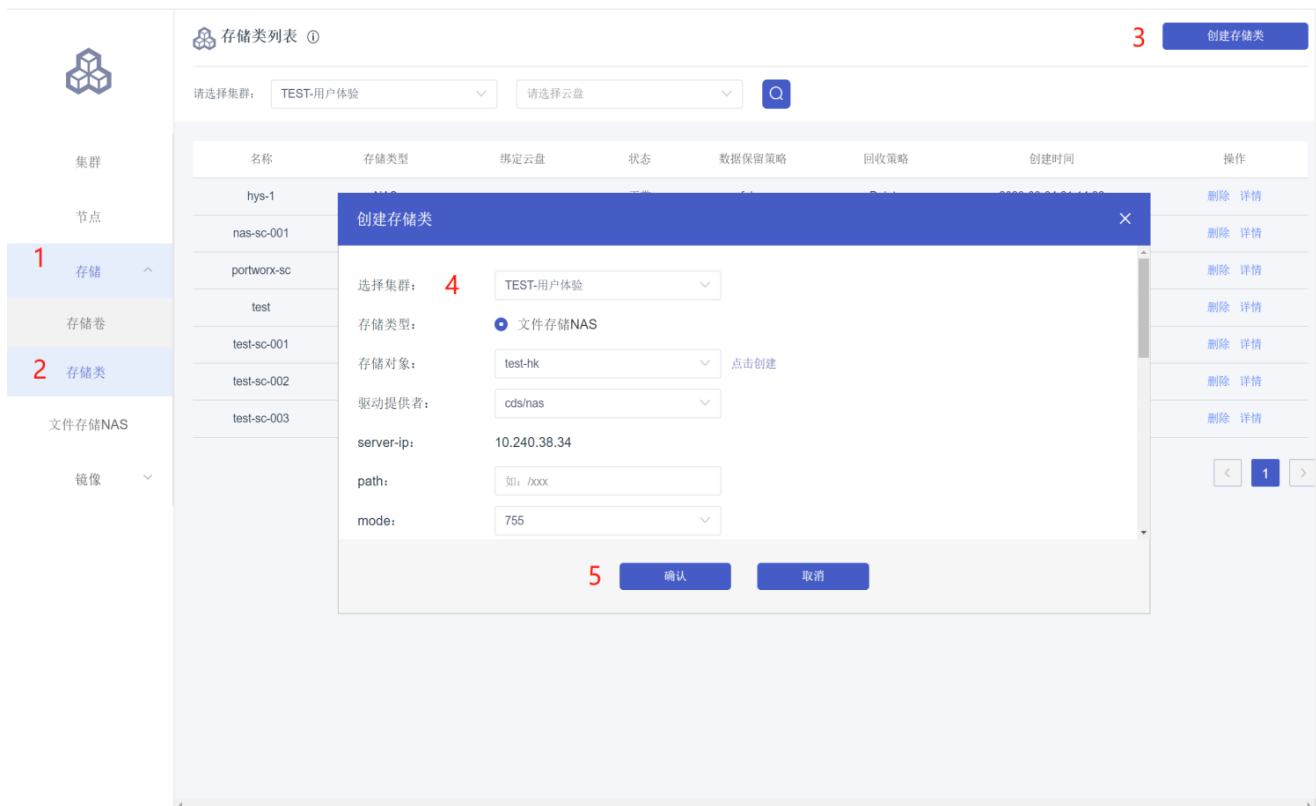
- 对于已经卸载的NAS盘，可以永久删除
- 删除后的NAS盘，所有数据均会被删除，且无法找回，请谨慎操作
- 删除需要手机或邮箱验证

文件存储NAS

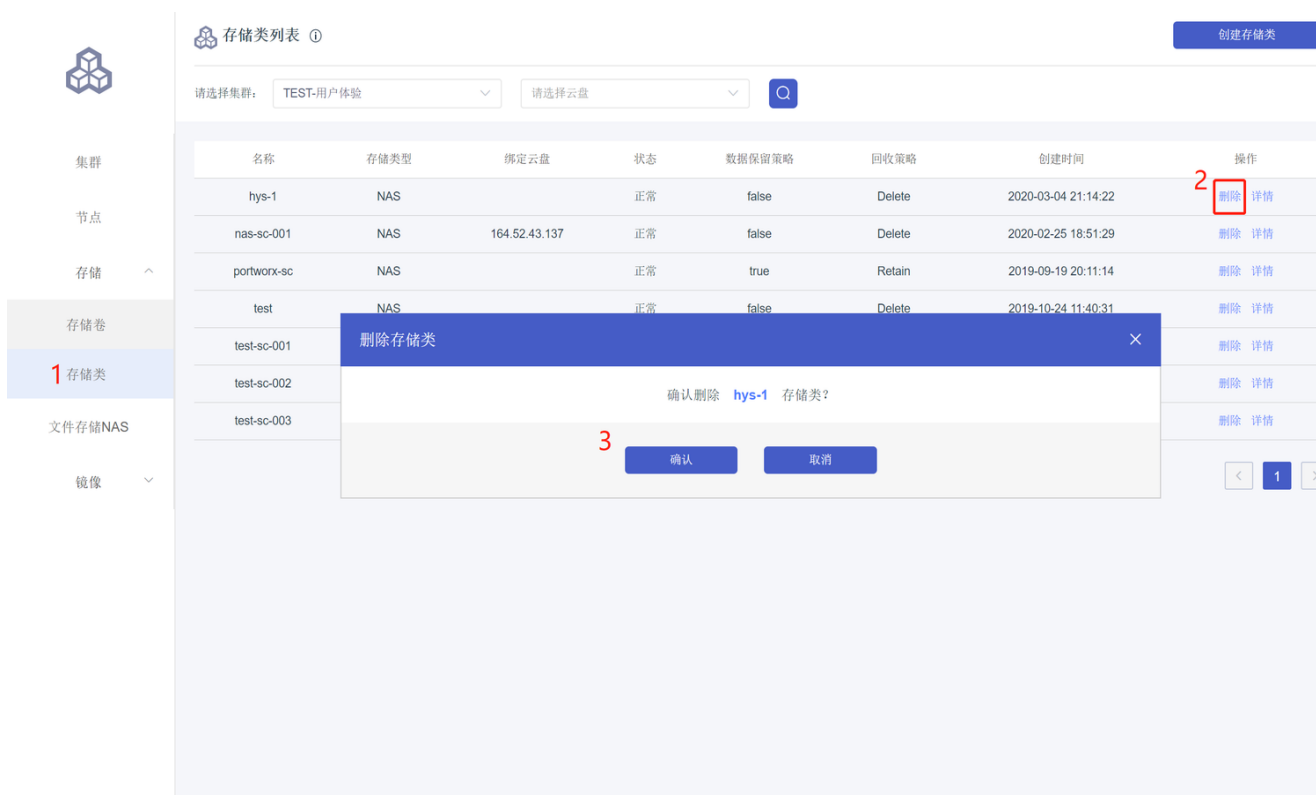
NAS存储ID	名称	区域	集群	类型	最高IOPS	容量	使用量	状态	创建时间	私有IP	操作
0f3238de-6f3d-11ea-a4c2-0242ac110129	test3261	中国大陆-广州-可用区A	未挂载	性能型	3000	2000G	0.00%	正常	2020-03-26 16:37:42	--	挂载 存储卷管理 扩容 删除
b9579d64-6f3c-11ea-8e8a-0242ac110129	nas2002	中国大陆-广州-可用区A	未挂载	性能型	3000	500G	0.01%	正常	2020-03-26 16:35:18	--	挂载 存储卷管理 扩容 删除
22ba581d-6f32-11ea-bee8-0242ac110129	test326	中国大陆-广州-可用区A	未挂载	性能型	3000	900G	0.01%	正常	2020-03-26 15:19:30	--	挂载 存储卷管理 扩容 删除
8175fa72-6e73-11ea-984a-0242ac110123	testwuxi	中国大陆-无锡-可用区A	未挂载	性能型	3000	500G	0.01%	正常	2020-03-25 14:14:30	10.241.23.168	卸载 存储卷管理 扩容 删除
79995b0c-6e73-11ea-96a5-0242ac110123	testwuxi	中国大陆-无锡-可用区A	未挂载	性能型	3000	500G	0.01%	正常	2020-03-25 14:14:30	10.241.23.45	卸载 存储卷管理 扩容 删除
145359b6-6e6d-11ea-b7da-0242ac110123	djkas	中国大陆-无锡-可用区A	未挂载	性能型	3000	500G	0.01%	正常	2020-03-25 14:14:30	--	挂载 存储卷管理 扩容 删除
67ad9e2a-6d8a-11ea-807a-0242ac11010f	op-test-	中国大陆-无锡-可用区A	未挂载	性能型	3000	500G	0.01%	正常	2020-03-24 29:55	--	挂载 存储卷管理 扩容 删除
717884fc-6db8-11ea-b5a4-0242ac11010f	k8s-test-	中国大陆-无锡-可用区A	未挂载	性能型	3000	500G	0.01%	正常	2020-03-24 15:53	--	挂载 存储卷管理 扩容 删除
b8c0448e-6d96-11ea-a95c-0242ac110104	op-test-2	中国大陆-无锡-可用区A	zhh-cc-03182	性能型	3000	500G	0.01%	已挂载	2020-03-24 14:14:30	10.241.23.88	卸载 存储卷管理 扩容 删除
ad7f642e-6d96-11ea-a36c-0242ac110104	op-test-1	中国大陆-无锡-可用区A	未挂载	性能型	3000	500G	0.01%	正常	2020-03-24 14:14:11	--	挂载 存储卷管理 扩容 删除
7b0d1a78-6cfa-11ea-b7c2-0242ac110003	guangzhou-nas001	中国大陆-广州-可用区A	未挂载	性能型	3000	500G	0.02%	正常	2020-03-23 19:36:05	--	挂载 存储卷管理 扩容 删除
581e44ac-6cd4-11ea-9ea1-0242ac110003	sdfs	中国大陆-无锡-可用区A	未挂载	性能型	3000	500G	0.01%	正常	2020-03-23 18:52:09	--	挂载 存储卷管理 扩容 删除
7e61dc4a-6ce1-11ea-8ca4-0242ac1100e7	sdfsdf	中国大陆-无锡-可用区A	未挂载	性能型	3000	600G	0.01%	正常	2020-03-23 16:37:13	--	挂载 存储卷管理 扩容 删除
92e35b48-6c08-11ea-9811-0242ac1100cf	zhh0323-nas	中国大陆-无锡-可用区A	未挂载	性能型	3000	500G	0.02%	正常	2020-03-23 15:33:22	--	挂载 存储卷管理 扩容 删除

• 创建存储类

- 点击存储->存储类->新建存储类，可以在弹出的对话框中新建一个使用NAS盘的存储类
- 创建存储类的参数如下配置：
 - 选择集群：选择配置存储类到哪个容器集群
 - 存储类型：选择所支持的存储类型，目前仅支持NAS文件存储
 - 存储对象：已创建并且挂在到该集群的NAS盘
 - 驱动提供者：文件NAS存储选择 `cds/nas`
 - server-ip: 系统会自动读取NAS盘的挂载点IP
 - path: NAS盘对应的远端挂在目录，默认为 `/nfsshare`
 - mode: pv使用文件夹的mode，一般选择 `755` 或 `777`
 - 存储类名称：配置该存储类的名称，当创建pvc时，需要引用该名称
 - 存储卷回收策略：Retain, PVC被删除后，保留对应的PV；Delete, PVC被删除后，删除对应的PV
 - 数据保留策略：当PV被删除后，如果数据保留策略为 `true`，则归档相关数据，归档名称为 `archived-原pv使用目录名称-时间戳`；如果为 `false`，则直接删除该pv所使用的数据



删除存储类



查看存储类

- 在存储类列表中，点击详情，可以查看存储类相应信息
- 可以在基本信息和yaml文件中切换不同的显示方式

存储类详情

返回存储类列表

基本信息 查看Yaml

当前存储类: hys-1

存储类型	NAS
存储对象	
驱动提供者	hys-1
server-ip	
path	
mode	
存储类名称	hys-1
存储卷回收策略	Delete
数据保留策略	false
创建时间	2020-03-04 21:14:22

- 查看存储卷列表

- 点击存储->存储卷，然后从集群选择下拉列表中，可以查看所选的集群中所有的存储卷（PV）

存储卷列表 ①

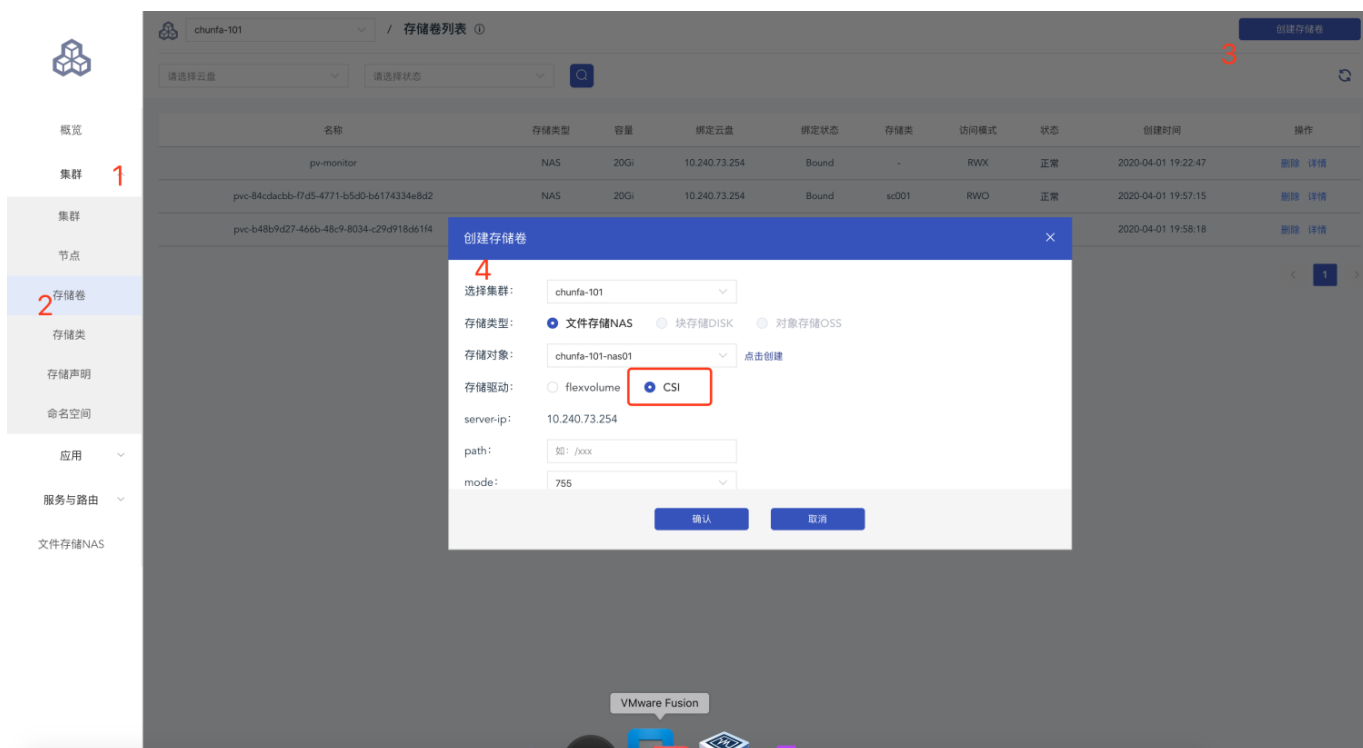
创建存储卷

请选择集群: TEST-用户体验 请选择云盘 请选择状态

名称	存储类型	容量	绑定云盘	绑定状态	存储类	访问模式	状态	创建时间	操作
flex-nas-pv-001	NAS	2Gi	164.52.43.137	Bound	-	RWX	正常	2020-02-26 11:33:37	删除 详情
flex-nas-pv-002	NAS	2Gi	164.52.43.137	Bound	-	RWX	正常	2020-02-26 11:33:41	删除 详情
iamapv	NAS	1Gi	10.240.38.9	Available	-	RWX	正常	2019-12-05 18:59:34	删除 详情
pv-demo-001	NAS	2Gi	164.52.43.137	Bound	-	RWX	正常	2020-03-18 17:15:12	删除 详情
pvc-047ac277-e81e-11e9-b558-005056bf732c	NAS	1Gi	10.240.38.9	Bound	portworx-sc	RWX	正常	2019-10-06 17:45:23	删除 详情
pvc-0543ccb4-5849-11ea-9429-005056bf0969	NAS	2Gi	164.52.43.137	Bound	nas-sc-001	RWX	正常	2020-02-26 11:35:23	删除 详情
pvc-074ffd5c-5849-11ea-9429-005056bf0969	NAS	2Gi	164.52.43.137	Bound	nas-sc-001	RWX	正常	2020-02-26 11:35:26	删除 详情
pvc-3f167f37-5301-11ea-89c2-005056bf732c	NAS	1Gi	10.240.38.9	Bound	test	RWX	正常	2020-02-19 18:19:01	删除 详情
pvc-5b0dca1a-e283-11e9-b558-005056bf732c	NAS	1Gi	10.240.38.9	Bound	portworx-sc	RWX	正常	2019-09-29 14:35:41	删除 详情
pvc-6816fd7-e298-11e9-b558-005056bf732c	NAS	1Gi	10.240.38.9	Bound	portworx-sc	RWX	正常	2019-09-29 17:06:22	删除 详情

- 创建存储卷

- 点击集群->存储卷->新建存储卷，可以在弹出的对话框中新建一个使用NAS盘的存储卷
- 创建存储类的参数如下配置：
 - 选择集群：选择配置存储类到哪个容器集群
 - 存储类型：选择所支持的存储类型，目前仅支持NAS文件存储
 - 存储对象：已创建并且挂在到该集群的NAS盘
 - 存储驱动：请选择CSI（Flexvolume今后会停止支持）
 - server-ip: 系统会自动读取NAS盘的挂载点IP
 - path: NAS盘对应的远端挂在目录，默认为 /nfsshare
 - mode: pv使用文件夹的mode，一般选择 755 或 777
 - 存储卷名称：配置该存储类的名称，当创建pvc时，需要引用该名称
 - 容量：所要创建存储卷的大小，单位GB
 - 访问模式：ReadWriteMany：可以多个节点同时读写，ReadWriteOnce：仅单个节点读写
 - 标签：可以添加自定义的标签



- 查看存储卷详情
 - 点击单个存储卷的详情，即可查看该存储卷的详细信息
 - 可以在基本信息和yaml文件中切换不同的显示方式

存储卷详情 返回存储卷列表

基本信息 查看Yaml

当前存储卷: **flex-nas-pv-001**

存储类型	NAS
存储对象	
存储驱动	ods/nas
server-ip	164.52.43.137
path	/nfsshare/openshift_static_1
mode	755
存储卷名称	flex-nas-pv-001
容量	2Gi
访问模式	RWX
标签	{ "name": "pv001" }
创建时间	2020-02-26 11:33:37
状态	Bound

• 删除存储卷

存储卷列表 创建存储卷

请选择集群: TEST-用户体验 | 请选择云盘 | 请选择状态 🔍

名称	存储类型	容量	绑定云盘	绑定状态	存储类	访问模式	状态	创建时间	操作
flex-nas-pv-001	NAS	2Gi	164.52.43.137	Bound	-	RWX	正常	2020-02-26 11:33:37	删除 详情
flex-nas-pv-002	NAS	2Gi	164.52.43.137	Bound	-	RWX	正常	2020-02-26 11:33:41	删除 详情
iamapv	NAS	1Gi	10.240.38.9	Available	-	RWX	正常	2019-12-05 18:59:34	删除 详情
pv-demo-001								18:17:15:12	删除 详情
pvc-047ac277-e81e-11e9-4								06:17:45:23	删除 详情
pvc-0543ccb4-5849-11ea-6								26:11:35:23	删除 详情
pvc-074ffd5c-5849-11ea-9								26:11:35:26	删除 详情
pvc-3f167f37-5301-11ea-8								19:18:19:01	删除 详情
pvc-5b0dca1a-e283-11e9-4								29:14:35:41	删除 详情
pvc-6816fdd7-e298-11e9-4								29:17:06:22	删除 详情

删除存储卷

确认删除 **pv-demo-001** 存储卷?

PV没有绑定任何StorageClass, PV对应的NAS云盘数据需要手动处理

删除存储卷后, 数据默认未清除, 可根据NAS盘的ip, 将NAS云盘的/nfsshare目录挂载到k8s集群中任意目录, 然后开始管理数据。

确认
取消

< 1 2 >

应用管理

1. 简介

首云支持创建无状态（Deployment）和有状态（StatefulSet）应用，并提供相应的容器组管理。

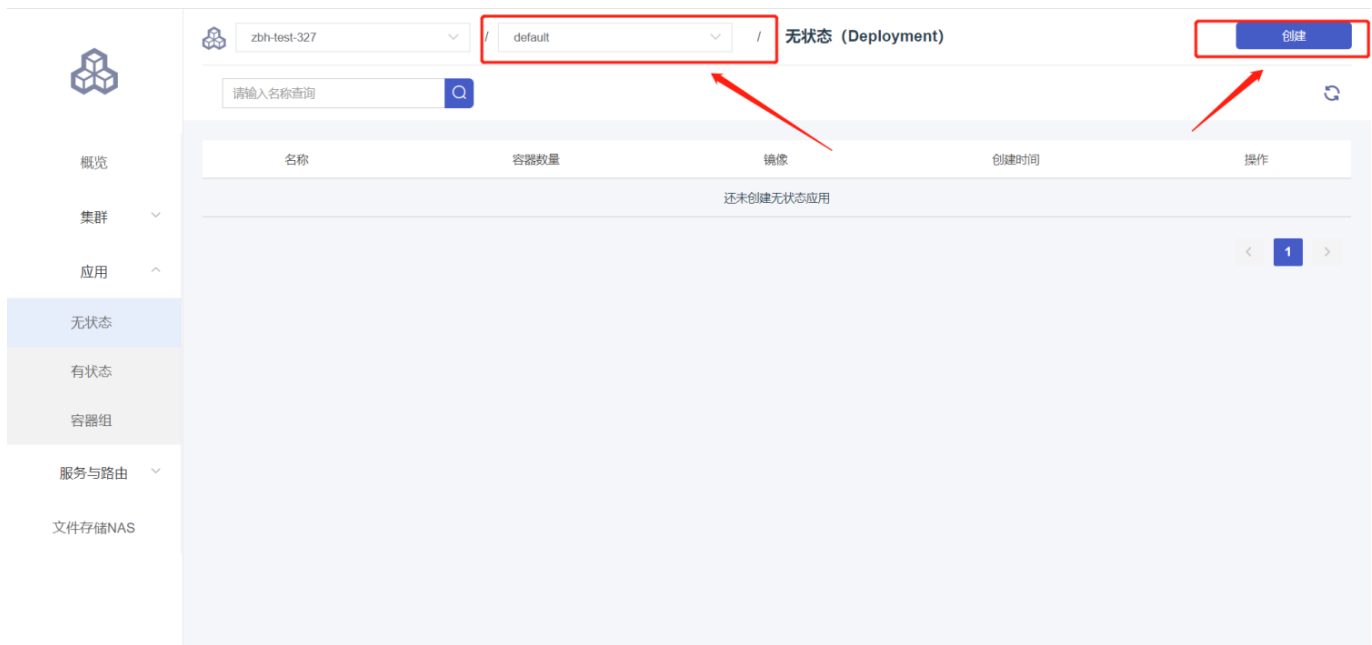
2. 前提条件

创建一个 Kubernetes 集群。详情请参见集群管理->创建集群。

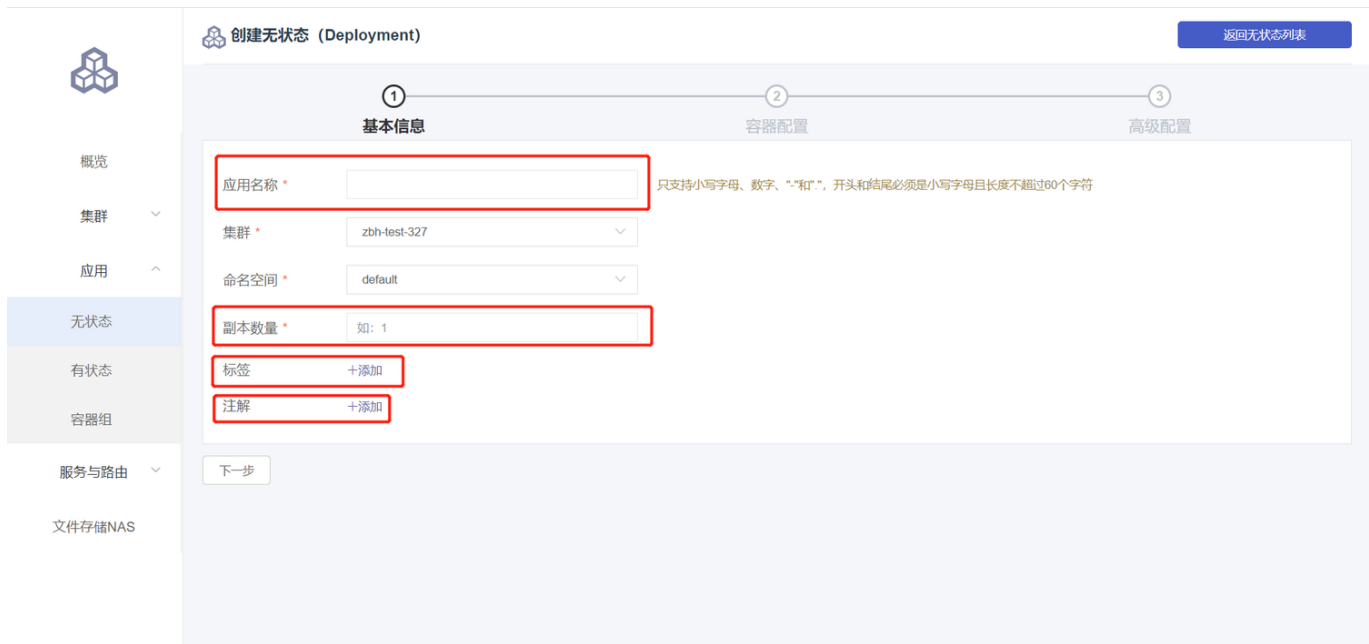
3. 操作说明

- 创建无状态（Deployment）应用

a) 在容器服务菜单下，单击左侧导航栏中的**应用>无状态**，选择所需的命名空间（默认为default），然后单击页面右上角的**创建**。



b) 设置应用名称、集群、命名空间、副本数量（即应用包含的 Pod 数量）、标签和注解。然后单击下一步进入容器配置页面



c) 设置容器配置

i. 基本配置

- **镜像名称**：填写所用镜像名称，本例中为hello-world。格式为domain/imagename。
- **镜像版本**：填写所需镜像版本，如不指定，默认为latest。
- **最小申请**：为该应用所需最小资源额度，包括 CPU 和内存两种资源。该资源由容器独占，以防资源不足而被其他服务或进程争占资源，导致应用不可用。其中，CPU 资源的单位为 Core（即一个核）、内存的单位为 MiB。
- **最大申请**：可指定该应用所能使用的最大资源额度，包括 CPU 和内存两种资源，防止占用过多资源。其中，CPU 资源的单位为 Core（即一个核）、内存的单位为 MiB。
- **Init Container**：勾选该项，表示创建一个 Init Container，在主容器启动前执行，进行初始化工作，详情参考<https://kubernetes.io/docs/concepts/workloads/pods/init-containers/>。
- **添加容器**：支持添加多个容器

① 基本信息 ② 容器配置 ③ 高级配置

容器1 容器2 × 添加容器

镜像名称 *

镜像版本

最小申请 CPU Core 内存 MiB

最大限制 CPU Core 内存 MiB

容器启动项 stdin tty

Init Container

ii. 可选配置

- **端口映射**：配置所启动容器使用的协议（支持TCP和UDP）、端口
- **环境变量**：配置所启动容器所需的环境变量（key/value的形式）
- **健康检查和就绪检查**：支持健康检查（liveness）和就绪检查（Readiness）。健康检查用于检测何时重启容器；就绪检查用于确定容器是否已经就绪，且可以接受流量。更多信息，请参见<https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-probes>。

健康检查

开启

路径 *

端口

http头

运行多久后开始检测 秒

检查间隔 秒 检查超时 秒

不健康阈值 次

就绪检查

开启

路径 *

端口

http头

运行多久后开始检测 秒

检查间隔 秒 检查超时 秒

健康阈值 次 不健康阈值 次

请求类型	配置说明
HTTP/HTTPS	<p>即向容器发送一个 HTTPget 请求，支持的参数包括：</p> <ul style="list-style-type: none"> • 路径：访问 HTTP server 的路径。 • 端口：容器暴露的访问端口或端口名，端口号必须介于 1~65535。 • HTTP 头：即 HTTPHeaders，HTTP 请求中自定义的请求头，HTTP 允许重复的 header。支持键值对的配置方式。 • 运行多久后开始检测（秒）：即 initialDelaySeconds，容器启动后第一次执行检测时需要等待多少秒，默认为 3 秒。 • 检查间隔（秒）：即 periodSeconds，指执行检查的时间间隔，默认为 10 秒，最小为 1 秒。 • 不健康阈值：探测成功后，最少连续探测失败多少次才被认定为失败。默认是 3，最小值是 1。
TCP	<p>即向容器发送一个 TCP Socket，kubelet 将尝试在指定端口上打开容器的套接字。如果可以建立连接，容器被认为是健康的，如果不能就认为是失败的。支持的参数包括：</p> <ul style="list-style-type: none"> • 端口：容器暴露的访问端口或端口名，端口号必须介于 1~65535。

	<ul style="list-style-type: none"> • 延迟探测时间（秒）：即 initialDelaySeconds，容器启动后第一次执行探测时需要等待多少秒，默认为 15 秒。 • 执行探测频率（秒）：即 periodSeconds，指执行探测的时间间隔，默认为 10 秒，最小为 1 秒。 • 超时时间（秒）：即 timeoutSeconds，探测超时时间。默认 1 秒，最小 1 秒。 • 不健康阈值：探测成功后，最少连续探测失败多少次才被认定为失败。默认是 3，最小值是 1。
<p>命令行</p>	<p>通过在容器中执行探针检测命令，来检测容器的健康情况。支持的参数包括：</p> <ul style="list-style-type: none"> • 命令行：用于检测容器健康情况的探测命令。 • 延迟探测时间（秒）：即 initialDelaySeconds，容器启动后第一次执行探测时需要等待多少秒，默认为 5 秒。 • 执行探测频率（秒）：即 periodSeconds，指执行探测的时间间隔，默认为 10 秒，最小为 1 秒。 • 超时时间（秒）：即 timeoutSeconds，探测超时时间。默认 1 秒，最小 1 秒。 • 不健康阈值：探测成功后，最少连续探测失败多少次才被认定为失败。默认是 3，最小值是 1。

- **生命周期**：为容器的生命周期配置容器启动执行、启动后处理和停止前处理。具体参见 <https://kubernetes.io/docs/tasks/configure-pod-container/attach-handler-lifecycle-event/>。
 - **启动执行**：为容器设置预启动命令和参数。
 - **启动后处理**：为容器设置启动后的命令。
 - **停止前处理**：为容器设置预结束命令。
- **数据卷**：支持配置本地存储和云存储。
 - **本地存储**：支持主机目录（hostpath）、配置项（configmap）、保密字典（secret）和临时目录四种存储卷类型，将对应的挂载源挂载到容器路径中。更多信息参见 [volumes](#)。
 - **云存储**：支持云存储。挂载源为该集群可用存储卷（PVC），详情参见左侧菜单栏**集群->存储卷**

开启

生命周期

启动执行 命令

 参数

启动后处理 命令

停止前处理 命令

数据卷

+添加本地存储

存储卷类型 *	挂载源 *	存储路径 *	操作
主机目录	<input type="text"/>	<input type="text"/>	删除

+添加云存储

存储卷类型 *	挂载源 *	存储路径 *	操作
云存储	请选择	<input type="text"/>	删除

iii. 完成容器配置后，单击下一步。

iv. 进行高级设置。

- **可选：水平伸缩。** 您可勾选是否开启水平伸缩，为了满足应用在不同负载下的需求，容器服务支持容器组（Pod）的弹性伸缩，即根据容器 CPU 和内存资源占用情况自动调整容器副本数量。
 - **指标：** 可选 CPU 使用量和内存使用量，需要和设置的所需资源类型相同。
 - **触发条件：** 资源使用率的百分比，超过设置的Pod request值，容器开始扩容。
 - **最大副本数：** 该 Deployment 可扩容的副本（即容器组Pod）数量上限。
 - **最小副本数：** 该 Deployment 可缩容的副本（即容器组Pod）数量下限。

注：若要启用水平伸缩，您必须为容器设置所需资源，否则容器自动伸缩无法生效。参见容器基本配置环节。

- **可选：升级方式。** 升级方式包括滚动升级（rollingupdate）和替换升级（recreate），详细请参见<https://kubernetes.io/zh/docs/concepts/workloads/controllers/deployment/>
 - **不可使用Pod最大数量：** 用于指定 Deployment在更新过程中不可用状态的Pod数量的百分比上限
 - **超过期望的Pod数量：** 用于指定在Deployment 更新Pod的过程中Pod总数超过Pod期望副本数的百分比上限

- 可选：**Pod标签和注解**。设置容器组的标签和注解。可用于Pod的调度设置。

- 可选：**调度设置**（亲和性调度依赖节点和 Pod 标签，您可使用内置的标签进行调度；也可预先为节点、Pod 配置相关的标签）
 - **节点亲和性**。通过已有节点 Node 的标签（键）、操作符和标签值（值）进行设置。
 - **必须满足**，即硬约束，一定要满足，对应 `requiredDuringSchedulingIgnoredDuringExecution`，效果与 `NodeSelector` 相同。您可以定义多条硬约束规则，但只需满足其中一条。
 - **尽量满足**，即软约束，不一定满足，对应 `preferredDuringSchedulingIgnoredDuringExecution`。您可为软约束规则设定权重，具体调度时，若存在多个符合条件的节点，权重最大的节点会被优先调度。您可定义多条软约束规则，但必须满足全部约束，才会进行调度。
 - **约束条件（操作符）**：包括 `In`, `NotIn`, `Exists`, `DoesNotExist`

创建
×

必要满足

选择器
+添加

键 *	操作符 *	值 *	操作
<input style="width: 100%;" type="text"/>	In ▼	<input style="width: 100%;" type="text"/>	删除

尽量满足

+添加规则
×

权重 *

选择器

选择器
+添加

键 *	操作符 *	值 *	操作
<input style="width: 100%;" type="text"/>	In ▼	<input style="width: 100%;" type="text"/>	删除

确认

取消

- **应用亲和性。**通过节点上运行的 Pod 的标签（键）、操作符和标签值（值）进行设置，支持硬约束和软约束。决定应用的 Pod 可以和哪些 Pod 部署在同一拓扑域。例如，对于相互通信的服务，可通过应用亲和性调度，将其部署到同一拓扑域（如同一个主机）中，减少它们之间的网络延迟。
 - **必须满足**，即硬约束，一定要满足，对应 `requiredDuringSchedulingIgnoredDuringExecution`，Pod 的亲和性调度必须要满足后续定义的约束条件。
 - **命名空间**：该策略是依据 Pod 的 Label 进行调度，所以会受到命名空间的约束。
 - **拓扑域**：即 `topologyKey`，指定调度时作用域，这是通过 Node 节点的标签来实现的。
 - **选择器**：单击选择器右侧的添加按钮，可添加多条硬约束规则。
 - **尽量满足**，即软约束，不一定满足，对应 `preferredDuringSchedulingIgnoredDuringExecution`。Pod 的亲和性调度会尽量满足后续定义的约束条件。对于软约束规则，您可配置每条规则的权重，其他配置规则与硬约束规则相同。
 - **约束条件（操作符）**：包括 In, NotIn, Exists, DoesNotExist

创建

+添加规则

命名空间 *

拓扑域 *

选择器 +添加

键 *	操作符 *	值 *	操作
<input type="text"/>	In ▼	<input type="text"/>	删除

+添加规则

必要满足

确认 取消

创建

+添加规则

权重 *

命名空间 *

拓扑域 *

选择器 +添加

键 *	操作符 *	值 *	操作
<input type="text"/>	In ▼	<input type="text"/>	删除

+添加规则

尽量满足

确认 取消

- 设置应用非亲和性调度。决定应用的 Pod 不与哪些 Pod 部署在同一拓扑域。应用非亲和性调度的场景包括：
 - 将一个服务的 Pod 分散部署到不同的拓扑域（如不同主机）中，提高服务本身的稳定性。
 - 给予 Pod 一个节点的独占访问权限来保证资源隔离，保证不会有其它 Pod 来分享节点资源。
 - 把可能会相互影响的服务的 Pod 分散在不同的主机上。

说明：应用非亲和性调度的设置方式与亲和性调度相同，但相同的调度规则代表的意义不同，请按需进行选择。

- 设置调度容忍（Toleration）。与节点设置的污点（Taints）配套使用，且Pod的 Toleration声明中的key和effect需要与Taint的设置保持一致。详情参见节点管理，<https://kubernetes.io/docs/concepts/configuration/taint-and-toleration/>

调度容忍
✕

规则 +添加

标签键 *	操作符 *	标签值 *	效果 *	时间(秒) *	操作
<input type="text"/>	Equal ▾	<input type="text"/>	NoSchedt ▾		删除

确认

取消

v. 点击创建。

- 在应用->无状态可看到创建成功的应用。

chunfa-009 / default / 无状态 (Deployment)
创建

Q
↻

名称	容器数量	镜像	创建时间	操作
hello-world	1/1	tifayuki/hello-world.cds	2020-03-28 18:26:00	详情 伸缩 编辑 移除

1

- 点击详情，可看到该应用的详情页面。

基本信息

名称: hello-world
命名空间: default
创建时间: 2020-03-28 18:26:00
标志:
注解: deployment.kubernetes.io/revision: 1
选择器: cdsapp: hello-world
策略: RollingUpdate
滚动升级策略: 超过期望的Pod数量: 25%
不可用Pod最大数量: 25%
状态: 已更新: 1个, 不可用: 0个, 计划1个

状态

类型	状态	更新时间	原因	消息
Available	True	2020-03-28 18:26:15	MinimumReplicasAvailable	Deployment has minimum availability.
Progressing	True	2020-03-28 18:26:15	NewReplicaSetAvailable	ReplicaSet "hello-world-5cb7b5c8bb" has successfully progressed.

事件

容器组水平伸缩器 历史版本

类型 (全部)	对象 (全部)	信息	原因	时间
Normal	{ "kind": "Deployment", "name": "hello-world" }	Scaled up replica set hello-world-5cb7b5c8bb to 1	ScalingReplicaSet	2020-03-28 18:25:55

- 点击**伸缩**，可修改当前应用的副本数量。

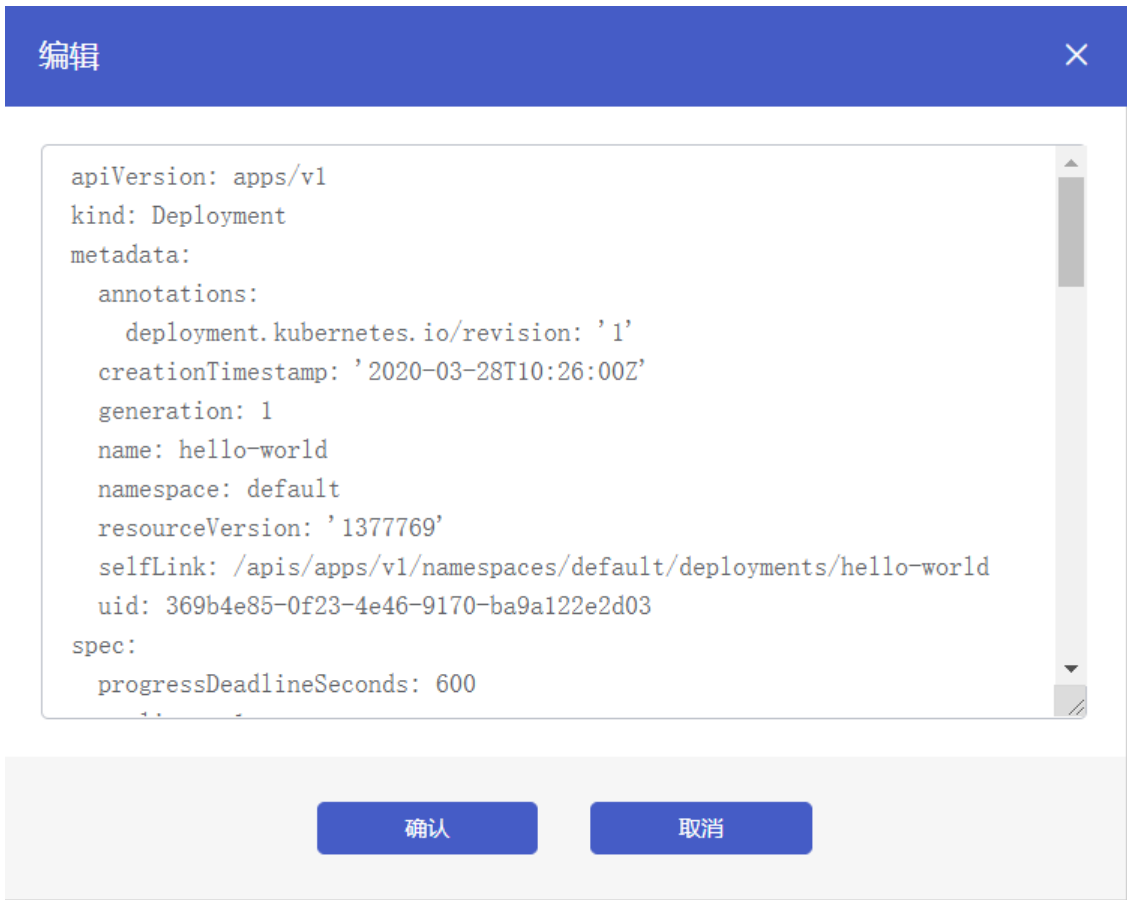
伸缩

所需容器组的数量

Pod可用副本数: 1

Pod配置副本数:

- 点击**编辑**，可修改该应用的yaml文件，点击**确认**即可生效。



- 点击**移除**，可删除该应用

- 创建**有状态 (StatefulSet)** 应用

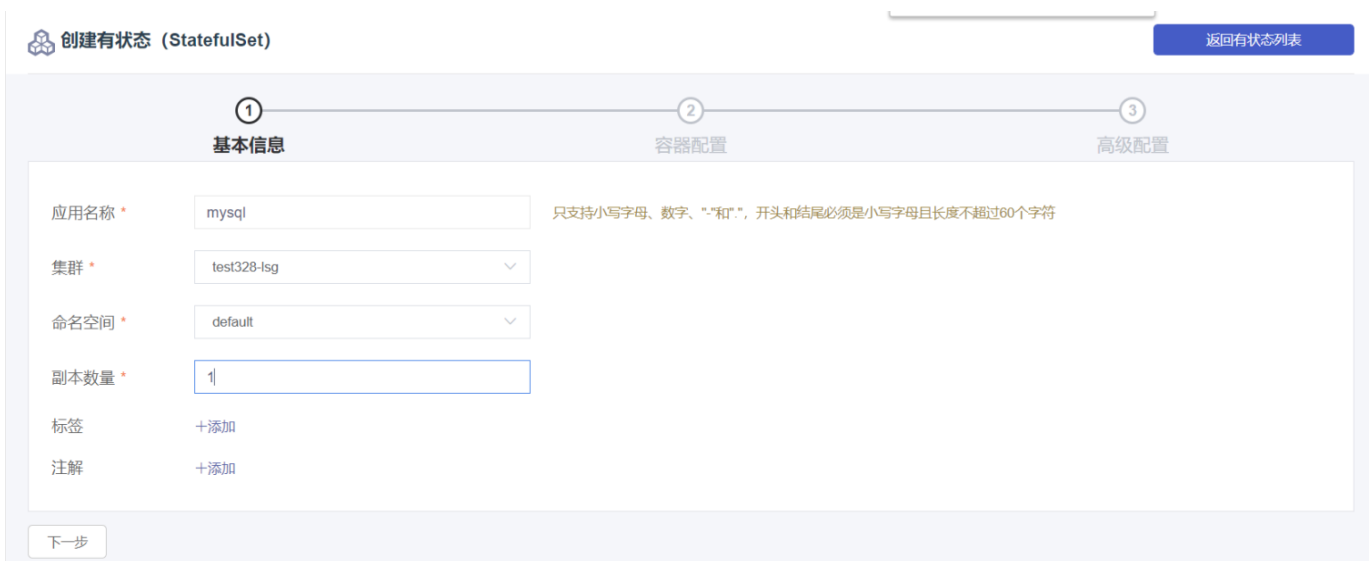
- 特性，详情参见<https://kubernetes.io/docs/tutorials/stateful-application/basic-stateful-set/>

应用场景	说明
稳定的持久化存储	基于PVC实现数据的持久化存储，即Pod重新调度后还是能访问到相同的持久化数据
稳定的网络标志	基于Headless Service（即没有Cluster IP的Service）来实现，即Pod重新调度后其PodName和HostName不变
有序部署，有序扩展	即Pod是有顺序的，在部署或者扩展的时候要依据定义的顺序依次依次进行（即从0到N-1，在下一个Pod运行之前所有之前的Pod必须都是Running和Ready状态），基于init containers来实现

a) 在容器服务菜单下，单击左侧导航栏中的**应用>有状态**，选择所需的命名空间（默认为default），然后单击页面右上角的**创建**。



b) 设置应用名称、集群、命名空间、副本数量（即应用包含的 Pod 数量）、标签和注解。然后单击下一步进入容器配置页面



c) 设置容器配置

i. 基本配置

- **镜像名称**：填写所用镜像名称，本例中为mysql。格式为domain/imagename。
- **镜像版本**：填写所需镜像版本，本例中为5.7。如不指定，默认为latest。
- **最小申请**：为该应用所需最小资源额度，包括 CPU 和内存两种资源。该资源由容器独占，以防资源不足而被其他服务或进程争占资源，导致应用不可用。其中，CPU 资源的单位为 Core（即一个核）、内存的单位为 MiB。
- **最大申请**：可指定该应用所能使用的最大资源额度，包括 CPU 和内存两种资源，防止占用过多资源。其中，CPU 资源的单位为 Core（即一个核）、内存的单位为 MiB。
- **Init Container**：勾选该项，表示创建一个 Init Container，在主容器启动前执行，进行初始化工作，详情参考<https://kubernetes.io/docs/concepts/workloads/pods/init-containers/>。
- **添加容器**：支持添加多个容器

① 基本信息
② 容器配置
③ 高级配置

容器1
容器2 ×
添加容器

镜像名称 *

镜像版本

最小申请 CPU Core 内存 MiB

最大限制 CPU Core 内存 MiB

容器启动项 stdin tty

Init Container

ii. 可选配置

- **端口映射**: 配置所启动容器使用的协议 (支持TCP和UDP)、端口
- **环境变量**: 配置所启动容器所需的环境变量 (key/value的形式)

+添加

端口映射	协议 *	容器端口 *	操作
	TCP		删除

+添加

环境变量	变量名称 *	变量/变量引用 *	操作
			删除

- **健康检查和就绪检查**: 支持健康检查 (liveness) 和就绪检查 (Readiness)。健康检查用于检测何时重启容器; 就绪检查用于确定容器是否已经就绪, 且可以接受流量。更多信息, 请参见 <https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-probes>。

健康检查

开启

路径 *

端口

http头

运行多久后开始检测 秒

检查间隔 秒 检查超时 秒

不健康阈值 次

就绪检查

开启

路径 *

端口

http头

运行多久后开始检测 秒

检查间隔 秒 检查超时 秒

健康阈值 次 不健康阈值 次

请求类型	配置说明
HTTP/HTTPS	<p>即向容器发送一个 HTTPget 请求，支持的参数包括：</p> <ul style="list-style-type: none"> • 路径：访问 HTTP server 的路径。 • 端口：容器暴露的访问端口或端口名，端口号必须介于 1~65535。 • HTTP 头：即 HTTPHeaders，HTTP 请求中自定义的请求头，HTTP 允许重复的 header。支持键值对的配置方式。 • 运行多久后开始检测（秒）：即 initialDelaySeconds，容器启动后第一次执行检测时需要等待多少秒，默认为 3 秒。 • 检查间隔（秒）：即 periodSeconds，指执行检查的时间间隔，默认为 10 秒，最小为 1 秒。 • 不健康阈值：探测成功后，最少连续探测失败多少次才被认定为失败。默认是 3，最小值是 1。
TCP	<p>即向容器发送一个 TCP Socket，kubelet 将尝试在指定端口上打开容器的套接字。如果可以建立连接，容器被认为是健康的，如果不能就认为是失败的。支持的参数包括：</p> <ul style="list-style-type: none"> • 端口：容器暴露的访问端口或端口名，端口号必须介于 1~65535。

	<ul style="list-style-type: none"> • 延迟探测时间（秒）：即 initialDelaySeconds，容器启动后第一次执行探测时需要等待多少秒，默认为 15 秒。 • 执行探测频率（秒）：即 periodSeconds，指执行探测的时间间隔，默认为 10 秒，最小为 1 秒。 • 超时时间（秒）：即 timeoutSeconds，探测超时时间。默认 1 秒，最小 1 秒。 • 不健康阈值：探测成功后，最少连续探测失败多少次才被认定为失败。默认是 3，最小值是 1。
命令行	<p>通过在容器中执行探针检测命令，来检测容器的健康情况。支持的参数包括：</p> <ul style="list-style-type: none"> • 命令行：用于检测容器健康情况的探测命令。 • 延迟探测时间（秒）：即 initialDelaySeconds，容器启动后第一次执行探测时需要等待多少秒，默认为 5 秒。 • 执行探测频率（秒）：即 periodSeconds，指执行探测的时间间隔，默认为 10 秒，最小为 1 秒。 • 超时时间（秒）：即 timeoutSeconds，探测超时时间。默认 1 秒，最小 1 秒。 • 不健康阈值：探测成功后，最少连续探测失败多少次才被认定为失败。默认是 3，最小值是 1。

- **生命周期**：为容器的生命周期配置容器启动执行、启动后处理和停止前处理。具体参见 <https://kubernetes.io/docs/tasks/configure-pod-container/attach-handler-lifecycle-event/>。
 - **启动执行**：为容器设置预启动命令和参数。
 - **启动后处理**：为容器设置启动后的命令。
 - **停止前处理**：为容器设置预结束命令。
- **数据卷**：支持配置本地存储和云存储。
 - **本地存储**：支持主机目录（hostpath）、配置项（configmap）、保密字典（secret）和临时目录四种存储卷类型，将对应的挂载源挂载到容器路径中。更多信息参见 **volumes**。
 - **云存储**：支持云存储。挂载源为该集群可用存储卷（PVC），详情参见左侧菜单栏**集群->存储卷**

开启

生命周期

启动执行 命令

参数

启动后处理 命令

停止前处理 命令

+ 添加本地存储

存储卷类型 *	挂载源 *	存储路径 *	操作
<input type="text" value="主机目录"/>	<input type="text"/>	<input type="text"/>	删除

+ 添加云存储

存储卷类型 *	挂载源 *	存储路径 *	操作
<input type="text" value="云存储"/>	<input type="text" value="请选择"/>	<input type="text"/>	删除

上一步

下一步

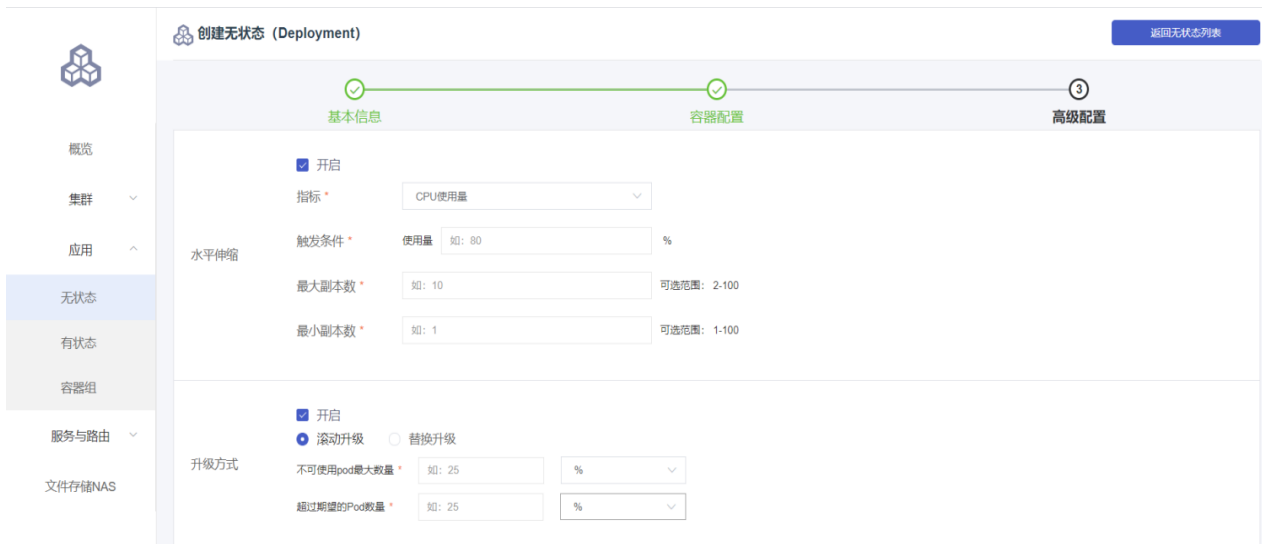
iii. 完成容器配置后，单击下一步。

iv. 进行高级设置。

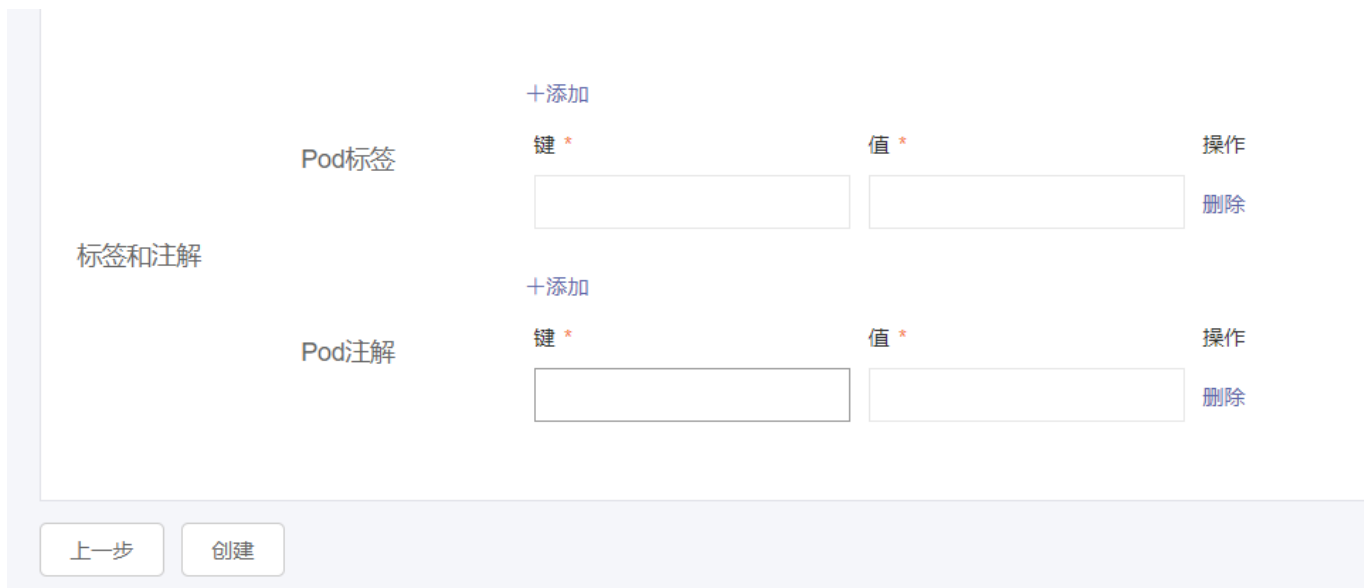
- **可选：水平伸缩。**您可勾选是否开启水平伸缩，为了满足应用在不同负载下的需求，容器服务支持容器组（Pod）的弹性伸缩，即根据容器 CPU 和内存资源占用情况自动调整容器副本数量。
 - **指标：**可选 CPU 使用量和内存使用量，需要和设置的所需资源类型相同。
 - **触发条件：**资源使用率的百分比，超过设置的Pod request值，容器开始扩容。
 - **最大副本数：**该 Deployment 可扩容的副本（即容器组Pod）数量上限。
 - **最小副本数：**该 Deployment 可缩容的副本（即容器组Pod）数量下限。

注：若要启用水平伸缩，您必须为容器设置所需资源，否则容器自动伸缩无法生效。参见容器基本配置环节。

- **可选：升级方式。**升级方式包括滚动升级（rollingupdate）和替换升级（recreate），详细请参见<https://kubernetes.io/zh/docs/concepts/workloads/controllers/deployment/>
 - **不可使用Pod最大数量：**用于指定 Deployment在更新过程中不可用状态的Pod数量的百分比上限
 - **超过期望的Pod数量：**用于指定在Deployment 更新Pod的过程中Pod总数超过Pod期望副本数的百分比上限



- 可选：**Pod标签和注解**。设置容器组的标签和注解。可用于Pod的调度设置。



- 可选：**调度设置**（亲和性调度依赖节点和 Pod 标签，您可使用内置的标签进行调度；也可预先为节点、Pod 配置相关的标签）
 - **节点亲和性**。通过已有节点 Node 的标签（键）、操作符和标签值（值）进行设置。
 - **必须满足**，即硬约束，一定要满足，对应 `requiredDuringSchedulingIgnoredDuringExecution`，效果与 `NodeSelector` 相同。您可以定义多条硬约束规则，但只需满足其中一条。
 - **尽量满足**，即软约束，不一定满足，对应 `preferredDuringSchedulingIgnoredDuringExecution`。您可为软约束规则设定权重，具体调度时，若存在多个符合条件的节点，权重最大的节点会被优先调度。您可定义多条软约束规则，但必须满足全部约束，才会进行调度。
 - **约束条件（操作符）**：包括 `In`, `NotIn`, `Exists`, `DoesNotExist`

创建
×

必要满足

选择器
+添加

键 *	操作符 *	值 *	操作
<input style="width: 95%;" type="text"/>	In ▼	<input style="width: 95%;" type="text"/>	删除

尽量满足

+添加规则
×

权重 *

选择器

选择器
+添加

键 *	操作符 *	值 *	操作
<input style="width: 95%;" type="text"/>	In ▼	<input style="width: 95%;" type="text"/>	删除

确认

取消

- **应用亲和性。**通过节点上运行的 Pod 的标签（键）、操作符和标签值（值）进行设置，支持硬约束和软约束。决定应用的 Pod 可以和哪些 Pod 部署在同一拓扑域。例如，对于相互通信的服务，可通过应用亲和性调度，将其部署到同一拓扑域（如同一个主机）中，减少它们之间的网络延迟。
 - **必须满足**，即硬约束，一定要满足，对应 `requiredDuringSchedulingIgnoredDuringExecution`，Pod 的亲和性调度必须要满足后续定义的约束条件。
 - **命名空间**：该策略是依据 Pod 的 Label 进行调度，所以会受到命名空间的约束。
 - **拓扑域**：即 `topologyKey`，指定调度时作用域，这是通过 Node 节点的标签来实现的。
 - **选择器**：单击选择器右侧的添加按钮，可添加多条硬约束规则。
 - **尽量满足**，即软约束，不一定满足，对应 `preferredDuringSchedulingIgnoredDuringExecution`。Pod 的亲和性调度会尽量满足后续定义的约束条件。对于软约束规则，您可配置每条规则的权重，其他配置规则与硬约束规则相同。
 - **约束条件（操作符）**：包括 In, NotIn, Exists, DoesNotExist

创建
×

+添加规则

命名空间 *

×

拓扑域 *

选择器 +添加

键 *	操作符 *	值 *	操作
<input style="width: 100%;" type="text"/>	In ▼	<input style="width: 100%;" type="text"/>	删除

+添加规则

确认

取消

创建
×

+添加规则

权重 *

×

命名空间 *

拓扑域 *

选择器 +添加

键 *	操作符 *	值 *	操作
<input style="width: 100%;" type="text"/>	In ▼	<input style="width: 100%;" type="text"/>	删除

+添加规则

确认

取消

- 设置应用非亲和性调度。决定应用的 Pod 不与哪些 Pod 部署在同一拓扑域。应用非亲和性调度的场景包括：
 - 将一个服务的 Pod 分散部署到不同的拓扑域（如不同主机）中，提高服务本身的稳定性。
 - 给予 Pod 一个节点的独占访问权限来保证资源隔离，保证不会有其它 Pod 来分享节点资源。

- 把可能会相互影响的服务的 Pod 分散在不同的主机上。

说明：应用非亲和性调度的设置方式与亲和性调度相同，但相同的调度规则代表的意义不同，请按需进行选择。

- 设置**调度容忍 (Toleration)**。与节点设置的污点 (Taints) 配套使用，且Pod的 Toleration声明中的key和effect需要与Taint的设置保持一致。详情参见节点管理，<https://kubernetes.io/docs/concepts/configuration/taint-and-toleration/>

调度容忍
✕

规则 +添加

标签键 *	操作符 *	标签值 *	效果 *	时间(秒) *	操作
	Equal ▾		NoSchedt ▾		删除

确认

取消

v. 点击创建。

- 在应用->有状态可看到创建成功的应用。

🏠

chunfa-009

/

default

/

有状态 (StatefulSet)

创建

🔍

↻

名称	容器数量	镜像	创建时间	操作
mysql	1/1	daocloud.io/library/mysql:latest	2020-03-28 22:45:49	详情 伸缩 编辑 移除

<

1

>

- 点击详情，可看到该应用的详情页面。

41

基本信息

名称 mysql
命名空间 default
创建时间 2020-03-28 22:45:49
标签
注解
选择器 cdsapp: mysql
策略 RollingUpdate
状态 已更新: 1个, 不可用: 0个, 计划1个

事件

类型 (全部)	对象 (全部)	信息	原因	时间
Normal	{ "kind": "StatefulSet", "name": "mysql" }	create Pod mysql-0 in StatefulSet mysql successful	SuccessfulCreate	2020-03-28 22:45:44

- 点击**伸缩**，可修改当前应用的副本数量。

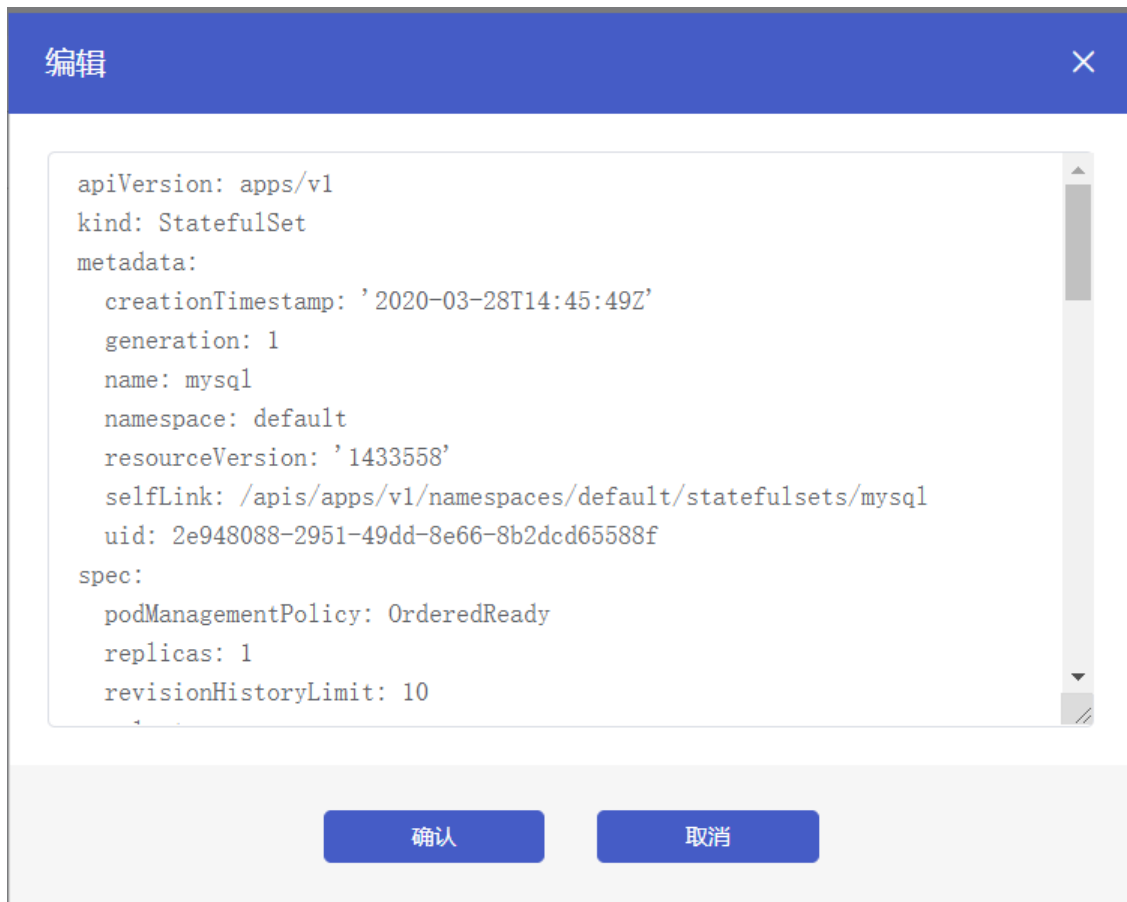
伸缩

所需容器组的数量

Pod可用副本数: 1

Pod配置副本数:

- 点击**编辑**，可修改该应用的yaml文件。



- 点击**移除**，可删除该应用
- 进入master节点控制台，可查看创建成功的mysql应用

```
[root@master001 ~]# kubectl get pods -n default
NAME                                READY  STATUS   RESTARTS  AGE
hello-world-6d774b787c-f76zq       1/1    Running  0          70m
mysql-0                              1/1    Running  0          32m
[root@master001 ~]# kubectl exec -it mysql-0 bash
root@mysql-0:/#
```

```

[root@master001 ~]# kubectl exec -it mysql-0 bash
root@mysql-0:/# mysql -h 127.0.0.1 -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 19
Server version: 8.0.19 MySQL Community Server - GPL

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> |

```

• 查看容器组（应用->容器组）

- 该页面可以看到集群对应命名空间下的Pod

The screenshot shows a Kubernetes dashboard interface. At the top, there are dropdown menus for 'chunfa-009' and 'default', followed by the text '容器组 (Pod)'. Below this is a search bar with the placeholder '请输入内容' and a search icon. The main content is a table with the following columns: 名称, 状态, 重启次数, Pod IP, 节点, 创建时间, CPU, 内存, and 操作. Two pods are listed: 'hello-world-6d774b787c-f76zq' and 'mysql-0'. The 'mysql-0' pod is highlighted with a blue background. Its status is 'Running', it has 0 restarts, IP 10.244.3.10, is on node 'worker001', and was created on 2020-03-28 22:45:44. Its CPU usage is 3m and memory usage is 450Mi. There are '详情' and '删除' links for each pod. At the bottom right, there is a pagination control showing '1'.

名称	状态	重启次数	Pod IP	节点	创建时间	CPU	内存	操作
hello-world-6d774b787c-f76zq	Running	0	10.244.3.8	worker001	2020-03-28 22:07:43	0m	9Mi	详情 删除
mysql-0	Running	0	10.244.3.10	worker001	2020-03-28 22:45:44	3m	450Mi	详情 删除

- 点击详情，可以查看该容器的详情

基本信息				
名称:	mysql-0	命名空间:	default	
状态:	Running	创建时间:	2020-03-28 22:45:44	
节点:	worker001	Pod IP:	10.244.3.10	
标签:	cdsapp: mysql controller-revision-hash: mysql-78d449f568 statefulset.kubernetes.io/pod-name: mysql-0			

状态				
类型	状态	更新时间	原因	消息
Initialized	True	2020-03-28 22:45:45	--	--
Ready	True	2020-03-28 23:04:41	--	--
ContainersReady	True	2020-03-28 23:04:41	--	--
PodScheduled	True	2020-03-28 22:45:49	--	--

网络管理

1.简介

本文介绍首云容器服务Kubernetes支持的网络类型。

首云支持ClusterIp

ClusterIp 是 Kubernetes 中默认的服务类型（ServiceType），选择此种类型，对应的 Service 将被分配一个集群内部的 IP 地址，只能在集群内部被访问。

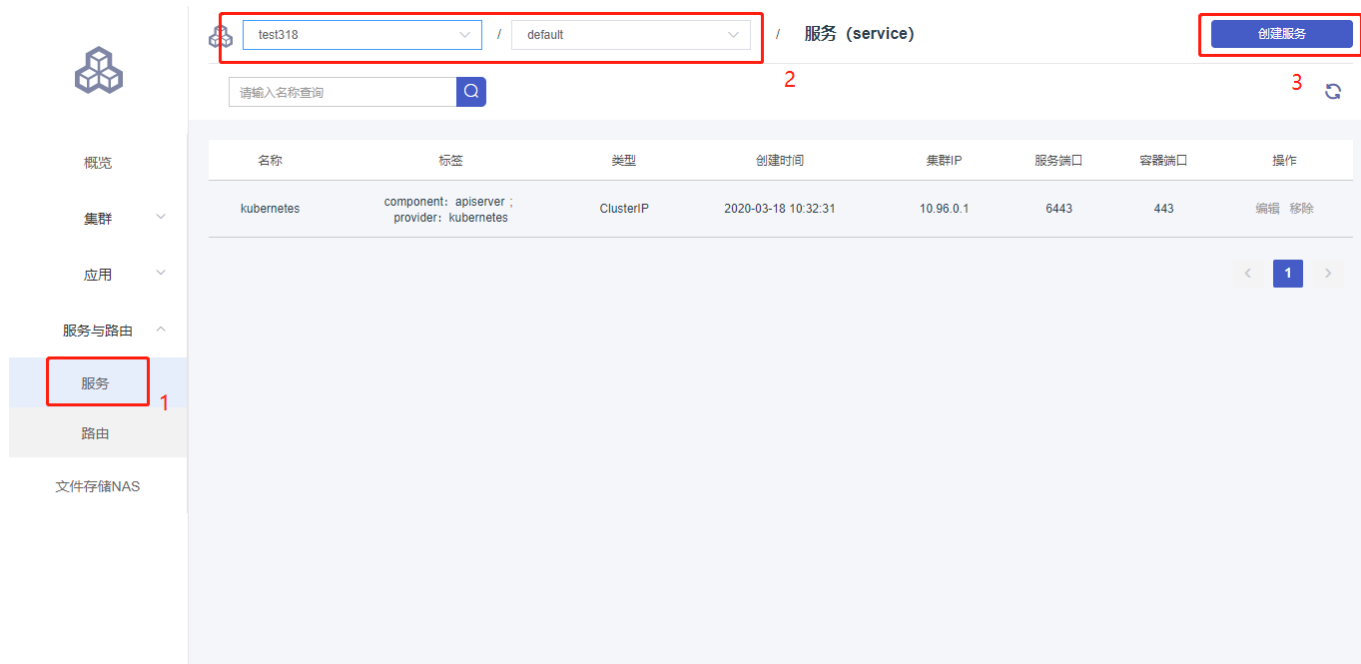
NodePort

在每台 Node 的固定端口上暴露服务，选择 NodePort 的服务类型，集群会自动创建一个 ClusterIp 类型的服务，负责处理Node接收到的外部流量。集群外部的 Client 可以通过:的方式访问该服务。

2.操作说明

- 创建服务

- 1.在容器集群菜单下，单击左侧导航栏中的服务与路由 > 服务，进入服务列表页面。
- 2.选择所需的集群和命名空间，单击页面右上角的**创建服务**。



test318 / default / 服务 (service) 创建服务

请输入名称查询

名称	标签	类型	创建时间	集群IP	服务端口	容器端口	操作
kubernetes	component: apiserver ; provider: kubernetes	ClusterIP	2020-03-18 10:32:31	10.96.0.1	6443	443	编辑 移除

1 2 3

3.在弹出的**创建服务**对话框中，进行配置。

添加服务
✕

名称: *

类型: ▾

关联: ▾

外部流量策略: ▾

端口映射: +添加

名称	协议	服务端口	容器端口	操作
<input type="text" value="tcp"/> *	<input type="text" value="TCP"/> ▾	<input type="text" value="80"/> *	<input type="text" value="80"/> *	

注解: +添加

键	值	操作
<input type="text" value="cds"/> *	<input type="text" value="test"/> *	* 删除

标签: +添加

键	值	操作
<input type="text" value="cds"/> *	<input type="text" value="test"/> *	* 删除

确认
取消

名称：输入服务的名称，本例中为hello。

类型：选择服务类型，即服务访问的方式。包括：

- **ClusterIP：**指通过集群的内部 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：**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。

关联：选择服务要绑定的后端对象。若不进行关联部署，则不会创建相关的 Endpoints 对象，您可自己进行绑定。

外部流量策略：可选值为Local或Cluster。

说明 服务类型为NodePort时，才能设置外部流量策略。

端口映射：添加服务端口和容器端口，容器端口需要与后端的 Pod 中暴露的容器端口一致。

注解：为该服务添加一个注解（annotation）。

标签：您可为该服务添加一个标签，标识该服务。

4.单击确认，hello 服务出现在服务列表中。

名称	标签	类型	创建时间	集群IP	服务端口	容器端口	操作
kubernetes	component: apiserver ; provider: kubernetes	ClusterIP	2020-03-23 15:28:46		6443	443	编辑 移除
hello	-	NodePort	2020-03-27 12:38:52		8080	80	编辑 移除

• 查看服务

在容器集群菜单下，单击左侧导航栏中的服务与路由 > 服务，进入服务列表页面。

名称	标签	类型	创建时间	集群IP	服务端口	容器端口	操作
kubernetes	component: apiserver ; provider: kubernetes	ClusterIP	2020-03-23 15:28:46	10.96.0.1	6443	443	编辑 移除
hello	-	ClusterIP	2020-03-27 12:46:27	10.96.131.3	80	80	编辑 移除

• 编辑服务

1.在容器集群菜单下，单击左侧导航栏中的服务与路由 > 服务，进入服务列表页面。

2.选择集群和命名空间，选择所需的服务（本示例中选择hello），单击右侧的编辑。

名称	标签	类型	创建时间	集群IP	服务端口	容器端口	操作
kubernetes	component: apiserver ; provider: kubernetes	ClusterIP	2020-03-23 15:28:46	10.96.0.1	6443	443	编辑 移除
hello	-	ClusterIP	2020-03-27 12:46:27	10.96.131.3	80	80	编辑 移除

3.在弹出的更新对话框中，进行配置修改，然后单击确认。

编辑服务 ×

名称: *

类型: ▼

关联: ▼

端口映射: +添加

名称	协议	服务端口	容器端口	操作
<input style="width: 80px;" type="text" value="tcp"/> *	<input style="width: 100px;" type="text" value="TCP"/> ▼	<input style="width: 60px;" type="text" value="80"/>	<input style="width: 60px;" type="text" value="80"/> *	

注解: +添加

标签: +添加

确认
取消

• 删除服务

- 1.在容器集群菜单下，单击左侧导航栏中的服务与路由 > 服务，进入服务列表页面。
- 2.选择集群和命名空间，选择所需的服务（本示例中选择 hello），单击右侧的**移除**。

- 3.在弹出的对话框中点击确认，即可删除服务。

提示 ×

请确认删除hello服务吗!

确认
取消

• 创建路由

- 1.在容器集群菜单下，单击左侧导航栏中的服务与路由 > 路由，进入路由列表页面。
- 2.选择所需的集群和命名空间，单击页面右上角的**创建路由**。



3. 在弹出的路由创建对话框中对路由规则进行配置。

路由规则是指授权进站到达集群服务的规则，支持 http/https 规则，配置项包括域名、服务名称、服务端口、服务路径、注解和标签等。

添加路由

名称: *

规则: +添加

域名: *

服务: +添加

服务	端口	路径
<input type="text" value="hello"/> ▼	<input type="text" value="80"/> ▼	<input type="text"/>

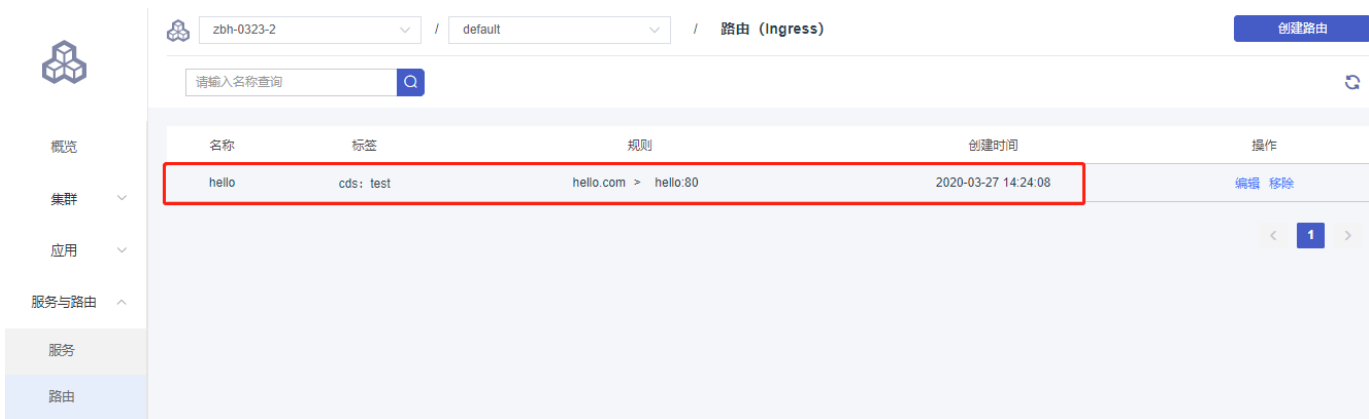
注解: +添加

键	值	操作
<input type="text"/>	<input type="text"/>	* 删除

标签: +添加

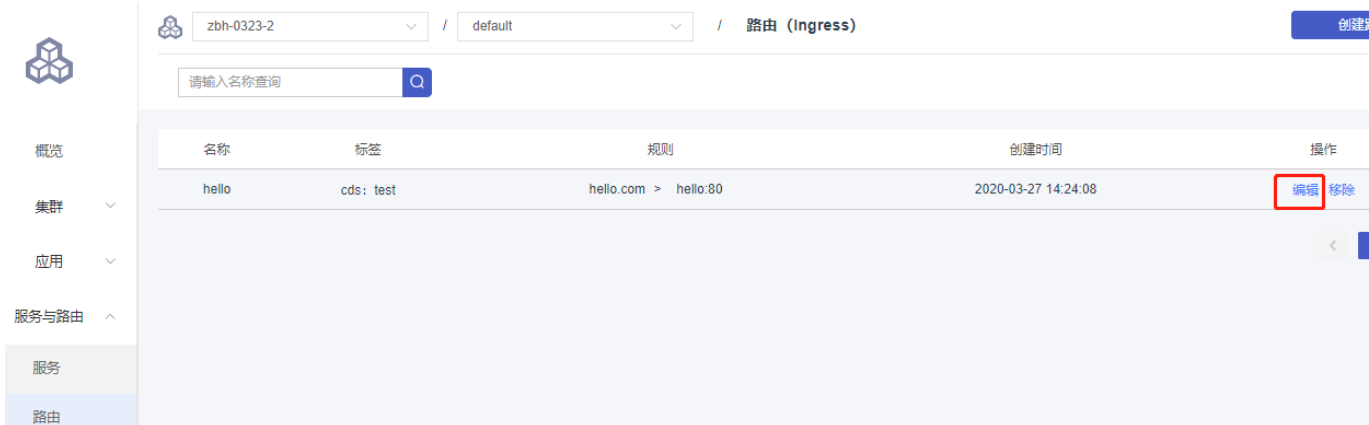
键	值	操作
<input type="text"/>	<input type="text"/>	* 删除

4. 最后单击**确认**，返回路由列表。等待一段时间，可以看到一条路由。



• 编辑路由

1. 在容器集群菜单下，单击左侧导航栏中的服务与路由 >路由，进入路由列表页面。
2. 选择集群和命名空间，选择所需的路由（本示例中选择hello），单击右侧的编辑。



3. 在弹出的对话框中，对路由的相关参数进行变更，然后单击**确认**，完成编辑

编辑路由
✕

名称: *

规则: +添加

域名: *

服务: +添加

服务	端口	路径
<input style="width: 60px;" type="text" value="hello"/> ▾	<input style="width: 40px;" type="text" value="80"/> ▾	<input style="width: 80px;" type="text"/>

注解: +添加

键	值	操作
<input style="width: 100px;" type="text" value="cds"/> *	<input style="width: 100px;" type="text" value="test"/> *	删除

标签: +添加

键	值	操作
<input style="width: 100px;" type="text" value="cds"/> *	<input style="width: 100px;" type="text" value="test"/> *	删除

确认
取消

- 移除路由

- 1.在容器集群菜单下，单击左侧导航栏中的服务与路由 >路由，进入路由列表页面。
- 2.选择集群和命名空间，选择所需的路由（本示例中选择hello），单击右侧的**移除**。



- 3.在弹出的对话框中点击确认，即可删除路由。



监控管理

1.简介

Prometheus是一套开源的系统监控报警框架，它具有灵活的数据模型：监控数据由值、时间戳、标签；源数据记录在标签中，支持采集时对标签进行修改，从而使得其具有强大的扩展能力。

2.操作说明

(1) 开启监控

说明：**集群创建后监控服务需要手动开启，开启过程如下：**

1. 在集群界面找到要开启监控的集群，点击右侧更多，会有开启监控的功能，如下图：



2. 点击开启监控后，会提示选择监控所需的pv，如下图：



创建此过程按照提示可能需要创建NAS存储或SC，PVC等,可参考上文存储管理

3. 点击确定，页面上方会显示会显示创建任务已下发的提示，同时集群状态转变为更新，如下图：

✔ 创建监控服务任务已下发，请等待完成！

1ce56738-68be-11ea-8c9d-0242ac11006d

test318

模板传输测试-无播

5

2020-03-18 10:13:53

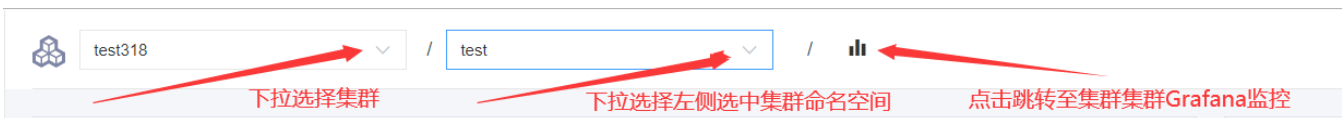
更新中

管理节点 集群扩容 仪表盘 更多

- 上述步骤完成后，等待集群更新，更新完成后，集群状态显示**正常**后可查看使用监控。

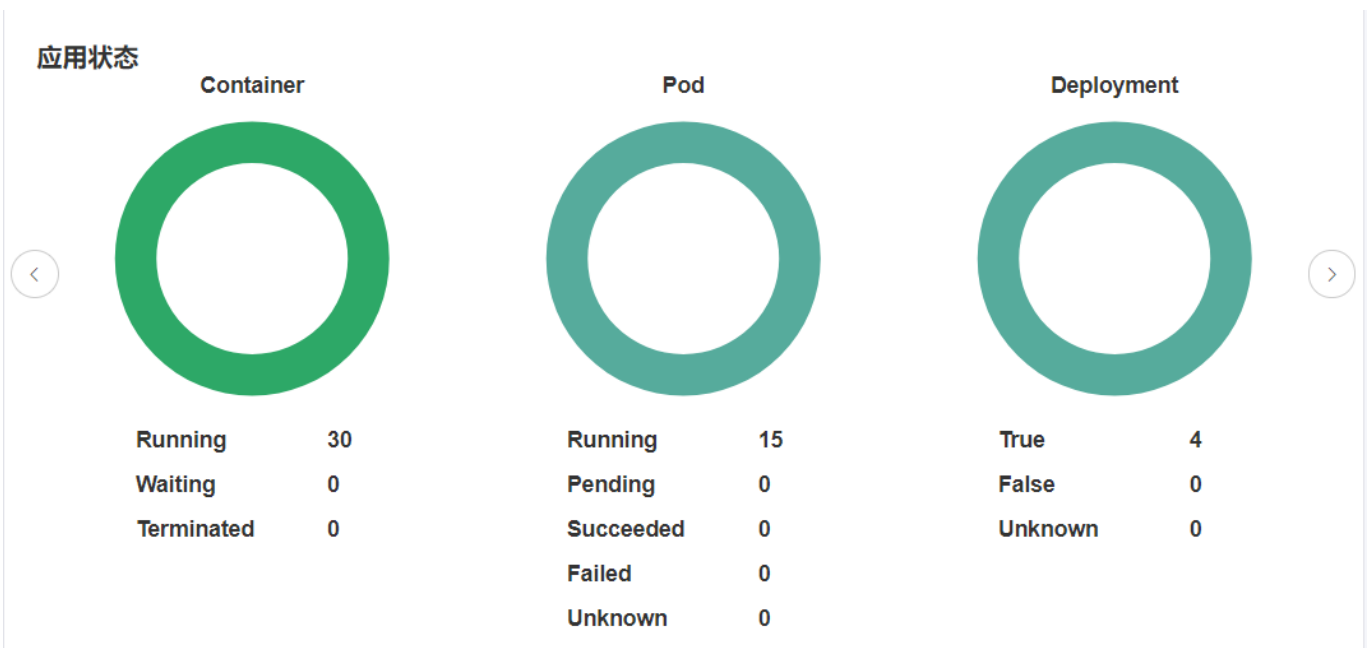
(2) 概览页监控查看以及资源介绍

- 选择集群，选择集群命名空间



- 监控仪表盘说明

- 应用状态





说明：

i. Container:

- Running:容器处于正在运行状态的容器个数
- Waiting:容器处于初始状态的容器个数
- Terminated:容器处于结束运行时状态的容器个数

ii. Pod(可在左侧功能栏[应用/容器组](#)查看详细信息):

- Running: 该Pod被绑定至一个节点，且该Pod内的所有容器均成功创建。
- Pending: K8s已经创建该Pod，但还没有进入运行状态，如Pod未完成调度，或正在拉取镜像等。
- Succeeded: Pod中所有容器都已成功终止，并且不会被重启。
- Failed: Pod中所有容器都已经终止，且至少有一个容器异常终止。
- Unknow: k8s的master节点与worker可能存在通信故障，无法确定Pod状态。

iii. Deployment(无状态应用，可在左侧功能栏[应用/无状态](#)查看详细信息)

- True: Deployment处于正常 (Complete) 状态的个数
- False: Deployment处于 (Failed) 状态的个数
- Unknown: Deployment状态未知的个数

iv. Statefulset (有状态应用，可在左侧功能栏[应用/有状态](#)查看详细信息)

- True:有状态应处于正常的个数
- False:有状态应用处于异常的个数

- **节点状态 (可在左侧功能栏[集群/节点](#)查看详细信息)**



- i. 就绪：当前处于就绪（可用）状态的节点(包括master节点和worker节点)。
- ii. 不可调度：显示Worker节点处于不可调度的节点数量。
- iii. PID压力：显示PID资源不足的节点个数。
- iv. 内存压力：显示内存过低的节点个数。
- v. 磁盘压力：显示磁盘容量低的节点个数。

资源监控



- i. CPU：显示当前集群的CPU总量，已使用，使用率。
- ii. 内存：显示当前集群的内存总量，已使用，使用率。（单位: GiB）
- iii. 文件系统使用统计：显示当前集群文件存储总量，已使用，使用率。（单位: GiB）

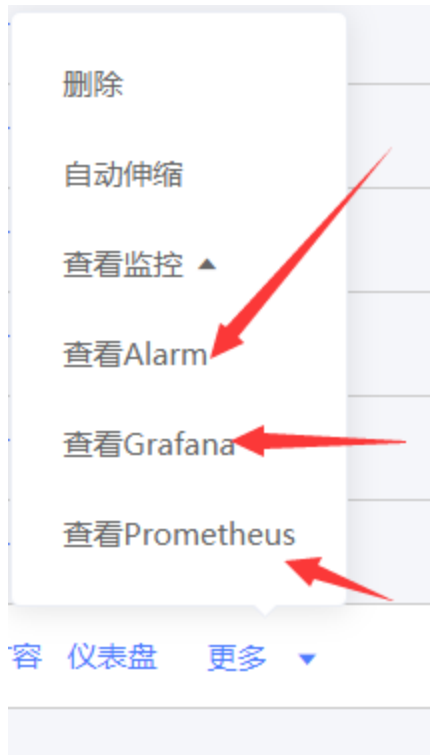
组件状态



- i. **etcd**: 分布式键值存储系统，用于保存集群状态数据，比如Pod，Service等对象信息。
- ii. **Kube-scheduler**：运行在Master节点上，调度器的职责主要是为新创建的pod在集群中寻找最合适的node，并将pod调度到Node上。
- iii. **Kubectl**: kubectl 是 Kubernetes 的命令行工具，是 Kubernetes 用户和管理员必备的管理工具。

(3) 查看详细监控

1. 点击集群的更多按键，之前的开启监控转变为查看监控，如图：



2. 点击查看Grafana可将prometheus监控抓取的监控数据转化为可视化的各类仪表盘进行显示

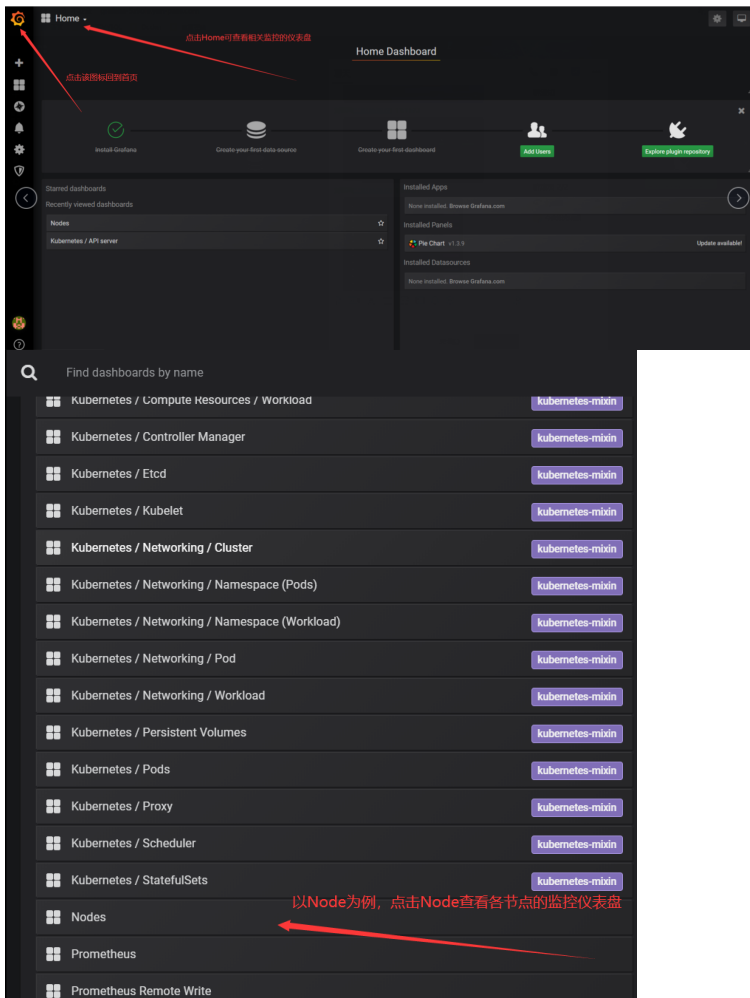
- 点击查看Grafana



- 跳转至Grafana登录首页



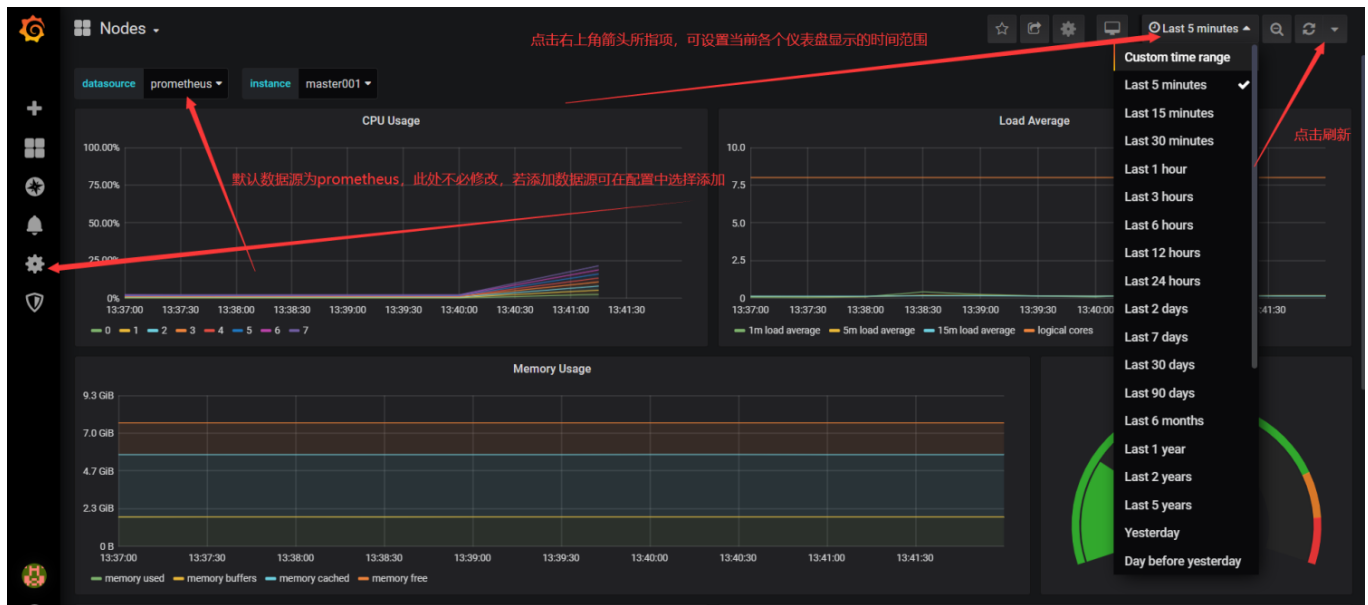
- 登录进首页后，即可查看相关仪表盘（折磨默认配置prometheus数据库作为数据源，无需用户自己配置数据源）



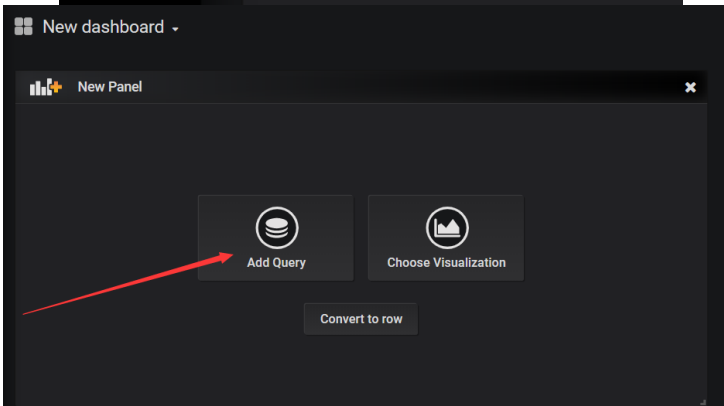
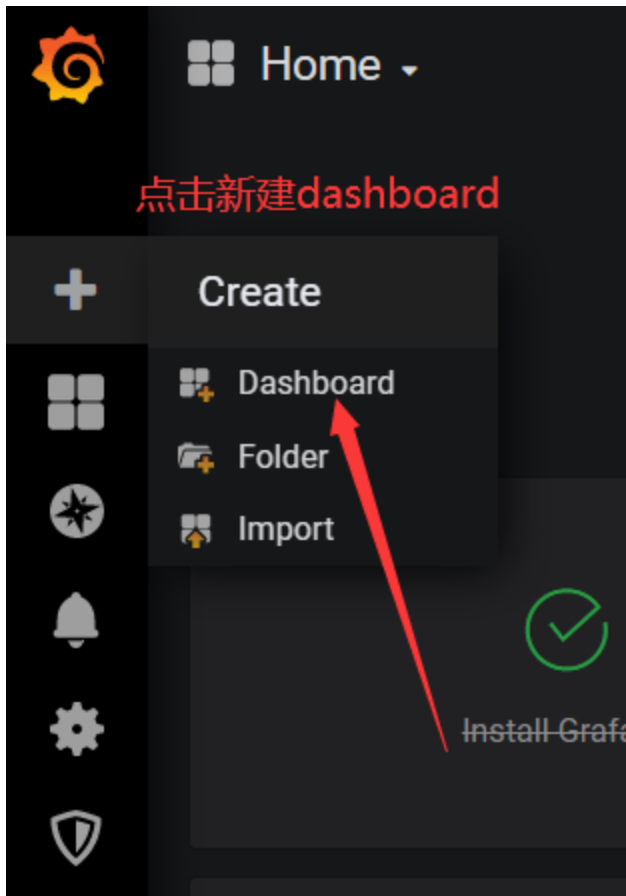
- 点击Nodes查看仪表盘，这里默认数据源已关联至prometheus，用户无需操作，可查看当前集群对应实例



- 左侧导航为Grafana功能栏，功能丰富，可自定义仪表盘等操作。
- 右上角为其他功能，如全屏，分享，打星，设置当前仪表盘显示时间范围，刷新当前仪表盘等功能



- 自定义仪表盘，点击左侧功能导航栏的+，点击Dashboard,页面转换为下图右侧NewPanel



- 点击上图右侧中的AddQuery,跳转至设置仪表盘主界面

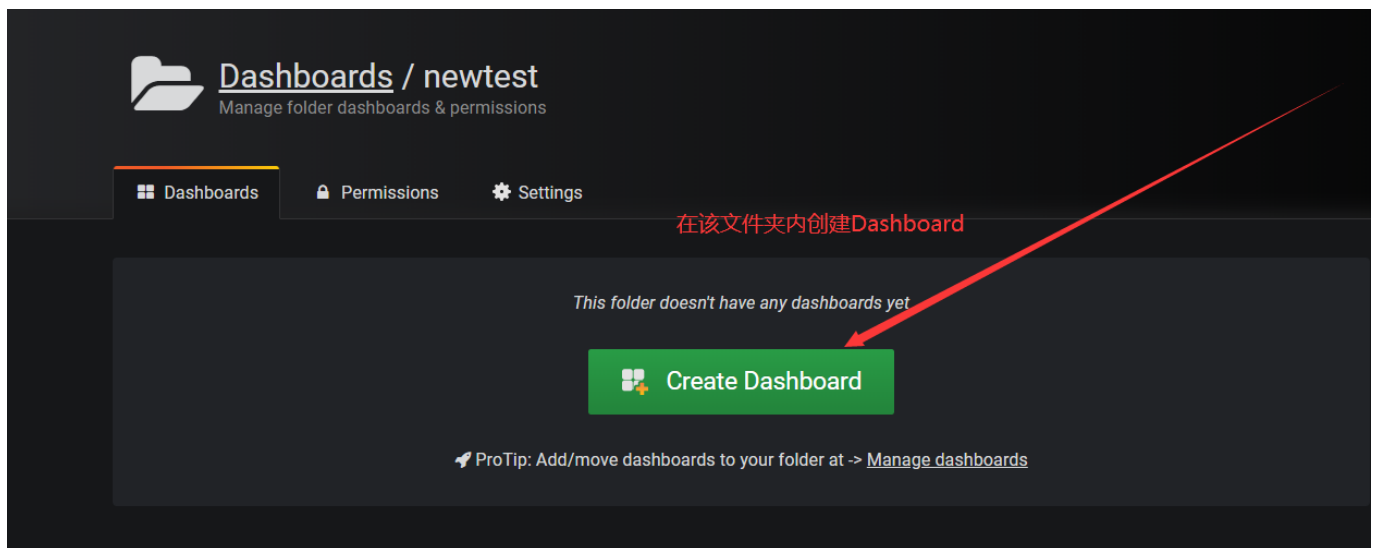
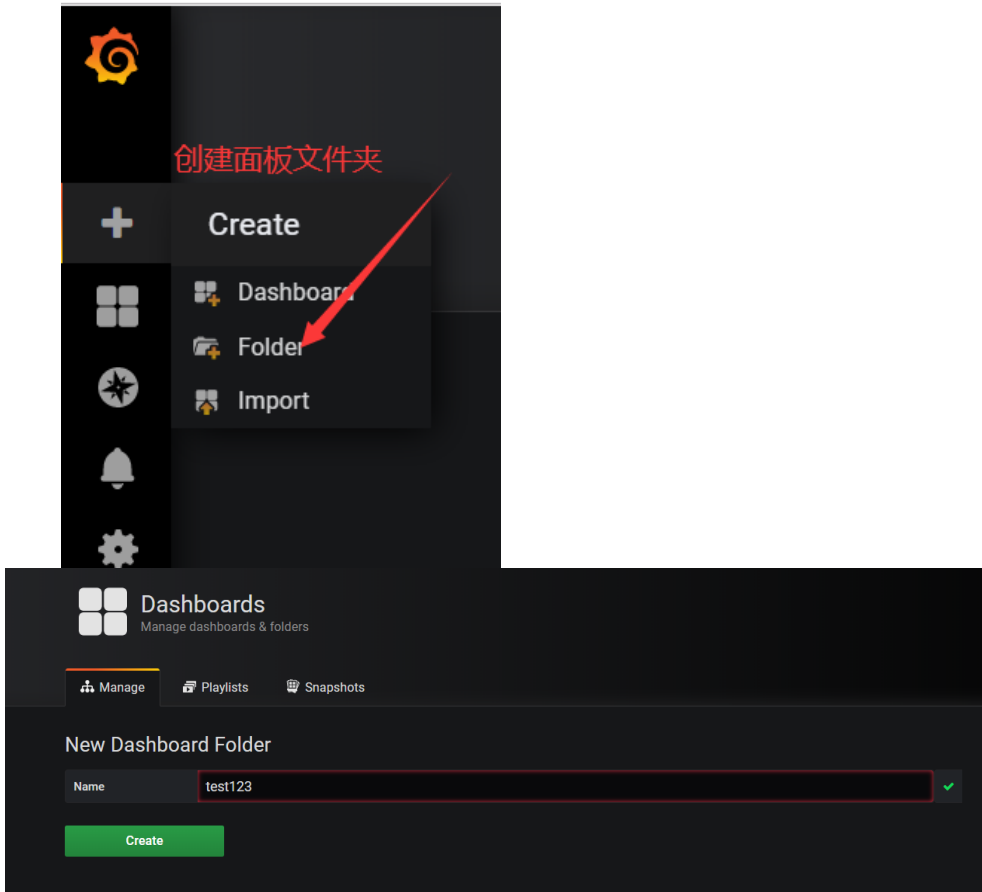


上图中查询数据源选择prometheus，Metrics里输入PormQL表达式（为prometheus监控系统中用于检索监控数据的查询语言），还可以设置其他比如图例，抓取时间等。可以添加多个查询，点击上图右侧AddQuery，操作同上。在通用配置里可以设置该仪表盘名称，最后点击右上方左侧功能按钮保存。

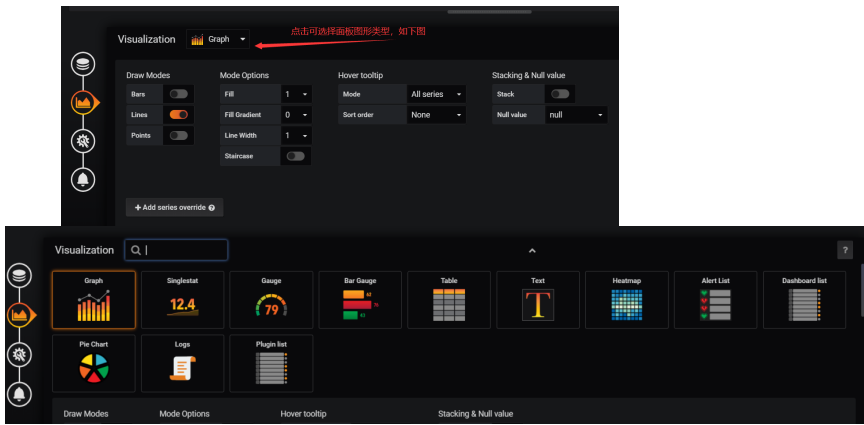
- 如图为上述步骤创建的简单仪表盘



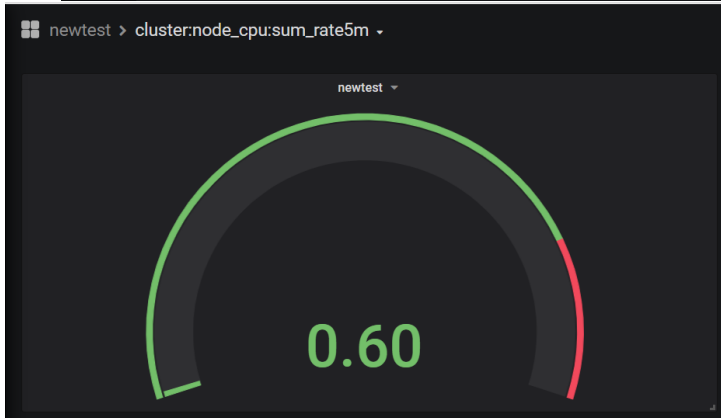
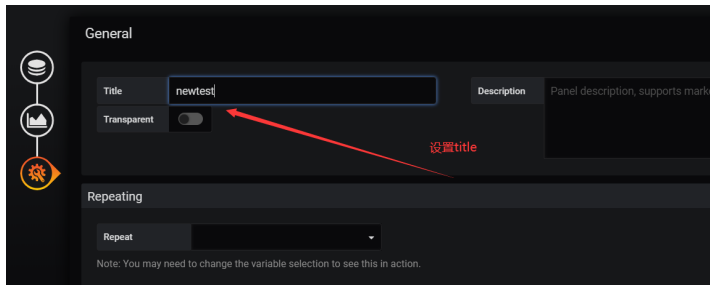
- 方便仪表盘管理，用户可创建文件夹，在该文件夹下创建仪表盘



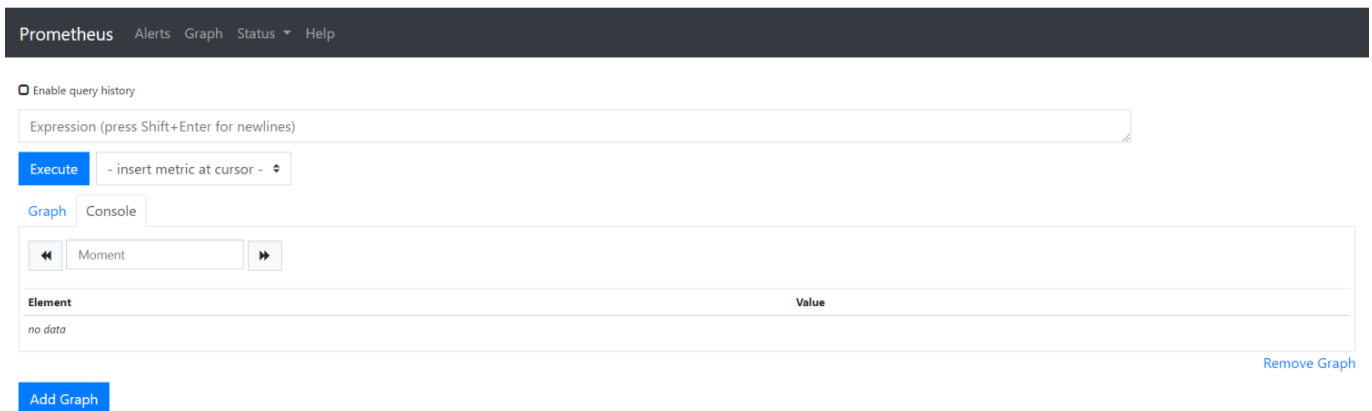
步骤前面自定义仪表盘相同
选择其他样式的仪表盘



这里以Gauge类型仪表盘为例，设置完成后如下图右侧



3. 点击查看prometheus跳转至prometheus监控主页



4. 点击查看Alarm查看Alertmanage报警主页

Filter Group Receiver: All Silenced Inhibited

+ Silence

Custom matcher, e.g. `env="production"`

+ Expand all groups

- + Not grouped 1 alert
- + namespace="monitoring" + 3 alerts
- + namespace="test" + 5 alerts

命名空间管理

1.简介

在 Kubernetes 集群中，您可使用 Namespaces（命名空间）功能创建多个虚拟的空间，在集群用户数量较多时，多个命名空间可以有效划分工作区间，将集群资源划分为多个用途。

2.操作说明

- 创建命名空间

创建命名空间

名称:

长度为1-63个字符，只能包含数字、字母、和“-”，且首尾只能是字母或数字

确认 取消

命名空间名称需唯一，长度为1-63个字符，只能包含数字、字母、和“-”，且首尾只能是字母或数字

- 删除命名空间

若命名空间下有资源，不允许删除命名空间

弹性伸缩

首云容器服务的自动伸缩能力是通过节点自动伸缩组件实现的，可以按需弹出实例，支持多实例规格、多种伸缩模式，满足不同的节点伸缩场景。

1. 工作原理

节点自动伸缩组件是基于kubernetes资源调度的分配情况进行伸缩判断的，节点中资源的分配是通过资源请求（Request）进行计算的。当Pod由于资源请求（Request）无法满足并进入等待（Pending）状态时，节点自动伸缩组件会根据配置的弹性伸缩组配置信息中的资源规格以及约束配置，计算所需的节点数目，如果可以满足伸缩条件，则会触发伸缩组的节点加入。当一个节点在弹性伸缩组中且节点上Pod的资源请求低于阈值时，节点自动伸缩组件会将节点进行缩容。因此资源请求（Request）的正确、合理设置，是弹性伸缩的前提条件。

2. 创建自动伸缩

1. 登录首云控制台
2. 在容器集群菜单下，单击左侧导航栏的集群 > 集群，进入集群列表页面。
3. 选择所需的集群并单击操作列的更多 > 自动伸缩。



4. 在创建自动伸缩配置页面，填写以下信息，并单击提交。

The dialog box '创建自动伸缩配置' contains the following fields and controls:

- 集群: A dropdown menu.
- 缩容阈值: A numeric input field set to '50' followed by a '%' sign and up/down arrows.
- 缩容触发时延: A numeric input field set to '10' followed by '分钟' and up/down arrows.
- 静默时间: A numeric input field set to '10' followed by '分钟' and up/down arrows.
- AccessKeyId: An empty text input field.
- AccessKeySecret: An empty text input field.

Below the fields is a note: '请到“用户中心-用户安全-密钥管理”中查看AccessKeyId和AccessKeySecret'. At the bottom are two buttons: '提交' (Submit) and '取消' (Cancel).

- 缩容阈值：cluster-autoscaler管理的伸缩组中，每一个节点的资源申请值（Request） / 每一个节点的资源容量。当低于配置的阈值时，节点会进行缩容。
- 缩容触发时延：集群满足配置的缩容阈值时，在配置的缩容触发时延到达后，集群开始缩容。单位：分钟。默认情况下是10分钟。
- 静默时间：在集群删除节点后，在静默时间内，集群不会再次触发缩容，单位：分钟。默认情况下是10分钟。
- AccessKeyId:首云用户key id
- AccessKeySecret：首云用户key 密钥

5.提交成功后出现如下页面，可在该页面中点击**修改**，修改**缩容阈值**、**缩容触发时延**和**静默时间**。

基本信息		
集群ID: [模糊]	集群名称: [模糊]	伸缩状态: ● 正常
缩容阈值: 50%	缩容触发时延: 10分钟	静默时间: 10分钟

伸缩组	实例规格	运行中	最小实例数	最大实例数	加入中	移除中	操作
+ 创建伸缩组							

注意：自动伸缩配置完成后需立即配置伸缩组，否则自动伸缩不生效！

6.创建伸缩组

在当前页面点击**创建伸缩组**，在弹框中进行创建伸缩组配置。

节点类型 Worker

计费方式 按需计费

计算类型与规格

8核8G
8核16G
8核32G
8核64G
16核16G

16核32G
16核64G
16核128G

云硬盘类型	容量 (GB)	最高IOPS	操作
性能型	60	600	

Worker还可以添加14块硬盘

添加云硬盘

实例数量

最小实例数	0	^	v	最大实例数	1	^	v
-------	---	---	---	-------	---	---	---

密码(cck用户)

为保证容器集群的稳定及安全，您可以通过cck用户操作集群。

请输入密码

8 - 30 个字符，且同时包含三项（大写字母、小写字母、数字、特殊符号）

请再次输入密码

确认
取消

- 节点类型：默认为worker，即为负载节点。
- 计费方式：同创建集群时所选计费方式。

- 计算类型与规格：伸缩组内实例的规格。
- 本地盘：可为伸缩组内的实例添加外部磁盘。
- 实例数量：伸缩组所包含的实例数量。
- 密码：用户cck登录实例所需的密码。

7. 点击**确认**创建伸缩组。

基本信息		
集群ID: [redacted]	集群名称: [redacted]	伸缩状态: ●更新中
缩容阈值: 50%	缩容触发时延: 10分钟	静默时间: 10分钟

伸缩组	实例规格	运行中	最小实例数	最大实例数	加入中	移除中	操作
[redacted]	8C8G	0	0	1	0	0	修改 删除

● 创建伸缩组

8. 修改伸缩组

在当前页面点击伸缩组右侧**修改**，可对**计算类型与规格**、**本地盘**和**实例数量**进行修改

基本信息		
集群ID: [redacted]	集群名称: [redacted]	伸缩状态: ●正常
缩容阈值: 50%	缩容触发时延: 10分钟	静默时间: 10分钟

伸缩组	实例规格	运行中	最小实例数	最大实例数	加入中	移除中	操作
[redacted]	8C8G	0	0	1	0	0	修改 删除

● 创建伸缩组

9. 删除伸缩组

在当前页面点击伸缩组右侧**删除**，即可删除当前伸缩组。当自动伸缩中没有伸缩组时，自动伸缩不生效！

自动伸缩配置 返回集群列表

基本信息		
集群ID: [redacted]	集群名称: [redacted]-009	伸缩状态: ●正常
缩容阈值: 50%	缩容触发时延: 10分钟	静默时间: 10分钟

伸缩组	实例规格	运行中	最小实例数	最大实例数	加入中	移除中	操作
[redacted]	8C8G	0	0	1	0	0	修改 删除

● 创建伸缩组

10. 启用/禁用自动伸缩

在伸缩组状态为**已禁用**的情况下可启用自动伸缩，启用完成后自动伸缩正常工作。

基本信息			修改	禁用
集群ID: [模糊]	集群名称: [模糊]	伸缩状态: ● 正常		
缩容阈值: 50%	缩容触发时延: 10分钟	静默时间: 10分钟		

伸缩组	实例规格	运行中	最小实例数	最大实例数	加入中	移除中	操作
[模糊]	8C8G	0	0	1	0	0	修改 删除

[创建伸缩组](#)

在伸缩组状态为已禁用的情况下可启用自动伸缩，启用完成后自动伸缩正常工作。

案例--如何创建一个Nginx

1. 创建无状态应用nginx

- 打开无状态应用界面，点击右上角创建

test328-lsg / test / 无状态 (Deployment) 创建

请输入名称查询

- 填写应用基本信息

创建无状态 (Deployment) 返回无状态列表

① 基本信息 ② 容器配置 ③ 高级配置

应用名称 * 只支持小写字母、数字、“-”和“_”，开头和结尾必须是小写字母且长度不超过60个字符

集群 *

命名空间 *

副本数量 * 必填项

+添加

标签键 *	值 *	操作
<input type="text" value="appname"/>	<input type="text" value="nginx"/>	删除

注解 +添加 可选项

下一步

2. 配置容器

- 配置容器基本信息

1 基本信息

2 容器配置

3 高级配置

容器1 添加容器

镜像名称 * 输入镜像名称为nginx

镜像版本 输入镜像版本, 不填写则默认为latest, 这里取默认值

最小申请 CPU Core 内存 MiB

最大限制 CPU Core 内存 MiB 配置容器的期望信息

容器启动项 stdin tty

Init Container

+添加

协议 *	容器端口 *	操作
TCP	80	删除

设置nginx容器端口为 80

开启

选择健康检查方式为tcp

端口 * 设置健康检查端口为80, 此端口要与上面容器端口一致

健康检查

延迟探测频率 (秒)	<input type="text" value="3"/>
执行探测频率 (秒)	<input type="text" value="10"/>
超时时间 (秒)	<input type="text" value="1"/>
不健康阈值	<input type="text" value="3"/>

这里取默认值, 也可自定义设置

就绪检查 开启

操作同健康检查一致, 可选操作 (可以不设置)
注: 初始化容器选项与就绪检查不可同时设置, 只能二者选其一或均不设置

• 设置生命周期

开启

生命周期

启动执行 命令

参数

启动后处理 命令 可选, 设置启动后进入容器创建hello目录

停止前处理 命令

• 设置健康检查, 就绪检查 (可选)

开启

选择健康检查方式为tcp

健康检查
 端口 * 设置健康检查端口为80，此端口要与上面容器端口一致
 延迟探测频率 (秒)
 执行探测频率 (秒) 这里取默认值，也可自定义设置
 超时时间 (秒)
 不健康阈值

就绪检查 开启
操作同健康检查一致，可选操作（可以不设置）
 注：初始化容器选项与就绪检查不可同时设置，只能二者选其一或均不设置

• 挂载数据卷（可选）

数据卷

存储卷类型 *	挂载源 *	存储路径 *	操作
云存储	test328pvc	/nfsshare/data/nginx	删除

数据卷挂载，可选

选择提前创建好的pvc 自定义一个存储路径

3. 高级配置

• 设置水平伸缩策略和升级方式

开启

水平伸缩
 指标 * CPU使用量
 触发条件 * 使用量 %
 最大副本数 * 可选范围: 2-100
 最小副本数 * 可选范围: 1-100

升级方式
 开启
 滚动升级 替换升级
 不可使用pod最大数量 * %
 超过期望的Pod数量 * %

4. 创建成功

• 查看容器启动状态

test328-lsg / test / 无状态 (Deployment) 创建

请输入名称查询 刷新

名称	容器数量	镜像	创建时间	操作
nginx	2/3	nginx:latest	2020-03-28 23:06:13	详情 伸缩 编辑 移除

< 1 >

- 点击编辑 可查看或者编辑 当前应用的yaml文件

编辑
✕

```

apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
    deployment.kubernetes.io/revision: '1'
  creationTimestamp: '2020-03-28T14:25:57Z'
  generation: 36
  labels:
    appname: nginx
  name: nginx
  namespace: test
  resourceVersion: '121867'
  selfLink: /apis/apps/v1/namespaces/test/deployments/nginx
  uid: 089a3556-d6fc-4e99-aa8f-3a99b1701b02

```

确认

取消

- 在节点列表进入master控制台可查看

1. 输入命令 `kubectl get pods -n test`

```

[root@master001 ~]# kubectl get pods -n test
NAME                READY STATUS      RESTARTS AGE
nginx-6b9966bb86-ffd7 1/1   Running      0      8m1s
nginx-6b9966bb86-klgg2 0/1   ContainerCreating 0       0s
nginx-6b9966bb86-p6n82 1/1   Running      0      8m1s

```

2. 输入命令 `kubectl --namespace=test exec -it nginx-6b9966bb86-p6n82 sh` 进入容器

```
[root@master001 ~]# kubectl --namespace=test exec -it nginx-6b9966bb86-p6n82 sh
# ls
bin dev hello lib media nfsshare proc run srv tmp var
boot etc home lib64 mnt opt root sbin sys usr
```

可以看到容器内成功创建hello目录

案例--如何创建一个WordPress

方法一：nodePort + Haproxy 外网访问 WordPress 配置方法

1. 创建 StorageClass、Namespace 和 pvc 资源

- 创建 StorageClass

- 创建命令

```
1 $ kubectl create -f sc.yaml
2 $ kubectl get sc
3 NAME                                PROVISIONER          AGE
4 wordpress-sc-001                   cds/nas              30m
```

- sc.yaml 内容如下

```
1 apiVersion: storage.k8s.io/v1
2 kind: StorageClass
3 metadata:
4   name: wordpress-sc-001
5 provisioner: nas.csi.cds.net
6 parameters:
7   archiveOnDelete: "false"
8   server: "10.10.10.12"           # 替换为集群配置的存储NAS云盘ip地址
9   path: "/nfsshare/wordpress"   # 目录可自行修改为 /nfssahre/<dirName>
10  vers: "4.0"
11  mode: "777"
12 reclaimPolicy: "Delete"
```

- 创建Namespace

- 创建命令

```
1 $ kubectl create -f wordpress-storageclass-pvc.yaml
```

- wordpress-storageclass-pvc.yaml 内容如下

```
1 apiVersion: v1
2 kind: Namespace
3 metadata:
4   name: wordpress
5   labels:
6     app: wordpress
7 ---
8 apiVersion: v1
9 kind: PersistentVolumeClaim
10 metadata:
11   name: mysql-pv-claim
12   namespace: wordpress
13   labels:
14     app: wordpress
15 spec:
16   accessModes:
17     - ReadWriteOnce
18   resources:
19     requests:
20       storage: 20Gi
21   storageClassName: wordpress-sc-001
22 ---
23 apiVersion: v1
24 kind: PersistentVolumeClaim
25 metadata:
26   name: wp-pv-claim
27   namespace: wordpress
28   labels:
29     app: wordpress
30 spec:
31   accessModes:
32     - ReadWriteOnce
33   resources:
34     requests:
35       storage: 20Gi
36   storageClassName: wordpress-sc-001
```

2. 部署MySQL容器组

- 创建一个Secret变量存放MySQL密码

- 创建命令

```
1 $ kubectl create secret generic mysql-pass --from-literal=password=<YOUR_PASSWORD> -n <NAMESPACE>
```

- 检查创建结果

```
1 $ kubectl get secret -n wordpress
2 NAME                                TYPE                                DATA  AGE
3 default-token-5k6fs                 kubernetes.io/service-account-token  3      43m
4 mysql-pass                           Opaque                              1      41m
```

- 部署MySQL容器

- 部署命令

```
1 $ kubectl create -f mysql-deployment.yaml
```

- mysql-deployment.yaml 内容如下

```
1 apiVersion: v1
2 kind: Service
3 metadata:
4   name: wordpress-mysql
5   namespace: wordpress
6   labels:
7     app: wordpress
8 spec:
9   ports:
10    - port: 3306
11   selector:
12     app: wordpress
13     tier: mysql
14   clusterIP: None
15 ---
16 apiVersion: v1
17 kind: PersistentVolumeClaim
18 metadata:
19   name: mysql-pv-claim
20   labels:
21     app: wordpress
22 spec:
23   accessModes:
```

```
24     - ReadWriteOnce
25   resources:
26     requests:
27       storage: 20Gi
28   ---
29   apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
30   kind: Deployment
31   metadata:
32     name: wordpress-mysql
33     namespace: wordpress
34     labels:
35       app: wordpress
36   spec:
37     selector:
38       matchLabels:
39         app: wordpress
40         tier: mysql
41     strategy:
42       type: Recreate
43     template:
44       metadata:
45         labels:
46           app: wordpress
47           tier: mysql
48       spec:
49         containers:
50         - image: mysql:5.6
51           name: mysql
52           env:
53             - name: MYSQL_ROOT_PASSWORD
54               valueFrom:
55                 secretKeyRef:
56                   name: mysql-pass
57                   key: password
58           ports:
59             - containerPort: 3306
60               name: mysql
61         volumeMounts:
62         - name: mysql-persistent-storage
63           mountPath: /var/lib/mysql
```

```
64     volumes:
65     - name: mysql-persistent-storage
66       persistentVolumeClaim:
67         claimName: mysql-pv-claim
```

- 检查MySQL部署情况

```
1 $ kubectl get deployment -n wordpress
2 NAME                READY   UP-TO-DATE   AVAILABLE   AGE
3 wordpress-mysql    1/1     1             1           39m
4 $ kubectl get pods -n wordpress
5 NAME                READY   STATUS    RESTARTS   AGE
6 wordpress-mysql-5b697dbbfc-5rwr1  1/1     Running   0           39m
```

3. 部署WordPress容器组

- 部署命令

```
1 $ kubectl create -f wordpress-deployment.yaml
```

- wordpress-deployment.yaml 内容如下

```
1 apiVersion: v1
2 kind: Service
3 metadata:
4   name: wordpress
5   namespace: wordpress
6   labels:
7     app: wordpress
8 spec:
9   type: NodePort                # 设置 NodePort 方式访问 Wordpress
10  ports:
11    - port: 80
12      targetPort: 80
13      nodePort: 30080            # 设置映射端口为 30080
14  selector:
15    app: wordpress
16    tier: frontend
17 ---
18 apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
19 kind: Deployment
20 metadata:
```



```
21 name: wordpress
22 namespace: wordpress
23 labels:
24   app: wordpress
25 spec:
26   selector:
27     matchLabels:
28       app: wordpress
29       tier: frontend
30   strategy:
31     type: Recreate
32   template:
33     metadata:
34       labels:
35         app: wordpress
36         tier: frontend
37     spec:
38       containers:
39       - image: wordpress:4.8-apache
40         name: wordpress
41         env:
42         - name: WORDPRESS_DB_HOST
43           value: wordpress-mysql
44         - name: WORDPRESS_DB_PASSWORD
45           valueFrom:
46             secretKeyRef:
47               name: mysql-pass
48               key: password
49         ports:
50         - containerPort: 80
51           name: wordpress
52         volumeMounts:
53         - name: wordpress-persistent-storage
54           mountPath: /var/www/html
55         volumes:
56         - name: wordpress-persistent-storage
57           persistentVolumeClaim:
58             claimName: wp-pv-claim
```

- 部署结果检查

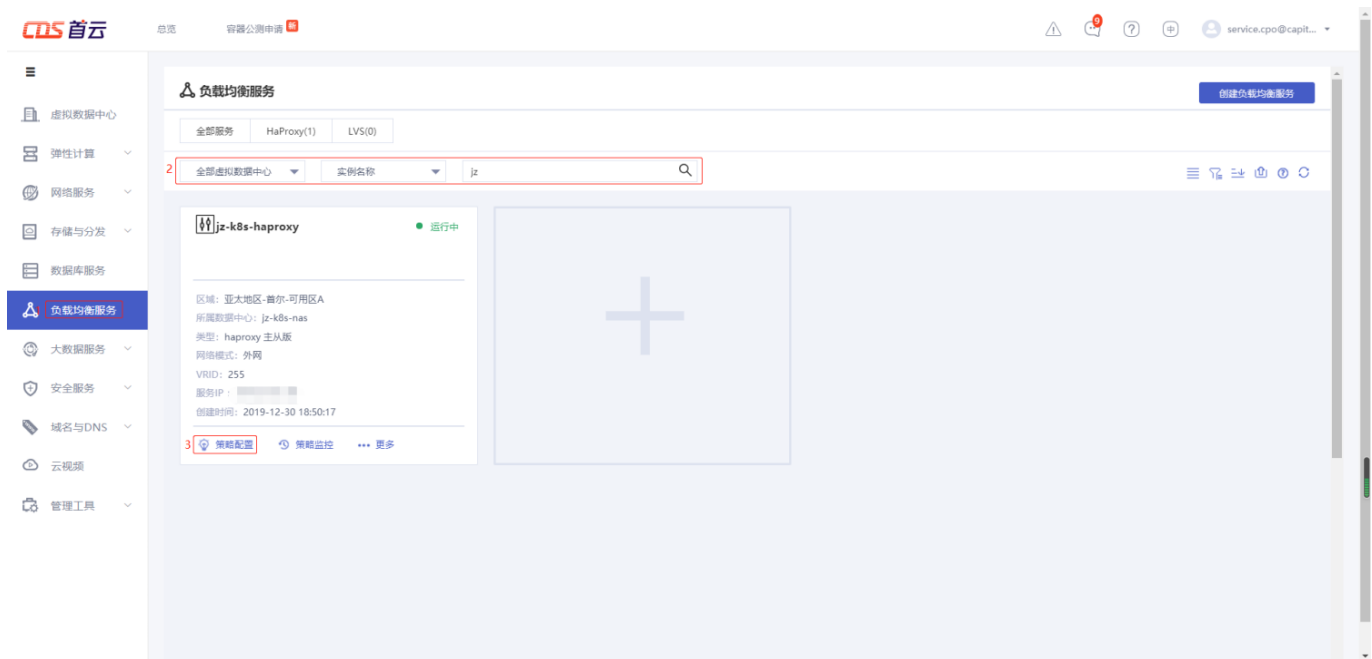
```

1 $ kubectl get deployment -n wordpress
2 NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
3 wordpress                            1/1     1             1           41m
4 wordpress-mysql                       1/1     1             1           44m
5 $ kubectl get pods -n wordpress
6 NAME                                READY   STATUS    RESTARTS   AGE
7 wordpress-578744754c-1hq5d           1/1     Running   0           41m
8 wordpress-mysql-5b697dbbfc-5rwr1    1/1     Running   0           44m

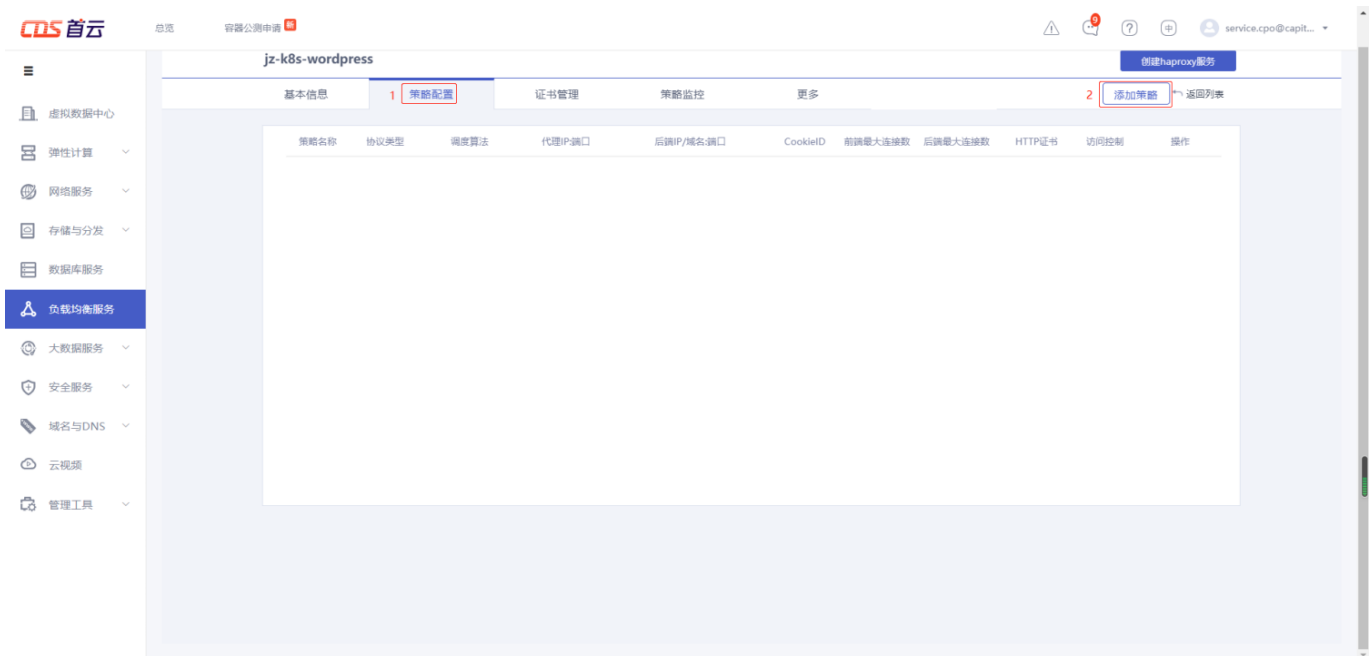
```

4. 配置 Haproxy 负载均衡

- 复用部署 WordPress 集群使用的 Haproxy -> 点击策略配置



- 点击添加策略



- 填写 Haproxy 策略名称 -> 选择 HTTP 类型 -> 填写上述暴露的 NodePort -> 选择负载均衡算法 -> 添加服务器（集群的工作节点） -> 点击确认

备注：将上述暴露的 NodePort：30080 端口在 Haproxy 策略中打开，这样可以通过 HaproxyIP:NodePort 即可访问集群内的服务，同时实现负载均衡

- 虚拟数据中心
- 弹性计算
- 网络服务
- 存储与分发
- 数据库服务
- 负载均衡服务**
- 大数据服务
- 安全服务
- 域名与DNS
- 云视频
- 管理工具

创建haproxy服务
添加策略
返回列表

基本信息
策略配置
证书管理
策略监控
更多

策略配置名称: 最多输入十五个字符, 由英文、数字及_组成, 并且以英文开头

协议类型: TCP HTTP

代理端口: 端口输入范围为1-65535, 22和1080端口已被使用。

调度算法: 简单轮询, 每个服务器根据权重轮流使用。此算法是动态的, 对于实例启动慢的服务器权重会在运行中调整。

最大连接数: 支持自定义, 所有策略最大连接数总和不能超过50000, 超过50000后, 会出现丢失的情况。

HTTP证书: 开启证书验证功能, 所有应用请求通过证书验证后在进行连接

Cookie头: 启用后端服务器cookie的持久连接功能

Keep-Alive: Keep-Alive功能使客户端到服务器的连接持续有效, 当出现对服务器的后续请求时, Keep-Alive功能避免了建立或者重新建立连接。

访问控制:

获取源IP: 默认已开启 后端服务器配置, [点击查看帮助文档](#)

Option设置:

httplog 启用记录HTTP请求、会话状态和计时的功能。

forwardfor 允许在发往服务器的请求首部中插入 "X-Forwarded-For" 首部。

abortonclose 当服务器负载过高时, 将自动关闭队列中处理时间较长的连接请求。

httppchk 开启健康检查并设置心跳检测的URL, 允许用http协议检查后端服务器的健康状态

连接超时时间: 设置请求连接超时的时间

时间设置:

客户端超时: 设置客户端连接超时的时间

服务器超时: 设置服务器端连接超时的时间

序号	IP地址/域名	端口	连接数	权重	操作
1	██████████	30080	2000	1	修改 删除
2	██████████	30080	2000	1	修改 删除
3	██████████	30080	2000	1	修改 删除
4	██████████	30080	2000	1	修改 删除
5	██████████	30080	2000	1	修改 删除
6	██████████	30080	2000	1	修改 删除

添加服务器
增加【或修改】配置会重自负载均衡的服务, 请确认增加【或修改】。
确认
取消

• 查看 Haproxy 服务的内外网 IP 地址

- 虚拟数据中心
- 弹性计算
- 网络服务
- 存储与分发
- 数据库服务
- 负载均衡服务**
- 大数据服务
- 安全服务
- 域名与DNS
- 云视频
- 管理工具

创建haproxy服务
返回列表

1 基本信息
策略配置
证书管理
策略监控
更多

jz-k8s-wordpress 修改

编号: 9a12a387-2126-47f3-aced-eebf57fc1bdc

状态: 运行中 数据中心: jz-k8s-nas

区域: 亚太地区-首尔-可用区A 创建时间: 2020-04-03 09:43:00

运行时间: 0.01天 计费方式: 按需计费

规格配置 修改

计算规格: 1核 | 2G 最大连接数: 50000, 超出50000后, 会出现丢失的情况。

引擎版本: HaProxy 服务高可用: 主从

服务信息

网络类型: 外网 连接服务IP: ██████████

内网IP(主): ██████████ 内网IP(从): ██████████

VRID: 110

10.241.46.7

当前配置信息

类型: HaProxy 主从

规格: 1核 | 2G

总计: ¥4.68/天

5. 访问 WordPress

- 集群中查看映射的 nodePort

```
1 $ kubectl get svc -n wordpress
2 NAME                TYPE                CLUSTER-IP          EXTERNAL-IP          PORT(S)
3 wordpress           NodePort            10.104.56.86        <none>                80:30080/TCP
4 wordpress-mysql     ClusterIP           None                 <none>                3306/TCP
5 $ kubectl get nodes -o wide
6 NAME                STATUS    ROLES    AGE   VERSION   INTERNAL-IP    EXTERNAL-IP
7 master001           Ready    master   87d   v1.16.3   ***
8 master002           Ready    master   87d   v1.16.3   ***
9 master003           Ready    master   87d   v1.16.3   ***
10 worker001          Ready    <none>   87d   v1.16.3   ***
11 worker002          Ready    <none>   87d   v1.16.3   ***
12 worker003          Ready    <none>   87d   v1.16.3   ***
```

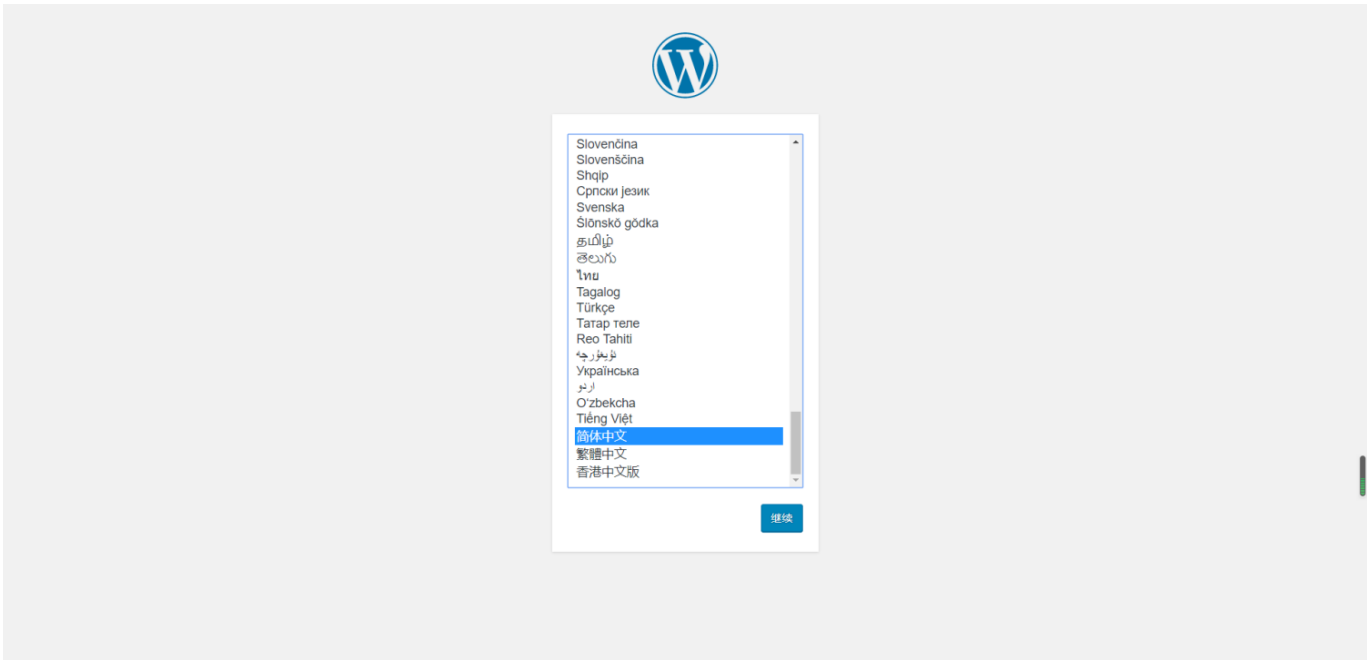
- 本机验证访问（无负载均衡），NodeIP 可以是任意Node节点IP地址，nodePort 为设置的 30080

```
1 $ curl <NodeIP>:<nodePort>
```

- 浏览器访问（使用负载均衡），开始安装WordPress

url = http://:<haproxyIP>:<nodePort>, haproxyIP 为上步创建的负责均衡服务的 IP, nodePort为设置的30080

- 选择语言



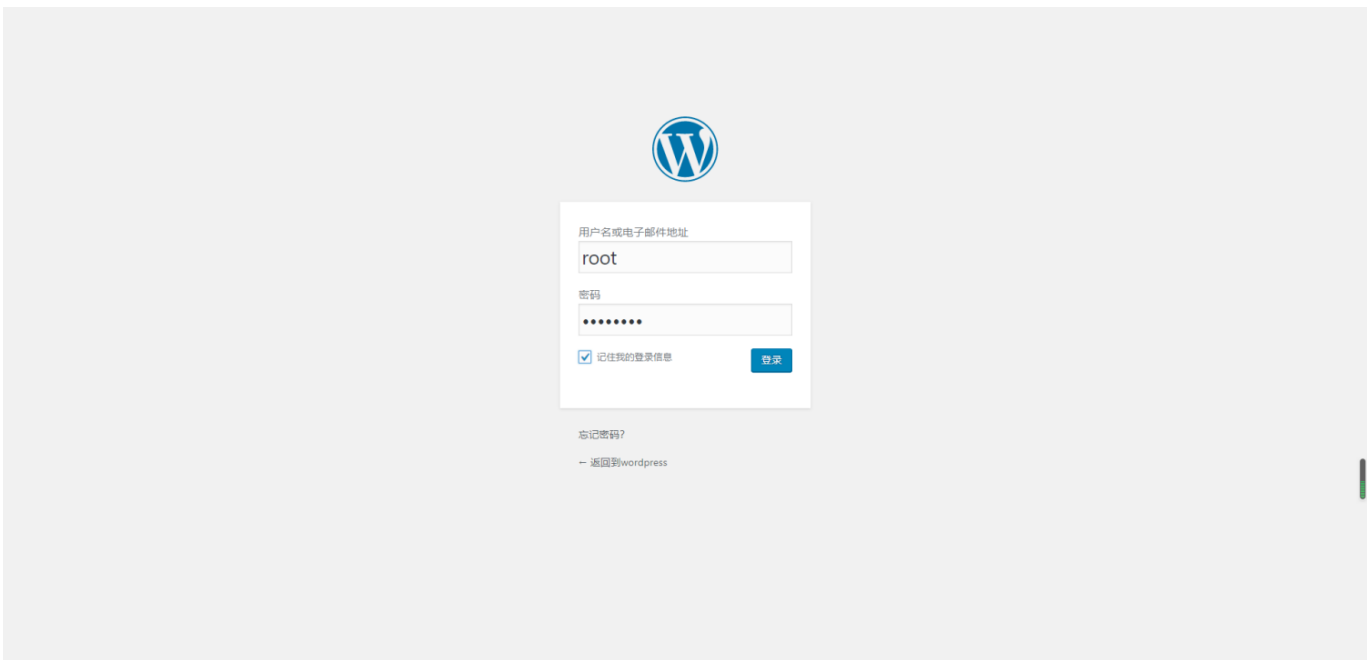
- 填写必要信息（账号密码用于后续登录WordPress）



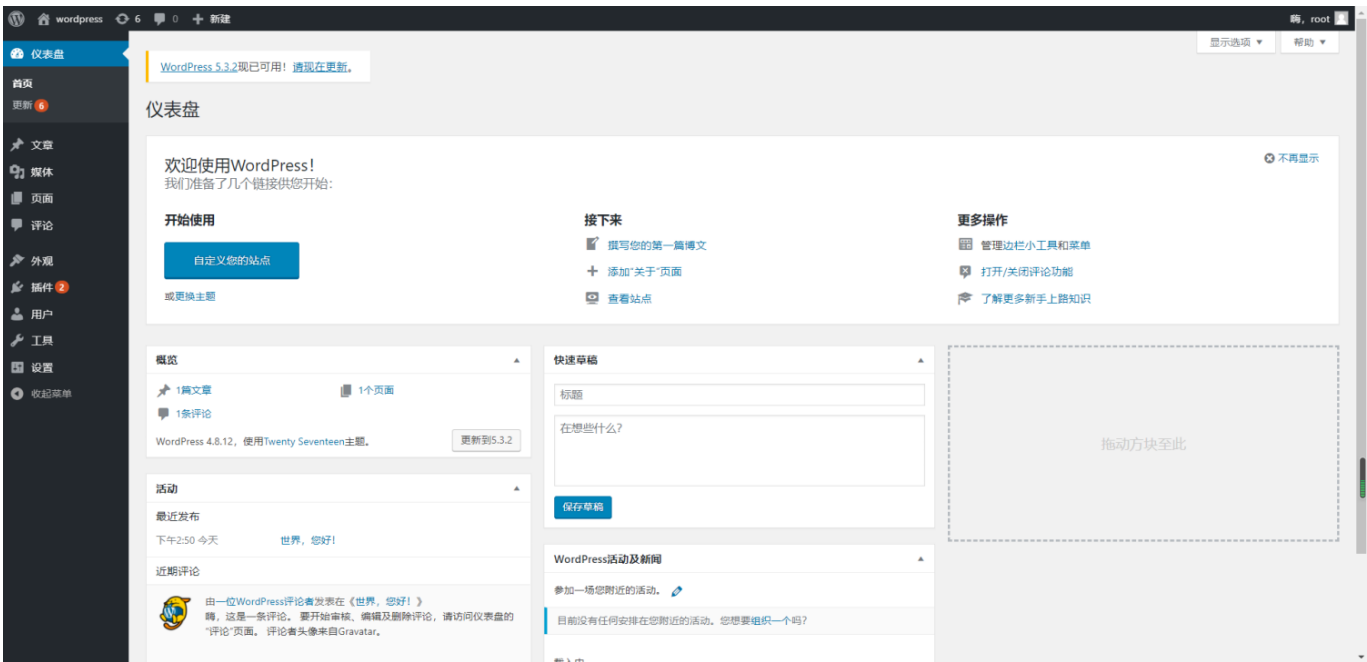
- 设置完成



- 输入之前设置的登录账号和密码，点击登录



- 成功进入WordPress页面



方法二：Ingress 外网访问 WordPress 配置方法

1. 创建 Namespace 和 PVC 资源

- 创建 StorageClass
 - 创建命令

```

1 $ kubectl create -f sc.yaml
2 $ kubectl get sc
3 NAME                                PROVISIONER          AGE
4 wordpress-sc-001                    cds/nas               30m

```

- sc.yaml 内容如下

```

1 apiVersion: storage.k8s.io/v1
2 kind: StorageClass
3 metadata:
4   name: wordpress-sc-001
5 provisioner: nas.csi.cds.net
6 parameters:
7   archiveOnDelete: "false"
8   server: "10.10.10.12"           # 替换为集群配置的存储NAS云盘ip地址
9   path: "/nfsshare/wordpress"   # 目录可自行修改为 /nfssahre/<dirName>
10  vers: "4.0"
11  mode: "777"

```



```
12 reclaimPolicy: "Delete"
```

- 创建Namespace

- 创建命令

```
1 $ kubectl create -f wordpress-storageclass-pvc.yaml
```

- wordpress-storageclass-pvc.yaml 内容如下

```
1 apiVersion: v1
2 kind: Namespace
3 metadata:
4   name: wordpress
5   labels:
6     app: wordpress
7 ---
8 apiVersion: v1
9 kind: PersistentVolumeClaim
10 metadata:
11   name: mysql-pv-claim
12   namespace: wordpress
13   labels:
14     app: wordpress
15 spec:
16   accessModes:
17     - ReadWriteOnce
18   resources:
19     requests:
20       storage: 20Gi
21   storageClassName: wordpress-sc-001
22 ---
23 apiVersion: v1
24 kind: PersistentVolumeClaim
25 metadata:
26   name: wp-pv-claim
27   namespace: wordpress
28   labels:
29     app: wordpress
30 spec:
31   accessModes:
32     - ReadWriteOnce
```

```
33 resources:
34   requests:
35     storage: 20Gi
36   storageClassName: wordpress-sc-001
```

2. 部署MySQL容器组

- 创建一个Secret变量存放MySQL密码
 - 创建命令

```
1 $ kubectl create secret generic mysql-pass --from-literal=password=<YOUR_PASSWORD> -n <NAMESPACE>
```

- 检查创建结果

```
1 $ kubectl get secret -n wordpress
2 NAME                                TYPE                                DATA  AGE
3 default-token-5k6fs                 kubernetes.io/service-account-token  3      43m
4 mysql-pass                           Opaque                              1      41m
```

- 部署MySQL容器
 - 部署命令

```
1 $ kubectl create -f mysql-deployment.yaml
```

- mysql-deployment.yaml 内容如下

```
1 apiVersion: v1
2 kind: Service
3 metadata:
4   name: wordpress-mysql
5   namespace: wordpress
6   labels:
7     app: wordpress
8 spec:
9   ports:
10    - port: 3306
11   selector:
12     app: wordpress
13     tier: mysql
14   clusterIP: None
15 ---
```

```

16 apiVersion: v1
17 kind: PersistentVolumeClaim
18 metadata:
19   name: mysql-pv-claim
20   labels:
21     app: wordpress
22 spec:
23   accessModes:
24     - ReadWriteOnce
25   resources:
26     requests:
27       storage: 20Gi
28 ---
29 apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
30 kind: Deployment
31 metadata:
32   name: wordpress-mysql
33   namespace: wordpress
34   labels:
35     app: wordpress
36 spec:
37   selector:
38     matchLabels:
39       app: wordpress
40       tier: mysql
41   strategy:
42     type: Recreate
43   template:
44     metadata:
45       labels:
46         app: wordpress
47         tier: mysql
48     spec:
49       containers:
50         - image: mysql:5.6
51           name: mysql
52           env:
53             - name: MYSQL_ROOT_PASSWORD
54               valueFrom:
55                 secretKeyRef:

```

```

56         name: mysql-pass
57         key: password
58     ports:
59     - containerPort: 3306
60       name: mysql
61     volumeMounts:
62     - name: mysql-persistent-storage
63       mountPath: /var/lib/mysql
64     volumes:
65     - name: mysql-persistent-storage
66       persistentVolumeClaim:
67         claimName: mysql-pv-claim

```

- 检查MySQL部署情况

```

1 $ kubectl get deployment -n wordpress
2 NAME                READY   UP-TO-DATE   AVAILABLE   AGE
3 wordpress-mysql    1/1     1             1           39m
4 $ kubectl get pods -n wordpress
5 NAME                READY   STATUS    RESTARTS   AGE
6 wordpress-mysql-5b697dbbfc-5rwr1  1/1     Running   0           39m

```

3. 部署WordPress容器组

- 部署命令

```

1 $ kubectl create -f wordpress-deployment.yaml

```

- wordpress-deployment.yaml 内容如下

```

1 apiVersion: v1
2 kind: Service
3 metadata:
4   name: wordpress
5   namespace: wordpress
6   labels:
7     app: wordpress
8 spec:
9   ports:
10    - port: 80
11      targetPort: 80
12   selector:

```

```
13   app: wordpress
14   tier: frontend
15 ---
16 apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
17 kind: Deployment
18 metadata:
19   name: wordpress
20   namespace: wordpress
21   labels:
22     app: wordpress
23 spec:
24   selector:
25     matchLabels:
26       app: wordpress
27       tier: frontend
28   strategy:
29     type: Recreate
30   template:
31     metadata:
32       labels:
33         app: wordpress
34         tier: frontend
35     spec:
36       containers:
37       - image: wordpress:4.8-apache
38         name: wordpress
39         env:
40         - name: WORDPRESS_DB_HOST
41           value: wordpress-mysql
42         - name: WORDPRESS_DB_PASSWORD
43           valueFrom:
44             secretKeyRef:
45               name: mysql-pass
46               key: password
47         ports:
48         - containerPort: 80
49           name: wordpress
50       volumeMounts:
51       - name: wordpress-persistent-storage
52         mountPath: /var/www/html
```

```

53     volumes:
54     - name: wordpress-persistent-storage
55       persistentVolumeClaim:
56         claimName: wp-pv-claim

```

- 部署结果检查

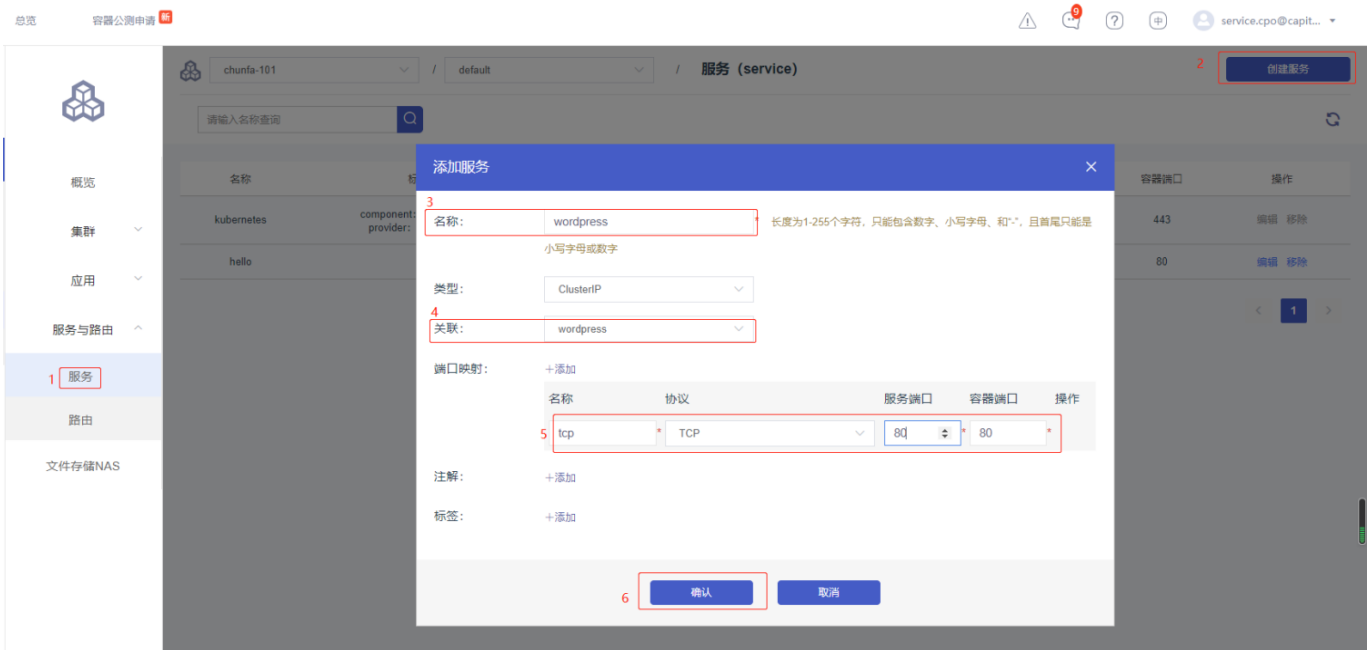
```

1 $ kubectl get svc -n wordpress
2 NAME                                TYPE                CLUSTER-IP          EXTERNAL-IP          PORT(S)              AGE
3 wordpress                            ClusterIP           10.108.144.175     <none>               80/TCP               19s
4 wordpress-mysql                      ClusterIP           None                <none>               3306/TCP             10d
5 $ kubectl get pods -n wordpress
6 NAME                                READY               STATUS              RESTARTS             AGE
7 wordpress-578744754c-6snkd           1/1                Running            0                    30s
8 wordpress-mysql-5b697dbbfc-5rwr1    1/1                Running            0                    10d

```

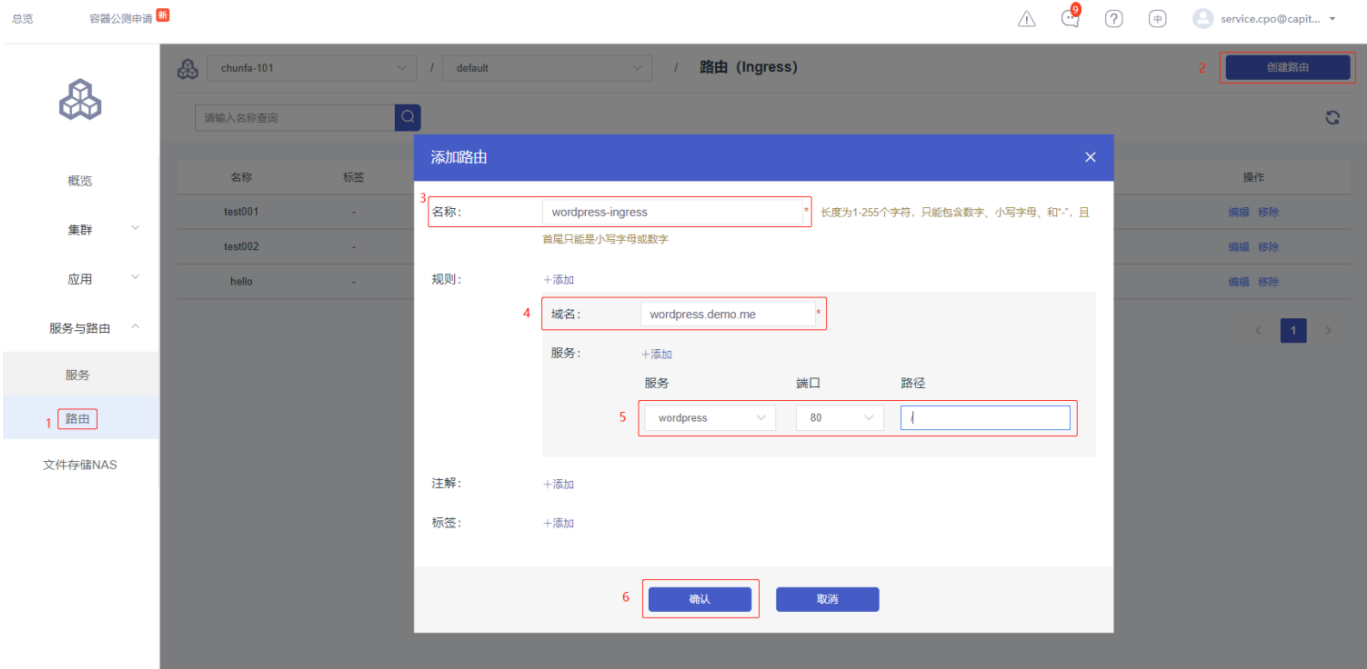
4. 创建 Service

- 选择服务 -> 右上角点击创建 -> 填写名称 -> 选择关联的后端服务 -> 设置映射端口（80） -> 无误后，点击确认



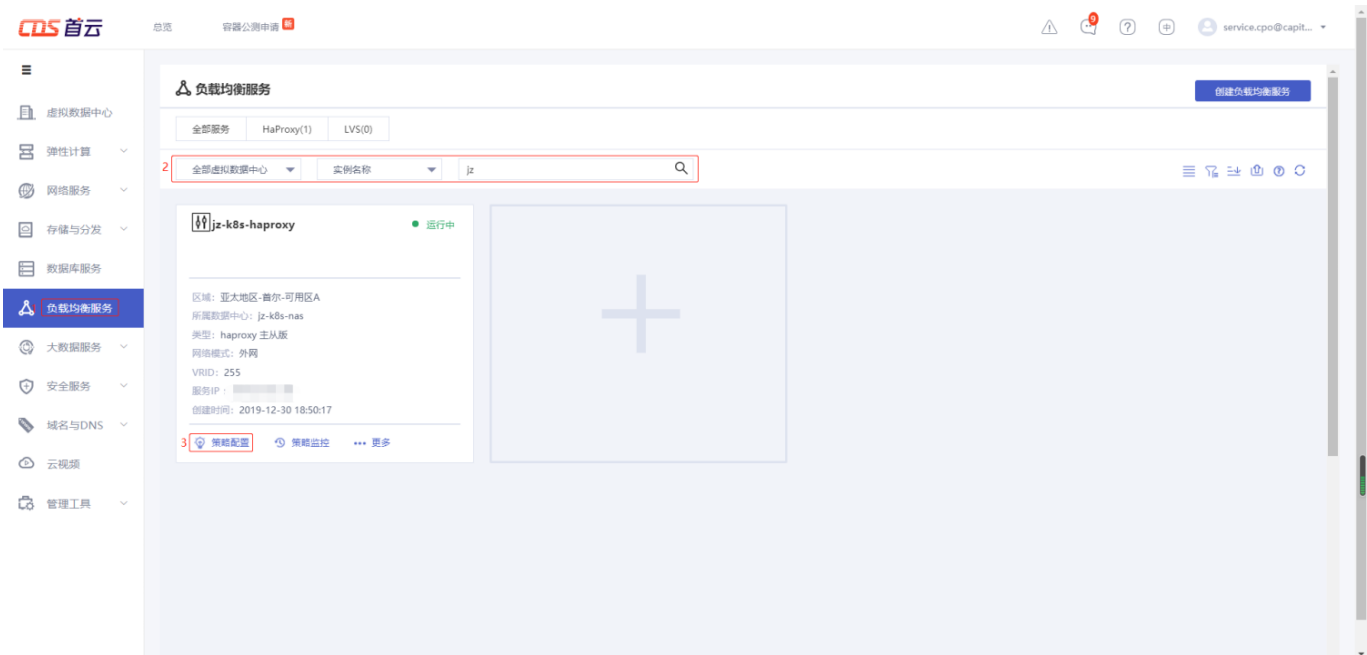
5. 创建 Ingress

- 选择路由 -> 右上角点击创建 -> 填写名称 -> 填写域名 -> 选择关联的服务和端口映射（80） -> 无误后，点击确认

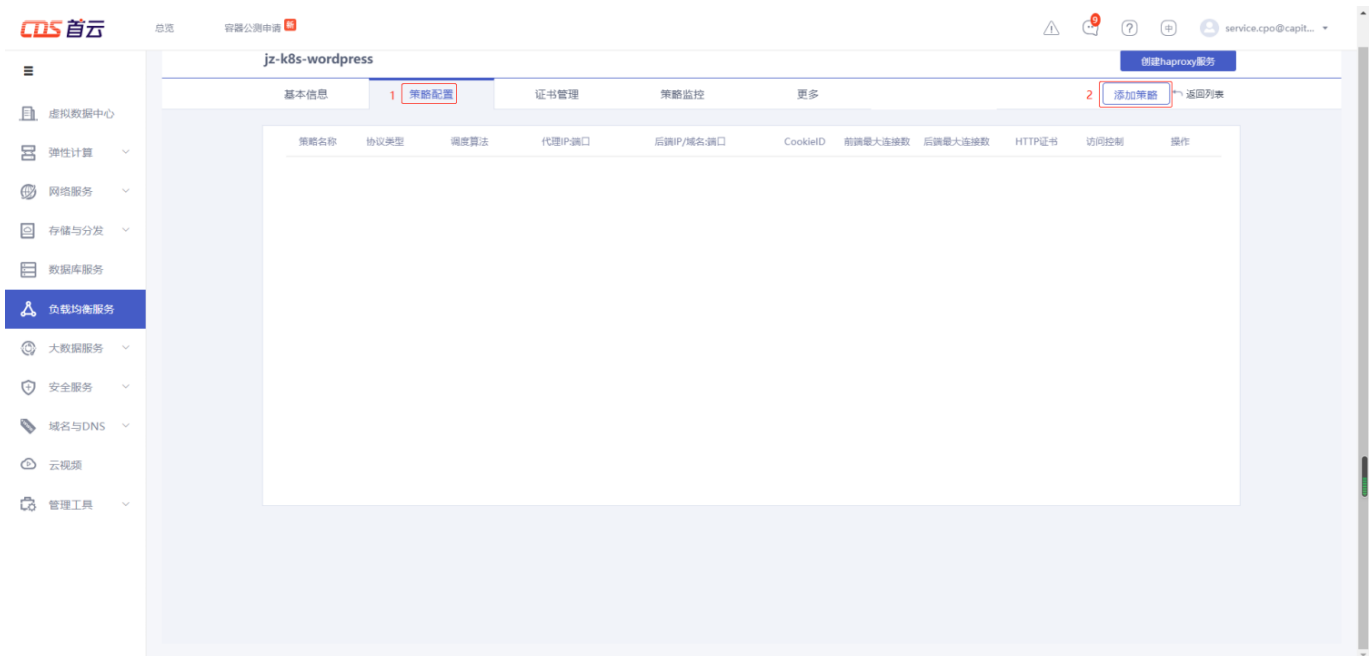


6. Haproxy 策略配置上述 Ingress 与服务映射的 80 端口

- 复用部署 WordPress 集群使用的 Haproxy -> 点击策略配置



- 点击添加策略



- 填写 Haproxy 策略名称 -> 选择 HTTP 类型 -> 填写上述暴露的 NodePort -> 选择负载均衡算法 -> 添加服务器（集群的工作节点） -> 点击确认

- 虚拟数据中心
- 弹性计算
- 网络服务
- 存储与分发
- 数据库服务
- 负载均衡服务**
- 大数据服务
- 安全服务
- 域名与DNS
- 云视频
- 管理工具

创建haproxy服务

基本信息
策略配置
证书管理
策略监控
更多

添加策略 返回列表

策略配置名称: 最多输入十五个字符, 由英文、数字及_组成, 并且以英文开头

协议类型: TCP HTTP

代理端口: 端口输入范围为1-65535, 22和1080端口已被使用。

调度算法: 简单轮询, 每个服务器根据权重轮流使用。此算法是动态的, 对于实例启动慢的服务器权重会在运行中调整。

最大连接数: 支持自定义 所有策略最大连接数总和不能超过50000, 超过50000后, 会出现丢失的情况。

HTTP证书: 开启证书验证功能, 所有应用请求通过证书验证后进行连接

Cookie头: 启用后端服务器cookie的持久连接功能

Keep-Alive: Keep-Alive功能使客户端到服务器端的连接持续有效, 当出现对服务器的后继请求时, Keep-Alive功能避免了建立或者重新建立连接。

访问控制:

获取源IP: 默认已开启 后端服务器配置, [点击查看帮助文档](#)

Option设置:

httplog 启用记录HTTP请求、会话状态和计时器的功能。

forwardfor 允许在发往服务器的请求首部中插入 "X-Forwarded-For" 首部。

abortonclose 当服务器负载过高时, 将自动关闭队列中处理时间较长的连接请求。

httpchk 开启健康检查并设置心跳检测的URL, 允许用http协议检查后端服务器的健康状态

连接超时时间: 设置请求连接超时的时间

时间设置:

客户端超时: 设置客户端连接超时的时间

服务器超时: 设置服务器端连接超时的时间

序号	IP地址/域名	端口	连接数	权重	操作
1	██████████	80	2000	1	修改 删除
2	██████████	80	2000	1	修改 删除
3	██████████	80	2000	1	修改 删除
4	██████████	80	2000	1	修改 删除
5	██████████	80	2000	1	修改 删除
6	██████████	80	2000	1	修改 删除

添加服务器

增加【或修改】配置会重负载均衡的服务, 请确认增加【或修改】。

确认
取消

7. 访问 Wordpress

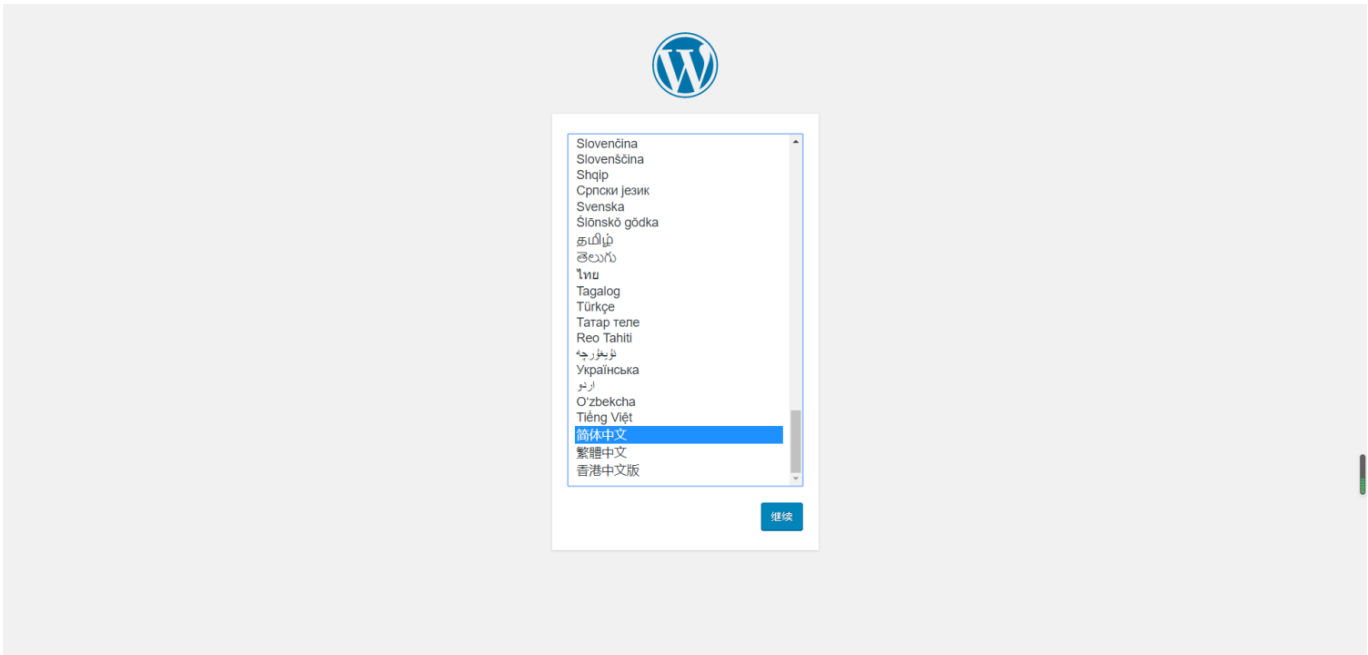
- 查看集群的 Haproxy 服务的外网和内网 IP



- 本机配置 hosts 域名解析

```
1 <HaproxyIP> wordpress.demo.me
```

- 浏览器访问：http://wordpress.demo.me，开始安装 WordPress



- 后续安装步骤请参照方法一中的安装步骤即可