K8S学习教程(二):在 PetaExpress KubeSphere容器平台部署高可用 Redis 集群

前言

`Redis` 是在开发过程中经常用到的**缓存中间件**，为了考虑在生产环境中**稳定性**和**高可用**，Redis通常采用集群模式的部署方式。

在制定Redis集群的部署策略时，常规部署在虚拟机上的方式配置繁琐并且需要手动重启节点，相较之下，使用 `PetaExpress 提供的 Kubernetes（k8s） 服务` 进行 `Redis` 集群的部署，则展现出了显著的优势：

1、安装便捷：使用镜像或者 `yaml` 配置文件即可一件安装，极大地简化了安装流程

2、缩扩容方便：在 `扩容`、`缩容` 方面的优点一键伸缩，无需复杂的配置和繁琐的步骤

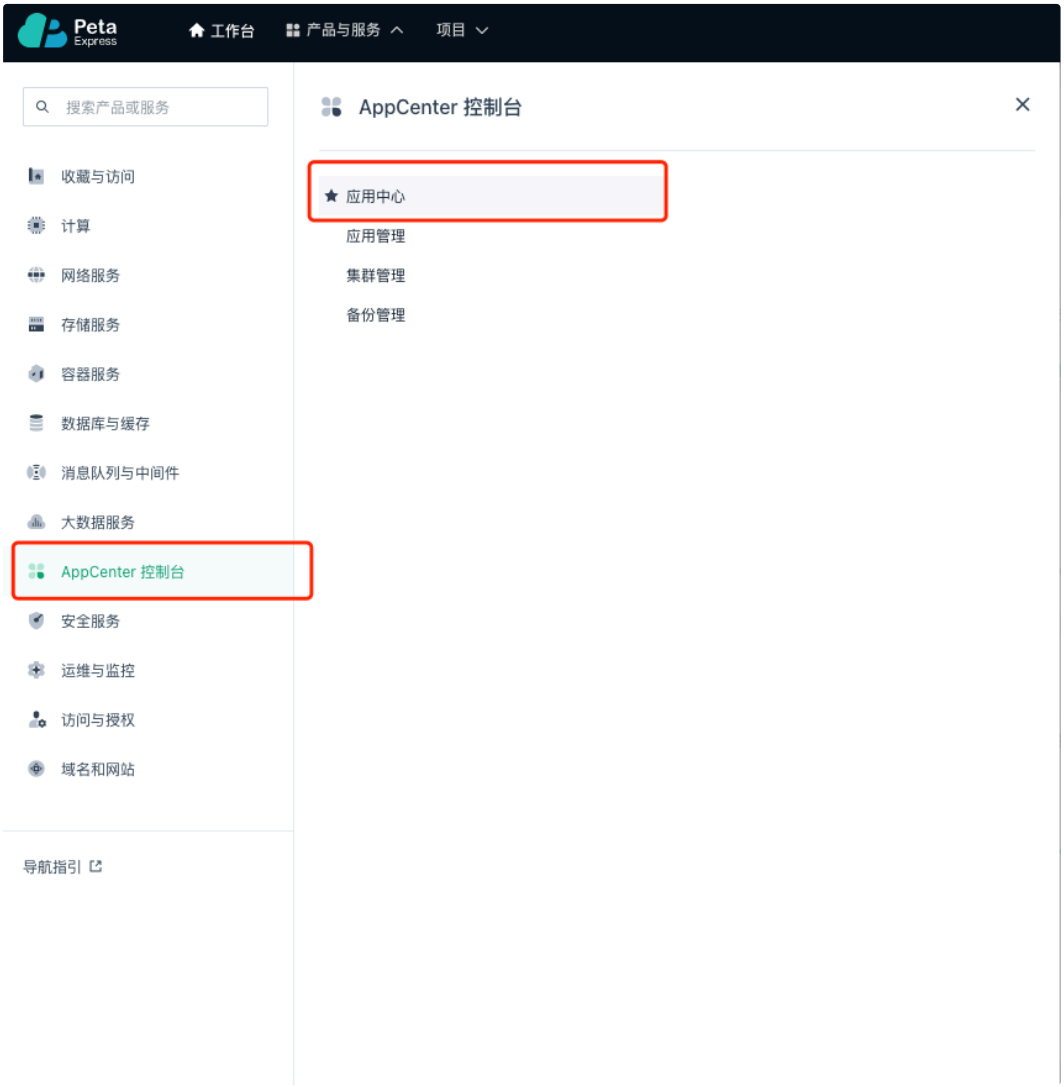
3、智能自动调度：容器意外挂掉后能够迅速进行自动调度重启和资源分配

4、高效且稳定：`Kubernetes` 在整个集群上进行调度，只要整个集群不挂掉总会调度到合适节点重启容器服务

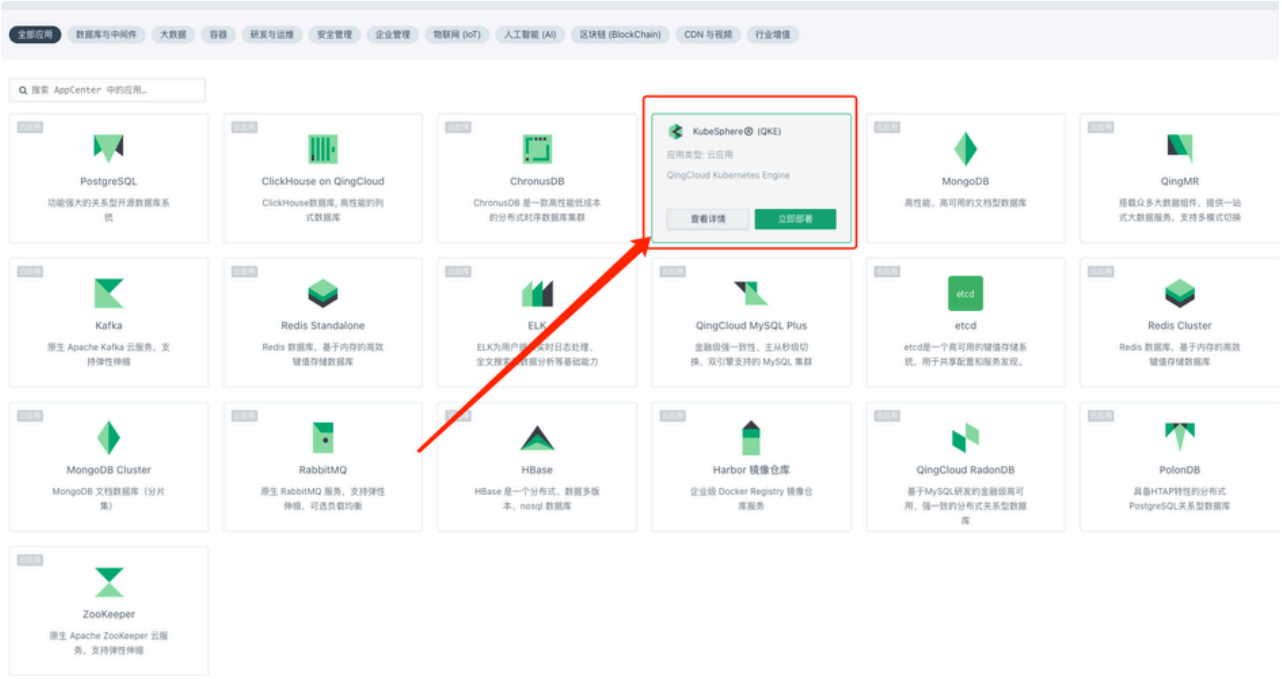
阅读全文，后面告诉你怎么免费白拿红包

在 PetaExpress KubeSphere容器平台部署 Kubernetes集群

在 Peta Express 中部署 Kubernetes 非常简单，直接使用 Peta Express 中内置的 QKE 即可，登录到 Peta Express 控制台，在产品与服务中找到 AppCenter 控制台 → 应用中心。



找到 QKE 立即部署即可。



按照提示输入名称，选择集群规模等信息，直接提交就行了。但需要注意集群的配置，如果是开发测试可以选择 “基础型开发环境” 或 “企业型测试环境”，如果是生产的话则可以选择 “基础型生产环境” 或 “企业型生产环境”，也可以自定义集群规模和HA。

根据集群的规模，部署时间大致2分钟到10分钟不等，还是非常快的。部署完 Kubernetes, 接下来我们就可以开始进入正题 安装 Redis 了。

安装 Redis 集群

我这里新建了一个 `test-project` 的项目空间来做 `Redis` 集群所有安装资源的放置，后续在 `DNS` 上会用到项目空间名称，会标注这一部分，需要注意用自己的项目空间名。

Redis 集群的安装流程大致分为以下几个关键步骤：

①配置 `redis.conf` 字典；

②创建 `redis` 服务；

③容器组配置；

④存储设置；

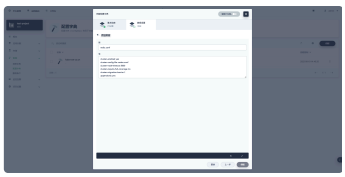
⑤高级设置。

接下来，我们将从第一步开始，逐步完成 Redis 集群的安装和配置过程。

配置 redis.conf 字典

在项目空间的 `配置` → `配置字典` → `创建` 进行配置字典的创建。

名称就叫 `redis-conf` 然后下一步 添加键值对数据。

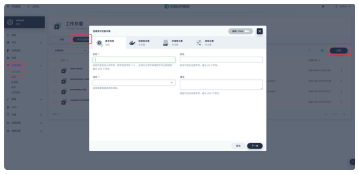


`key` 值的内容为 `redis.conf` ，`value` 值为：

```
cluster-enabled yes
cluster-config-file nodes.conf
cluster-node-timeout 5000
cluster-require-full-coverage no
cluster-migration-barrier 1
appendonly yes

```

创建 Redis 服务

在项目空间的 `应用负载` → `工作负载` → `有状态副本集` → `创建` 进行 **Redis** 服务的创建。



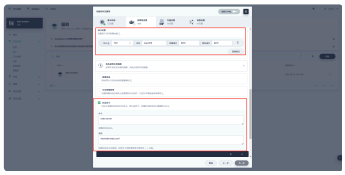
基本设置里名称就叫 `redis-cluster` 然后进行重头戏，下一步的`容器组配置`。

容器组配置

这一步的核心就是配置 Redis 的容器，集群数量我们通常采用三主三从的集群配置，那容器的副本数量就是 6 个，这样的配置不仅保证了系统的稳定性，也提升了数据的安全性。

容器组副本数量调到 6 个，点击添加容器。

镜像选择 `docker hub` 中 `redis` ，并选择使用默认端口，CPU 和内存可以选择性预留，如果不预留就是调度公共资源。

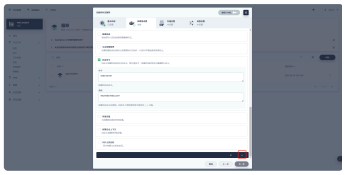


选择 `使用默认端口` 的话下面的端口设置就是如上图一样都会使用 `6379` ，还有就是配置启动命令。

如上图配置：

- 命令： `redis-server`
- 参数： `/etc/redis/redis.conf`

参数指向的就是之前字典配置的内容，但是需要下一步 `存储设置` 里进行配置字典才能使用。



其他内容没有什么需要配置的，选择对勾完成容器配置。

更新策略就是推荐的 `滚动更新` ，其他也没什么需要修改的，点击下一步配置 `存储设置` 。

存储设置

在这一步有两个操作

- `添加存储卷模板`
- `挂载配置字典或保密字典`

添加存储卷模板

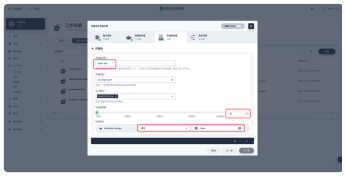
PVC 名称前缀：redis-pvc

容量：10G

挂载路径：

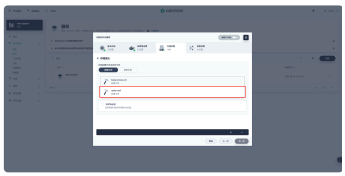
- **权限**：读写
- **地址**：/data

  主要是挂载路径选好，配置好后点击对勾完成配置



挂载配置字典或保密字典

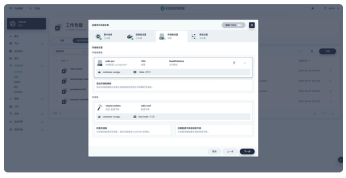
这一步是挂载我们之前配置的字典 `redis-conf`，也是我们 `redis` 启动命令的参数内容。



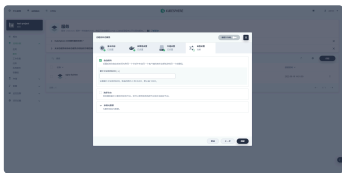
选择 `redis` 的配置字典。

挂载权限为：`只读`，地址为：`/etc/redis`；跟上面的命令参数的配置相对应。

特定键选择 `redis.conf` 后面同名 `redis.conf`，完成后点击对勾回到存储设置。



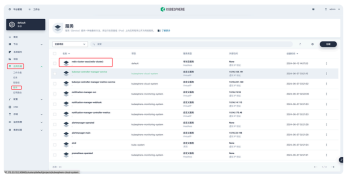
配置好后就入上图，点击下一步进入最后的高级设置。



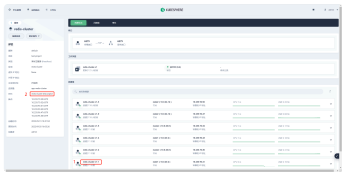
`高级设置`里是一些额外配置，可以根据自己场景选择调整配置，调成完成后点击`创建`**进行 Redis 集群容器的创建**。

初始化 Redis 集群

创建完 **Redis 服务**后 在项目空间的`应用负载` → `服务` → `指定redis服务` 进入 `redis` 服务详情，详情如下图：



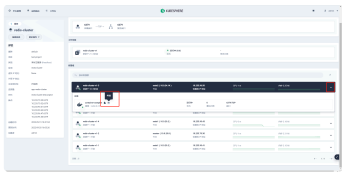
6 个 `redis` 的容器组都启动成功了，接下来就是初始化集群；因为我们配置的 `redis` 的服务是`有状态服务 (Headless)` 所以访问模式可以通过内部 DNS，访问格式是：**(容器名称).(容器 DNS).svc.cluster.local**。



按上图示例 比如访问**集群 1 节点**访问地址就是 `redis-cluster-v1-1` 加 `DNS` 地址 `redis-cluster.test-project` 加 `svc.cluster.local`，完整地址如下：

```
redis-cluster-v1-1.redis-cluster.test-project.svc.cluster.local

```

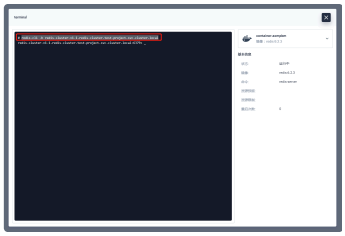
在 redis 集群的非第一节点的其他节点终端内通过这个地址进行访问验证他们是否互通，进入 3 节点的终端，如下图：



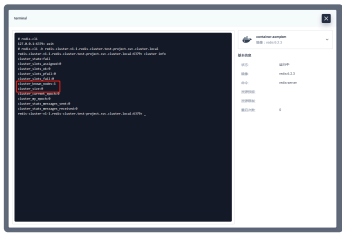
进入终端，执行命令：

```
redis-cli -h redis-cluster-v1-1.redis-cluster.test-project.svc.clusterredis.local

```

如果能如下图一样跳转到 `v1-1` 的节点上就代表这两个节点互通。



执行命令：`cluster info` 查看节点的集群情况。



主要看上图的这两个参数，`nodes` 为 `1` 表明当前节点只有 `1` 个，`cluster_size` 表明当前没有 `master` 节点，所以目前还不是集群结构，`info` 属性的详解在此列出：

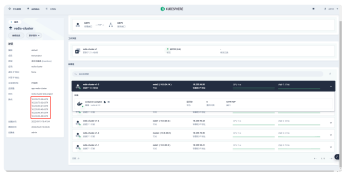
- `cluster_state`：ok 状态表示集群可以正常接受查询请求。fail 状态表示，至少有一个哈希槽没有被绑定（说明有哈希槽没有被绑定到任意一个节点），或者在错误的状态（节点可以提供服务但是带有 FAIL 标记），或者该节点无法联系到多数 master 节点。
- `cluster_slots_assigned`：已分配到集群节点的哈希槽数量（不是没有被绑定的数量）。16384 个哈希槽全部被分配到集群节点是集群正常运行的必要条件。
- `cluster_slots_ok`：哈希槽状态不是 FAIL 和 PFAIL 的数量。
- `cluster_known_nodes`：集群中节点数量，包括处于握手状态还没有成为集群正式成员的节点。
- `cluster_slots_pfail`：哈希槽状态是 PFAIL 的数量。只要哈希槽状态没有被升级到 FAIL 状态，这些哈希槽仍然可以被正常处理。PFAIL 状态表示我们当前不能和节点进行交互，但这种状态只是临时的错误状态。
- `cluster_slots_fail`： 哈希槽状态是 FAIL 的数量。如果值不是 0，那么集群节点将无法提供查询服务，除非 cluster-require-full-coverage 被设置为 no。
- `cluster_current_epoch`：集群本地 Current Epoch 变量的值。这个值在节点故障转移过程时有用，它总是递增和唯一的。
- `cluster_my_epoch`：当前正在使用的节点的 Config Epoch 值。这个是关联在本节点的版本值。
- `cluster_size`：至少包含一个哈希槽且能够提供服务的 master 节点数量。
- `cluster_stats_messages_sent`：通过 node-to-node 二进制总线发送的消息数量。
- `cluster_stats_messages_received`：通过 node-to-node 二进制总线接收的消息数量。

IP 地址初始化集群(初始化方案一)

先尝试使用 `ip + port` 的方式初始化集群，但是在 Kubernetes（`K8s`）中启动服务 `ip` 都会变化，所以最终的结果还是要用 `DNS` 方式进行集群初始化。

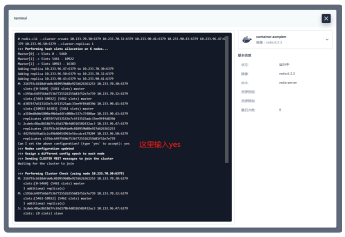
执行本步后再想修改为 DNS 地址初始化需要从来一遍，如果不想麻烦的同学可以直接跳过。

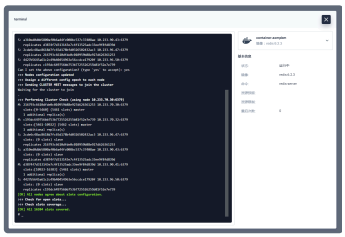
记录 redis 集群的所有 ip+port，初始化命令如下：



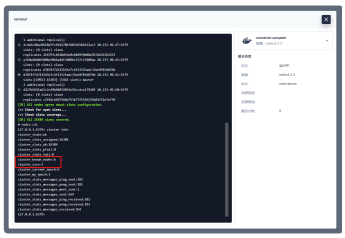
```
redis-cli --cluster create 10.233.70.30:6379 10.233.70.32:6379 10.233.90.41:6379 10.233.90.43:6379 10.233.96.47:6
```

进入 `redis` 集群随意一个节点的`终端`执行上面的命令。

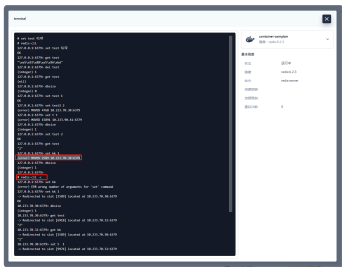




如上图集群初始化就完成了，再输入命令 `redis-cli` 进入命令端，再执行 `cluster info` 查看集群信息。



现在我们的集群节点有了 `6` 个，`master` 节点也有了三个，集群建立完成，后面的操作选择 `master` 节点进行操作。



在对集群节点进行验证的时候如果遇到上图的错误 `(error) MOVED 2589 10.233.70.30:6379` 是因为 `redis-cli` 没有开启集群模式，将命令修改为 `redis-cli -c` 就切换为集群模式了。

使用内部 DNS 初始化(初始化方案二)

使用 `ip` 地址的方式在每次 `K8s` 调度 `redis` 后 `ip` 都会发生变化，所以在 `K8s` 集群中使用 `ip` 方式初始化集群并不太合适，但是如果使用内部 `DNS` 直接跟上面一样初始化集群会出现错误，因为 `redis` 对域名的支持并不太好，所以这时候可以用 `Redis-tribe`。

创建 Redis-tribe 服务

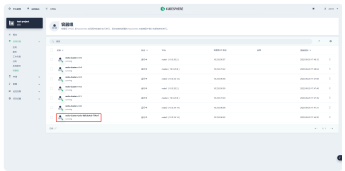
在项目空间的 `应用负载` → `工作负载` → `创建` → `编辑 YAML` 进行Redis-tribe服务的创建。

参数 `namespace` 就写项目名称：

具体 `YAML` 内容如下：

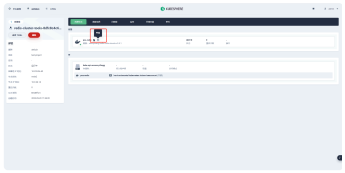
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: test-project
  labels:
    app: redis-cluster-tools
  name: redis-cluster-tools
spec:
  replicas: 1
  selector:
    matchLabels:
      app: redis-cluster-tools
  template:
    metadata:
      labels:
        app: redis-cluster-tools
      name: pos-redis
    spec:
     containers:
     - name: pos-redis
       image: sunnywang/redis-tools-ubuntu:v0.5.1
       imagePullPolicy: IfNotPresent
       args:
       - /bin/bash
       - -c
       - sleep 3600

```

创建好后在容器组内找到 `redis-cluster-tools`。



初始化集群

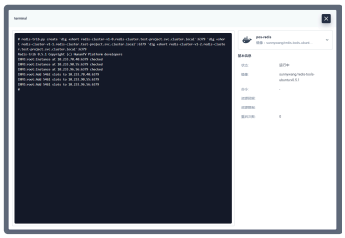
点击容器名称进入容器详情再进入到终端里。



先执行以下命令初始化 `master` 节点，这时候之前的内部 `DNS` 的域名就有用了。

```
redis-trib.py create `dig +short redis-cluster-v1-0.redis-cluster.test-project.svc.cluster.local`:6379 `dig +shor

```

执行结果如下图：



接下来给每个 `master` 节点绑定对应的副本节点，总共三个：

0 节点->3 节点

```
redis-trib.py replicate --master-addr `dig +short redis-cluster-v1-0.redis-cluster.test-project.svc.cluster.local

```

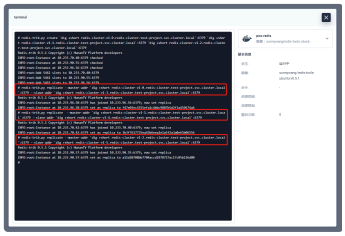
1 节点->4 节点

```
redis-trib.py replicate  --master-addr `dig +short redis-cluster-v1-1.redis-cluster.test-project.svc.cluster.loca

```

2 节点->5 节点

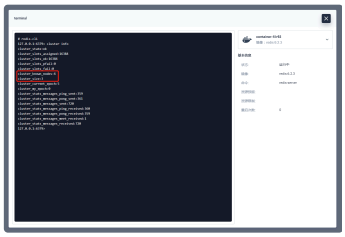
```
redis-trib.py replicate --master-addr `dig +short redis-cluster-v1-2.redis-cluster.test-project.svc.cluster.local

```

执行结果如下：

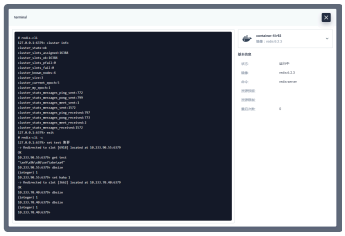


验证

随便进入一个集群节点的终端，还是执行 `cluster info` 命令，查看集群信息。



使用基础命令进行验证，验证集群模式的 `redis-cli` 需要加 `-c`。



验证集群模式可以正常使用。

凭此文章可以去petaexpress官网发工单**免费白拿10美元红包**，数量有限先到先得。申领步骤：注册→登录→发工单回复“**文章网址+文章标题+申请奖励**”