Hpa

|  |
| --- |
|  |

范例解释

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Cpu超过80%进行缩容    Scaletargetref、minReplicas & maxReplicas、metrics、behavior   每隔 --horizontal-pod-autoscaler-sync-period（默认 15 秒）检查目标 Deployment Pod 的 CPU 使用率   **CPU 使用率 > 80%** → 扩容，Pod 数量增加，但不会超过 maxReplicas   **CPU 使用率 < 80%** → 缩容，但 HPA 会等待 stabilizationWindowSeconds 秒，防止 Pod 频繁被删除   HPA 保证 Pod 数量始终在 [minReplicas, maxReplicas] 之间  havior:  scaleUp:  stabilizationWindowSeconds: 0 # 扩容不延迟  policies:  - type: Pods  value: 4 # 每次最多增加 4 个 Pod  periodSeconds: 60 # 每 60 秒最多增加这么多  scaleDown:  stabilizationWindowSeconds: 300  scaleUp 可控制：  扩容速度（一次最多增加多少 Pod）  扩容延迟（避免瞬间大量扩容）   |  |  | | --- | --- | | **情况** | **默认行为** | | CPU > target | 扩容，立即按比例增加 Pod | | CPU < target | 缩容，立即按比例减少 Pod（如果没有 stabilizationWindowSeconds） | | YAML 设置 behavior.scaleDown.stabilizationWindowSeconds | 缩容延迟（比如 300 秒），避免频繁缩容 |   Cpu内存混合指标：  Cpu70%+内存80%      metrics.resource 中，**检查的是 Pod 级别的指标，而不是 Node 级别**，   1. HPA 获取目标 Deployment 的所有 Pod 2. 每个 Pod 查询 **容器 CPU 使用率**（来自 Metrics Server） 3. 计算平均值 → 与 target.averageUtilization 比较 4. 决定增加或减少副本数   自定义指标：       |  |  |  |  | | --- | --- | --- | --- | | **指标类型** | **作用对象** | **数据来源** | **示例** | | Resource | Pod 容器 CPU/内存 | Metrics Server | CPU Utilization | | Pods | Pod 自定义指标 | 自定义 Metrics Adapter / Prometheus Adapter | HTTP 请求数、队列长度等 | |

应用示例

|  |
| --- |
| 主节点导入metrices-server  执行compunnets.yml  创建deployment：    创建hpa：          创建service：  默认的clusterip    模拟负载触发扩容：  一个窗口获pod创建情况：  一个窗口获取hpa监控情况：    执行模拟：    Cpu达到10%就会扩容：            终端请求：  看到缩容了：    到了最小2个 |

KPA和VPA

|  |
| --- |
| 区别：  VPA用于自动调整Pod的CPU和内存资源（requests/limits），而非调整副本数。需结合目标控制器（如Deployment）和VPA配置文件使用。 |