实验二：Service job in cloud

UNIkeEN 2023/11/30

Part A：认识 Kubernetes 集群

运行结果

各软件环境的配置安装结果图见附录图 1~3；本题命令运行结果截图见附录图 4、5。

对 Kubernetes 集群的认识

Kubernetes 是一个用于管理容器化的工作负载和服务的自动化运维平台，可以实现容器集群的大规模自动化部署、扩缩容、负载均衡、维护等功能。解决了传统容器化手动部署、伸缩效率低且容易出错的痛点。

架构

Kubernetes 集群主要包括控制平面（Master）和多个 Node 组件。

- Master 组件是集群的控制中心，管理集群的全局决策（如编排调度、检测和响应集群事件等）。主要包含 kube-apiserver（通讯核心、集群接口），ectd（数据库），kube-controller-manager（保证维持集群工作状态一致），kube-scheduler（调度器）等。
- Node 组件维护多个运行的 Pod 并提供 Kubernetes 运行环境——即 Kubernetes 通过将容器放入在 Node 上运行的 Pod 中来执行工作负载。主要包含 kubelet（节点代理），kube-proxy（网络代理），container-runtime 等。
- Pod：Kubernetes 中最基本的调度单位，通常只包含一个容器。

除此之外还有一些插件如 Dashboard（Web 控制台）、Weave Scope（资源监控）等。

部分特性

- 自动部署/回滚：以一定速率将状态变更到用户描述的状态，如果出现问题则自动回滚。
- 自动伸缩：根据预设的规则和实时负载自动增减容器实例数量，以优化资源使用和应对负载变化。
- 服务发现：用 DNS 名称或 IP 地址暴露服务，使服务消费者能够找到服务提供者。
- 存储编排：允许自动挂载存储系统，包含本地存储、公共云提供商的存储，或网络存储系统。
- 自我修复：重启失败容器；替换失效节点并重新部署容器；关闭不满足健康检查的容器。
- 声明式配置：用户声明想要的环境状态，Kubernetes 负责实现。

kube-scheduler 组件功能

kube-scheduler 是 Kubernetes 集群的默认调度器，将未调度的 Pod 调度到一个合适的节点上来运行。在挑选与调度的过程中，scheduler 会按顺序运行一系列指定算法。主要包含三个阶段：

- 排序阶段：在 k8s 中，待调度的 Pod 会放在一个叫 activeQ 的优先队列中，scheduler 按设置的优先级（若无则以创建时间）取出待调度 Pod。
- 过滤阶段：将所有满足 Pod 调度需求的节点选出来。如 PodFitsResources 过滤函数会检查候选节点的可用资源能否满足 Pod 的资源请求。
- 打分阶段：调度器根据当前启用的打分规则，给每一个可调度节点进行打分。

过滤（具体可再分为三个扩展点）和打分阶段可以放入多个系统自带或用户提供的插件（相关算法）以执行。最后，kube-scheduler 会将 Pod 调度到得分最高的节点上，多个最高分则随机调度，执行 reserve 阶段和绑定阶段的插件流程。

Part B：在 Kubernetes 上搭建博客系统

名词解释

- `Service`（服务）是一种定义一组 Pod 访问策略和方法的抽象。将运行在一组 Pod 上的应用程序抽象为网络接口，提供统一的 IP 地址和端口来访问底层 Pod。外部用户和内部 Pod 都使用 Service 与其他 Pod 通信。它还可以作为内部负载均衡器使用。

- `PersistentVolumeClaim`（PVC，持久卷请求）即请求存储。它允许用户以抽象的方式请求特定大小和访问模式（如只读或读写）的存储。PVC 仅被用于申请存储资源，而无需了解底层存储的具体实现细节。PV 提供实际的存储，而 PVC 是对这些资源的请求。

- `Deployment` 是对 Pod 和 ReplicaSets（副本集）的声明式更新的管理。它允许你描述应用的期望状态，比如应用应该运行哪些镜像和代码版本。用于无状态应用的部署和扩展。它支持声明式的应用更新、回滚、扩展和自我修复功能。例如，如果一个 Pod 崩溃，`Deployment` 会自动替换它。

集群状态与创建博客

状态见图 6，自建博客的文章截图见图 7。

部署 Kubernetes 应用与部署传统应用的不同之处

- 部署方法：Kubernetes 应用使用容器进行部署，依赖 Docker 等容器技术并使用声明式配置（如 yaml）定义应用部署，且部署可通过命令行工具自动完成；传统应用直接部署在操作系统上或使用虚拟化技术，配置和部署通常更手动且具有依赖性。

- 资源利用：Kubernetes 部署灵活分配资源，提高资源利用率。而传统部署无法较快速地将剩余资源分配给其他的应用。

- 扩展性：Kubernetes 应用容易水平扩展，可以自动或手动增加或减少实例数量。传统应用扩展通常更复杂和耗时，可能需要更多的硬件或资源重新配置，且扩展通常是垂直的（增加单个节点的资源）。

- 更新维护：Kubernetes 应用支持滚动更新，无需停机即可更新应用，对于 CI/CD 更加敏捷，且可以自动回滚到之前的版本。传统应用新可能需要停机维护窗口，回滚复杂，通常需要手动干预。

- 跨平台一致性：传统部署通常对物理机环境有特定要求（或在不同环境需进行不同操作），而 Kubernetes 部署可以跨平台。

附录

```
root@master:/# curl -Lo minikube https://kubernetes.oss-cn-hangzhou.aliyuncs.com/r
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 65.0M  100 65.0M    0     0  32.3M      0  0:00:02  0:00:02 --:--:-- 32.3M
root@master:/# chmod +x minikube
root@master:/# sudo mv minikube /usr/local/bin/
root@master:/# minikube version
minikube version: v1.23.1
commit: 9e2f8cb489d9b3e871ba206d40ae92c7521b7e76-dirty
root@master:/#
```

图 1：Minikube 配置运行结果

```
root@master:/# sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
719385e32844: Pull complete
Digest: sha256:c79d06dfdfd3d3eb04cafd0dc2bacab0992ebc243e083cabe208bac4dd7759e0
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

root@master:/#
```

图 2：Docker 配置运行结果

```
root@master:/# kubectl version
Client Version: version.Info{Major:"1", Minor:"22", GitVersion:"v1.22.1", GitCommit:"632ed300f2c34f6d6d15ca4cef3d3c7073412212",
GitTreeState:"clean", BuildDate:"2021-08-19T15:45:37Z", GoVersion:"go1.16.7", Compiler:"gc", Platform:"linux/amd64"}
The connection to the server localhost:8080 was refused - did you specify the right host or port?
root@master:/#
```

图 3：Kubectl 配置运行结果

```
root@master:/# minikube start --image-mirror-country='cn' --registry-mirror="https://d955d3883f694633bdad8561831288f0.mirror.swr
.myhuaweicloud.com" --force
😄  minikube v1.23.1 on Ubuntu 18.04 (amd64)
❗  minikube skips various validations when --force is supplied; this may lead to unexpected behavior
✨  Automatically selected the docker driver. Other choices: ssh, none
🛑  The "docker" driver should not be used with root privileges.
💡  If you are running minikube within a VM, consider using --driver=none:
📘    https://minikube.sigs.k8s.io/docs/reference/drivers/none/
🎉  minikube 1.28.0 is available! Download it: https://github.com/kubernetes/minikube/releases/tag/v1.28.0
💡  To disable this notice, run: 'minikube config set WantUpdateNotification false'

✅  Using image repository registry.cn-hangzhou.aliyuncs.com/google_containers
👍  Starting control plane node minikube in cluster minikube
🚜  Pulling base image ...
    > registry.cn-hangzhou.aliyun...: 355.40 MiB / 355.40 MiB  100.00% 8.12 MiB
🔥  Creating docker container (CPUs=2, Memory=2200MB) ...
    > kubeadm.sha256: 64 B / 64 B [--------------------------] 100.00% ? p/s 0s
    > kubelet.sha256: 64 B / 64 B [--------------------------] 100.00% ? p/s 0s
    > kubectl.sha256: 64 B / 64 B [--------------------------] 100.00% ? p/s 0s
    > kubeadm: 43.71 MiB / 43.71 MiB [------------] 100.00% 62.21 MiB p/s 900ms
    > kubectl: 44.73 MiB / 44.73 MiB [------------] 100.00% 48.64 MiB p/s 1.1s
    > kubelet: 146.25 MiB / 146.25 MiB [------------] 100.00% 8.54 MiB p/s 17s

    ▪ Generating certificates and keys ...
    ▪ Booting up control plane ...
    ▪ Configuring RBAC rules ...
🔎  Verifying Kubernetes components...
    ▪ Using image registry.cn-hangzhou.aliyuncs.com/google_containers/storage-provisioner:v5
🌟  Enabled addons: storage-provisioner, default-storageclass
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
root@master:/#
```

图 4：Minikube 运行结果

```
root@master:/# kubectl get pod --all-namespaces
NAMESPACE     NAME                               READY   STATUS    RESTARTS      AGE
kube-system   coredns-7d89d9b6b8-lfwn7           1/1     Running   0             57s
kube-system   etcd-minikube                      1/1     Running   0             71s
kube-system   kube-apiserver-minikube            1/1     Running   0             69s
kube-system   kube-controller-manager-minikube   1/1     Running   0             72s
kube-system   kube-proxy-kt4nl                   1/1     Running   0             57s
kube-system   kube-scheduler-minikube            1/1     Running   0             70s
kube-system   storage-provisioner                1/1     Running   1 (25s ago)   69s
root@master:/#
```

图 5：检查组件运行情况

```
root@master:/home/lab2# kubectl get service --all-namespaces
NAMESPACE     NAME              TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)                  AGE
default       kubernetes        ClusterIP      10.96.0.1        <none>        443/TCP                  10m
default       wordpress         LoadBalancer   10.104.193.170   <pending>     80:30181/TCP             83s
default       wordpress-mysql   ClusterIP      None             <none>        3306/TCP                 83s
kube-system   kube-dns          ClusterIP      10.96.0.10       <none>        53/UDP,53/TCP,9153/TCP   10m
root@master:/home/lab2# kubectl get deployment --all-namespaces
NAMESPACE     NAME              READY   UP-TO-DATE   AVAILABLE   AGE
default       wordpress         1/1     1            1           2m15s
default       wordpress-mysql   1/1     1            1           2m15s
kube-system   coredns           1/1     1            1           11m
root@master:/home/lab2# kubectl get pod --all-namespaces
NAMESPACE     NAME                                 READY   STATUS    RESTARTS      AGE
default       wordpress-d69744f79-6zqpv            1/1     Running   0             2m22s
default       wordpress-mysql-6dc6f4b65b-2txkl     1/1     Running   0             2m22s
kube-system   coredns-7d89d9b6b8-lfwn7             1/1     Running   0             11m
kube-system   etcd-minikube                        1/1     Running   0             11m
kube-system   kube-apiserver-minikube              1/1     Running   0             11m
kube-system   kube-controller-manager-minikube     1/1     Running   0             11m
kube-system   kube-proxy-kt4nl                     1/1     Running   0             11m
kube-system   kube-scheduler-minikube              1/1     Running   0             11m
kube-system   storage-provisioner                  1/1     Running   1 (10m ago)   11m
root@master:/home/lab2#
```

图 6：集群状态

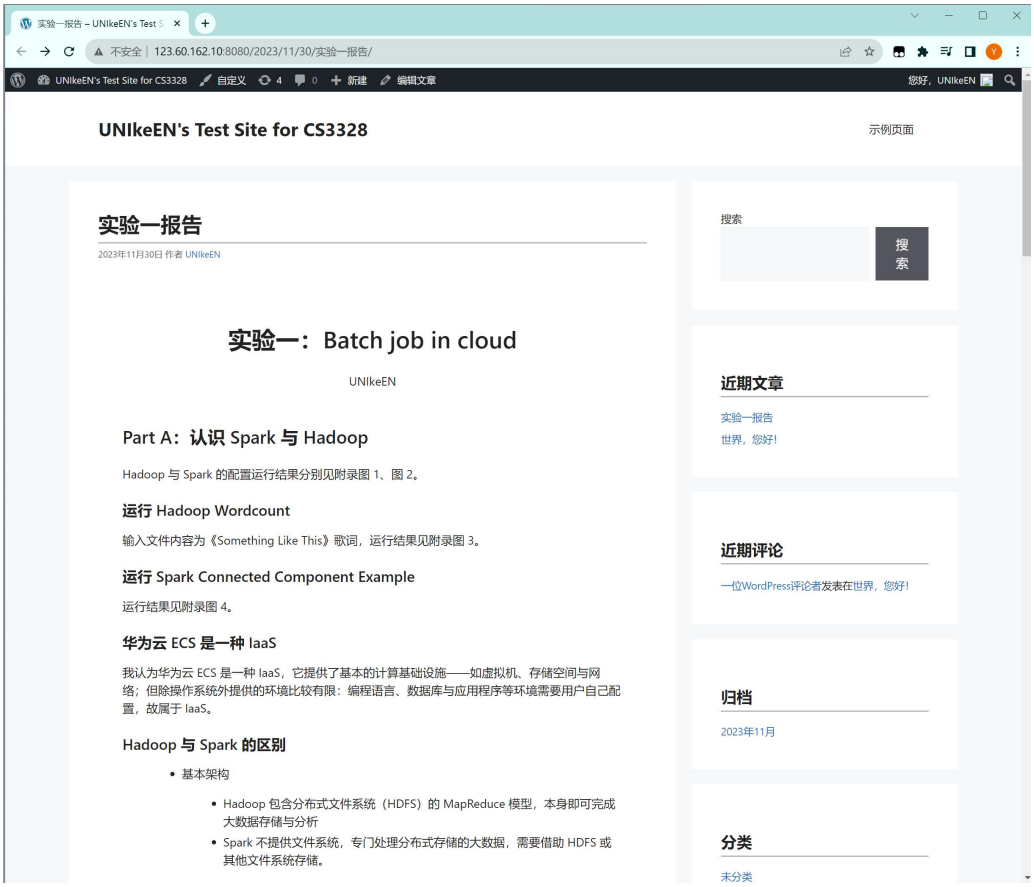

图 7：测试博客截图