实验二报告

Part A

执行命令截图

如附录图 1、2 所示

描述对 kubernets 的认识

- Kubernetes（简称 K8s）是一个开源的容器编排平台，旨在简化和自动化容器化应用的部署、扩展和运维。Kubernetes 的主要**用途**之一是容器编排。在传统的应用开发中，开发人员需要考虑应用的部署、运行和扩展等问题，而容器技术的出现解决了这些问题。Kubernetes 提供了强大的容器编排功能，使得开发人员可以将应用打包成容器，而无需担心底层基础设施的细节。

- kubernets **架构**可以描述为：一个集群上由 Control Plane 和 n 个 Node 组成，其中 **Control Plane** 由以下部分组成——kube-apiserver: 提供 API 服务，用于与集群交互。etcd: 分布式键值存储，用于存储集群的配置数据。scheduler: 负责调度容器到集群中的节点。controller-manager: 包含多个控制器，负责管理集群的状态和行为。loud-controller-manager: 与云服务提供商相关，负责处理与云平台相关的控制逻辑，例如负责云资源的创建、删除和监控。**Node** 由以下部分构成——kubelet: 负责与 Master 节点通信，管理容器的生命周期。kube-proxy: 负责维护网络规则，实现服务发现和负载均衡。Pod：包含一个或多个容器， 共享相同的网络命名空间和存储卷，使得容器之间可以方便地进行通信和数据共享。

- 其主要**特性**有：**自动化容器部署和调度：** Kubernetes 提供了自动化的容器编排功能，负责将应用程序的容器部署到集群中的各个节点，并根据资源需求和约束进行智能调度。**自动伸缩：** Kubernetes 允许根据应用程序的负载动态地调整容器的数量，实现自动伸缩。这提高了系统的弹性和资源利用效率。**服务发现与负载均衡：** Kubernetes 提供了内置的服务发现机制，通过定义 Service 对象，容器可以通过服务名称相互通信。同时，"kube-proxy"负责维护网络规则，实现负载均衡，确保流量被均匀分配到后端的容器中。**自我修复：** Kubernetes 具有自我修复机制，能够自动检测并替换失败的容器实例，确保应用程序的高可用性。**声明式配置：** 用户可以使用 YAML 或 JSON 格式的声明式配置文件来描述应用程序的状态和要求，Kubernetes 负责实现所需的状态。这种声明式配置简化了应用程序的管理。**多环境和多云支持：** Kubernetes 的设计理念支持在不同的环境和云平台上运行应用程序，使其具有较好的可移植性。**滚动更新和回滚：** Kubernetes 允许用户通过滚动更新机制轻松地升级应用程序，同时还提供了回滚功能，以应对不可预测的问题等

学习组件 scheduler

" scheduler" 是 Kubernetes 集群中的一个核心组件，负责将新创建的 Pod 调度到集群中的合适节点上。其功能为：

- Pod 的调度决策："kube-scheduler" 负责根据集群中节点的资源利用情况、硬件特性、网络拓扑等因素，以及用户定义的调度策略，决定将新创建的 Pod 调度到哪个节点上。这个调度决策考虑了集群的整体负载均衡和资源利用效率。

- 调度策略的定制："kube-scheduler" 允许用户通过调度策略（Scheduling Policies）来定制 Pod 的调度行为。这包括通过 NodeSelector、Affinity 和 Anti-Affinity 等机制，以及通过资源限制和请求等方式来影响调度决策。

- 预选和优选过程："kube-scheduler" 在进行调度决策时，采用预选（preemption）和优选（priority）的过程。在预选阶段，调度器会剔除不符合条件的节点，以缩小候选节点的范围。在优选阶段，调度器为每个候选节点计算一个分数，最终选择得分最高的节点进行调度。

- 可扩展性："kube-scheduler" 具有可扩展性，允许用户根据自己的需求实现和添加自定义的调度器扩展。这样可以为集群提供更多定制化的调度策略，适应不同的应用场景。

- 多调度器架构：Kubernetes 支持多调度器架构，意味着可以同时运行多个调度器，并且每个调度器可以独立地决策 Pod 的调度。这为用户提供了更大的灵活性，可以根据不同的调度需求选择合适的调度器。

Part B

service, persistentvolumeclaim 与 deployment 的含义与用途

- **Service（服务）：** 在 Kubernetes 中，Service 定义了一组 Pod 的访问方式，为这些 Pod 提供了一个稳定的网络端点。它充当了服务的负载均衡器，将请求分发到后端的 Pod。Service 可以是 ClusterIP（仅在集群内部可访问）、NodePort（在每个节点上暴露端口）、LoadBalancer（使用云服务提供商的负载均衡器）等类型。

- **PersistentVolumeClaim（持久卷声明）：** PersistentVolumeClaim（PVC）是对持久卷（Persistent Volume）的声明，用于请求存储资源。它允许应用程序声明对持久化存储的需求，并由 Kubernetes 动态地提供符合需求的存储卷。PVC 提供了一种抽象，使得应用程序无需关心底层的存储细节。

- **Deployment（部署）：** Deployment 是 Kubernetes 中用于定义、创建和更新应用程序的资源对象。它允许用户描述应用程序的期望状态，Kubernetes 将负责管理应用程序的实际状态，确保它与期望状态一致。Deployment 中包含了容器的镜像、副本数量、Pod 模板等信息。

service、deployment 和 pod 的状态以及创建的第一个博客内容

如附录图 3、4 所示

部署 kubernetes 应用与部署传统应用的不同之处

- **容器化：** Kubernetes 应用通常被容器化，即封装到容器中。容器是独立、轻量级的运行环境，具有良好的隔离性和可移植性。传统应用可能以更为传统的方式运行，例如直接安装在物理服务器或虚拟机上，依赖主机的操作系统和配置。

- **声明式配置：** 部署和管理 Kubernetes 应用通常采用声明式配置，通过 YAML 或 JSON 文件描述应用的期望状态，由 Kubernetes 控制平面负责实现。 传统应用的部署可能涉及手动配置和管理，例如在服务器上安装和配置软件、设置环境变量等。

- **弹性和伸缩性：** Kubernetes 提供自动化的伸缩机制，能够根据负载动态地调整应用的实例数量。这使得应对变化的工作负载更为灵活。传统应用的伸缩性可能需要手动调整实例数量或依赖特定的自动化工具。

- **服务发现与负载均衡：** Kubernetes 提供内置的服务发现和负载均衡机制，通过 Service 对象将服务抽象化，使得应用可以通过服务名称相互访问。 传统应用可能需要手动配置和管理服务发现和负载均衡，使用专用的硬件或软件解决方案。

- **持久化存储：** Kubernetes 提供了持久卷（Persistent Volumes）和持久卷声明（Persistent Volume Claims）来处理应用程序的持久化存储需求。 传统应用可能需要手动管理存储，配置和维护文件系统或数据库。

- **部署和更新：** Kubernetes 使用 Deployment 对象可以实现滚动更新、版本回退等功能，而不中断应用的可用性。传统应用的部署和更新可能需要停机时间或手动干预，具体取决于应用的设计和部署策略。

附录

```
root@master:~# minikube start --image-mirror-country='cn' --registry-mirror="htt
ps://e780fff690fe443cb8a110683778a4aa.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 t
o unexpected behavior
✨  Using the docker driver based on existing profile
🛑  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/
💡  Tip: To remove this root owned cluster, run: sudo minikube delete
👍  Starting control plane node minikube in cluster minikube
🚜  Pulling base image ...
🔄  Restarting existing docker container for "minikube" ...
🐳  Preparing Kubernetes v1.22.1 on Docker 20.10.8 ...
🔎  Verifying Kubernetes components...
    ▪ Using image registry.cn-hangzhou.aliyuncs.com/google_containers/storage-pr
ovisioner:v5
🌟  Enabled addons: storage-provisioner, default-storageclass
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" name
space by default
```

图 1 执行 minikube 启动命令

```
root@master:~# kubectl get pod --all-namespaces
NAMESPACE     NAME                               READY   STATUS    RESTARTS      AGE
kube-system   coredns-7d89d9b6b8-kx6xz           1/1     Running   1 (11h ago)   11h
kube-system   etcd-minikube                      1/1     Running   1 (11h ago)   11h
kube-system   kube-apiserver-minikube            1/1     Running   1 (11h ago)   11h
kube-system   kube-controller-manager-minikube   1/1     Running   1 (11h ago)   11h
kube-system   kube-proxy-r6hzq                   1/1     Running   1 (11h ago)   11h
kube-system   kube-scheduler-minikube            1/1     Running   1 (56s ago)   11h
kube-system   storage-provisioner                0/1     Error     2 (11h ago)   11h
```

图 2 执行 kubectl get pod 命令

```
zhouxudong — root@master: ~/experiment2 — ssh root@124.71.158.38 — 105×23
~ — root@master: ~/experiment2 — ssh root@124.71.158.38
~ — root@master: ~ — ssh root@124.71.158.38
root@master:~/experiment2# 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                  12h
default       wordpress         LoadBalancer   10.103.44.9    <pending>     80:30090/TCP             18m
default       wordpress-mysql   ClusterIP      None           <none>        3306/TCP                 18m
kube-system   kube-dns          ClusterIP      10.96.0.10     <none>        53/UDP,53/TCP,9153/TCP   12h
root@master:~/experiment2# kubectl get deployment --all-namespaces
NAMESPACE     NAME              READY   UP-TO-DATE   AVAILABLE   AGE
default       wordpress         1/1     1            1           18m
default       wordpress-mysql   1/1     1            1           18m
kube-system   coredns           1/1     1            1           12h
root@master:~/experiment2# kubectl get pod --all-namespaces
NAMESPACE     NAME                               READY   STATUS    RESTARTS      AGE
default       wordpress-6894487c6c-j8rmr         1/1     Running   0             28m
default       wordpress-mysql-74859db64d-gx851   1/1     Running   0             28m
kube-system   coredns-7d89d9b6b8-kx6xz           1/1     Running   1 (12h ago)   12h
kube-system   etcd-minikube                      1/1     Running   1 (12h ago)   12h
kube-system   kube-apiserver-minikube            1/1     Running   1 (12h ago)   12h
kube-system   kube-controller-manager-minikube   1/1     Running   1 (12h ago)   12h
kube-system   kube-proxy-r6hzq                   1/1     Running   1 (12h ago)   12h
kube-system   kube-scheduler-minikube            1/1     Running   1 (34m ago)   12h
kube-system   storage-provisioner                1/1     Running   3 (33m ago)   12h
root@master:~/experiment2#
```

图 3 运行的所有 service、deployment 和 pod 的状态

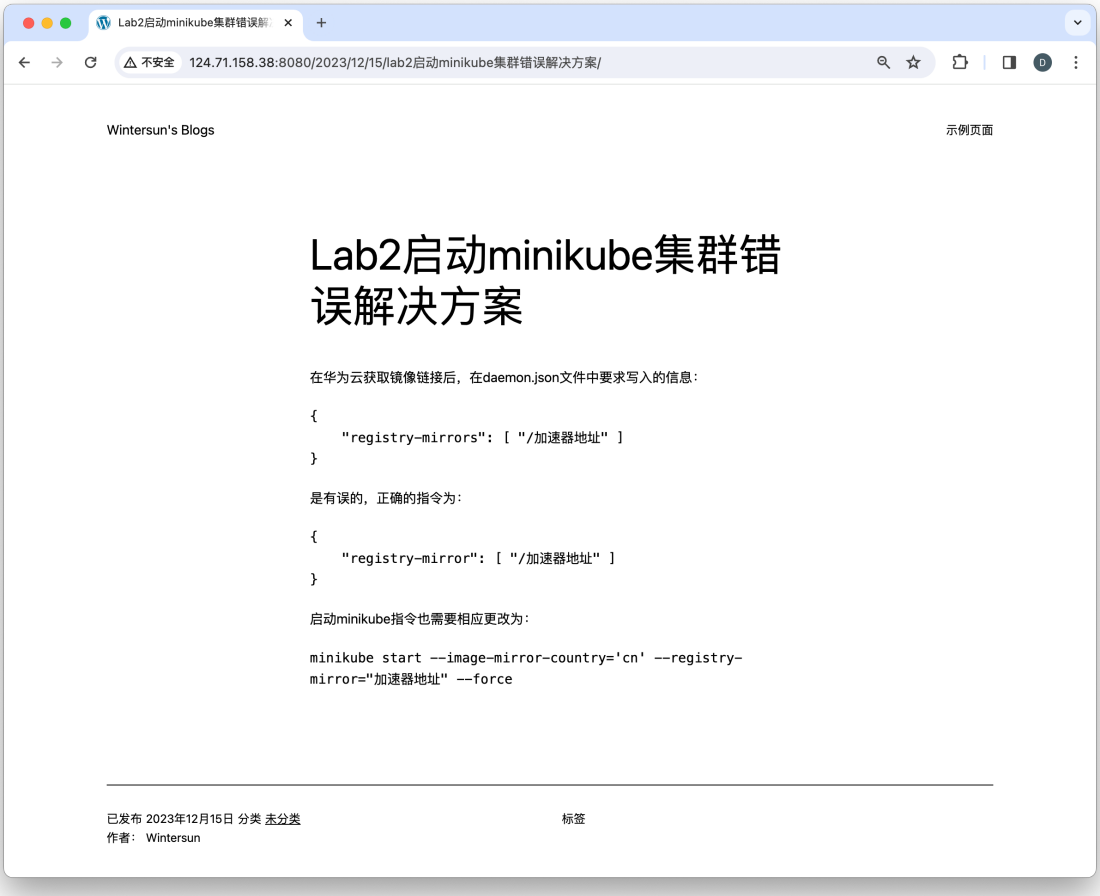

图 4 创建的第一个博客内容