基于 Kubernetes 的容器化 Web 应用的自动化部署方案研究与实现*

刘健民　　林宁**

南宁学院信息工程学院，南宁　530200

摘　要　在文章针对企业快速增长的业务需求和高并发访问给服务器带来的各种问题，设计了一套基于 k8s 的容器化 web 应用自动化部署方案，该案基于 Kubernetes（简称 K8s）为基本框架，采用 Linux 系统搭建 Master 节点和 node 节点，利用 Deployment 技术实现容器化的应用部署，MySQL 数据库作为用户数据储存，同时，结合了 Jenkins、docker、k8s 等开发技术，实现了容器化 Web 应用自动化部署方案。系统由应用模板模块、配置管理模块、自动化部署模块组成；用户通过集群获取对应的配置文件，下载和固定对应的 chart 包再通过我们集群环境下 gitlab+jenkins+harbor 实现自动化部署的自动生成需要的网页，通过实验证明，自动化部署方案能够大大提高运维效率，为企业业务服务器架设提供参考方案。

关键字　**Kubernetes**，Linux，企业，自动化部署

Research and Implementation of Automated Deployment Solution for Containerized Web Applications based on Kubernetes

Liu JianMin　Lin Ning

School of Information Engineering of Nanning University
Nanning 530200, China

Abstract—The article introduces a Kubernetes-based automatic deployment solution for containerized web applications, designed to address the challenges of rapid business growth and high concurrent access to servers in an enterprise context. Built on the Kubernetes (K8s) framework, the solution utilizes Linux systems to set up Master and node nodes, employs Deployment technology for containerized application deployment, and uses MySQL as the data storage solution for user data. By integrating Pods, Docker, K8s, and other development technologies, the solution achieves automated deployment of containerized web applications. The system comprises application template module, configuration management module, and automated deployment module. Users can access configuration files through the cluster, download and lock the corresponding chart package, and leverage Gitlab, Jenkins, and Harbor in the cluster environment for automated deployment to generate the required web page. Experimental results demonstrate that the automated deployment solution significantly enhances operational efficiency, and provides a reference solution for enterprise business service deployment.

Keywords—Kubernetes, Linux, enterprise, Automated deployment

1　引　言

现如今互联网已经广泛渗透生活各领域，在这样的背景下，中小型企业对拥有公司网站有了一定的需求。但企业搭建和管理一个网站对于会给他们会带来很多麻烦和高额成本。同时随着云计算技术的发展，容器化技术如Docker等开始兴起，容器技术实现了对虚拟技术的“轻量化”，容器能够独立运行于宿主机之上，与其他进程互不干扰[1]。然而，当容器数量较多时，手动管理它们是不切实际的。使用自动化工具对容器进行编排管理是必要的[2]。Kubernetes作为一款开源的容器编排工具，可以帮助企业快速部署和管理web应用。

因此，基于Kubernetes的容器化Web应用自动化部署方案系统应运而生。作为一款开源的容器编排工具，为企业提供了自动化部署、扩展和管理容器化应用程序的解决方案。K8s作为一款领先的容器编排工具，提供了强大的容器管理功能，包括自动化部署、自动扩展、自我修复等特性它的出现极大地简化了容器化应用的管理和部署过程，受到了广泛关注和应用。它可以为企业提供了自动化部署、扩展和管理容器化以此来解决小型企业搭建自家网站而带来不必要的麻

*基金资助：南宁学院 2023 年校企联合共建示范课程项目（2023XJYYX03）资助。
**通讯作者：林宁　bgy_2009@163.com

烦。满足企业快速迭代，系统高可用性和弹性伸缩的需求和因为手动操作的失误。

2 相关技术研究

2.1 Kubernetes 容器编排平台

Kubernets是专用于部署、管理容器的一个开源软件系统。它利用了linux的上述特性来运行迥异的程序，无需关注程序内部的构成，同时让管理员免于将容器逐一部署到集群中每个节点之苦[3]。它的出现极大地简化了容器化应用的管理和部署过程。

2.2 Docker 容器技术

Docker 是一种开源的容器化平台，将所需应用程序、依赖库及所有配置文件等打包压缩到一个标准化的开发单元之中，并允许开发人员和运维人员创建、部署和管理这些虚拟化单元[4]。通过 Docker 开发者可以使用轻量级、独立的容器，即使在不同的环境中运行应用程序，也不需要担心环境的不匹配的问题。

2.3 监控和日志工具

Prometheus 是一种开源的系统监控和警报工具，非常适合用于 Kubernetes 的集群监控。Prometheus 能够用于提供近实时的、基于动态云环境和容器的微服务、服务和应用程序的内省监控。同时，Promtheus 将历史监控数据存储在时序数据库中，通过充分发挥时序数据库的特性，可以更好的聚合数据进行分析[5]。Kibana 是一款开源的数据可视化工具，用于对数据进行查询、分析和可视化展示。并通过用户设定相关筛选数据，筛选出有用的信息，以柱状图的形式展现给用户，用户将得到的有用信息进行整理并指定相应的预防性措施[6]。

2.4 自动化流水线部署

自动化流水线通过 GitLab+Jenkins+Harbor 来实现，Gitlab CI 是 Gitlab 提供的 CI/CD 方案和能力，可以进行持续集成、持续交付、持续部署[7]。Jenkins 主要作用是可以将测试、编译、打包、部署软件项目流程化，使这一套流程可以通过某种事件自动触发[8]。通过与 GitLab 集成，实现自动化的持续集成和持续部署流程。Harbor 作为镜像仓库，用于管理和存放生产的 Docker 镜像同时加快容器应用的部署速度，并保证应用程序镜像的安全性。

3 Web 应用的自动化部署方案设计

（1）Kubernetes 集群架构设计

本 web 自动化部署方案使用 Docker 容器化进行研究开发，使用 k8s 容器编排工具对集群架构进行设计，包括 Master 节点和多个 Node 节点。Master 节点作为集群的核心，用于实现整个集群环境的运行和管理，Node 作为集群的工作节点负责运行容器应用。集群架构如图 1 所示。

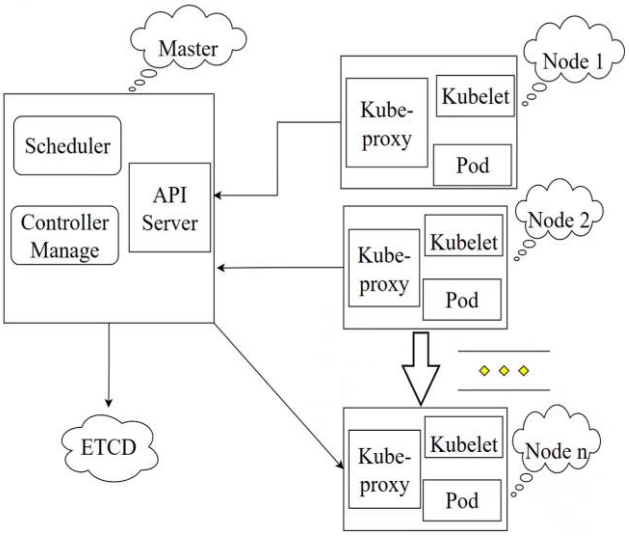

图 1 Kubernetes 集群架构图

（2）集群中企业用户权限与资源进行交互设计

企业用户该模块使用 kubeconfig 文件连接到集群，并通过 RoleBinding 或 ClusterRoleBinding 接收角色权限。Role 限制在特定命名空间内的权限，ClusterRole 提供跨多个命名空间的权限。通过这些绑定，用户可以在其被授权的区域内对资源进行操作。

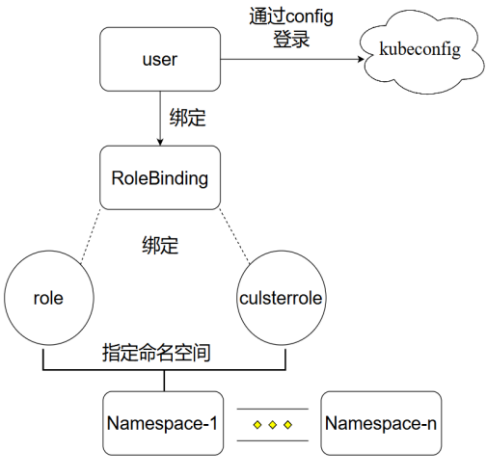

图 2 通过 RBAC 权限模型与资源进行交互的示意图

（3）Web 模块设计

模块使用 heml 部署 WordPress 实现，企业用户通过 WordPress 来定义应用程序的配置参数，并将这些参数注入到应用程序的部署配置中，实现应用程序的参数化配置。

（4）配置管理模块设计

在 Kubernetes 中，daemonset 确保在集群的每个节点上运行指定的 Pod 副本，statefulset 用于管理有状态的应用，deployment 负责无状态应用的声明式更新，而 secret 用于存储敏感数据，如密码。configmap 用于存储和管理非机密性配置数据。组件交互如图 3 所示。

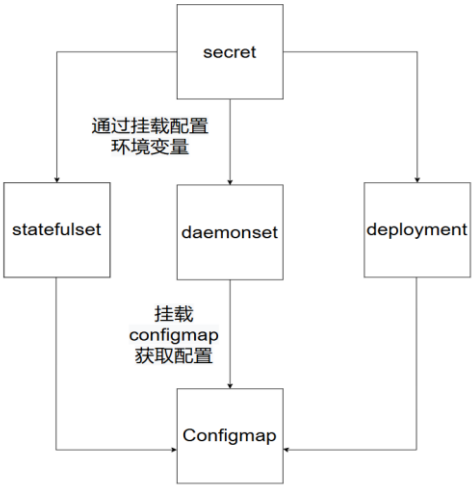

图 3　管理组件交互图

（5）自动化配置管理设计

集成开发环境使用了 Git 进行版本控制，GitLab 用于代码托管与协作平台，Harbor 用于管理和存放 Docker 镜像，Jenkins 于自动化构建、测试和部署软件项目。它可以与 GitLab 集成，通过 GitLab 触发 Jenkins 的构建任务，实现自动化的持续集成和持续部署流程，并最终被部署到由四个实例组成的环境中。该模块可以根据用户的配置自动创建和管理相关的 k8s 资源。

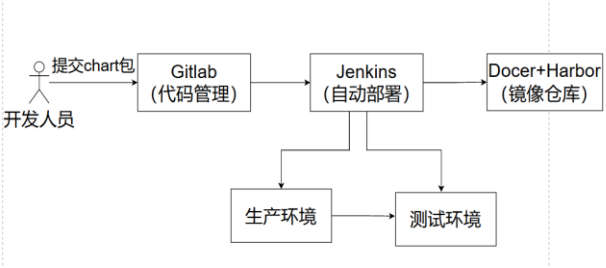

图 4　gitlab+jenkins+harbor 自动化部署运行结构图

（6）监控和日志模块设计

监控模块用于实时监视应用程序的运行状态、性能指标和健康状态，以便及时发现和解决问题。日志模块用于收集、存储和分析应用程序的日志信息，帮助开发人员和运维团队更好的进行故障排查，该模块以 Prometheus 和 kibana 技术为基础实现，结构如图所示。

（7）系统稳定性设计

通过 Ingress 实现负载均衡从而实现系统的拓展性。Ingress 可以帮助集群在不同的负载情况下自动的调节资源并保证系统在出些一些小故障的时候还能正常运行从而提升系统的稳定性。配置 hpa 资源，设置 pod 的 cpu 和内存的利用率，hpa 会按照配置文件定时的监控 pod 资源利用率并按照配置的资源实现自动扩展 pod 以实现高可用。结构如图 6,7 所示。

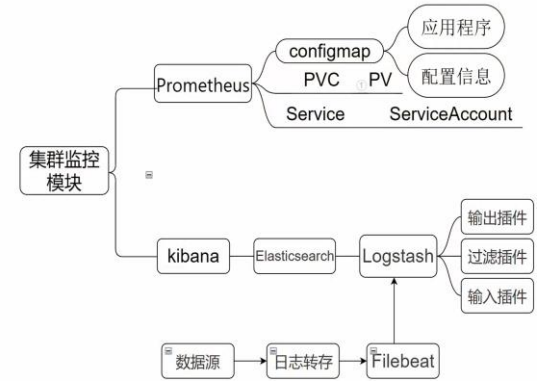

图 5　监控和日志模块结构图

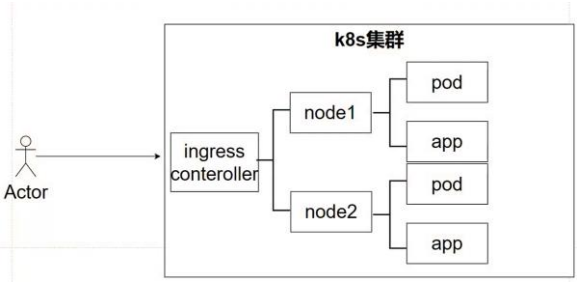

图 6　系统扩展结构图

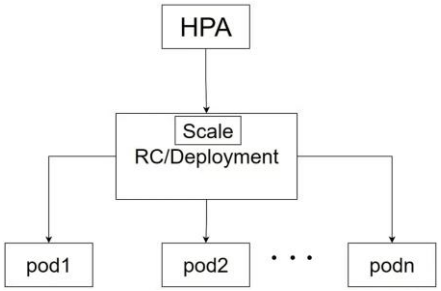

图 7　系统高可用结构图

4　Web 应用的自动化部署方案实现

本方案基于 Kubernetes 容器编排工具进行容器化进行研究开发，使用持续集成工具（Jenkins、GitLab、harbor）来实现自动化部署流程，包括代码构建、测试、打包和部署等环节。

4.1　搭建 Kubernetes 集群

根据上述设计，明确了方案的大体框架。需要设计一个基于 k8s 的系统架构，其中包括 Master 节点和多个 node 节点。Master 节点是 k8s 集群的控制节点，负责整个集群的管理和控制，node 节点是 k8s 集群的工作负载节点，负责运行容器和作为系统冗余。当某

个 node 节点宕机，其他的 pod 会被 Master 分配到其他节点上。

（1） Kubernetes 集群步骤如下

① 基础环境配置

```
# service iptables stop //关闭防火墙
# setenforce 0                  // selinux 安全策略
# swapoff -a          //关闭 swap 分区
```

② 安装 Docker

安装最新版 docker

```
#yum install -y docker-ce //安装最新版 Docker
//cri 环境操作
```

③ Kubernetes 集群安装与初始化。

```
#所有节点定制阿里云的关于kubernetes的软件源
#yum install kubeadm-1.27.3 kubectl-1.27.3 kubelet-1.27.3 -y
//master 环境软件部署
#yum install kubeadm-1.27.3 kubectl-1.27.3 kubelet-1.27.3 -y
//node 环境软件部署
# master 节点初始化
kubeadm init --kubernetes-version=1.27.3 \
--apiserver-advertise-address= <Master 节点 IP> \
--image-repository registry.aliyuncs.com/ google_
containers \
#node 节点加入集群
kubeadm join <Master IP>:6443 --token <token>
\--discovery-token-ca-cert-hash sha256:<hash>
```

④ Kubernetes 集群互通。

```
# 安装 calico 网络组件
kubectl apply -f
https://raw.githubusercontent.com/projectcalico/calico/master/
manifests/calico.yaml
```

4.2 用户管理

用户通过 config 登录进入使用 kubeconfig 并连接到 k8s 集群中，集群通过 RoleBinding 来对每一个用户进行权限绑定并给予相对于的权限操作。角色绑定分为集群角色绑定和命名空间角色绑定。集群角色绑定可以使指定用户能在整个集群内执行特定的命令操作。本模块使用到 openssl 技术，该项提供了一系列的加密算法和安全功能，以该技术的数字证书生成技术为基础实现该模块。

（1） 用户管理模块实现步骤如下

① # mkdir/etc/kubernetes/pki/user/test //创建一个存放用户证书目录

② 配置 openssl 文件并创建生成用户秘钥文件和用户证书。

```
# openssl genrsa -out 用户名.key 2048 //使用 openssl 工具
创建用户秘钥文件
# openssl req -new -key example.key -out example.csr -config
openssl.cnf //用 openssl 工具生成用户证书请求文件
# openssl x509 -req -days 365 -in example.csr -signkey
example.key -out example.crt -extensions req_ext -extfile
openssl.cnf //
```

使用 openssl 生成好用户秘钥文件和用户证书的请求文件后，使用以下命令来定义 CA 文件路径和秘钥文件以此来签署用户证书并生成用户证书，以此实现权限隔离。

4.3 Web 应用模板部署

本模版以 WordPress 作为 web 应用模板来实现用户自动化部署 web 界面，Helm 是 Kubernetes 的包管理工具，安装后可以简化程序的部署方式和管理。创建 HelmChart 用于定义 WordPress 所需的 k8s 资源，同时为了使得 HelmChart 更灵活，配置一个配置文件 values 文件用于定制部署 WordPress 应用程序的配置。WordPressHelmChart 用于进行参数配置和根据实际需求进行拓展。WordPress 结构如图所示。

```
[root@k8s-master opt]# tree WordPress/
WordPress/
├── charts
├── Chart.yaml
├── templates
│   ├── mysql-deployment.yaml
│   ├── wordpress-deployment.yaml
│   ├── wordpress-mariadb-service.yaml
│   └── wordpress-service.yaml
└── values.yaml

2 directories, 6 files
```

图 8 WordPress 目录结构图

（1） Web 应用模板部署步骤如下。

① 部署 helm，实现 k8s 集群中软件包的管理。

```
#下载 helm 文件
wget
https://get.helm.sh/helm-v3.14.2-linux-amd64.tar.gz
```

② 配置 mysql，将网站数据存入到指定的表列行中，确保网站的加载和运行和存储和保存博客的内容。

```
spec:
containers:
- name: mysql
image: mysql:5.7  # MySQL 镜像版本为 5.7
env:
- name: MYSQL_ROOT_PASSWORD
value: {{ .Values.mysqlRootPassword }}  # 通过 Helm
Values 指定 MySQL 的 root 密码
- name: MYSQL_DATABASE
value: {{ .Values.mysqlDatabase }}  # 通过 Helm Values
指定要创建的数据库名称
ports:
- containerPort: 3306  # MySQL 默认端口号为 3306
```

③ wordpress 配置,charts 存放 Helm 的图表依赖，Chart.yaml 包含了 helm 的基本信息，template 作为 k8s 的 yaml 模板文件用于部署 MySQL 数据库，WordPress 应用，存储 WordPress 数据和定义服务访问 WordPress。

```
apiVersion:v1     //指定正在使用的 KubernetesAPI 版本
kind:Service
metadata:
name:{{.Release.Name}}-wordpress      //使用 Helm 模板变
量创建的服务的名称
spec:
```

```
selector:
app:wordpress                    //选择 wordpress 的 pod
type:NodePort                    //在所有节点的特定范围内随机选择的端口上显示服务
```

4.4 配置管理模块部署

配置管理负责管理和维护应用程序在集群中的配置信息，用户可以通过 ConfigMap 和 Secret 获取配置信息。StatefulSet、DaemonSet 和 Deployment 可以直接使用 ConfigMap 来配置应用程序，Secret 用于存储相对敏感的配置信息，确保配置数据的安全性和可靠性。

（1） 配置管理模块实现步骤如下

① 创建 configmap 服务

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: mysql-config    # ConfigMap 的名称为 mysql-config
data:
  my.cnf: |
    [mysqld]
    innodb_log_files_in_group=3   # InnoDB 日志文件组中的文件数为 3
    lower_case_table_names=1   # 表名大小写不敏感
    innodb_buffer_pool_size=4G   # InnoDB 缓冲池大小为 4GB，用于缓存存储引擎操作中的数据和索引
    max_allowed_packet=30M   # MySQL 数据库接收的最大数据包大小为 30MB
```

② 创建 secret 服务用于存储敏感信息，例如密码、OAuth 令牌和 ssh 密钥等，把 pod 访问的加密数据存入 etcd 中。apiVersion: v1

```
kind: Secret
metadata:
  name: mysql-root-secret #Secret 的名称为 mysql-root-secret
  type: Opaque   # Secret 类型为 Opaque 表示密文数据
data:password: cGFzc3dvcmQxMjM=   #加密后的密码
```

4.5 自动化配置管理设计

自动化配置模块通过 gitlab+jenkins+harbor 实现自动化部署。Gitlab 与 jenkins 集成加上配置插件，使代码进入环境后自动激活并且 jenkins 进入构建操作并开始自动化构建流程。构建完成后 jenkins 把生成的 docker 镜像放入 harbor 中进行存储，最后存储在 harbor 中的镜像可以在环境中进行自动化部署。

（1）自动化配置部署步骤如下

① 部署 Gitlab，在自动化部署中负责管理存放代码，用户将代码提交到 GitLab 中，并通过 GitLab 触发 Jenkins 进行自动化构建。

```
# helm repo add gitlab https://charts.gitlab.io //添加 gitlab 的 Helm 仓库
#helm install gitlab gitlab/gitlab --namespace gitlab //GitLab 应用安装到 Kubernetes 集群中
#通过查看集群内的 Gitlab 服务端口登录
```

② 部署 jenkins 服务。在自动化部署中负责自动化构建流程的执行。Jenkins 接收到 Gitlab 信号后会根据配置，自动构建流程，拉起代码，打包应用等操作并把生成的 docker 镜像推送到 harbor 中。

```
# replicas: 1
selector:
  matchLabels:
    app: jenkins // #  指定部署的 Pod 副本数量为 1，并通过标签选择器匹配该 Deployment 管理的 Pod
image: jenkins/jenkins:latest
imagePullPolicy: IfNotPresent
name: jenkins
  privileged: true #  指定镜像为 jenkins，设置镜像拉取策略为 IfNotPresent，并设置容器的安全上下文，其中 runAsUser 和 runAsGroup 都为 0，privileged 为 true
#通过查看集群内的 Jenkins 服务端口登录
```

③在 Gitlab 添加个人访问令牌后进入 Jenkins 页面配置 Jenkins Job，换 Gitlab api 令牌 添加 Gitlab 主机 url,使得 Jenkins 集成 GitLab 实现自动化。

④#查看解锁 Jenkins 密码

```
kubectl exec jenkins-xxx-xxxx -- cat /var/jenkins_home/secrets/initialAdminPassword
```

⑤ 部署 harbor 服务 Harbor 作为 docker 镜像仓库，被用于存储和管理 docker 镜像。在自动化部署中负责存储由 jenkins 构建生成的 docker 镜像同时部署时可以从 harbor 中拉取镜像。

```
#Wget https://github.com/goharbor/harbor/releases/download/v2.8.5/harbor-online-installer-v2.8.5.tgz //下载 harbor 文件
```

⑥ 修改 harbor.Yaml 配置，设置成主机一致配置。

```
Hostname：  //主机 ip
http：
Port：80 //http 配置
```

4.6 监控和日志部署

通过 Prometheus 对集群进行实时监控并利用 elkstack 实现对日志的收集管理，把集群内的容器状态以网页的方式直观的展示出来。Filebeat 负责从各个节点收集日志文件，从而提高日志处理效率。以这些程序配置能确保对集群的全面监控，能有效的提高运维人员的排错效率。

（1） 监控服务实现步骤如下

① 配置 configmap，configmap 作为 k8s 两种特殊类型的卷之一，configmap 对象用于管理配置容器的配置提供配置文件，在这用于支持 Prometheus 的运行和配置。

```
#配置 Prometheus 抓取的目标为 localhost:9090
  scrape_configs:
- job_name: 'prometheus'
static_configs:
```

```
- targets: ['localhost:9090']
#抓取 cadvisor 目标的 Kubernetes SD 配置，以 https 协议进行通信，并配置 TLS 证书相关信息
 - job_name: "cadvisor"
kubernetes_sd_configs:
- role: node
scheme: https
tls_config:
ca_file:/var/run/secrets/kubernetes.io/serviceaccount/ca.crt //
# 将配置应用到集群中
kubectl apply -f configmap.yaml //
```

② 部署 pv 于 pvc，pv 作为集群中的存储，属于集群资源，生命周期和它们的 pod 时相互独立，pvc 作为用户对存储的请求，pvc 和 pv 一对一的映射。在这的用处是解决数据持久化问题。以下是 Master 节点的操作。

```
# 安装 NFS 相关软件包
yum -y install nfs-utils rpcbind
# 创建存储 Prometheus 的目录
mkdir /data/Prometheus
# 编辑 exports，将/data/prometheus 目录暴露给 NFS 客户端并设置读写权限和禁用 root 用户映射
vim /etc/exports
/data/prometheus*(rw,sync,no_root_squash)
# 重启 NFS 服务
systemctl restart nfs
# 查看 exports 文件配置
exports -r
# 显示 NFS 服务器上共享的目录及权限信息
showmount -e
```

③ Node 节点操作

```
# 安装所有 nfs 开头的软件
yum -y install nfs*
```

④ 创建一个 Prometheus 配置文件，以下是主要配置。

```
# 挂载 Prometheus 配置目录，规则目录，目标目录
- mountPath: "/etc/prometheus"
name: prometheusconfig
- mountPath: "/etc/prometheus/rules"
name: rules
- mountPath: "/etc/prometheus/targets"
name: targets
```

⑤ 配置 service，Prometheus 服务文件，确保服务正常运行。

```
# 定义一个 prometheus 的服务，运行在集群中的 prometheus 程序
kind:Service
metadata:
creationTimestamp:null
labels:
Prometheusapp:prometheus
name:prometheus
namespace:Prometheus
```

（2） 日志服务实现步骤如下

① 配置 elasticsearch-logging，Elasticsearch 作为开源的分布式搜索和分析引擎，用于存储和搜索大量的结构化和非结构化数据。在本模块中用于存储日志数据。

```
#部署的数据信息
apiVersion:apps/v1
kind:Deployment
metadata:
generation:1
labels:
app:elasticsearch-logging
version:v1
```

② 配置 logstash，用于实时处理和解析日志数据，并把数据传输到 Elasticsearch 中。

```
# Logstash 的配置文件内容
apiVersion:v1
kind:ConfigMap
metadata:
name:logstash-config
data:
```

③ 部署 nginx 和 filebeat。Nginx 内置了丰富的性能指标和状态信息，通过监控 Nginx 可以实时了解服务的运行状态。filebeat 作为轻量型日志传输和分析，用于收集和分析日志，然后输出到 Elasticsearch 中，最后数据通过 kibana 可视化展示出来。

```
# Filebeat 的配置文件内容
apiVersion:v1
kind:ConfigMap
metadata:
name:filebeat-config-to-logstash
data:
#定义容器名称为 nginx
spec:
containers:
-name:nginx
#设置容器监听端口和使用 TCP 协议
ports:
-containerPort:80//容器监听端口为 80
protocol:TCP//使用 TCP 协议
volumeMounts:
-name:logm
```

④ 部署 kibana，kibana 为 Elasticsearch 提供数据可视化界面，通过查看网页数据能直面的了解到每个 pod 的状态。

```
# name:kibana//定义 Deployment 的名称为 kibana
labels:
name:kibana//设置标签为 kibana
spec:
replicas:1//设置副本数为 1
selector:
matchLabels:
name:kibana//选择器匹配标签为 kibana
template:
metadata:
---
```

⑩ 查看 Kibana Service 的详情。

```
#查看 Kibana 服务端口 kubectl get svc
# 登录 elastic 192.168.120.60:32633
```

4.7 系统的扩展性和高可用性实现

Ingress 是 Kubernetes 中用于管理外部访问的资源，通过 Ingress Controller 将外部请求路由到集群内部的服务。HPA 是 Kubernetes 提供的一种自动扩展机制，能够根据监控指标动态调整 Pod 的副本数量。当监控指标达到设定的阈值时，HPA 会自动增加 Pod 的副本数量，以应对流量增加的情况，保证系统高可用性。

（1） Ingress 负载均衡实现步骤如下

① #下载 Ingress 配置文件 kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress nginx/controller-v1.8.2/deploy/static/provider/cloud/deploy.yaml

② 修改配置文件，官方使用的是 loadBalancer 需要公网 ip,如果国内来取不到镜像，把 registry.k8s.io 地址改为 k8s.dockerproxy.com 即可。

③ # 使用 wait 命令检测是否为 ready kubectlwait--namespaceingress-nginx\
--for=condition=readypod\
--selector=app.kubernetes.io/component=controller\
--timeout=120s

④ 检查 Ingress 容器运行状态 kubectl get pod -n ingress-test

（2） HPA 高可用实现步骤如下

① # 下载 metrics-server 组件配置文件

#Wget https://github.com/Kubernetes-sigs/metricsserver/releases/latest/download/components.yaml-Ometrics-servercomponents.yaml

② 修改进行地址为国内的地址

5 结束语

本方案基于 Kubernetes 容器编排工具进行容器化进行研究开发，实践证明，自动化部署方案能够大大提高运维效率，实现了服务器的高可用性，方案可以为企业服务器架设提供参考。

参 考 文 献

[1] 刘洋,赵瑞锋,李波,等.基于 Docker 技术的静态安全分析云计算应用[J].电力科学与 技术学报,2021,36(04):181-187.

[2] 杜金涛,董建刚,承华青.Kubernetes 水平伸缩策略的改进研究[J].现代电子技术,2023,46(10):129-136.

[3] 罗梓桉. 基于kubernetes的高可用文件传输系统[D].电子科技大学,2023.

[4] 栗晓晗. Docker镜像安全及运行异常检测研究与应用[D].西南交通大学,2024.

[5] 刘启东. 基于Prometheus的边缘机房监控系统设计与实现[D].浙江大学,2022.

[6] 张艳清.基于Kibana系统的机器人故障数据分析[J].科技与创新,2021(19):23-25.

[7] 黄鹏,陈聪,高雅濛.基于Gitlab流水线的可视化编排工具[J].电脑编程技巧与维护,2024(03):144-146+157.

[8] 苏开印. 基于Jenkins的自动化单元测试平台研究与实现[D].北京邮电大学,2024