结束语

欢迎同学们历经千辛万苦，终于来到了最后一章。这一章，我们将串联起前面所学的知识点，和 `Jenkins + Nexus` 做一套流程的集成。

在这里，我准备了一套简单的增删改查项目。前端技术栈为 `React + craco` ，后端技术栈为 `MySQL + eggjs` 。

让我们直接开始吧！

操作步骤

在开始之前，我们需要在 `Kubernetes` 集群内再加一台 `Node` ，起名为 `node2` 。具体流程请参考之前的章节。`Node2` 的主要用途是用于部署 `MySQL` 使用。

1. 项目仓库

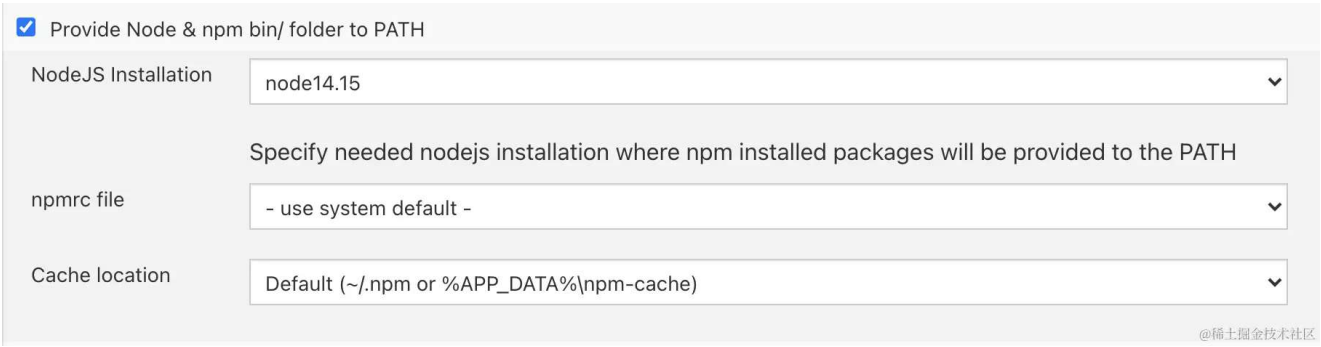
先来看下这次项目部署所需要的仓库：gitee.com/organizatio...。 其中， `k8s-demo-frontend` 是前端项目， `k8s-demo-backend` 是后端项目。

2. 构建 & 部署前端应用

第一步我们先部署前端应用，先将前端跑起来。

我们前往 `Jenkins` ，新建一个任务，起名为 `demo-frontend` 。接着配置任务的Git代码源，让 Jenkins 可以拉取代码。因为我们目前是公开项目，所以还不需要配置私有仓库认证。

按照之前的方式，勾选 构建环境 => Provide Node & npm bin/ folder to PATH 选项，给你执行的任务增加 Nodejs 运行环境



继续添加构建脚本，让 Jenkins 构建镜像。找到 构建 => 添加构建步骤 => Excute Shell ，填写以下脚本：

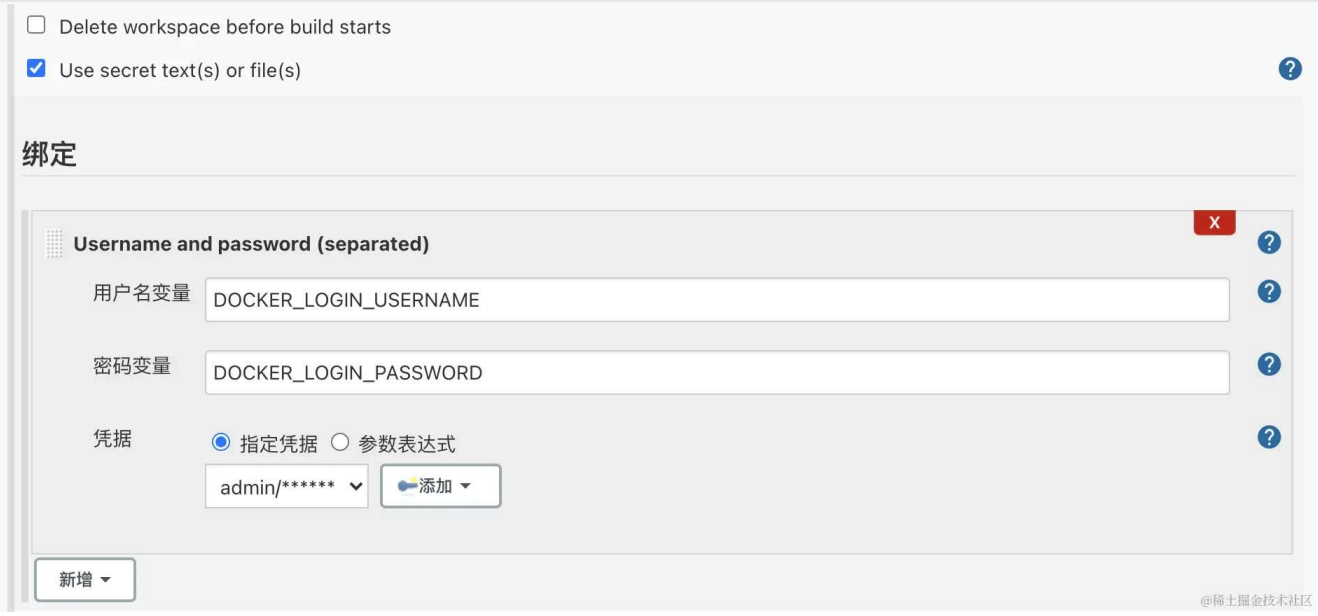
脚本首先使用 npm run build 对代码进行编译打包，随后使用 docker build 命令构建镜像。最后推送镜像到镜像库内。

shell 复制代码

```
#!/bin/sh -l

time=$(date "+%Y%m%d%H%M%S")
npm install --registry=https://registry.npm.taobao.org
npm run build
docker build -t 172.16.81.7:8082/frontend-app:$time .
docker login -u $DOCKER_LOGIN_USERNAME -p $DOCKER_LOGIN_PASSWORD 172.16.81.7:8082
docker push 172.16.81.7:8082/frontend-app:$time
```

因为推送镜像需要 docker login ，我们还需要在 Jenkins 端配置下 docker 登录信息。配置文件方式如下图，和之前的章节无异。



保存后执行，即可生成前端镜像。

镜像生成后，我们还需要去k8s集群内部署下这个镜像。

前往集群节点，新建一个文件。叫做 `demo-frontend.yaml` ，输入以下内容。**镜像地址换成刚才 Jenkins 构建后的镜像地址。**

yaml 复制代码

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: demo-frontend
spec:
  selector:
    matchLabels:
      app: demo-frontend
  replicas: 1
  template:
    metadata:
      labels:
        app: demo-frontend
    spec:
      imagePullSecrets:
      - name: private-registry
      containers:
      - name: frontend-app
        imagePullPolicy: Always
        image: 172.16.81.7:8082/frontend-app:20210117162137
        ports:
          - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: demo-frontend-service
```

```
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
  type: NodePort
```

保存后退出，使用 `kubectl apply` 命令部署前端服务。部署完毕后，使用 `kubectl get svc` 命令来获取下服务的端口。

```
demo-frontend-service    NodePort    10.110.202.160    <none>        80:32384/TCP    4h5m
```


可以看到，此时前端已经部署成功了。使用浏览器打开即可看到页面。

3. 部署 & 初始化MySQL

我们在开头时，添加了一台全新的 `Node` 节点，这台节点机器用于部署MySQL服务。我们可以给节点加污点，让除了特定的服务，其他服务都不可以部署上去。

这里添加一条污点， `key` 等于 `MySQL` ， `value` 等于 `true` 。

shell 复制代码

```
kubectl taint nodes node2 mysql=true:NoSchedule
```

添加完毕后，我们就可以放心部署 `MySQL` 了。不过在开始部署之前，我们还需要去 `Node2` 节点给 `MySQL` 的数据创建一个文件夹。我们会将本地的文件夹挂载进 `MySQL` 容器内，以方便 `MySQL` 数据可以持久化。

shell 复制代码

```
mkdir /var/lib/mysql && mkdir /var/lib/mysql/data
```

还需要给 `MySQL` 容器添加挂在访问密码。这里我们将密码存入 `secret` 内保存。

shell 复制代码

```
kubectl create secret generic demo-mysql-auth \
--from-literal=password=367734
```

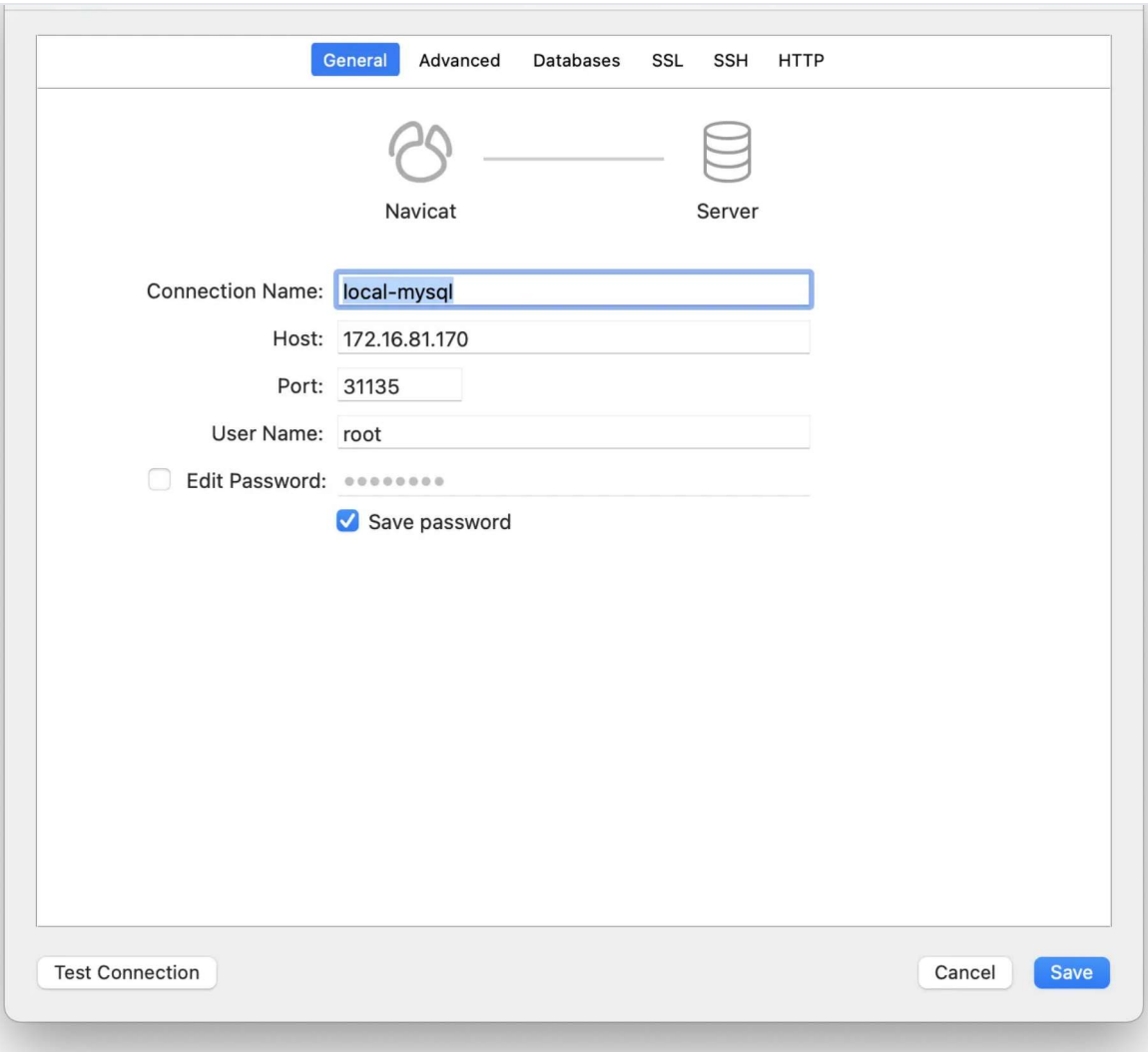
此时我们就可以开始部署MySQL了。新建一个YAML文件，输入以下内容。这里给 `MySQL` 容器添加了污点对应的容忍度，密码也挂载了进去，设置了默认端口 `3306` 。

yaml 复制代码

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: demo-mysql
spec:
  replicas: 1
  selector:
```

```
    metadata:
      labels:
        app: demo-mysql
    spec:
      tolerations:
      - key: "mysql"
        operator: "Equal"
        value: "true"
        effect: "NoSchedule"
      containers:
      - name: demo-mysql
        image: mysql:5.6
        imagePullPolicy: IfNotPresent
        args:
        - "--ignore-db-dir=lost+found"
        ports:
        - containerPort: 3306
        volumeMounts:
        - name: mysql-data
          mountPath: "/var/lib/mysql"
        env:
        - name: MYSQL_ROOT_PASSWORD
          valueFrom:
            secretKeyRef:
              name: demo-mysql-auth
              key: password
      volumes:
      - name: mysql-data
        hostPath:
          path: /var/lib/mysql
          type: Directory
---
apiVersion: v1
kind: Service
metadata:
  name: demo-mysql-service
spec:
  type: NodePort
  ports:
  - port: 3306
    protocol: TCP
    targetPort: 3306
  selector:
    app: demo-mysql
```

部署成功后，我们可以使用 `Navicat` 等工具访问数据库了。数据库的 `host` 是 `service` 的地址，用户是 `root` ，密码则是我们挂载进去的密码。



可以访问数据库后，使用我们的初始化 `sql` 文件，初始化以下数据库和表结构。这里的 `sql` 创建了一个名称为 `demo-backend` 的数据库，数据库内创建了 `user` 表。并加入了4个数据库字段。

sql 复制代码

```sql
SET NAMES utf8mb4;
SET FOREIGN_KEY_CHECKS = 0;

CREATE DATABASE IF NOT EXISTS `demo-backend` DEFAULT CHARSET utf8 COLLATE utf8_general_ci;
USE `demo-backend`;

DROP TABLE IF EXISTS `users`;
CREATE TABLE `users` (
  `id` int(11) NOT NULL AUTO_INCREMENT COMMENT 'ID',
  `name` varchar(255) NOT NULL COMMENT '姓名',
  `age` int(11) NOT NULL COMMENT '年龄',
  `sex` varchar(255) NOT NULL COMMENT '性别：1男 2女',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=9 DEFAULT CHARSET=utf8;

SET FOREIGN_KEY_CHECKS = 1;
```

sql	message
SET NAMES utf8mb4	OK, Time: 0.001000s
SET FOREIGN_KEY_CHECKS = 0	OK, Time: 0.002000s
CREATE DATABASE IF NOT EXISTS `demo-bac...	OK, Time: 0.002000s
USE `demo-backend`	OK, Time: 0.002000s
DROP TABLE IF EXISTS `users`	OK, Time: 0.003000s
CREATE TABLE `users` (	OK, Time: 0.009000s
SET FOREIGN_KEY_CHECKS = 1	OK, Time: 0.002000s

4. 构建 & 部署后端应用

最后一步就是部署后端服务了。首先第一步，也是在 `Jenkins` 端新建项目，具体流程和前端应用一样。构建脚本需要进行修改：

因为这里没有静态资源需要构建，所以直接将源码目录拷贝进容器即可：

sql 复制代码

```sql
#bin/bash
time=$(date "+%Y%m%d%H%M%S")
npm install --registry=https://registry.npm.taobao.org
docker build -t 172.16.81.7:8082/backend-app:$time .
docker push 172.16.81.7:8082/backend-app:$time
```

执行任务，镜像 `push` 完成代表成功。

镜像准备好后，我们需要在k8s端部署下服务。在部署之前，我们先将数据库相关信息存入 `configmap` ，然后挂载进后端服务：

yaml 复制代码

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: mysql-config
data:
  host: 'demo-mysql-service'
  port: 3306
  username: 'root'
  database: 'demo-backend'
```

存好后就可以部署后端服务了，以下是配置文件。内容拉取了一个后端服务镜像，并将数据库账号和端口服务地址通过 `configmap` 传入了进去。

yaml 复制代码

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: demo-backend
spec:
  selector:
    matchLabels:
      app: demo-backend
  replicas: 1
  template:
    metadata:
      labels:
        app: demo-backend
    spec:
      imagePullSecrets:
      - name: private-registry
      containers:
      - name: backend-app
        imagePullPolicy: Always
        image: [镜像地址]
        ports:
          - containerPort: 7001
        env:
        - name: MYSQL_HOST
          valueFrom:
            configMapKeyRef:
              name: mysql-config
              key: host
        - name: MYSQL_PORT
          valueFrom:
            configMapKeyRef:
              name: mysql-config
              key: port
        - name: MYSQL_USER
          valueFrom:
            configMapKeyRef:
              name: mysql-config
              key: username
        - name: MYSQL_DATABASE
          valueFrom:
            configMapKeyRef:
              name: mysql-config
              key: database
---
apiVersion: v1
kind: Service
metadata:
  name: demo-backend-service
spec:
  selector:
    app: demo-backend
  ports:
  - protocol: TCP
    port: 7001
```

保存后，使用 `kubectl apply` 即可让服务生效。

接着访问下前端界面，功能正常代表部署成功。

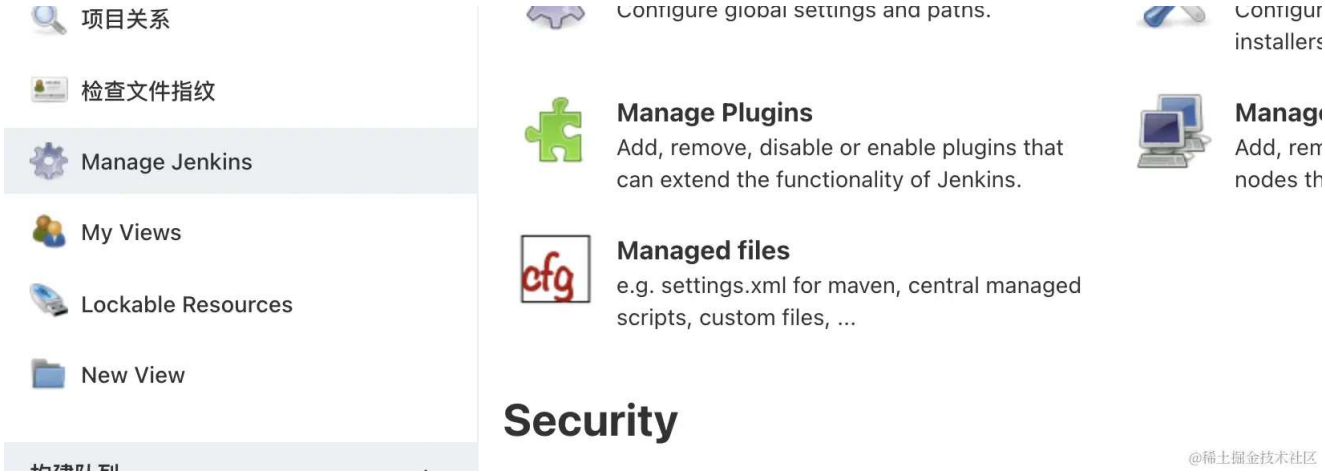
5. 集成 Jenkins

在前面的服务部署成功后，我们还需要使用 `Jenkins` 直接一键执行构建和部署。

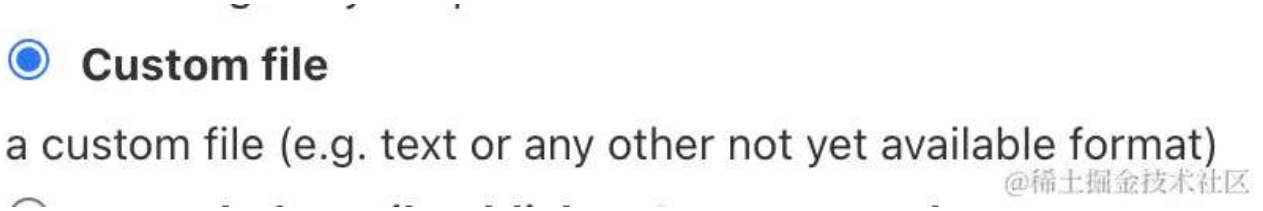
我们在前面部署镜像时，都是在集群内直接操作。可是一般情况下，Jenkins 和 k8s 并不在一台机器上。那我们如何远程操作集群呢?

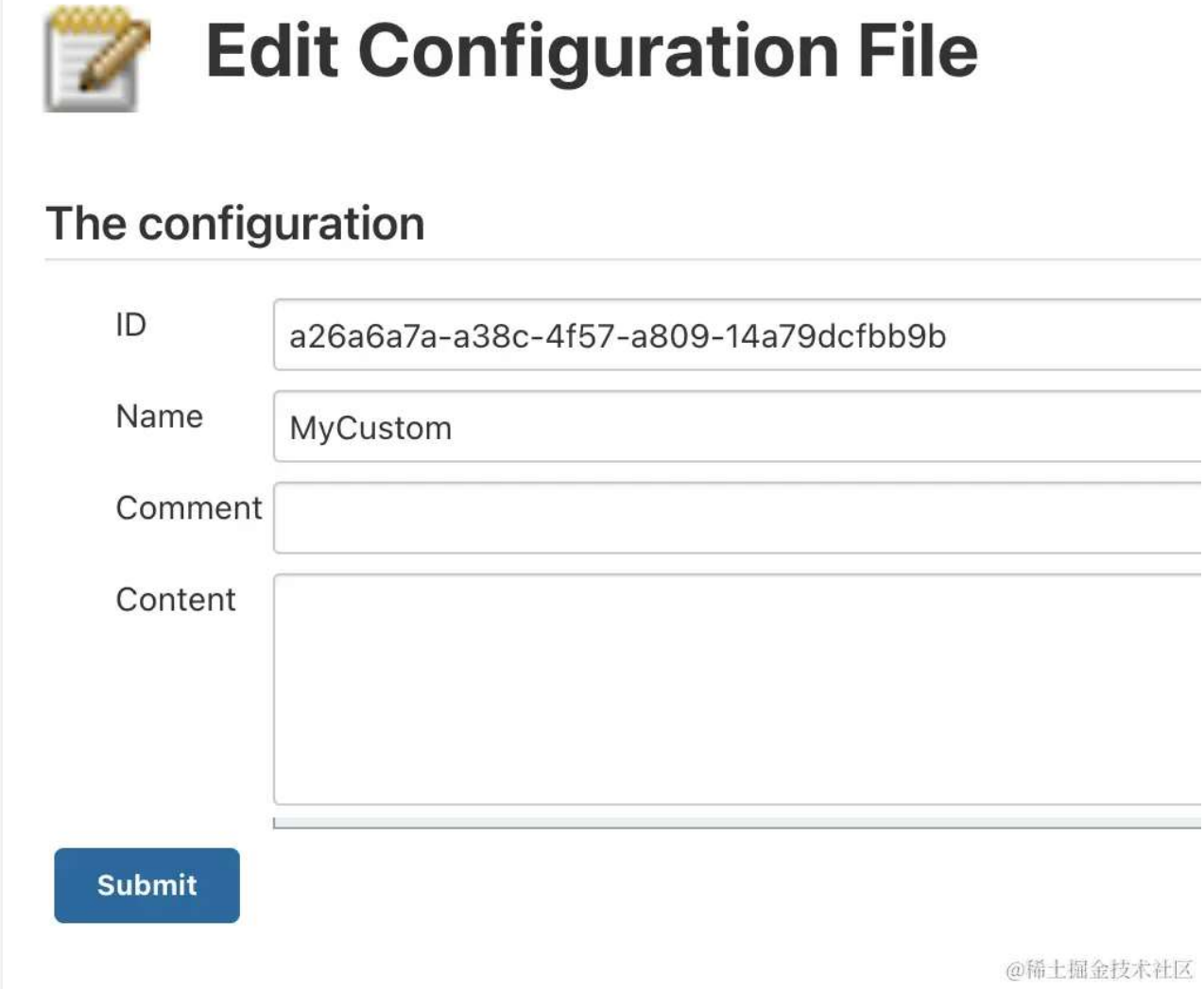
这里可以使用 `kubectl 的 --kubeconfig` 命令，传入集群的配置文件即可远程操作。只要保证Jenkins和k8s集群网络互通即可。配置文件的路径也很好找，位于集群机器的 `~/.kube/config` 文件。

这样，我们在 `Jenkins` 端添加一个全局配置文件，方便任务使用。



找到 `Manage Jenkins => Managed files` 。选择右边的 `Add a new Config` ：



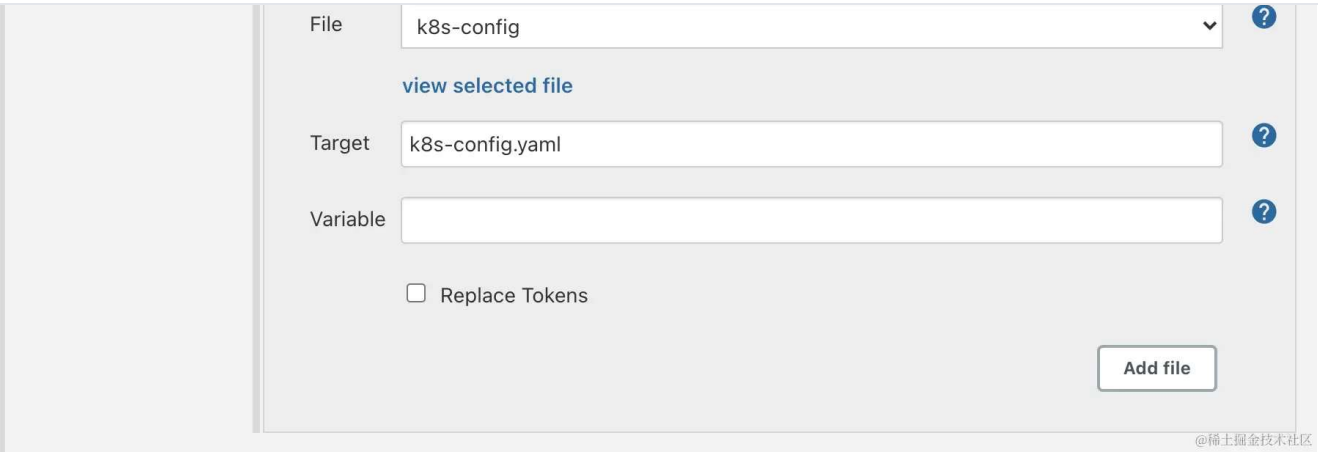


随后，我们还需要在 `Jenkins` 机器上安装 `kubectl`，只安装 `kubectl` 即可。

shell 复制代码

```
cat <<EOF > /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=http://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86_64
enabled=1
gpgcheck=0
repo_gpgcheck=0
gpgkey=http://mirrors.aliyun.com/kubernetes/yum/doc/yum-key.gpg
        http://mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg
EOF
yum install -y kubectl
```

回到任务编辑界面，找到 `绑定` 一栏，选择我们刚刚配置的配置文件。填写 `target` 一栏，让配置文件输出为文件。



找到命令界面。以前端任务为例，我们在 `docker push` 的命令后，加一条 `kubectl` 执行命令。在这里，直接使用 `kubectl --kubeconfig` 制定配置文件，即可远程操作

shell 复制代码

```shell
kubectl --kubeconfig=k8s-config.yaml set image deployment/demo-frontend demo-frontend=172.16.81.7:8082/frontend-app:$
```

> 我们可以使用 `kubectl set image` 命令快速设置镜像地址版本 格式为：kubectl set image deployment/[deployment名称] [容器名称]=[镜像版本]

保存后执行，提示 `deployment.apps/[deployment名称] image updated` 代表更新完毕。

上一章 下一章

留言

输入评论（Enter换行，Ctrl + Enter发送）

发表评论

全部评论（12）

beckyyy LV.3 JY.5 啊这 1年前

mysql需要配置允许远程连接

点赞 回复 ···

叶_同学 LV.3 JY.4 前端切图仔 1年前

2 回复 ···

SometimesNever JY.3 1年前

argo做cd会不会更方便点

点赞 回复 ···

叶_同学 LV.3 JY.4 前端切图仔 1年前

jenkins 新建任务的时候可以复制旧的项目 cloud.tencent.com 复制之前部署的 demo 就可以，稍微按作者的配置改改就好

点赞 回复 ···

沛公 JY.1 2年前

我的阿里云余额快扣完了，就看你了啊小哥 😂

1 1 ···

王圣松 （作者） 2年前

久等了～已完毕

点赞 回复 ···

用户7242346774... JY.1 2年前

小哥，加油啊，期待这一节

点赞 1 ···

王圣松 （作者） 2年前

久等了～已完毕

点赞 回复 ···

大妈ZoomQuiet JY.1 2年前

是也乎,(￣▽￣)
何时完成?

点赞 3 ···

这太难了 2年前

😄 这难道是大妈

点赞 回复 ···

大妈Zoom... 回复 这太难了 2年前

是也乎,(￣▽￣) 替红薯来查岗了

“😄 这难道是大妈”

点赞 回复 ···

查看更多回复