東方通 | TongTech®

400-650-7088

东方通应用服务器软件
TongWeb V7.0.6
容器云版
用户使用手册

版本 V7-1

北京东方通科技股份有限公司

目录

图片目录

表格目录

版本变更说明

本手册的更新是累积的。因此，最新的手册版本包含对以前版本所做的所有更改。

本手册版本 V7-1 适用于 TongWeb V7.0.6 容器云版。

手册版本	更新内容
V7-1	统一使用 TongWeb V7.0.6 用户使用手册模板

前言

本文档是 TongWeb V7.0.6 容器云版产品的用户手册之一，详细介绍了容器云版的安装及手工配置和管理。本手册的提供了 TongWeb V7.0.6 及云环境安装流程，随后的章节详细说明了如何使用及管理容器云版 TongWeb V7.0.6。

适合的对象

本手册主要适用对象为使用容器云版应用服务器进行应用开发的开发人员，以及生产环境的系统管理人员，应用发布人员等具有 TongWeb V7.0.6 容器云版的使用和配置经验的用户。因此，具备如下的技能可能会有助于使用者更好的理解文档的内容：

1. 熟悉 Linux 常用命令
2. 对 Docker 和 K8s 环境有一定的了解

容器云版用户手册集

容器云版 TongWeb V7.0.6 为您提供的用户手册集包含的文档有：

1. TongWeb V7.0.6 容器云版用户使用手册：详细介绍如何在各个操作系统上安装 TongWeb V7.0.6 容器云版，以及手动配置的操作步骤。
2. TongWeb V7.0.6 容器云版控制台用户使用手册：提供快速安装启动配置等说明
3. TongWeb V7.0.6 授权中心用户使用手册：提供 License 授权的具体说明

技术支持

东方通产品将为您提供全方位的技术支持，您可以通过以下方式获得技术支持：

网址：www.tongtech.com

Support Tel：400-650-7088

邮箱：pqmo@tongtech.com

您在取得技术支持时，请提供如下信息：

1. 您的姓名
2. 您的公司信息
3. 您的联系方式
4. 操作系统及其版本
5. 产品版本号
6. 日志等错误的详细信息

术语说明

HTTP	超文本传输协议(Hypertext Transfer Protocol)
SSL	安全套接层(Secure Sockets Layer)
HTTPS	基于 SSL 的超文本传输协议(Hypertext Transfer Protocol over SSL)
JDK	Java 开发工具包(Java Development Kit)
JDBC	Java 数据库连接(Java Database Connectivity)
XML	可扩展标记语言(Extensible Markup Language)
K8s	Kubernetes

第一章　容器云版 TongWeb V7.0.6

1.1 TongWeb V7.0.6 概述

中间件平台为企业提供了应用开发、部署、运行、维护所需的基础设施，既能减少企业的开发维护成本，又能保障企业应用的安全性和可靠性，同时还能提高企业运营维护的整体效率。

应用服务器 TongWeb V7.0.6 作为满足上述需求的中间件平台，广泛地应用于电信、金融、政府、交通、能源等各种领域的企业应用中。

应用服务器作为基础架构软件，位于操作系统与应用之间，帮助企业将业务应用集成在一个基础平台上，为应用高效、稳定、安全运行提供关键支撑。包括便捷的开发、随需应变的灵活部署、丰富的运行时监视、高效的管理等。

TongWeb V7.0.6 容器云版坚持自主研发，与国产芯片、操作系统、数据库等产业生态伙伴开展广泛兼容性适配，为党政、金融、运营商等核心业务应用搭建高性能、高可用系统提供有力支持。

TongWeb V7.0.6 容器云版提供容器化、云原生等更多新技术功能，提供优质的相关解决方案，在 IT 发展领先行业应用广泛。

1.2 TongWeb V7.0.6 容器云版

TongWeb V7.0.6 容器云版是一款针对容器化及云原生的应用服务器，支持最新的 JavaEE 规范标准如 Servlet4.0、JSP2.3、websocket1.1 等。它为企业应用提供了可靠、可伸缩、可管理和高安全的基础平台。同时具有功能完善、支持国密算法，支持开放标准和基于组件开发、统一配置、轻量等特点，为开发和部署企业应用提供了必需的底层核心功能。

1.3 系统架构

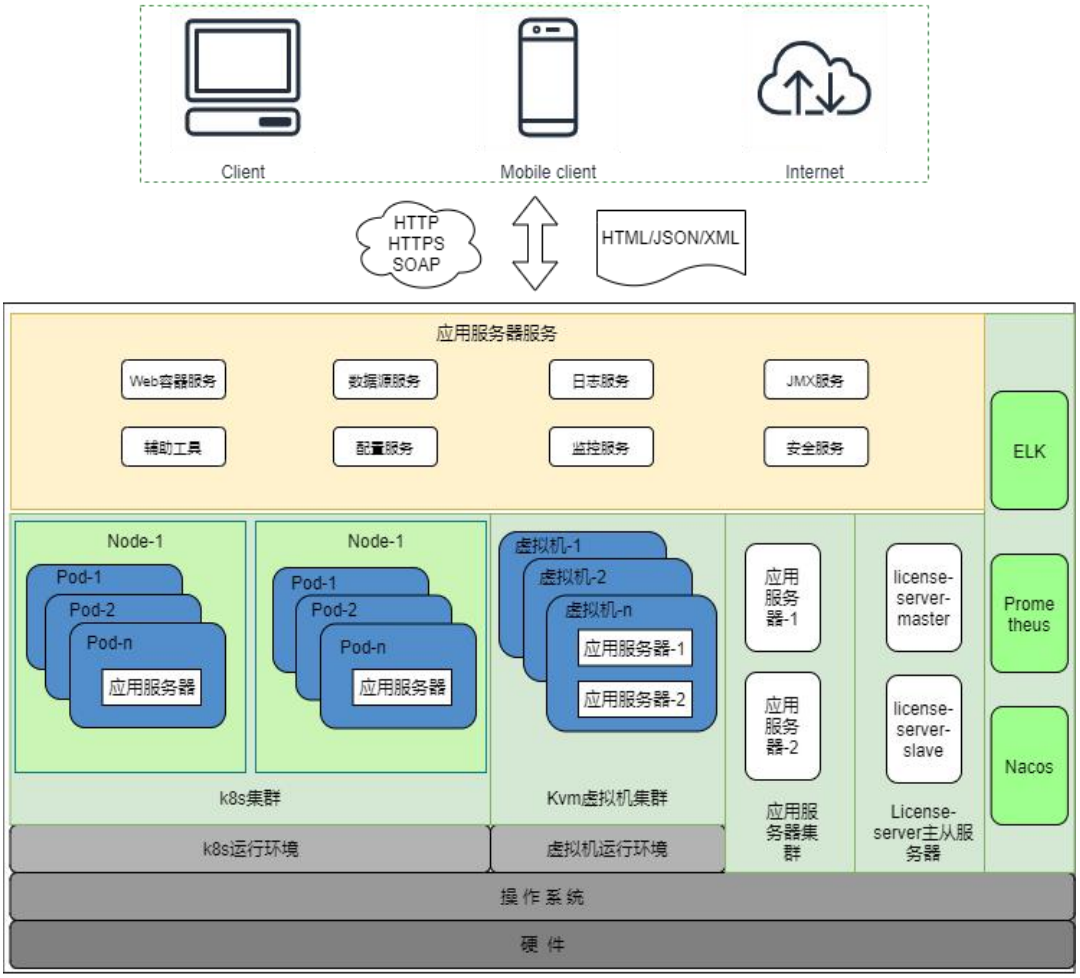


图 1.3- 1：系统架构图

1.4 容器云版特性

TongWeb V7.0.6 容器云版提供了稳定的 Web 应用部署功能。除了体量小，运行快之外，还具有以下功能特性：

1. 丰富的基础功能。
2. 优秀的容器适配。
3. 灵活的配置服务。
4. 成熟的日志服务。
5. 多维度监控服务。
6. 稳定的产品性能。
7. 完善的安全防护。

1.5 支持的运行环境

表 1.5- 1：支持的运行环境

环境	说明
Windows	Windows7,windows10，windows server
Linux	CentOS、Ubuntu、Linux、SUSE
国产平台	龙芯系列、飞腾系列、华为系列、申威系列、海光系列
JDK 版本	Jdk8 以上；针对 HTTP2 需要 8u252 以上
云环境	Docker,K8S（1.13 以上）

1.6 规范支持

表 1.6- 1：支持的 JavaEE 规范

规范/API	版本
Servlet	4.0
JSP	2.3
Expression Language	3.0
WebSocket	1.1

第二章 安装指南

2.1 目录结构说明

表 2.1- 1：目录结构说明

目录名称	说明
autodeploy	服务器默认的自动部署监听目录
bin	服务器启动、停止等脚本文件目录
conf	服务器配置文件所在目录
lib	服务器运行所需的类文件所在目录，主要以 Jar 文件形式存在。
lib/host	用于存放 jar 包，如 nacos 相关的 jar 包，kafka 相关的 jar 包。

lib/app	用于存放用户应用共享的 jar 包
logs	服务器存放日志文件的目录，日志文件包括访问日志文件和服务器日志文件。
tmp	用来存放服务器在运行时的编译后文件

2.2 Windows 平台安装

1. 解压：

   通过解压工具解压 **tongweb-cloud-7.0.C.x.tar.gz**

2. 参数配置：

   在 **env.options** 中有可以设置配置中心的类型，可以根据实际情况修改以下参数:

表 2.2- 1：TongWeb V7.0.6 配置中心参数说明

配置项	说明	默认值
CONFIG_REMOTE_ENABLE	是否启用远程配置	false
CONFIG_REMOTE_TYPE	远程配置中心类型	Nacos
CONFIG_SERVER	配置中心 ip 和端口	127.0.0.1:8848
LOG_NACOS_DATA_ID	Nacos 日志配置的 DATAID	logging
APP_NACOS_DATA_ID	Nacos 应用配置的 DATAID	application
TONGWEB_NACOS_DATA_ID	Nacos TongWeb V7.0.6 配置的 DATAID	tongweb
NACOS_GROUP	日志，应用，Tongweb V7.0.6 的 Group	DEFAULT_GROUP

   在 application.properties 中有 license 的方式配置，具体配置参考 *3.3 License 信息*

3. 运行：

   TongWeb V7.0.6 服务器提供了在 Windows 环境下启动的命令行方式，使用 **${TW_HOME}/bin** 目录下的 **startserver.bat** 启动 TongWeb V7.0.6 服务器。

2.3 Linux 平台安装

1. 解压压缩包：

   tar -zxvf tongweb-cloud-7.0.C.x.tar.gz

2. 参数配置：

   在 **env.options** 中有可以设置配置中心的类型,可以根据实际情况修改参数, 参数配置项如*表 2.2-1 TongWeb V7.0.6 配置中心参数说明*。

   在 application.properties 中有 license 的方式配置，具体配置参考 *3.3 License 信息*

3. 运行：

   TongWeb V7.0.6 容器云版服务器提供了在 Linux 环境下启动的命令行方式，使用 **${TW_HOME}/bin** 目录下的 **startserver.sh** 启动 TongWeb V7.0.6 服务器。

2.4 Docker 环境安装

2.4.1 环境准备

用户需要提前准备好 Docker 环境，Docker 环境在 docker-17.5.0-ce 以上版本。

2.4.2 镜像构建

如果用户拿到的是 TongWeb V7.0.6 容器云版的解压包并且需要运行在 Docker 容器中，就需要进行镜像的构建。

1. TongWeb V7.0.6 容器云版 Docker 环境镜像构建需要使用的软件包清单如下：

tongweb：TongWeb V7.0.6 容器云版根目录。

build_tongweb.sh 脚本文件：用于镜像构建。

Dockerfile 文件：用于镜像构建。

2. 镜像构建

在当前目录下，通过运行如下脚本进行镜像构建。镜像名称格式为**tongweb/tongweb[:tag]**。脚本默认缺省值是 x86 的指令集。如果是 ARM 指令集需要加入第二个参数 arm 指定。指令集和基础镜像的关系见附录 F。

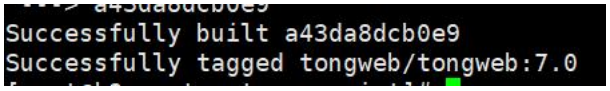
build_tongweb.sh 脚本格式为：

./build_tongweb.sh <镜像名称> [base]

base 是可选参数，缺省值是 x86。如果是华为和飞腾等 ARM 指令集的，需要把 base 传入 arm，[base]具体对应的基础镜像名，参考附录 F 指令集和基础镜像关系。

./build_tongweb.sh tongweb/tongweb:7.0

镜像构建成功后会打印 Successfully 信息



3. 运行容器

1) 用户指定挂载目录

自动部署挂载目录是对应用挂载到容器中运行，需要把应用复制到挂载目录下，日志挂载目录是日志文件挂载的位置。

特别注意：

如果用户使用的是远程配置中心的方式需要在**${TW_HOME}/bin/docker-set-env.env** 修改 nacos 相关的配置，若用户使用的是本地配置的方式无需修改 docker-set-env.env。

docker-set-env.env 的配置参数如下：

CONFIG_REMOTE_ENABLE：是否启用远程配置

CONFIG_REMOTE_TYPE：远程配置中心类型

CONFIG_SERVER：远程配置中心地址和端口

LOG_NACOS_DATA_ID：Nacos 日志功能的 Data ID

APP_NACOS_DATA_ID：Nacos 应用功能的 Data ID

TONGWEB_NACOS_DATA_ID：Nacos TongWeb V7.0.6 配置的 Data ID

NACOS_GROUP=DEFAULT_GROUP：Nacos 日志，应用，TongWeb V7.0.6 的 Group

如果未采用 license server，采用的是 **license.dat** 文件的方式，需要把 license.dat 挂载

进去。docker 运行是额外加**-v license.dat** 挂载目录**:/opt/TongWeb/license.dat:rw**

采用 license server，需要去${TW_HOME}/conf/application.properties 下修改并重新制作镜像或者是修改 Nacos 中心。具体 license 配置可参考第三章配置文件总体说明、Nacos 配置可参考 4.6.3 容器云环境配置。

在当前目录下，通过运行如下命令启动容器(未加 license.dat 的挂载):

docker run --env-file tongweb/bin/docker-set-env.env --name tongweb -p 9060:8088 -v <挂载目录> -d tongweb/tongweb:7.0

自动部署挂载目录: /opt/TongWeb/autodeploy:rw

日志挂载目录: /opt/TongWeb/logs:rw

2) 通过如下命令查看是否启动成功。

docker logs 容器 ID

```
[2020-09-16 04:12:28] [INFO] [license] [许可证过期日期为 2020-12-26]
[2020-09-16 04:12:29] [INFO] [config] [TongWeb初始化成功]
[2020-09-16 04:12:30] [INFO] [web-container] [初始化协议处理器 ["http-nio-8088"]]
[2020-09-16 04:12:30] [INFO] [web-container] [正在启动服务[TongWeb]]
[2020-09-16 04:12:30] [INFO] [web-container] [正在启动 Servlet 引擎: [Tongweb/7.0.C.0]]
[2020-09-16 04:12:30] [INFO] [web-container] [开始协议处理句柄["http-nio-8088"]]
[2020-09-16 04:12:30] [INFO] [boot] [Tongweb服务器启动成功]
```

2.4.3 本地镜像运行

如果用户已经拿到打包好了的镜像，可以直接通过脚本运行,镜像名称必须为 **tongweb-image.tar** 且放置在**${TW_HOME}/bin** 目录下。

特别注意：

docker-image-run.sh 中未进行 license 的挂载，如果用户采用的是本地 license.dat 的认证方式，需要修改 **docker-image-run.sh** 脚本。在 docker run 中增加**-v license.dat** 挂载目录: **/opt/TongWeb/license.dat:rw**，用于 licnese.dat 的挂载。

如果采用的是 License Server 的认证方式，有如下两种场景:

- 使用本地配置文件。
- 使用 Nacos 配置中心。

1. 本地配置文件方式：需要修改**${TW_HOME}/conf/application.properties** 文件，建议用户修改${TW_HOME}/conf/application.properties 文件并制作重新制作镜像。如果不重新制作镜像需要把 application.properties 文件挂载到容器中，在 docker run 中增加**-v application.properties** 挂载目录: **/opt/TongWeb/conf/application.properties:rw**，用于配置文件的挂载。
2. Nacos 配置中心：nacos 中 license 的配置可参考 4.6.3 容器云环境配置。

docker-image-run.sh 脚本格式为:

./docker-image-run.sh <映射端口><自动部署挂载目录><日志挂载目录><导入的镜像名称>

进入${TW_HOME}/bin 文件夹下，通过如下命令来运行镜像:

```
./docker-image-run.sh 9060 自动部署挂载目录 日志文件挂载目录 tongweb/tongweb:7.0
```

2.4.4 镜像仓库拉取

如果用户有自己的镜像仓库，可以直接从东方通镜像仓库拉取进行并推送到用户的私有

镜像仓库。进入${TW_HOME}/bin 文件夹下，通过运行如下命令。

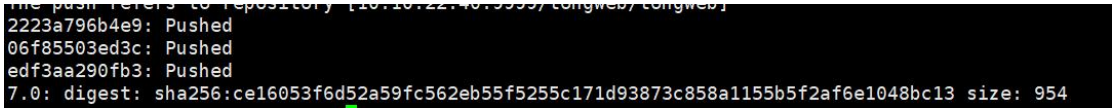
docker-repo-transfer.sh 脚本格式为:

./docker-repo-transfer.sh <东方通镜像仓库地址><用户名><密码><镜像名称><用户镜像仓库地址><用户名><密码><镜像名称>

```
./docker-repo-transfer.sh 东方通镜像仓库地址 username password tongweb/tongweb:7.0 用户镜像仓库地址 username password tongweb/tongweb:7.0
```

注：仓库地址格式为: **ip:port**

出现如下图表示已经成功 push 到用户镜像仓库



进入${TW_HOME}/bin 文件夹下，通过如下命令从用户镜像仓库中拉取镜像并运行:

特别注意：

docker-repo-run.sh 中未进行 license 的挂载，如果用户采用的是本地 license.dat 的认证方式，需要修改 **docker-repo-run.sh** 脚本，在 docker run 中增加**-v license.dat** 挂载目录: **/opt/TongWeb/license.dat:rw**，用于 license.dat 的挂载。

如果采用的是 License Server 的认证方式，有如下两种场景:

- 使用本地配置文件。
- 使用 Nacos 配置中心。

1. 本地配置文件方式：需要修改**${TW_HOME}/conf/application.properties** 文件，建议用户修改${TW_HOME}/conf/application.properties 文件并制作重新制作镜像。如果不重新制作镜像需要把 application.properties 文件挂载到容器中，在 docker run 中增加**-v application.properties** 挂载目录: **/opt/TongWeb/conf/application.properties:rw**，用于配置文件的挂载。
2. Nacos 配置中心：nacos 中 license 的配置可参考 4.6.3 容器云环境配置。

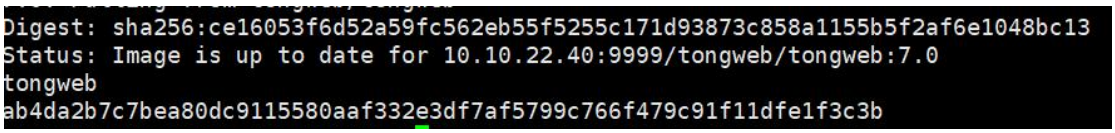
docker-repo-run.sh 脚本格式为:

./docker-repo-run.sh <用户镜像仓库地址><用户名><密码><镜像名称><映射端口><自动部署挂载目录><日志挂载目录><容器名称>

```
./docker-repo-run.sh 用户镜像仓库地址 usernamepassword tongweb/tongweb:7.0 9060 自动部署挂载目录 日志挂载目录 tongweb
```

自动部署挂载目录是对应用挂载到容器中运行，需要把应用复制到挂载目录下，日志挂载目录是日志文件挂载的位置。

出现如下图所示表示容器已经运行成功:



2.4.5 Swarm 集群环境

为了适应 Docker 集群环境下的构建，也提供了在 Swarm 集群环境下运行的简易版脚本文件，方便快速的搭建 TongWeb V7.0.6 容器云版 Swarm 集群，用户可根据实际情况修改，如果没有搭建 Swarm 集群环境跳过此章节。

Swarm 配置文件

1. 进入${TW_HOME}/bin 目录下，编辑打开 **docker-swarm-template-config.options** 文件进行配置参数的修改。
   IMAGE：镜像地址
   REPLICAS：副本个数
   PORT：端口
   SRV_PORT：暴露端口
   LOG_MOUNT_PATH：容器日志目录
   LOG_PATH：日志挂载目录,需要根据用户实际情况修改
   DEPLOY_MOUNT_PATH：容器自动部署应用目录
   DEPLOY_PATH：自动部署应用挂载目录，需要根据用户实际情况修改
   PARALLELISM：同一时间升级的容器数量
   DELAY：容器升级间隔时间
   CONDITION：自启动策略
2. 进入${TW_HOME}/bin 目录下，通过运行 **docker-swarm-template.sh** 脚本进行 docker-swarm-template-sample.yaml 文件的创建。
   ./docker-swarm-template.sh docker-swarm
3. 部署服务，通过运行一下命令部署服务（前提是已经搭建并初始化好了 Swarm 集群）
   docker stack deploy -c docker-swarm-template-sample.yaml 服务名称
4. 如果要停止服务通过以下命令
   docker service rm 服务名称
5. 如果要删除堆栈信息通过如下命令
   docker stack rm 服务名称

2.5 K8S 环境安装

2.5.1 环境准备

如果用户需要使用管理控制台，必须先搭建好 k8s 环境，k8s 的版本需要在 1.13 以上。

2.5.2 配置文件

进入${TW_HOME}/bin 文件夹下，用编辑器打开 **k8s-set-env.options** 文件进行配置参数的修改：

IMAGE:镜像地址；
REPLICAS：副本数量；
PORT_HTTP：HTTP 映射端口；
PORT_HTTPS:HTTPS 映射端口；
LOG_MOUNT_PATH：容器日志目录；
LOG_PATH：日志挂载目录，需要根据用户实际情况修改；
DEPLOY_MOUNT_PATH：容器自动部署应用目录；
DEPLOY_PATH：自动部署应用挂载目录，需要根据用户实际情况修改；
SRV_PORT_HTTP：HTTP K8s 集群内部使用端口；

SRV_PORT_HTTPS：HTTPS K8s 集群内部使用端口；
NODE_HTTP_PORT：HTTP 暴露给外部访问的端口；
NODE_HTTPS_PORT：HTTPS 暴露给外部访问的端口；
LOG_NFS_SERVER：日志 NFS 服务器地址；
DEPLOY_NFS_SERVER：自动部署目录 NFS 服务器地址；
CONFIG_REMOTE_ENABLE：是否启用远程配置；
CONFIG_REMOTE_TYPE：远程配置中心类型；
CONFIG_SERVER：远程配置中心地址和端口；
LOG_NACOS_DATA_ID：Nacos 日志功能的 Data ID；
APP_NACOS_DATA_ID：Nacos 应用功能的 Data ID；
TONGWEB_NACOS_DATA_ID：Nacos TongWeb V7.0.6 配置的 Data ID；
NACOS_GROUP=DEFAULT_GROUP：Nacos 日志，应用，TongWeb V7.0.6 的 Group；

2.5.3 创建 Namespace

1. 通过运行如下命令创建命名空间
   kubectl create namespace tongweb-namespace
2. 通过如下命令查看创建的命名空间
   kubectl get namespace tongweb-namespace

```
NAME                STATUS   AGE
tongweb-namespace   Active   64s
```

2.5.4 创建 Deployment

提供了 Hostpath 和 NFS 方式进行日志和应用的挂载。

hostpath 方式挂载

1. 进入 ${TW_HOME}/bin 文件夹下，运行 **k8s-template.sh** 脚本创建 k8s-template-deployment-hostpath.yaml 文件
   ./k8s-template.sh hostpath

```
[root@k8smaster bin]# ./k8s-template.sh hostpath
生成k8s-template-deployment-hostpath.yaml文件
```

2. 如果采用的是本地 license 文件挂载的方式且没有使用 license server。需要手动在 **k8s-template-deployment-hostpath.yaml** 文件中添加 license 文件的挂载，可参考如下示例：

```
        volumeMounts:
        - name: log-volume
          mountPath: /opt/Tongweb/logs
        - name: autodeploy-volume
          mountPath: /opt/Tongweb/autodeploy
        - name: license-volume
          mountPath: /opt/Tongweb/license.dat
          subPath: license.dat
      volumes:
      - name: log-volume
        nfs:
          path: /home/work/logs
          server: xxxx
      - name: autodeploy-volume
        nfs:
          path: /home/work/autodeploy
          server: xxxx
       - name: license-volume
        nfs:
          path: /home/work/license
          server: xxxx
```

3. 运行如下命令创建 pod:

kubectl create -f k8s-template-deployment-hostpath.yaml --namespace=tongweb-namespace

```
[root@k8smaster bin]# kubectl create -f k8s-template-deployment-hostpath.yaml --namespace=tongweb-namespace
deployment.apps/tongweb created
```

4. 通过如下命令查看创建的 pod:

kubectl get pod -n tongweb-namespace

```
[root@k8smaster tong_script]# kubectl get pod -n tongweb-namespace
NAME                       READY   STATUS    RESTARTS   AGE
tongweb-fc55b9f6b-xhtxd    1/1     Running   0          84s
```

5. 采用 hostpath 挂载利用的是本地 host 挂载的方式，当 pod 启动后，进入对应的 node 节点上面，把需要部署的应用放到自动部署的挂载目录下。在 master 节点下通过以下命令查看对应的 node:

kubectl get pod -n tongweb-namespace -o wide

nfs 挂载

1. 进入 ${TW_HOME}/bin 文件夹下，运行 **k8s-template.sh** 脚本创建 k8s-template-deployment-nfs.yaml 文件

./k8s-template.sh nfs

```
[root@k8smaster bin]# ./k8s-template.sh nfs
生成k8s-template-deployment-nfs.yaml文件
```

2. 如果采用的是本地 license 文件挂载的方式且没有使用 license server。需要手动在 **k8s-template-deployment-nfs.yaml** 文件中添加 license 文件的挂载，可参考如下示例:

```
        volumeMounts:
        - name: log-volume
          mountPath: /opt/Tongweb/logs
        - name: autodeploy-volume
          mountPath: /opt/Tongweb/autodeploy
        - name: license-volume
          mountPath: /opt/Tongweb/license.dat
          subPath: license.dat
      volumes:
      - name: log-volume
        nfs:
          path: /home/work/logs
          server: xxxx
      - name: autodeploy-volume
        nfs:
          path: /home/work/autodeploy
          server: xxxx
       - name: license-volume
        nfs:
          path: /home/work/license
          server: xxxx
```

3. 运行如下命令创建 pod：

kubectl create -f k8s-template-deployment-nfs.yaml --namespace=tongweb-namespace

```
[root@k8smaster bin]# kubectl create -f k8s-template-deployment-nfs.yaml --namespace=tongweb-namespace
deployment.apps/tongweb created
```

4. 通过如下命令查看创建的 pod：

kubectl get pod -n tongweb-namespace

```
[root@k8smaster tong_script]# kubectl get pod -n tongweb-namespace
NAME                       READY   STATUS    RESTARTS   AGE
tongweb-fc55b9f6b-xhtxd    1/1     Running   0          84s
```

2.5.5 创建 Service

1. 进入 ${TW_HOME}/bin 文件夹下，运行 **k8s-template.sh** 脚本创建 k8s-template-service.yaml 文件

./k8s-template.sh service

```
[root@k8smaster bin]# ./k8s-template.sh service
生成k8s-template-service.yaml文件
```

2. 运行如下命令创建 pod：

kubectl create -f k8s-template-service.yaml --namespace=tongweb-namespace

```
[root@k8smaster bin]# kubectl create -f k8s-template-service.yaml --namespace=tongweb-namespace
service/tongwebsrv created
```

3. 通过如下命令查看运行的 Service：

kubectl get svc -n tongweb-namespace

```
[root@k8smaster tong_script]# kubectl get svc -n tongweb-namespace
NAME         TYPE       CLUSTER-IP     EXTERNAL-IP   PORT(S)                        AGE
tongwebsrv   NodePort   10.96.140.18   <none>        9088:31529/TCP,9443:30067/TCP   13m
```

2.6 License Server 安装

参考《TongWeb V7.0.6 授权中心用户使用手册》

2.7 停止服务器

2.7.1 Windows 平台

TongWeb V7.0.6 容器云版服务器提供了在 Windows 环境下启动的命令行方式，使用 ${TW_HOME}/bin 目录下的 **stopserver.bat** 停止 TongWeb V7.0.6 容器云版服务器。

2.7.2 Linux 平台

TongWeb V7.0.6 容器云版服务器提供了在 Linux 环境下启动的命令行方式，使用 ${TW_HOME}/bin 目录下的 **stopserver.sh** 停止 TongWeb V7.0.6 容器云版服务器。

2.7.3 Docker 环境

1. 通过一下命令查看运行的容器，查看容器名称为 tongweb 的容器 id
   docker ps
2. 通过一下命令停止正在运行的容器
   docker stop 容器 ID
3. 如果要删除该容器，运行如下命令：
   docker rm 容器 ID

2.7.4 K8s 环境

1. 删除 Service
   kubectl delete svc/tongwebsrv -n tongweb-namespace

```
[root@k8smaster tong_script]# kubectl delete svc/tongwebsrv -n tongweb-namespace
service "tongwebsrv" deleted
```

2. 查看 Service 是否删除成功
   kubectl get svc -n tongweb-namespace

```
[root@k8smaster tong_script]# kubectl get svc -n tongweb-namespace
No resources found in tongweb-namespace namespace.
```

3. 删除 Pod

1) hostpath 挂载方式删除：

kubectl delete -f k8s-template-deployment-hostpath.yaml -n tongweb-namespace

```
[root@k8smaster bin]# kubectl delete -f k8s-template-deployment-hostpath.yaml -n tongweb-namespace
deployment.apps "tongweb" deleted
```

2) NFS 挂载方式删除

kubectl delete -f k8s-template-deployment-nfs.yaml -n tongweb-namespace

```
[root@k8smaster bin]# kubectl delete -f k8s-template-deployment-nfs.yaml -n tongweb-namespace
deployment.apps "tongweb" deleted
```

4. 查看 pod

kubectl get pod -n tongweb-namespace

```
[root@k8smaster tong_script]# kubectl get pod -n tongweb-namespace
NAME                       READY   STATUS        RESTARTS   AGE
tongweb-fc55b9f6b-9m7vj    0/1     Terminating   0          61s
```

第三章　配置文件总体说明

3.1 配置文件说明

在 TongWeb V7.0.6 容器云版中，主要的配置文件有 **logging.properties**、**application.properties**、**tongweb.xml**。

表 3.1- 1：配置文件说明

配置文件	说明
logging.properties	日志服务的参数配置，包含日志轮转配置、日志存放位置、日志级别、访问日志的配置等；存放位置：conf 目录下。
application.properties	应用配置参数，包括 license 信息配置等
tongweb.xml	Web 容器内配置的相关信息，如应用属性、通道、数据源等都在该文件里配置；存放位置：conf 目录下。

3.2 tongweb.xml 说明

TongWeb V7.0.6 容器云版 Web 容器内的配置都在**${TW_HOME}/conf/tongweb.xml** 文件里，这里对 tongweb.xml 配置文件进行部分说明。配置文件如下：

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<tongweb>
<server shutdown-port="8006">
<web-container>
<virtual-host name="localhost" listeners="tong-http-listener"
accesslog-enabled="false"  app-base="autodeploy" auto-deploy="true">
</virtual-host>
<http-listener name="tong-http-listener" port="8088" io-mode="nio">
</http-listener>
</web-container>
<monitor-service monitoring-enabled="false">
```

```
<monitor-config name="Memory" monitoring-enabled="false"/>
<monitor-config name="JVMMemoryPool"/>
<monitor-config name="GarbageCollector"/>
<monitor-config name="JVMThread"/>
<monitor-config name="Compilation"/>
<monitor-config name="ClassLoading"/>
<monitor-config name="Runtime"/>
<monitor-config name="OperatingSystem"/>
<monitor-config name="TWServer"/>
<monitor-config name="ConnectorAndThreadPool"/>
<monitor-config name="DataSource"/>
<monitor-config name="WebModule"/>
<monitor-config name="SessionManager"/>
<monitor-config name="Loader"/>
<monitor-config name="ResourceCache"/>
<monitor-config name="Request"/>
</monitor-service>
<!--  <jdbc-connection-pool  name="jdbc/test"-->
<!--  jdbc-driver="com.mysql.jdbc.Driver"-->
<!--
jdbc-url="jdbc:mysql://xxxx:3306/xxxx?useUnicode=true&amp;characterEncoding=utf-8&amp;serverTimezone=Asia/Shanghai"-->
<!--  user-name="xxxxx" password="xxxxx"-->
<!--  initial-size="10" max-active="100" min-idle="10" max-wait-time="30000"-->
<!--  validation-query="SELECT 1" validation-query-timeout="5"
test-on-borrow="false"-->
<!--  test-on-connect="false" test-on-return="false" test-while-idle="false"-->
<!--  time-between-eviction-runs="60000" min-evictable-idle-time="60000"
remove-abandoned="true"-->
<!--  remove-abandoned-timeout="2"/>-->
```

3.3 脚本文件说明

在${TW_HOME}/bin 目录下提供了帮助用户开发调试用的脚本，如下表所示。

表 3.3- 1：脚本文件说明

文件名称	说明
startdebug.bat	Window 系统 Debug 方式启动脚本
startdebug.sh	Linux 系统 Debug 方式启动脚本
startserver.bat	Window 系统启动脚本
startserver.sh	Linux 系统启动脚本
startservernohup.sh	Linux 系统后台启动脚本

stopserver.bat	Window 系统停止脚本
stopserver.sh	Linux 系统停止脚本
version.bat	Window 系统打印版本信息脚本
version.sh	Linux 系统打印版本信息脚本
docker-swarm-template.sh	Docker 创建 Swarm 模板文件脚本
docker-repo-run.sh	Docker 从用户镜像仓库拉取镜像并运行脚本
docker-image-run.sh	Docker 本地镜像运行脚本
docker-repo-transfer.sh	Docker 从东方通镜像仓库推送到用户镜像仓库脚本
docker-swarm-template-config.options	Docker Swarm 环境配置信息
docker-swarm-template-sample.tpl	Docker Swarms 示例模板文件
docker-set-env.env	Docker nacos 环境变量
k8s-template.sh	K8s 创建模板文件脚本
k8s-set-env.options	K8s 环境配置信息
k8s-template-deployment-hostpath.tpl	K8s Hostpath 挂载方式模板文件
k8s-template-deployment-nfs.tpl	K8s Nfs 挂载方式模板文件
k8s-template-service.tpl	K8s Service 模板文件
env.options	配置中心参数配置
cipher-tool.bat	Window 环境加密工具执行脚本
cipher-tool.sh	Linux 环境加密工具执行脚本
external.vmoptions	设置启动参数
licenseinfo.bat	Windows 环境查看 TongWeb V7.0.6 目录下存放的 license.dat 授权文件内的信息。 如果需要查看 license server 的信息需要把 license server 的 license.dat 放到 TongWeb V7.0.6 目录下。
licenseinfo.sh	Linux 环境查看 TongWeb V7.0.6 目录下存放的 license.dat 授权文件内的信息。 如果需要查看 license server 的 license 信息需要把 license server 的 license.dat 放到 TongWeb V7.0.6 目录下。

3.4 License 信息

3.4.1 概述

使用 TongWeb V7.0.6 容器云版时，首先需要经过 license 认证。TongWeb V7.0.6 容器云版的 license 认证方式分为两种。一种是 **license server** 认证，另一种是读取 **license.dat** 文件认证。TongWeb V7.0.6 容器云版默认采取读取 license.dat 文件认证的方式。

特别说明:

License 的配置方式需要在**${TW_HOME}/conf/application.properties** 中进行配置。

3.4.2 文件认证

TongWeb V7.0.6 容器云版默认采用的是 **license.dat** 文件认证的方式，用户可以直接将 license.dat 文件放到 TongWeb V7.0.6 目录下。系统会自动去 TongWeb V7.0.6 目录下查找是否有 license.dat 文件，如果存在就直接读取该 license.dat 文件。

用户也可以采取配置参数的方式指定 license.dat 文件的路径，在 **${TW_HOME}/conf/application.properties** 文件中配置如下参数：

- server.tongweb.license.type=file 表示通过 license.dat 文件的方式认证。
- server.tongweb.license.path=d:\\license.dat 表示读取 D 盘下的 license.dat 文件，该配置项是可选参数，不配置默认到 TongWeb V7.0.6 目录下去寻找。也就是说需要把 license.dat 文件放到 TongWeb V7.0.6 目录下。

3.4.3 License Server 认证

TongWeb V7.0.6 容器云版提供 License server 用于集群下的 License 认证。

特别说明：

License Server 的启动参考《授权中心用户使用手册》。

License Server 不建议部署在 K8s 或者 Docker 中。

采用 License server 认证方式，用户需要搭建 License server。用户拿到 License server 的压缩包，解压后其整体目录结构如下图：

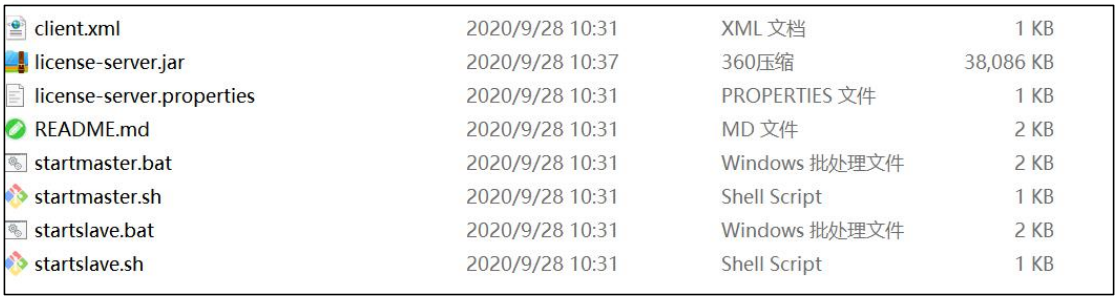


图 3.4- 1：LicenseServer 目录结构

License server 一共包含如下内容。分别是：

- client.xml 是记录 TongWeb V7.0.6 请求认证记录信息；
- license-server.properties 是应用程序配置文件，记录节点信息、超时、周期等信息；
- license.dat 是 TongWeb V7.0.6 的 License 文件，需要放置到该目录下；
- license-server.jar 是 License-server 的应用程序执行文件；
- start.bat Windows 环境下的启动脚本；
- start.sh Linux 环境启动脚本；
- startnohup.sh Linux 环境静默启动脚本；

具体 license server 的使用说明可参考《授权中心用户使用手册》。

用户需要在**${TW_HOME}/conf/appliction.properties** 文件中配置 license server 相关的参数信息。

- server.tongweb.license.master=127.0.0.1:8888 表示 license 的 ip 地址和访问端口。
- server.tongweb.license.type=server 表示采用 license server 方式认证。

${TW_HOME}/conf/application.properties 的配置详见附录 A

第四章　功能说明

4.1 应用管理

4.1.1 自动部署配置

TongWeb V7.0.6 容器云版支持文件和目录的方式的自动部署，**${TW_HOME}/autodeploy** 是 TongWeb V7.0.6 的默认自动部署目录，将应用拷贝到自动部署目录下即可部署应用。把应用从自动部署目录下删除即可解除部署应用。应用部署的文件只支持 war 包和目录的方式。

自动部署目录配置在**${TW_HOME}/conf/tongweb.xml** 的 **virtual-host** 节点中:

表 4.1- 1: 自动部署配置项

配置项名称	说明	默认值
app-base	自动部署目录	autodeploy

4.1.2 热部署

TongWeb V7.0.6 容器云版支持应用的热部署，用户只需要把应用的 war 包或者目录放到自动部署目录下，TongWeb V7.0.6 就可以进行自动部署到 TongWeb V7.0.6 服务器上。

4.2 Web 容器

4.2.1 Web 容器说明

在 Java EE 平台上，Web 应用运行在 Web 容器中。Web 容器提供了 Web 应用的运行时环境，包括生命周期管理、安全、请求转发等。同时 Web 容器为 Web 应用提供访问其他 API 的能力。

Web 应用通过 XML 格式的部署描述文件来定义自身的行为。一个容器可以同时运行多个 Web 应用，它们一般通过不同的 URI 来进行区分和访问。

如：http://<host>:<port>/contextroot/servletname

IPV6 访问格式:

http://<IPV6 地址>:<port>/contextroot/servletname

该 URI 各个部分的含义:

http(或 https)://虚拟主机名或别名:虚拟主机关联的通道的监听端口/虚拟主机上部署应用的访问前缀/应用中的 Servlet 名)。

4.2.2 Web 容器配置

容器配置里的属性，是对所有 Web 应用的通用属性。

Web 容器配置的配置项在 **${TW_HOME}/conf/tongweb.xml** 的 **<server>/<web-container>**节点中。

4.2.3 虚拟主机

虚拟机主机是 TongWeb V7.0.6 应用服务器管理的一组主机名。将单个物理主机分成多个“虚拟”的主机，即虚拟主机间可共享一台物理主机的资源。TongWeb V7.0.6 服务器上一个 Web 应用可以部署在多个虚拟主机上。

在使用虚拟主机时，首先应该确认操作系统的 hosts 表中，是否正确的配置了虚拟主机名与 IP 地址的映射，例如：admin 10.10.4.10。如果未配置主机名与 IP 地址的映射，则需要手工添加。要注意 hosts 表中配置的主机名必须配置虚拟主机时填写的“**主机名**”一致。访问格式为：**http://<virtualServer>:<port>/contextRoot**

虚拟主机配置项在 **${TW_HOME}/conf/tongweb.xml** 的 **<server>/<web-container>/<virtual-host>**节点中，各配置项说明如下：

表 4.2- 1: 虚拟主机配置项

配置项名称	说明	默认值
name	虚拟主机唯一名称	localhost
accesslog-enabled	虚拟主机的访问日志开关	false
accesslog-dir	访问日志目录，如为相对目录，则相对 tongweb 目录	logs/access
app-base	自动部署目录，相对 tongweb	autodeploy
auto-deploy	是否启用自动部署	true

4.2.4 HTTP 通道

Web 容器使用通道接收用户请求(每个通道提供自己的监听地址和监听端口)。TongWeb V7.0.6 默认提供 tong-http-listener 的 HTTP 监听器来监听用户请求，默认的端口号为 8088。部署在容器云上的 Web 应用访问 URL 为：http://<IP>:<port>/加上 web 应用指定的上下文路径，访问 Web 应用。

1. HTTP 通道配置项在 **${TW_HOME}/conf/tongweb.xml** 的 **<server>/<web-container>/<http-listener>**节点中，各配置项说明如下：

表 4.2- 2: 通道配置项

配置项名称	说明	默认值
name	监听器名称	tong-http-listener
port	监听器监听端口	8088
io-mode	通道使用的 io 模式：nio、nio2	nio
ssl-enabled	开启 ssl 功能	true
http2-enabled	开启 http2 功能	true
uri-encoding	指定 uri 参数编码	GBK

address	ip 地址或者域名	无

2. 当 ssl-enable=true 时，需要对 ssl 进行配置，在**${TW_HOME}/conf/tongweb.xml** 的**<server>/<web-container>/<http-listener>/<ssl>**节点中。各配置项说明如下：

表 4.2- 3: 通道 ssl 配置项

配置项名称	说明	默认值
keystore-file	秘钥库路径，支持绝对路径和相对路径；默认为当前根目录的相对路径	conf/ssl/tongweb.keystore
keystore-pass	秘钥库密码	123456
keystore-type	秘钥库类型	JKS
client-auth	是否开启客户端认证	true
truststore-file	信任证书路径，支持绝对路径和相对路径；默认为当前根目录的相对路径	conf/ssl/tongweb.keystore
truststore-pass	信任证书密码	123456
truststore-type	信任证书类型	JKS
ssl-protocol	设置使用的 SSL 协议	TLS

3. protocol 配置项在**${TW_HOME}/conf/tongweb.xml** 的**<server>/<web-container>/<http-listener>/<protocol>**节点中。各配置项说明如下：

表 4.2- 4: 通道 protocol 配置项

配置项名称	说明	默认值
max-header-count	容器允许的最大请求头个数。当请求中包含头字段个数大于该值时，请求会被拒绝，单位：个	100
connection-timeout	Socket 网络连接超时时间；单位：毫秒	6000
max-threads	通道可创建的最大线程数，当连接超过了最大连接数，之后的连接不被处理；	200
min-spare-threads	最小备用线程数，即初始化时创建的线程数	10
max-connections	在任何给定的时间服务器接受并处理的最大连接数。当这个数字已经达到了，服务器将不会接受任何连接，直到连接的数量降到低于此值。基于 acceptCount 的设置，操作系统可能仍然接受连接。	10000
accept-thread-count	最大排队数	1

4. http-options 配置项在**${TW_HOME}/conf/tongweb.xml** 的**<server>/<web-container>/<http-listener>/<http-options>**节点中，各配置项说明如下：

表 4.2- 5: 通道 http-options 配置项

配置项名称	说明	默认值
compression	压缩开关：开启后可以节省带宽；复制分别为（on:启用压缩数据；off:禁用压缩数据；force:在所有情况下强制压缩数据）	off
compressable-mime-type	用户 http 压缩的 MIME 类型	text/html,

		text/xml, text/plain
compression-min-size	启用传输压缩的内容最小值，单位：字节	2048
disable-upload-timeout	禁用上传超时	true
max-http-header-size	http 请求与请求头的最大值，单位为：字节	8192
max-keep-alive-requests	HTTP 请求最大长连接个数。将此属性设置为 1，将禁用 HTTP/1.0、以及 HTTP/1.1 的长连接。设置为-1，不禁用	100
connection-upload-timeout	文件上传超时时间	无，表示不限
server	覆盖 http 响应的服务器头。如果设置，此属性的值将覆盖 TongWeb V7.0.6 默认值和web 应用程序设置的任何服务器头。如果未设置，则使用应用程序指定的任何值。	无
restricted-user-agents	值是一个正则表达式（使用 java.util.regex），它与 HTTP/1.1 或 HTTP/1.0 保持活跃的 HTTP 客户机的用户代理头匹配，因此不应该使用它，即使客户机宣称支持这些特性。默认值是一个空字符串(禁用 regexp 匹配)。	无

4.2.5 WebApp 部署

TongWeb V7.0.6 容器云版支持 WebApp 的部署，用户可以在 **tongweb.xml** 中配置 webapp 的标签，设置类加载器的模式、虚拟主机名称、上下文路径。

特别说明：

WebApp 中的**vs-names**配置的虚拟主机名称需要和 tongweb.xml 中配置的虚拟主机名称一致。

1. webapp 在**${TW_HOME}/conf/tongweb.xml** 的**<tongweb>/<apps>/<web-app>**的节点中，参数配置如下表所示：

表 4.2- 6：web-app 配置说明

配置参数	说明	默认值
delegate	true，表示遵循 JVM 的 delegate 机制，即一个 WebAppClassLoader 在加载类文件时，会先递交给 SharedClassLoader 加载，SharedClassLoader 无法加载成功，会继续向自己的父类委托，一直到 BootstarpClassLoader，如果都没有加载成功，则最后由 WebAppClassLoader 自己进行加载。 false，表示将不遵循这个 delegate 机制，即 WebAppClassLoader 在加载类文件时，会优先自己尝试加载，如果加载失败，才会沿着继承链，	false

	依次委托父类加载。	
vs-names	虚拟主机名称	无
context-root	上下文路径	无
app-root	War 包文件名或应用目录名	无
parameter-encoding	请求参数解码的字符集	GBK
response-encoding	应答编码的字符集	GBK

示例：

```
<apps>
<web-app delegate="true" vs-names="localhost" context-root= "/xxx"
app-root="xxx.war">
</web-app>
</apps>
```

2. 在 WebApp 中支持对上下文全局资源的引用，在**${TW_HOME}/conf/tongweb.xml** 的**<tongweb>/<apps>/<web-app>/<resource>**的节点中，参数配置如下表所示：

表 4.2- 7：web-app 全局资源引用说明

配置项参数	说明	默认值
type	资源应用类型目前只支持 ref	ref
name	被创建的资源连接的名称，相对于 java:comp/env 上下文。	无
ref-name	在全局的 jndi 上下文中，连接的全局资源的名称。	无
class-name	资源类型，全类名。	无

示例：

```
<apps>
<web-app delegate="true" vs-names="localhost" context-root= "/xxx"
app-root="xxx.war">
<resource type="ref" ref-name="jdbc/test" name="jdbc/test_app"
class-name="javax.sql.DataSource"/>
</web-app>
</apps>
```

4.3 JDBC 配置

4.3.1 概述

JDBC 数据源的主要功能是为应用程序提供数据库连接。JDBC 数据源基于数据库连接池技术，因此连接复用便是 JDBC 数据源的基本功能。每个 JDBC 数据源使用一个连接池维护一定数量的连接。连接池预先建立多个数据库连接对象，然后将连接对象保存到连接池中，当客户请求到来时，从池中取出一个连接对象为客户服务，当请求完成时，客户程序调用 close()方法将连接对象放回池中。

4.3.2 连接池管理功能描述

为保证连接池的性能，服务器提供连接池的管理功能，主要包括定时处理空闲连接、检测连接的有效性等。

4.3.3 空闲超时处理

空闲连接即连接池中没有被使用的连接。开启空闲超时功能，当连接池中存在空闲的连接数大于连接池的最小连接数时（初始化连接数）时，服务器以用户配置的"检查连接的周期"时间为周期对连接池中的空闲连接进行检查，主要检查空闲连接是否超时。

具体检查步骤如下：

1. 如果连接空闲的时间超过用户配置的"空闲超时"时间，将从连接池中删除。
2. 检查连接池中的连接数，如果连接数小于用户配置的"最小连接数"，则创建新连接并放到连接池中，使连接池中的连接数达到用户配置的最小值。

4.3.4 连接有效性检查

服务器在将连接提供给应用程序之前验证连接的有效性，如果由于网络出现故障或数据库服务器崩溃等原因造成无法获取数据库连接，应用服务器将自动重新建立数据库连接。该功能会带来一定的性能开销，因此不是每次获取连接时都进行有效性检查。为了减少开销，服务器将按照一定的周期对连接进行有效性检查，除了对连接进行定期的有效性检查外，还提供获取连接时对连接进行有效性检查、创建连接时对连接进行有效性检查、归还连接时进行有效性检查。（获取连接时对连接进行有效性检查、创建连接时对连接进行有效性检查、归还连接时进行有效性检查的使用参见 4.3.6 Jdbc 数据源配置）

1. 连接有效性检查的条件如下：

- 开启连接有效性检查功能；
- 配置“测试连接的 SQL 语句"或者用户自定义的验证类；
- 用户配置的"检查连接的周期"大于 0，且从上一次检查至今，已经超过了配置的"连接验证时间间隔"；

2. 具体检查步骤如下：

- 检查连接是否有效，如果连接无效，将从连接池中删除。
- 检查连接池中的连接数，如果连接数小于用户配置的"最小连接数"，则创建新连接并放到连接池中，使连接池中的连接数达到用户配置的最小值。

4.3.5 驱动加载

用户需要在**${TW_HOME}/lib** 目录下添加数据库的驱动，并在 **tongweb.xml** 中配置数据库的驱动和连接信息，具体参考 4.3.6 Jdbc 数据源配置。

4.3.6 JDBC 数据源配置

TongWeb 提供了 jdbc 数据源的配置，为应用提供了连接数据库的方法。配置由 2 种方式，默认为全局资源，需要配置 **link-name**,应用通过 link-name 访问资源。

如果没有配置 link-name，需要增加 **webapp resource**。

1. 将数据库的驱动程序放到${TW_HOME}/lib 目录下。
2. 数据源配置项在${TW_HOME}/conf/tongweb.xml 的**<server>/<jdbc-connection-pool>**节点中。
3. java 获取 JNDI 参考代码：

```
Context ctx = new InitialContext();
Context envCtx = (Context) ctx.lookup("java:comp/env");
DataSource ds = (DataSource) envCtx.lookup("jdbc/test");
```

各配置项说明如下：

表 4.3- 1：JDBC 配置项

配置项名称	说明	默认值
link-name	JDBC 的 JNDI 资源名称	无
name	JDBC 的唯一名称	无
jdbc-driver	数据库驱动类名	无
jdbc-url	连接数据库的 url	无
user-name	连接数据库所需要的用户名，可以进行加密，加密方式可以参考 4.11.1 节 Cipher 加密工具的使用，加密后的字符串前缀需要带上 enc:或者 enc#+密文，例：enc:rwOym49c6QHQui1OcPHr9g==如果加密后的字符串包含:需要使用 enc#+密文的方式，例：enc#rwOym49c6QHQui1OcPHr:9g==	无
password	连接数据库用户名的密码，可以进行加密，加密方式可以参考 4.11.1 节 Cipher 加密工具的使用，加密后的字符串前缀需要带上 enc:或者 enc#+密文，例：enc:rwOym49c6QHQui1OcPHr9g==。如果加密后的字符串包含:需要使用 enc#+密文的方式，例：enc#rwOym49c6QHQui1OcPHr:9g==	无
initial-size	初始连接数：首次创建池或应用服务器初始启动时，数据库连接池里创建的连接数；单位：个	10
max-active	最大连接数：连接池允许创建的最大连接数；单位：个	100
min-idle	最小空闲连接数，单位：个	10
max-wait-time	当没有可用连接时，连接池等待连接被归还的最大时间，超时则抛出异常；单位：毫秒	30000
validation-query	用于测试连接数据库的 SQL 语句	无

validation-query-timeout	验证超时：测试连接活动的最长时间；单位：秒	5000
time-between-eviction-runs	检查连接的周期，单位：毫秒	5000

示例：

```
<jdbc-connection-pool name="jdbc/test" jdbc-driver="com.mysql.jdbc.Driver"
jdbc-url="jdbc:mysql://localhost:3306/jdbc?useUnicode=true&amp;characterEncoding=utf-8&amp;serverTimezone=Asia/Shanghai" user-name="xxxxx" password="xxxxx" initial-size="10" max-active="100" min-idle="10" max-wait-time="30000" validation-query="SELECT 1" validation-query-timeout="5" test-on-borrow="false" test-on-connect="false" test-on-return="false" test-while-idle="false"time-between-eviction-runs="60000" min-evictable-idle-time="60000" remove-abandoned="true"remove-abandoned-timeout="2"/>
```

Web-app 使用例子

```
<web-app vs-names="localhost" app-root="demo1">
<resource type="ref" ref-name="jdbc/test" name="jdbc/test_app"
class-name="javax.sql.DataSource">
</resource>
</web-app>
```

字段说明：

localhost 虚拟主机 默认值 localhost 不能为 null；

doc-base 应用名称；

type 字段内容固定；

ref-name 对应 jdbc-connection-pool 中的 name；

name 应用访问名称；

4.4 安全管理

4.4.1 国密 SSL

国密 SSL 协议包括握手协议、密码规格变更协议、报警协议、网关到网关协议和记录层协议。握手协议用于身份鉴别和安全参数协商；密码规格变更协议用于通知安全参数的变更；报警协议用于关闭通知和对错误进行报警；网关到网关协议用于建立网关到网关的传输层隧道；记录层协议用于传输数据的分段、压缩及解压缩、加密及解密、完整性校验等。

TongWeb V7.0.6 容器云版支持国密 SM2、SM3、SM4 通信双向认证，要使用国密 SSL 功能，需要使用到国密支持的 jar 文件：**tongweb-gmssl-1.0.X.jar**

以下为国密 ssl 功能使用的完整示例：

1. 首先将 **tongweb-gmssl-1.0.0.jar** 文件复制到**${TW_HOME}/lib/**

   注：tongweb-gmssl-1.0.0.jar 文件请联系相关人员获取。

2. 打开**${TW_HOME}/conf/tongweb.xml** 配置文件，添加国密的**<http-listener>**配置如下：

```
<!-- 国密 SSL 配置 ，开启此配置需要在 lib 加入支持国密的 tongweb-gmssl-1.0.0.jar -->
<http-listener name="https" port="8443"  io-mode="nio" ssl-enabled="true">
<ssl  ssl-protocol="GMSSLv1.1"
```

```
                keystore-type="PKCS12"
                keystore-file="conf/ssl/sm2.enc.pfx"
                keystore-pass="xxxxxxxx"
                truststore-file="conf/ssl/sm2.sig.pfx"
                truststore-pass="xxxxxxxx"
                truststore-type="PKCS12"
            />
    </http-listener>
```

国密 SSL 各配置项说明如下：

表 4.4- 1：国密配置项-1

配置项	说明	默认值
ssl-protocol	SSL 通信协议	GMSSLv1.1
keystore-type	配置加密密钥库文件的格式	PKCS12
keystore-file	配置加密密钥库文件	conf/ssl/sm2.enc.pfx
keystore-pass	配置加密密钥库文件的密码	xxxxxxxx
truststore-file	配置签名密钥库文件	conf/ssl/sm2.sig.pfx
truststore-pass	配置签名密钥库文件的密码	xxxxxxxx
truststore-type	配置签名密钥库文件的格式	PKCS12

注意：

加密密钥库：是指将加密私钥和加密证书打包成的加密密钥库文件。

签名密钥库：是指将签名私钥和签名证书打包成的密钥库文件。

密钥库文件格式 ： 密钥库文件类型有 JKS，JCEKS，PKCS12，BKS。

秘钥文件由用户向具有证书颁发资质的机构申请国密证书。

3. 启动TongWeb V7.0.6 容器云版，部署一个应用 demo。
4. 启动一个支持国密协议的浏览器，并在设置中启用国密 SSL 协议支持功能。（此处以奇安信浏览器为例）点击**设置-网页内容--国密 SSL 通信**，选中“**启用国密 SSL 通信**”选项。

图 4.4- 1：设置国密通讯协议

5. 访问应用路径 **https://<localhost>:8443/demo/index**

   注意：

   请求协议必须是 https;

   localhost 为应用部署地址;

   8443 为<http-listener>配置标签中配置的端口号;

   demo 为部署的应用名称;

   index 为具体请求的路径。

- 如果使用不支持国密协议的浏览器（例如 chrome），或者将步骤 4 中的设置取消，再次访问步骤 5 中的地址，则会提示无法访问。
- 如果需要在 http2 规范下使用国密 SSL,需要在配置项中增加 server.http2.enabled=true，servlet4.0 规范中，国密 SSL 不支持服务器推送特性。

6. 常见错误：

   US_export_policy.jar 找不到

```
        ... 28 more
Caused by: java.lang.SecurityException: Can not initialize cryptographic mechanism
        at com.tongweb.tianfu.a.a.x.<clinit>(Unknown Source)
        ... 34 more
Caused by: java.security.PrivilegedActionException: java.io.FileNotFoundException: /home/mc/jdk/jdk1.8.0_144/j
re/lib/security/policy/unlimited/US_export_policy.jar (No such file or directory)
        at java.security.AccessController.doPrivileged(Native Method)
        ... 35 more
Caused by: java.io.FileNotFoundException: /home/mc/jdk/jdk1.8.0_144/jre/lib/security/policy/unlimited/US_expor
t_policy.jar (No such file or directory)
        at java.util.zip.ZipFile.open(Native Method)
        at java.util.zip.ZipFile.<init>(ZipFile.java:225)
        at java.util.zip.ZipFile.<init>(ZipFile.java:155)
        at java.util.jar.JarFile.<init>(JarFile.java:166)
        at java.util.jar.JarFile.<init>(JarFile.java:130)
        at com.tongweb.tianfu.a.a.x.a(Unknown Source)
        at com.tongweb.tianfu.a.a.x.c(Unknown Source)
        at com.tongweb.tianfu.a.a.y.run(Unknown Source)
```

   问题原因：

Java SE 开发工具包和 Java SE 运行时环境附带的 Javaencryption 扩展（JCE）策略文件限定长度位 128 位。

国密 SM2S 是 256 位 ecc，超过了 jdk 限制长度。JDK 提供了 **local_policy.jar** 和 **US_export_policy.jar**，这两个文件可实现无限强度版本。找不到这两个文件就会报错。

解决方法：

只需要将这两个文件放在指定目录**${JAVA_HOME}/jre/lib/security/policy/unlimited/**下面。Jdk 会使用无限强度的文件，就不会出现上述错误。

文件的获取：

首先去找 jdk 安装包中**${JAVA_HOME}/jre/lib/security** 是否提供了这两个文件，如果提供了直接放在${JAVA_HOME}/jre/lib/security/policy/unlimited/下面，如果没有提供需要去官网下载安装 jdk 版本的对应的上述两个文件。

4.4.2 HTTPS

HTTPS 是一种安全超文本传输协议，通过 SSL/TLS 连接保护在线传输的任何通信。HTTPS 有助于在服务器和浏览器之间建立安全通信、可以保护网站免受篡改或窃听、可以保护用户免受中间人攻击等优点。

TongWeb V7.0.6 容器云版提供 HTTPS 的支持，JRE 版本必须要 JRE1.8_252 以上。如果要使用 https，必须开启 https 服务才行，启动 HTTPS 的支持需要在 tongweb.xml 文件的 **<server>/<web-container>/<http-listener>**中启动 **ssl** 和 **https**。

- 通过配置 ssl-enabled 启动 ssl

ssl-enabled=true 启动 ssl

- 通过配置 http2-enabled 启动 http2

http2-enabled=true 启动 http2

- 需要在 **${TW_HOME}/conf/tongweb.xml** 的 **<server>/<web-container>/<http-listener>**中添加 SSL 配置如下：

```
<ssl keystore-file="conf/ssl/tongweb.keystore"
                    keystore-pass="xxxx"
                    keystore-type="JKS"
                    client-auth="false"
                    truststore-file="conf/ssl/tongweb.keystore"
truststore-pass="xxxx"
truststore-type="JKS"
ssl-protocol="TLS" />
```

SSL 的配置参数说明如下表所示：

表 4.4- 2: 国密配置项-2

SSL 配置	说明	默认值
keystore-file	SSL 证书路径	无
keystore-pass	访问 key store 的密码	无
keystore-type	key store 类型	JKS
client-auth	客户端双向认证参数	false
truststore-file	Trust store 路径	无
truststore-pass	Trust store 密码	无

truststore-type	Trust store 类型	JKS
ssl-protocol	SSL 协议	TLS

4.5 日志配置

4.5.1 概述

TongWeb V7.0.6 容器云版集成了丰富的日志使用功能，支持自定义日志格式输出、支持日志轮转方式配置、支持自定义日志输出文件、模块级日志级别控制,日志国际化（只支持中英文）等，日志类型分为系统日志和访问日志。

特别说明：

在 Window 环境下日志输出可能会乱码，用户可以在 **external.vmoptions** 文件中通过设置参数**-Dproperties.file.encoding=UTF-8** 设置读取 properties 文件编码方式，如果还出现乱码的情况可以设置**-Dcommons.logger.encoding=GBK** 指定日志输出编码格式为 GBK。

TongWeb V7.0.6 容器云版在 Docker 或者 K8s 环境中运行时，涉及到日志挂载的问题，在容器中运行时日志文件名都是加了容器 ID 作为区分不同容器的日志，格式为 **filename.后缀名.容器 ID**。

设置参数 **server.tongweb.log-service.console.redirect** 为 **true** 后控制台日志将会转发到 **server.log** 文件中，控制台中将不会再输出应用日志信息。

4.5.2 系统日志

系统日志记录了 TongWeb V7.0.6 容器云版服务器的运行状态。通过分析系统日志的错误信息，可帮助查找系统出错原因，以及通过日志的时间间隔找到耗时较多的系统操作，以便诊断系统故障原因和性能瓶颈。

1. 系统日志轮转方式分为：不论转、按大小轮转、按天轮转、按小时轮转只能选择一种轮转方式。不轮转的则设置 **server.tongweb.log-service.rotation=fasle**，如果 **server.tongweb.log-service.rotation=true** 也就是开启轮转，按天轮转、按小时轮转、按大小轮转这三种轮转方式只能选择一种，如果全部开启其优先级为按天轮转>按小时轮转>按大小轮转。
2. 支持自定义日志输出格式
3. 在**${TW_HOME}/conf/logging.properties** 文件中对系统日志进行配置，系统日志配置如下表所示：

表 4.5- 1: 系统日志配置

系统日志配置	说明	默认值
server.tongweb.log-service.enable	是否启动服务日志,值为 boolean 类型,取值为 true 表示启动日志服务。	无，建议配置为 true
server.tongweb.log-service.file	服务日志存放路径，可以配置相对路径和绝对路径，格式为：日志路径/日志文件名，路径可以是相对路径也可以是绝对路径。相对路径只能相对于 TongWeb V7.0.6,如 logs/server.log	无

server.tongweb.log-service.log-format	输出日志打印格式，如 [%d{yyyy-MM-dd HH:mm:ss}] [%p] [%c] [%m]%n	无
server.tongweb.log-service.module.enable	是否打印模块名称，为 true 表示打印模块名称，false 表示不打印模块名称，打印具体的类名	无
server.tongweb.log-service.rotation-limit	按大小轮转：轮转大小限制，值为 0，表示不按大小轮转，如果要按大小轮转需要配置轮转的大小，如 20MB，表示按 20MB 的大小轮转，单位只能为 KB,MB,GB	0
server.tongweb.log-service.rotation-time-limit	轮转周期限制，值为 0 表示不按周期轮转，值为 h 表示按小时轮转，值只能为 h	无
server.tongweb.log-service.rotation-file-count	按大小轮转：轮转个数限制，值为 0 表示不限制个数（最大 21 个）。值大于 0 表示轮转个数达到设置的值就不轮转，且轮转最大个数不能超过 21 个。	0
server.tongweb.log-service.rotation-by-day	是否按天轮转，值为 boolean 类型,为 true 表示按天轮转，为 false 表示不按天轮转	false
server.tongweb.log-service.rotation	开启轮转开关，为 true 表示开启轮转，为 false 表示不轮转	false
server.tongweb.log-service.console.redirect	应用日志转发到 server.log 文件 建议配置系统日志输出格式，以避免迭代输出混乱的日志内容，示例如下： server.tongweb.log-service.log-format=%m%n	true
server.tongweb.log-service.verbose	是否启用控制台日志	false

4.5.3 访问日志

访问日志记录的是访问 Web 应用时 http 请求的相关信息，包括访问处理时长、访问链接、访问 IP、请求方式等。通过分析访问日志，可以找出处理耗时多的请求，以便诊断系统性能瓶颈。

1. 访问日志配置是每个 Web 应用的通用配置；
2. 访问日志开关在虚拟主机中配置；
3. 默认的访问日志名，如：access.localhost.20.08.26.txt；
4. 访问日志轮转方式分为：不轮转、按天轮转，按小时轮转；只能选择一种轮转方式，访问日志轮转通过参数 server.tongweb.web-container.access-log.file-date-format 进行设置，不配置该参数表示不轮转，配置该参数根据表 4-6-2 中的参数配置进行轮转。
5. 使用访问日志需要再 tongweb.xml 中 <server>/<web-container>/<virtual-host> 的 accesslog-enabled 参数设置为 true，表示开启访问日志。accesslog-dir 参数表示设置访问日志的路径。
6. 在${TW_HOME}/conf/logging.properties 文件中对访问日志进行配置，访问日志的配置项如下表所示。

表 4.5- 2: 访问日志配置

访问日志配置项	说明	默认值

server.tongweb.log-service.access-log.enable	访问日志开关	false
server.tongweb.web-container.access-log.pattern	访问日志格式，如%{yyyyMMddHHmmssSSS}t %U %m %a %D	无
server.tongweb.web-container.access-log.suffix	访问日志文件名后缀，默认为.txt，访问日志后缀格式只能设置为.txt 或者.log	无
server.tongweb.web-container.access-log.file-date-format	访问日志文件格式，设置%d{yy.MM.dd.HH}表示每小时轮转一次，%d{yy.MM.dd}表示每天轮转一次	无默认值，当值为%d{yy.MM.dd}表示按天轮转，当值为%d{yy.MM.dd.HH}表示按小时轮转
server.tongweb.web-container.access-log.history-day	访问日志保存天数，默认为 0 表示一直保存，大于 0 表示保留最近多少天，file-date-format 设置为%d{yy.MM.dd}才能使用	默认值为 0 表示不删除历史日志文件，大于 0 表示保留天前的日志，该参数需要日志轮转方式为按天轮转结合使用
server.tongweb.log-service.access-log.buffered	如果为 false，每一次请求写一次日志，为 true 时则会写入缓存队列，异步从缓存队列读取并写入日志，缓冲大小为 128000 B。	默认为 true。
server.tongweb.log-service.access-log.logExtend	选用传统日志格式，或者使用扩展的访问日志，它决定日志格式 pattern。	默认为传统格式，即 false。

7. 访问日志格式输出格式说明如下：

表 4.5- 3: 访问日志输出格式说明

字符	含义
%a	远程主机的 IP 地址。
%A	本地 IP 地址
%b	发送字节数，包含 HTTP 请求头，如果为 0 时其值为 '-'
%B	发送字节数，不包含 HTTP 请求头
%H	请求协议
%l	已认证的远程逻辑用户名，一般返回 '-'
%m	请求方法（GET, POST，等等）
%p	接受该请求的本地端口
%q	请求参数，（以'? '开头）
%r	请求的第一行（请求方法以及 request URI）

%s	HTTP 响应状态码
%S	用户的 session ID
%t	日期和时间，使用 Common Log 格式
%u	认证的远程用户,没有则返回 '-'
%U	请求的 URL 路径
%v	本地 server 名称
%D	处理请求所有时间，单位为毫秒
%T	处理请求所有时间，单位为秒（seconds）
%I	当前请求线程名

此外，它还支持写入请求/响应头，cookies，session，请求属性与时间戳格式的信息

访问日志格式	访问日志格式说明
% {xxx}i	请求头信息，xxx 为请求头属性。
%{xxx}o	响应头信息。
% {xxx}c	cookies 信息。
%{xxx}r	xxx 为 ServletRequest 的属性。
%{xxx}s	xxx 为 HttpSession 的属性。
%{xxx}t	xxx 是一个加强的 SimpleDateFormat 模式，支持所有的 SimpleDateFormat 格式，并且支持额外的属性。
%{xxx}t	额外支持的属性。

两个简写配置：

访问日志格式	访问日志格式说明
common	%h %l %u %t \"%r\" %s %b
combined	%h %l %u %t \"%r\" %s %b \"%{Referer}i\" \"%{User-Agent}i\"

8. 访问日志格式输出扩展格式说明如下：

扩展模式的访问日志其实并不是对原来访问日志的扩展，而是以不同的 pattern 对请求/响应进行记录。从属性来看，它只少了个"locale"属性，以及 pattern 采用不同的值。

这里的 pattern 是由一些格式符号组成，有一些符号须要一些额外的前缀。一般的前缀有： "c"表示客户端即远程。"s"表示服务器端即本地，"cs"表示客户端到服务器端。"sc"表示服务器端到客户端。"x"表示"application specific"

表 4.5- 4: 访问日志扩展模式的格式说明

访问日志格式	访问日志格式说明
bytes	发送字节，不包括 HTTP headers，如果为零显示为'-'
c-dns	远程主机名（或者 IP 地址，如果连接器的 enableLookups 为 false）
c-ip	远程（客户端）IP 地址
cs-method	请求方法（GET, POST 等）
cs-uri	请求的 URL

cs-uri-query	请求参数（如果存在，以"?"开头）
cs-uri-stem	请求的 URL 路径
date	yyyy-mm-dd format 格式的 GMT 日期
s-dns	本地主机名
s-ip	本机 ip 地址
sc-status	响应状态
time	该请求的时间
time-taken	从开始到结束，服务器在该请求上所花的时间（以秒为单位）
x-threadname	当前请求线程名

可以用 x-H(XXX)表示 HttpRequestServlet 的方法。

x-H(authType)
x-H(characterEncoding)
x-H(contentLength)
x-H(locale)
x-H(protocol)
x-H(remoteUser)
x-H(requestedSessionId)
x-H(requestedSessionIdFromCookie)
x-H(requestedSessionIdValid)
x-H(scheme)
x-H(secure)

也支持从 headers cookies, context,request 或 session 属性和 request 参数中获取信息：

访问日志格式	访问日志格式说明
cs(XXX)	请求头
sc(XXX)	响应头
x-A(XXX)	servlet 上下文属性
x-C(XXX)	具有专用名的首个 cookie 值。例如，要记录一个叫做 JSESSIONID，则指定该属性为 x-C(JSESSIONID)。
x-O(XXX)	将所有名称为 XXX 的响应头串接
x-P(XXX)	请求参数
x-R(XXX)	request 属性

x-S(XXX)	session 属性
cs(XXX)	请求头
sc(XXX)	响应头
x-A(XXX)	servlet 上下文属性
x-C(XXX)	具有专用名的首个 cookie 值。例如，要记录一个叫做 JSESSIONID，则指定该属性为 x-C(JSESSIONID)
x-O(XXX)	将所有名称为 XXX 的响应头串接

4.5.4 日志级别

TongWeb V7.0.6 容器云版支持的日志级别：分别是 ERROR、WARN、INFO、DEBUG、TRACE，支持模块级别的日志配置输出。可通过配置日志级别的参数设置不同的模块输出不同级别的日志信息，TongWeb V7.0.6 容器云版目前支持 core、web-container、config、boot、monitor、commons、tool、access、admin、other 这些模块的日志级别的配置。除了 TongWeb V7.0.6 容器云版自己的模块输出自己模块名，其他模块输出模块的输出模块名全为[other]。

在**${TW_HOME}/conf/logging.properties** 文件中对日志级别进行配置，日志级别的配置如下所示：

表 4.5- 5：日志级别配置

日志级别配置	说明	默认值
server.tongweb.log-service.level.core	core 模块日志级别设置	INFO
server.tongweb.log-service.level.web-container	web-container 模块日志级别设置	INFO
server.tongweb.log-service.level.config	config 模块日志级别设置	INFO
server.tongweb.log-service.level.boot	boot 模块日志级别设置	INFO
server.tongweb.log-service.level.monitor	monitor 模块日志级别设置	INFO
server.tongweb.log-service.level.commons	commons 模块日志级别设置	INFO
server.tongweb.log-service.level.tool	tool 模块日志级别设置	INFO
server.tongweb.log-service.level.access	access 模块日志级别设置	INFO
server.tongweb.log-service.level.admin	admin 模块日志级别设置	INFO
server.tongweb.log-service.level.other	other 模块日志级别设置	INFO
server.tongweb.log-service.level.license	license 模块日志级别设置	INFO
server.tongweb.log-service.level.session	session 模块日志级别设置	INFO

4.5.5 Kafka 日志

TongWeb V7.0.6 容器云版支持将日志信息输出到 Kafka 中，以便于日志的统一采集和日志分析等。

1. 若用户想输出日志到 KafKa 中需要手动搭建好 KafKa 消息系统。再在 **logging.properties** 中配置 KafKa 相关的信息。

   通过 **server.tongweb.log-service.kafka.enable** 启动日志输出到 Kafka 中

示例：

server.tongweb.log-service.kafka.enable=true

2. 如果 kafak 推送日志乱码，可以通过在 **external.vmoption**s 中添加 **-Dkafka.logger.encoding** 参数设置 Kafka 日志推送编码格式。配置可以参考 4.11 Java 启动参数配置。
3. 在**${TW_HOME}/conf/logging.properties** 文件中 Kafka 日志进行配置，Kafka 的相关配置如下表所示：

表 4.5- 6：Kafka 日志配置

KafKa 配置	说明	默认值
server.tongweb.log-service.kafka.enable	Kafka 开关	false
server.tongweb.log-service.kafka.topic	配置 kafka 订阅的 topic	无
server.tongweb.log-service.kafka.keying.strategy	传输通信策略，值为 ContextName 表示此策略使用登录的 CONTEXT_NAME 作为消息密钥、值为 HostName 表示此策略使用 HOSTNAME 作为消息密钥、值为 LoggerName 表示此策略使用记录器名称作为消息键、值为 NoKeyKeying 表示不生成消息密钥、值为 ThreadNameKeying 表示此策略使用调用线程名称作为消息键	无
server.tongweb.log-service.kafka.delivery.strategy	Kafka 消息提交策略，AsynchronousDeliveryStrategy 表示异步提交，BlockingDeliveryStrategy 表示阻塞提交策略。默认采用阻塞提交策略	无
server.tongweb.log-service.kafka.isAppendTimestamp	是否开启消息时间戳，false 表示不开启，为 true 表示开启	无
server.tongweb.log-service.kafka.partition	kafka 分区配置	无
server.tongweb.log-service.kafka.producer.bootstrap.servers	用于建立与 kafka 集群的连接，这个 list 仅仅影响用于初始化的 hosts，来发现全部的 servers。 格式：host2:port2；可以配置多个	无
server.tongweb.log-service.kafka.producer.acks	Server 完成 producer request 前需要确认的数量。 acks=0 时，producer 不会等待确认，直接添加到 socket 等待发送； acks=1 时，等待 leader 写到 local log 就行； acks=all 或 acks=-1 时，等待 isr 中所有副本确认	无
server.tongweb.log-service.kafka.producer.linger.ms	Producer 默认会把两次发送时间间隔内收集到的所有 Requests 进行一	无

	次聚合然后再发送，以此提高吞吐量，而 linger.ms 则更进一步，这个参数为每次发送增加一些 delay，以此来聚合更多的 Message。	
server.tongweb.log-service.kafka.producer.max.block.ms	控制 block 的时长，当 buffer 空间不够或者 metadata 丢失时产生 block	无
server.tongweb.log-service.kafka.producer.client.id	当向 server 发出请求时，这个字符串会发送给 server，目的是能够追踪请求源	无
server.tongweb.log-service.kafka.producer.metadata.fetch.timeout.ms	在第一次将数据发送到某 topic 时，需先 fetch 该 topic 的 metadata，得知哪些服务器持有该 topic 的 partition，该值为最长获取 metadata 时间（0.9 版以上已经废弃该参数）	无
server.tongweb.log-service.kafka.producer.buffer.memory	Producer 可以用来缓存数据的内存大小。该值实际为 RecordAccumulator 类中的 BufferPool，即 Producer 所管理的最大内存。 如果数据产生速度大于向 broker 发送的速度，producer 会阻塞 max.block.ms，超时则抛出异常	无
server.tongweb.log-service.kafka.producer.compression.type	Producer 用于压缩数据的压缩类型，取值：none, gzip, snappy, or lz4	无

支持的 Kafka Server 端版本：2.6.0；2.5.1；2.5.0；2.4.1；2.4.0；2.3.1；2.3.0；2.2.2；2.2.1；2.2.0；2.1.1；2.1.0；2.0.1；2.0.0；1.1.0

4.5.6 本地文件配置

TongWeb V7.0.6 容器云版单机环境下读取的是 **tongweb.xml**、**application.properties**、**logging.propreties** 这三个文件的配置。tongweb.xml 是进行 tongweb 核心功能的配置，application.properties 是对应用的配置（如 License）、logging.properties 是对日志模块功能的配置。

TongWeb V7.0.6 容器云版默认采用的是单机配置方式，要确保 **tongweb/conf** 下 tongweb.xml、application.properties、logging.properties 这三个文件存在。

4.5.7 异步日志

异步日志是指日志的输出方式不在是打印一条日志就输出一条日志，而是采用异步线程的方式，由后台线程去处理日志的输出。采用异步日志的输出能够提高系统的性能。

1. 如果要进行异步日志的输出，先要打开异步日志开关。设置 **server.tongweb.log-service.async.enable** 为 **true**，表示开启异步日志开关。

   示例：

server.tongweb.log-service.async.enable=true

2. 在**${TW_HOME}/conf/logging.properties** 文件中进行异步日志进行配置，异步日志的相关配置如下表所示：

表 4.5- 7：异步日志参数表

配置项	说明	默认值
server.tongweb.log-service.async.enable	异步日志开关	false
server.tongweb.log-service.async.queueSize	队列的最大容量	100000
server.tongweb.log-service.async.innerAppenderName	异步处理 Appender;值只能为 FILE,ROLLINGFILE	File
server.tongweb.log-service.async.discardingThreshold	默认，当队列还剩余 20% 的容量时，会丢弃级别为 TRACE, DEBUG 与 INFO 的日志，仅仅只保留 WARN 与 ERROR 级别的日志。想要保留所有的事件，可以设置为 0	0
server.tongweb.log-service.async.includeCallerData	获取调用者的数据相对来说比较昂贵。为了提高性能，默认情况下不会获取调用者的信息。默认情况下，只有像线程名或者 MDC 这种"便宜"的数据会被复制。设置为 true 时，appender 会包含调用者的信息	false
server.tongweb.log-service.async.maxFlushTime	根据所引用 appender 队列的深度以及延迟， AsyncAppender 可能会耗费长时间去刷新队列。当 LoggerContext 被停止时，AsyncAppender stop 方法会等待工作线程指定的时间来完成。使用 maxFlushTime 来指定最大的刷新时间，单位为毫秒。在指定时间内没有被处理完的事件将会被丢弃。	1000
server.tongweb.log-service.async.neverBlock	默认为 false，在队列满的时候 appender 会阻塞而不是丢弃信息。设置为 true，appender 不会阻塞你的应用而会将消息丢弃	false

4.6 配置中心

4.6.1 概述

TongWeb V7.0.6 容器云版提供单机和集群版的配置支持，在单机环境下通过 tongweb.xml、application.properties、logging.properties 文件进行参数的配置。在集群云环境下支持利用 Nacos 提供配置中心的支持。Nacos 具有支持基于 DNS 和 RPC 的服务发现机制、动态配置服务、动态 DNS 服务等特点，能够很好的在集群环境下提供配置服务。TongWeb V7.0.6 容器云版与 Nacos 进行了完美的集成，支持 xml 文件格式、properties 文件格式、Yml 文件格式等常用文件格式。用户只需要在 Nacos 控制台上进行配置就能为集群应用服务提

供配置支持。

4.6.2 非容器云环境使用配置中心

TongWeb V7.0.6 容器云版支持在单机或者非容器云环境下使用配置中心，这样一来方便用户在集群环境下使用统一配置中心，便于用户维护和升级。非容器云版使用配置中心需要先搭建好 Nacos 配置中心，然后修改**${TW_HOME}/bin/env.options** 文件。

环境变量参数如下：

CONFIG_REMOTE_ENABLE：是否启用远程配置

CONFIG_REMOTE_TYPE：远程配置中心类型

CONFIG_SERVER：远程配置中心地址和端口

LOG_NACOS_DATA_ID：Nacos 日志功能的 Data ID

APP_NACOS_DATA_ID：Nacos 应用功能的 Data ID

TONGWEB_NACOS_DATA_ID：Nacos TongWeb V7.0.6 配置的 Data ID

NACOS_GROUP=DEFAULT_GROUP：Nacos 日志，应用，TongWeb V7.0.6 的 Group

在 Windows/Linux 环境下修改 env.options，如下图所示：

CONFIG_REMOTE_ENABLE=false

CONFIG_REMOTE_TYPE=nacos

CONFIG_SERVER=127.0.0.1:8848

LOG_NACOS_DATA_ID=logging

APP_NACOS_DATA_ID=application

TONGWEB_NACOS_DATA_ID=tongweb

NACOS_GROUP=DEFAULT_GROUP

4.6.3 容器云环境配置

TongWeb V7.0.6 容器云版支持在容器云环境下的统一配置，无论用户是用的 Docker 进行的编排还是 K8s 进行的编排都支持统一配置。TongWeb V7.0.6 容器云版利用 Nacos 作为配置中心，用户需要先搭建好 Nacos 的环境。

4.6.3.1 Nacos 配置

Nacos 的安装见附录 B Nanos 安装说明, 登录 Nacos 控制台打开**配置管理**->**配置列表**, 可以看到添加的配置列表。如下图所示：

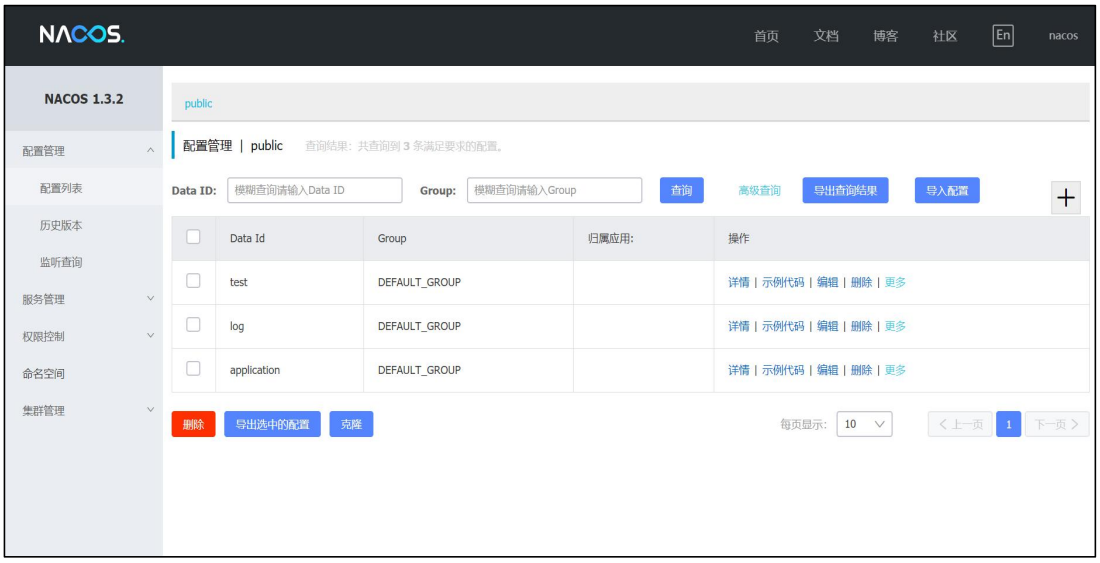


图 4.6- 1：Nacos 配置界面

点击**加号**进行配置添加，配置配置文件的 **Data ID** 和 **Group** 在配置内容里根据文件类型添加具体的配置，添加后就可以在配置列表中看到添加的配置，如下图示例新建配置：

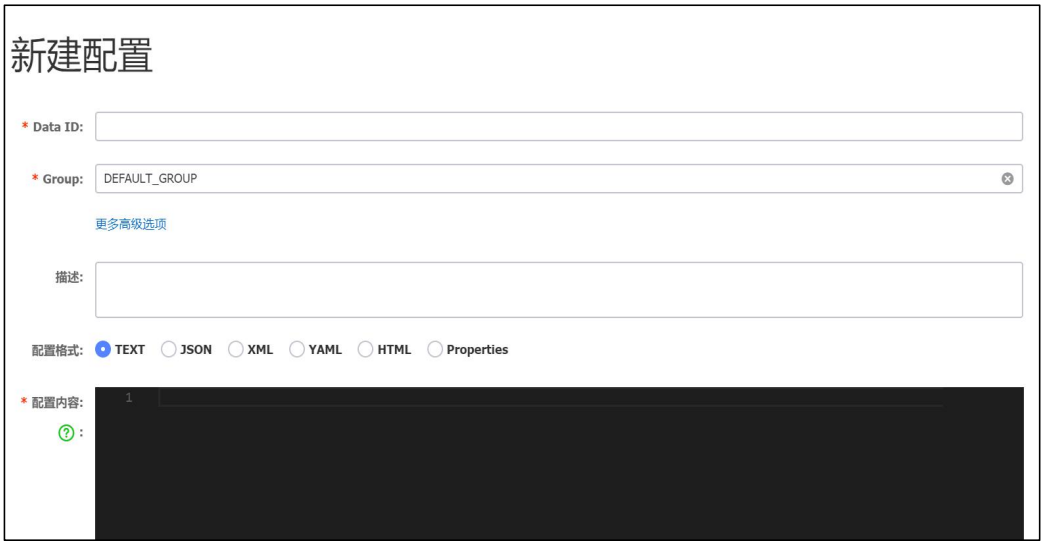


图 4.6- 2：Nacos 新建配置

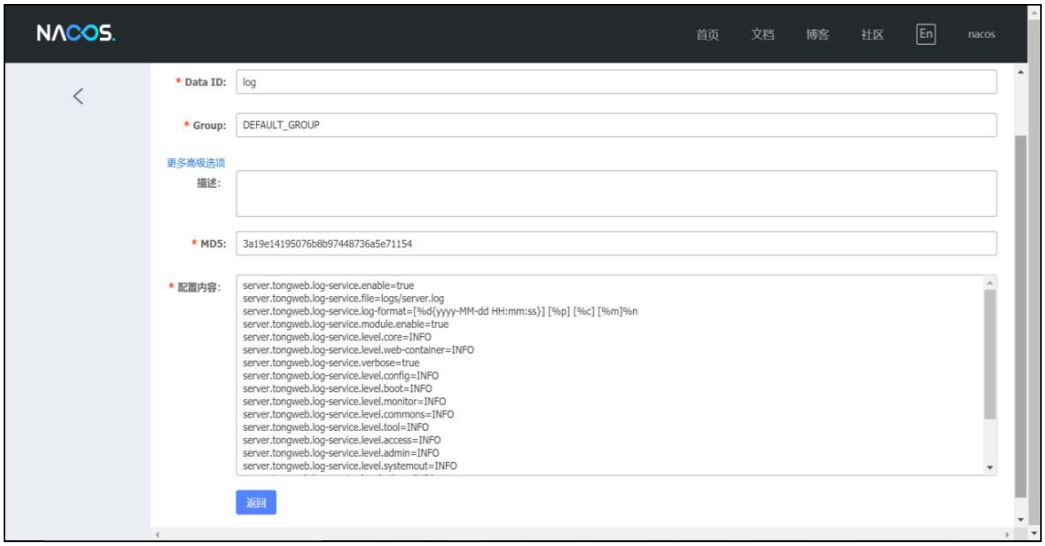


图 4.6- 3：Nacos 日志配置

图 4.6- 4：application 配置

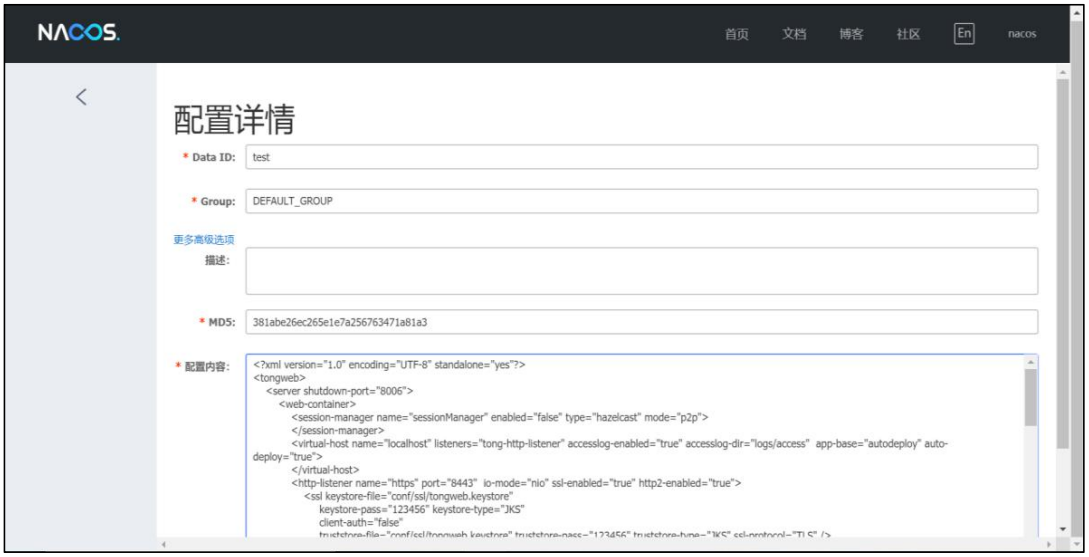


图 4.6- 5：tongweb.xml 配置

4.6.3.2 参数说明

容器云环境下需要对配置中心的参数进行配置，docker 在 **docker-tongweb.env** 文件中进行配置，k8s 在 **k8s-tongweb-env.options** 文件中进行配置。

1. 通过 CONFIG_REMOTE_ENABLE 启动远程配置。
   CONFIG_REMOTE_ENABLE=true
2. 通过 CONFIG_REMOTE_TYPE 指定配置中心类型，目前 TongWeb V7.0.6 容器云版采用的是 Nacos 配置中心。
   CONFIG_REMOTE_TYPE=nacos
3. 通过 nacos_group 参数指定 Nacos Group，目前日志配置、license 配置、TongWeb V7.0.6 核心模块配置采用的是同一个 Group。
   NACOS_GROUP=DEFAULT_GROUP
4. 通过 CONFIG_SERVER 指定 Nacos 配置中心的地址和访问端口。

CONFIG_SERVER=127.0.0.1:8848

5. 通过 LOG_NACOS_DATA_ID 指定日志配置的 Data ID，该项配置需要和 Nacos 的日志配置保持一致。
   LOG_NACOS_DATA_ID=logging
6. 通过 APP_NACOS_DATA_ID 指定应用的配置,该项配置需要和 Nacos 的应用配置保持一致
   APP_NACOS_DATA_ID=application
7. 通过 TONGWEB_NACOS_DATA_ID 配置 tongweb 配置的 Data ID，该项配置需要和 TongWeb V7.0.6 的配置保持一致。
   TONGWEB_NACOS_DATA_ID=tongweb

4.6.3.3 容器云配置

在 Docker 环境下需要在**${TW_HOME}/bin/docker-set-env.env** 进行 Nacos 配置中心的参数配置，在 K8S 环境下需要在**${TW_HOME}/bin/k8s-set-env.options** 进行 Nacos 配置中心的参数配置。配置参数的说明见 4.6.3.2 参数说明。

4.7 监控服务

4.7.1 概述

监视服务用于显示 TongWeb V7.0.6 服务器在运行时的内存、线程、数据源、操作系统等方面的资源占用情况。利用 Prometheus 对数据进行图表化的展示，以显示其历史变化轨迹等，监视服务的意义在于帮助维护人员实时准确地了解系统运行情况,并能据此对具体的参数进行相应的调节，以使得整个系统更加稳定和流畅地运行。

4.7.2 非容器云环境监控环境搭建

TongWeb V7.0.6 服务提供对常用的资源进行监控，采用 Prometheus 进行监控，用户需要搭建好 Prometheus，搭建好 Prometheus 后，在 Prometheus 根目录下的 **prometheus.yml** 中添加如下配置，多个 tongweb 需要添加多个 **job_name**：

```
scrape_configs:
  - job_name: 'prometheus'
    metrics_path: '/monitor/prometheus'
    static_configs:
- targets: ['[tongweb.host]:[port]']
```

4.7.3 容器云环境监控环境搭建

TongWeb V7.0.6 容器云版支持 Prometheus 与 Docker 和 k8s 的完美集成，利用 Prometheus 抓取 TongWeb V7.0.6 容器云版的数据进行监控展示，能够实时的监控每个 Node 及每个 Pod 上面容器运行的资源情况，方便运维人员及时清楚的知道容器运行的情况，

并作出响应。

4.7.4 Prometheus 与 Docker 集成

TongWeb V7.0.6 容器云版支持 Prometheus 与 Docker 的集成，Prometheus 从 TongWeb V7.0.6 容器中抓取数据并进行可视化展示，用户能够监控到每个 Docker 容器资源的运行情况。Prometheus Docker 化安装件见附录 C，下面将介绍 Prometheus 的使用。

1. 进入 **http://<ip>:<port>/graph**，选择对应的 **execute**，在 execute 按钮右边有下拉选择框，可以看到能够监控哪些资源。
2. 选择对应的资源，切换到 Graph，就能够看到 Prometheus 已经对该资源进行监控。

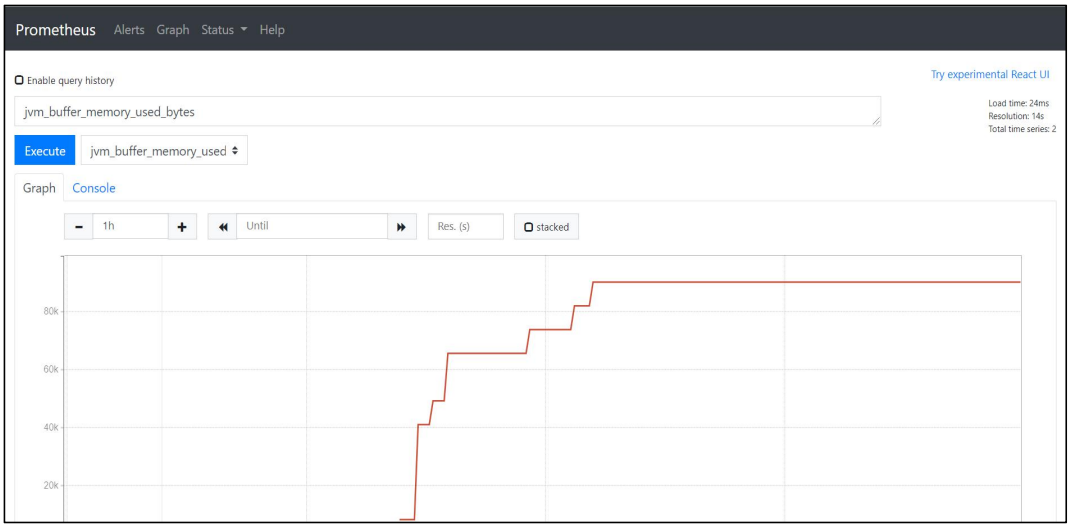


图 4.7- 1：Prometheus 监控的 Docker 资源

4.7.5 Prometheus 与 K8s 集成

TongWeb V7.0.6 容器云版支持 Prometheus 与 K8s 的集成，Prometheus 支持从 K8s 集群中读取数据并进行展示，利用这一特点在 K8s 环境下安装 Prometheus 可以对 TongWeb V7.0.6 集群进行实时监控，便于用户分析每个 Pod 上资源使用情况。利用 K8s 可以动态修改资源限制参数的特点，根据 Prometheus 监控到的数据，可以动态调整每个 Pod 的参数数据。Prometheus 在 K8s 环境下的安装见附录 C，下面介绍 Prometheus 的使用：

1. 进入 **http://<ip>:<port>/graph**，选择对应的 **execute**,在 execute 按钮右边有下拉选择框，可以看到能够监控哪些资源。
2. 选择对应的资源，切换到 Graph，就能够看到 Prometheus 已经对该资源进行监控。

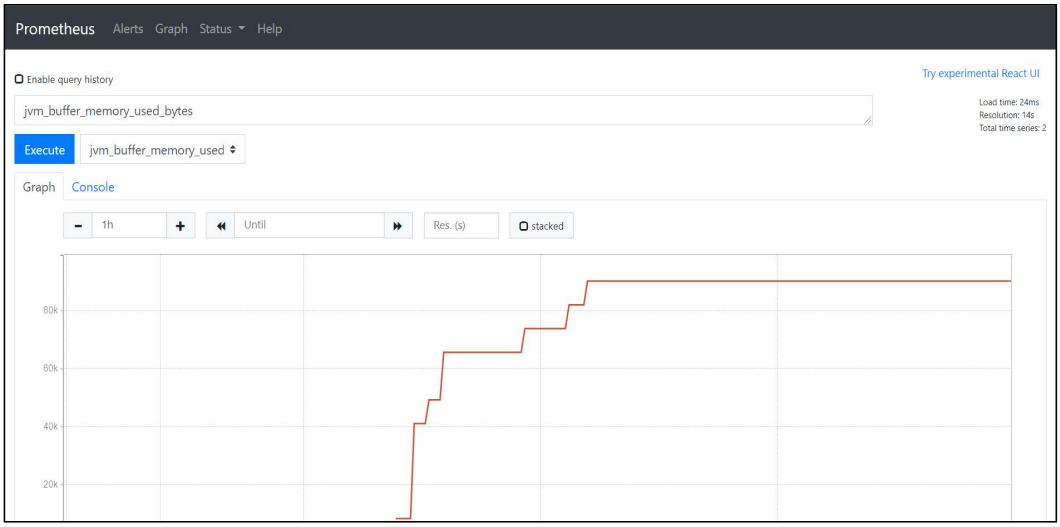


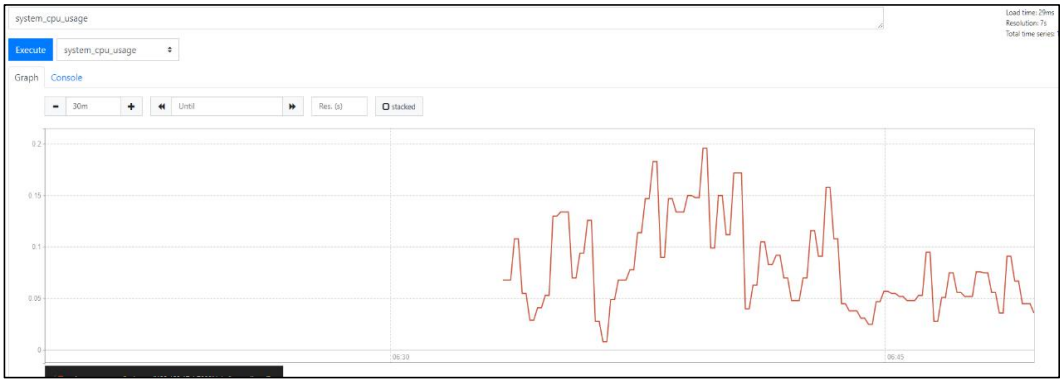
图 4.7- 2：Prometheus 监控的 K8s 资源

4.7.6 Prometheus 监控展示

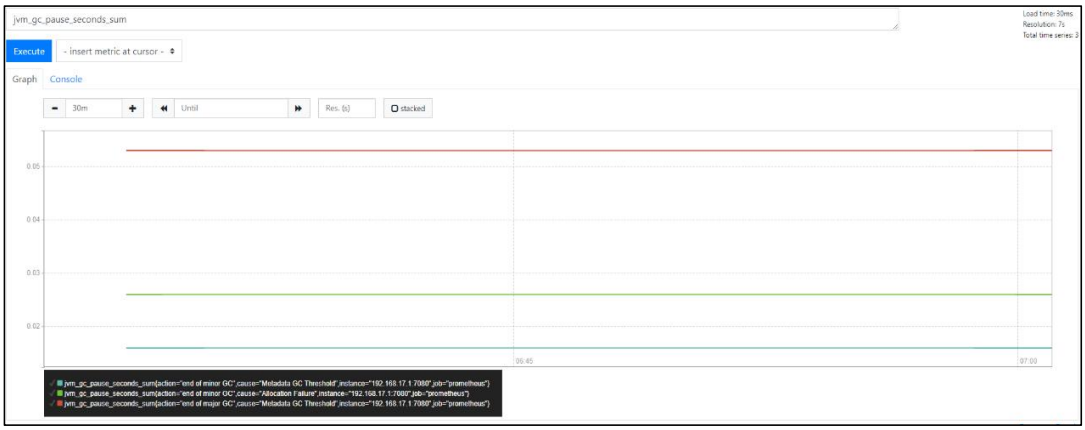
Prometheus 是一套开源的系统监控报警框架，是 Kubernetes 之后的第二个托管项目。Prometheus 具有时间序列的多维数据模型、PromQL（是一种灵活的查询语言）、不依赖分布式存储、时间序列收集通过 HTTP 上的拉取模型进行、通过中间网关支持推送时间序列、通过服务发现或静态配置发现目标、多种模式的图形化和仪表盘支持等特点，成为了监控系统的首选方案。

TongWeb V7.0.6 容器云版与 Prometheus 进行了完美集成，在容器云和单机都能够进行数据的实时监控。以下展示了 TongWeb V7.0.6 容器云版和 Prometheus 系统集成之后的部分效果图。

1. CPU 使用情况监控展示图，展示 Cpu 的使用情况。



2. GC 暂停时间监控展示图，展示 GC 的暂停情况。



3. JVM 各状态线程监控展示图，展示 JVM 里线程信息。

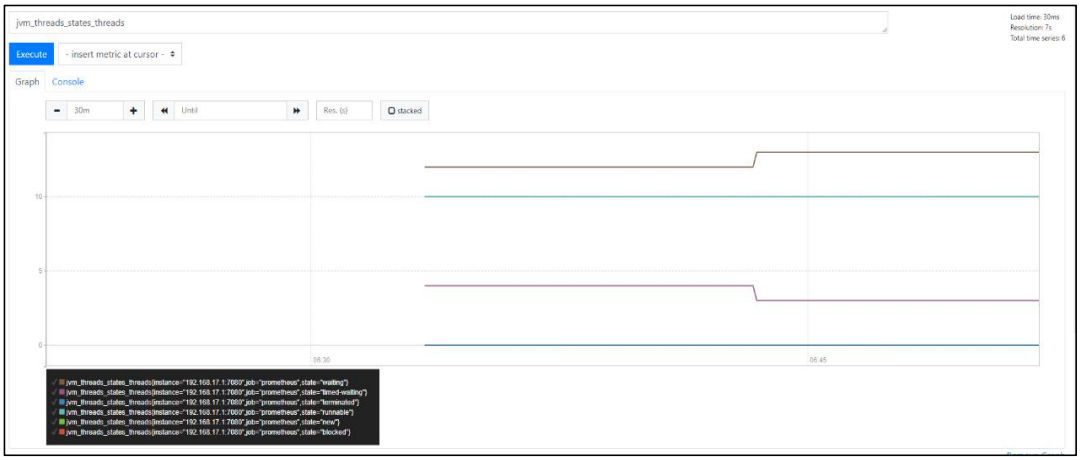


图 4.7- 3：JVM 各状态线程监控展示图

4. JVM 运行时内存监控展示图，实时监控 JVM 内存信息

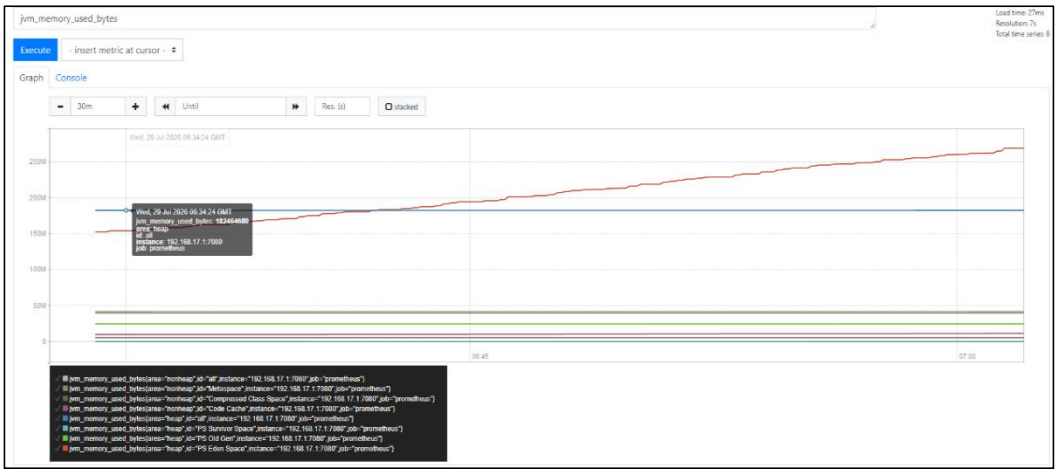


图 4.7- 4：JVM 运行时内存监控展示图

5. 请求平均响应时间监控展示图，监控 Http 请求的响应时间

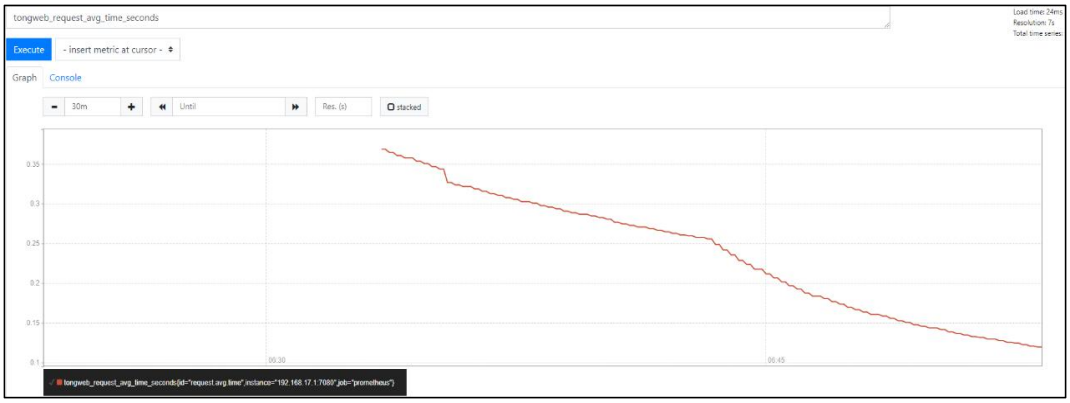


图 4.7- 5：请求平均响应时间监控展示图

6. 通道请求次数监控展示图，监控每个请求的访问次数

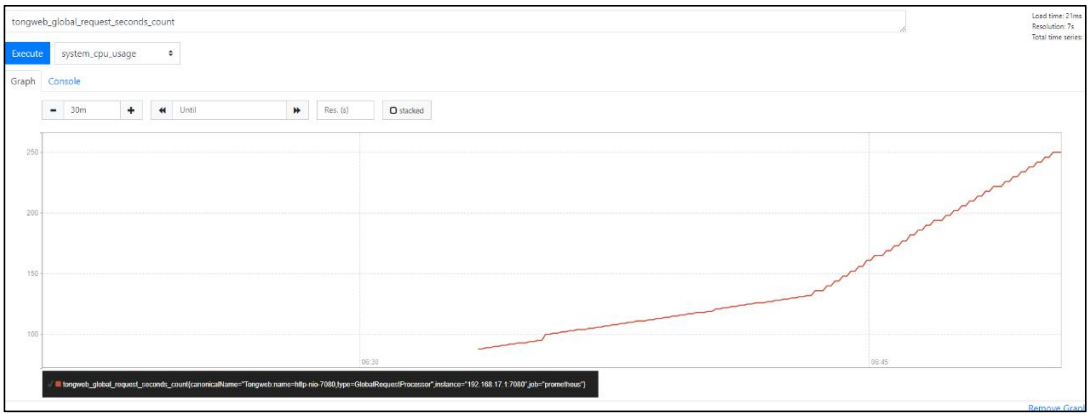


图 4.7- 6：通道请求次数监控展示图

4.7.7 参数配置

1. 监控服务的配置项在${TW_HOME}/conf/tongweb.xml 的**<server>/<monitor-service>**节点中，各配置项说明如下：

表 4.7- 1：监控服务配置项

配置项名称	说明	默认值
monitoring-enabled	是否启用监控服务	true

2. 监控配置项在${TW_HOME}/conf/tongweb.xml 的**<server>/<monitor-service>/<monitor-config>**节点中，各配置项说明如下：

配置项名称	说明
name	监控配置项名称
monitoring-enabled	是否启用该项配置
service-info	监控数据与第三方系统集成信息配置

Service-info 有如下配置：

配置项名称	说明	默认值
type	监控数据对接监控平台种类	无

context-root	监控模块在 host 下的路径	/monitor
request-url	监控数据暴露地址	/prometheus

示例：

```
<monitor-service monitoring-enabled="true"
service-info="type:prometheus,context-root:/monitor,request-url:/prometheus">
```

配置项参数如下：

监控项名称	说明
Memory	JVM 虚拟机运行时内存
JVMMemoryPool	JVM 虚拟机内存池情况
GarbageCollector	垃圾收集器
JVMThread	JVM 线程
Compilation	JVM 编译器
ClassLoading	JVM 类加载器
Runtime	运行时监控
OperatingSystem	操作系统信息
TWServer	TongWeb V7.0.6 服务器信息
ConnectorAndThreadPool	通道请求监控
DataSource	数据源
WebModule	服务器上的应用部署信息
SessionManager	会话信息
Loader	应用服务器类加载器
ResourceCache	应用服务器进程缓存资源
Request	全局请求数据(满请求，平均响应时间等)

示例：

```
<monitor-service monitoring-enabled="false">
<monitor-config name="Memory" monitoring-enabled="false"/>
<monitor-config name="JVMMemoryPool"/>
<monitor-config name="GarbageCollector"/>
<monitor-config name="JVMThread"/>
<monitor-config name="Compilation"/>
<monitor-config name="ClassLoading"/>
<monitor-config name="Runtime"/>
<monitor-config name="OperatingSystem"/>
<monitor-config name="TWServer"/>
<monitor-config name="ConnectorAndThreadPool"/>
```

```
<monitor-config name="DataSource"/>
<monitor-config name="WebModule"/>
<monitor-config name="SessionManager"/>
<monitor-config name="Loader"/>
<monitor-config name="ResourceCache"/>
<monitor-config name="Request"/>
</monitor-service>
```

4.8 JMX 连接配置

JMX 技术是 Java 关于应用和资源管理的标准技术，它为开发标准化、集中式的、安全的远程管理应用提供了方案。TongWeb V7.0.6 容器云版提供配置安全的 JMX 连接。

JMX 参数配置

1. 用户需要在**${TW_HOME}/bin/external.vmoptions** 文件添加如下 **jmx** 连接配置：

特别说明：

在 Docker 环境下使用 JMX 时如果 jmx 地址和端口经常变动，可以把 external.vmoptions 文件挂载出来方便修改参数值。尽量不要进行端口的修改，端口的修改需要重新进行映射，会涉及到容器的重启。

示例：

```
-Dcom.sun.management.jmxremote.port=8999
-Dcom.sun.management.jmxremote.authenticate=false
-Dcom.sun.management.jmxremote.ssl=false
-Djava.rmi.server.hostname=127.0.0.1
```

2. 在 K8s 环境下用户需要在 Deployment 的 yaml 文件中添加 **JAVA_OPTS** 环境变量，并设置如下参数值。

示例：

```
-Dcom.sun.management.jmxremote.port=8999
-Dcom.sun.management.jmxremote.authenticate=false
-Dcom.sun.management.jmxremote.ssl=false
-Djava.rmi.server.hostname=master 节点的地址
```

3. 在 k8s 环境下，如果远程访问 jmx 可以通过端口转发的方式进行访问。

示例：

```
kubectl port-forward -n namespace <pod-name> --address 0.0.0.0 8999:8999
```

4. 配置鉴权：

```
-Dcom.sun.management.jmxremote.ssl=false
-Dcom.sun.management.jmxremote.authenticate=true
-Dcom.sun.management.jmxremote.password.file=/usr/local/java/jmx/jmxremote.password
-Dcom.sun.management.jmxremote.access.file=/usr/local/java/jmx/jmxremote.access
```

其中

-Dcom.sun.management.jmxremote.authenticate=true-----修改为 true

5. 创建 **jmxremote.password** 文件：

```
cp $JAVA_HOME/jre/lib/management/jmxremote.password.template
/usr/local/java/jmx/jmxremote.password

chmod 700 jmxremote.password  (注意: 这个文件默认是不可写的)
```

修改 jmxremote.password，添加用户名和密码：

示例：

```
fox    123456
```

设置 jmxremote.password 的路径：

-Dcom.sun.management.jmxremote.password.file=jmxremote.password 的路径

6. 创建 **jmxremote.access** 文件：

```
cp $JAVA_HOME/jre/lib/management/jmxremote.access jmxremote.access
```

编辑 jmxremote.access，设置 jmx 用户权限：

示例：

```
fox      readwrite
```

设置 jmxremote.access 的路径：

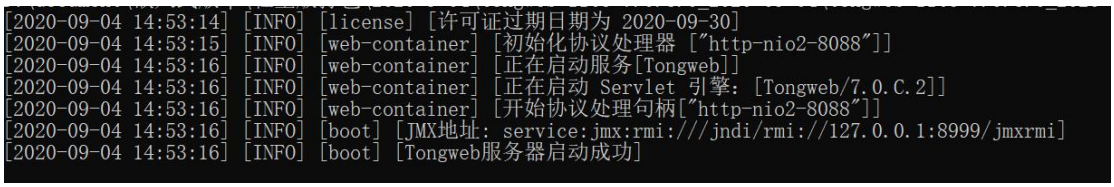
-Dcom.sun.management.jmxremote.access.file=/usr/local/java/jmx/jmxremote.access"

7. 查看 JMX URL：

在启动服务器时已将 JMX URL 提供给用户，具体信息可在日志文件 **tongweb/logs/server.log** 中查找。如 URL for the Standard JMXConnectorServer: [service:jmx:rmi:///jndi/rmi://testserver:8999/jmxrmi]

说明：中括号中的信息即为连接服务器的 JMX URL，其中 testserver 为服务器所在机器的主机名或 IP 地址(本地或者远程)。

8. 启动 tongweb 会打印图所示信息：



4.9 国际化配置

TongWeb V7.0.6 容器云版提供国际化支持，方便不同区域的用户都能够使用 TongWeb V7.0.6。为支持国际化配置用户需要在 external.vmoptions 文件中配置如下参数：

表 4.9- 1: 国际化支持配置

国际化参数	说明	默认值
user.language	语言	根据系统语言
user.region	区域	根据系统区域

4.10 配置加密工具

配置加密工具

TongWeb V7.0.6 容器云版提供加密工具的使用。如果需要对数据源的用户名和密码进行加密，需要进入${TW_HOME}/bin 目录下执行如下脚本进行加密：

使用工具运行命令

cipher-tool.bat/cipher-tool.sh [-mode] [s]

用法：

-mode：模式，-e 为可选参数，缺省值为-e

-e 为加密模式：对待处理字符串进行加密

s：待处理字符串

使用帮助说明：**cipher-tool.bat/cipher-tool.sh -h**

例如：加密字符串"root123456"

- Windows 环境加密:

cipher-tool.bat -e root123456

```
D:\tongweb\twnext\tw-next\release\target\tongweb-cloud-7.0.C.0_2020-09-15\tongweb\bin>cipher-tool.bat -e root123456
enc:2mSix4fYWg4pxnsmM1AIGg==
```

拷贝整个输出字符串到配置使用处。

- Linux 环境加密：

./cipher-tool.sh -e root123456

```
[root@localhost bin]# ./cipher-tool.sh root123456
enc:2mSix4fYWg4pxnsmM1AIGg==
```

拷贝整个输出字符串到配置使用处。

4.11 JAVA 启动参数配置

启动参数指的是服务器启动脚本中设置的参数，包括 JVM 参数，服务器参数。其中 JVM 参 数又包括常见的最大堆内存参数和最小堆内存参数、文件格式参数，日志格式参数等。非 k8s 环境在 **external.vmoptions** 文件中可以添加启动参数的配置。k8s 环境中在 Deployment 的 **yaml** 文件中添加 JAVA_OPTS 环境变量。

K8s 中 **JAVA_OPTS** 参数配置示例如下：

```
env:
- name: CONFIG_REMOTE_ENABLE
  value: "false"
- name: CONFIG_REMOTE_TYPE
  value: "nacos"
- name: CONFIG_SERVER
  value: "168.1.13.71:8848"
- name: LOG_NACOS_DATA_ID
  value: "log"
- name: APP_NACOS_DATA_ID
  value: "application"
- name: TONGWEB_NACOS_DATA_ID
  value: "test"
- name: NACOS_GROUP
  value: "DEFAULT_GROUP"
- name: ENV_TYPE
  value: "cloud"
- name: JAVA_OPTS
  value: "-Dconfig=XXX -Xmx512m"
```

启动参数配置说明如下:

-Dproperties.file.encoding=UTF-8 设置读取 properties 文件编码格式为 UTF-8

-Dcommons.logger.encoding=UTF-8 设置日志输出的编码格式为 UTF-8

-Dkafka.logger.encoding=UTF-8 设置 kfaka 日志推送编码格式

-Xms 初始堆大小

-Xmx 最大堆大小

-XX:NewSize 设置年轻代大小

-XX:PermSize 设置持久代初始值

-XX:MaxPermSize 设置最大持久代

-XX:NewRatio 年轻代与老年代的比值

-XX:SurvivorRatio Eden 区与 Survivor 区的大小比值

-XX:+UseParNewGC 设置年轻代为并行收集

-Duser.language 设置日志输出语言

-Duser.region 设置区域

第五章 容器常见操作

5.1 Docker 常见操作

5.1.1 修改参数配置

Docker 中修改参数配置分为修改内部参数值和修改外部参数值的方式，修改内部参数值是指用户只需要修改内部的参数配置，并进行重启即可生效。修改外部参数值是指用户需要运行 Docker 容器时指定参数值，如目录挂载，端口映射等。

5.1.1.1 修改内部参数值

如果用户只是修改配置文件，不涉及目录挂载的变动，端口的映射等需要运行时进行参数配置。可以通过进入到容器中修改参数配置文件，如日志配置、license 配置、tongweb 一些参数配置、JMX 地址信息等。修改完成后重启容器，即可生效。

通过如下命令进入到容器中：

docker exec -it 容器名/容器 ID /bin/bash

进入/opt/TongWeb V7.0.6 目录下，这就是容器中存放 TongWeb V7.0.6 的目录。根据用户的实际条件结合用户手册进行参数的修改，修改完成后通过运行如下命令进行容器的重启。

docker restart 容器名/容器 ID

5.1.1.2 修改外部参数值

如果用户需要在运行时指定参数，但又不想重新制作镜像并运行。用户可以根据当前的运行容器重新制作新的镜像并运行，即可保留当前容器的配置参数，也可通过运行时指定新的参数配置。

通过如下命令制作新的镜像：

docker commit 容器名称/容器 ID 新的镜像名称

然后通过 **docker run** 命令启动新的容器，这时可以自行制定挂载目录，映射端口。

5.1.2 运行时指定参数

Docker 运行时可以设置启动参数限制，如通过--cpus 设置使用 cpus 数量，通过-m 设置使用内存等。常见的 **docker run** 命令可选参数如下：

-d, --detach=false,: 后台运行容器，并返回容器 ID。

-i, --interactive=false: 以交互模式运行容器，通常与-t 同时使用。

-P, --publisch-all=false: 随机端口映射，容器内部端口随机映射到主机的端口。

-p, --publish=[ ]: 指定端口映射，格式为：主机(宿主)端口:容器端口。

-t, --tt=false,: 为容器重新分配一个伪输入终端，通常与-i 同时使用。

--name=" ": 为容器指定一个名称。

--dns-search=[ ]: 指定容器 DNS 搜索域名，默认和宿主一致。

-h, --hostname=[ ]: 指定容器的 hostname。

-e, --env=[ ]: 设置环境变量。

--env-file=[ ]: 从指定文件读入环境变量。

--cpuset-cpus: "0-2" or "0,1,2": 绑定容器到指定 CPU 运行。

-m, --memory=" ", :设置容器使用内存最大值。

--net="bridge": 指定容器的网络连接类型，支持 bridge/host/none/container 四种类型。

--link=[ ]: 添加链接到另一个容器。

--expose=[ ]: 开放一个端口或一组端口。

-v, --volume=[ ]: 绑定一个卷。

5.2 K8s 常见操作

K8s 支持容器的弹性伸缩，用户可以对应用进行扩容，提升 pod 的副本数来应对大量的流量，当负载小的时候可以对应用进行缩容，以避免资源浪费。也可以让应用根据资源情况自动的进行扩容和缩容。

5.2.1 手动扩容缩容

K8s 支持对应用进行手动的扩容和缩容，当应用访问量大时，可以提前增加应用的副本数来应对大流量。当应用访问量降下来后，可以手动减少副本数来达到缩容的目的。

通过制定副本数目达到扩容和缩容的目的，扩容、缩容命令如下。

kubectl scale deployment deployment 名称 --replicas=4

5.2.2 自动扩容缩容

HPA 与 Deployment、Service 一样，也属于一种 K8S 资源对象。HPA 的目标是希望通过追踪集群中所有 Pod 的负载变化情况，来自动化地调整 Pod 的副本数，以此来满足应用的需求和减少资源的浪费。

HAP 度量 Pod 负载变化情况的指标有两种：

- CPU 利用率（CPUUtilizationPercentage）。
- 自定义的度量指标，比如服务在每秒之内的请求数（TPS 或 QPS）。

CPU 利用率

如下示例通过 cpu 的变化来达到自动扩容和缩容，设置自动扩容和缩容有两种方式。第一种通过配置文件，第二种通过命令行。

1. 用户需要手动创建 yaml 文件，配置文件内容如下：

```
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: tongwebhpa
spec:
  scaleTargetRef:
    kind: Deployment
    name: tongweb
  minReplicas: 1
  maxReplicas: 10
  targetCPUUtilizationPercentage: 50
```

文件 kind 类型是 HorizontalPodAutoscaler，其中有 3 个地方需要额外注意下：

- scaleTargetRef 字段指定需要管理的 Deployment 的名字，也就是提前需要存在一个 Deployment 对象。
- minReplicas 和 maxReplicas 字段定义 Pod 可伸缩的数量范围。这个例子中扩容最高不能超过 10 个，缩容最低不能少于 1 个。

- targetCPUUtilizationPercentage 指定 CPU 使用率，也就是自动扩容和缩容的触发条件，当 CPU 使用率超过 50% 时会触发自动动态扩容

2. 通过运行如下命令可达到同样的效果

kubectl autoscale deployment Deployment 名称 --cpu-percent=50 --min=2 --max=10

自定义的度量指标

除了基于 CPU 来进行自动扩缩容之外，用户还可以根据自定义的监控指标来进行。这个就需要使用 Prometheus Adapter, Prometheus 用于监控应用的负载和集群本身的各种指标，Prometheus Adapter 可以帮我们使用 Prometheus 收集的指标并使用它们来制定扩展策略。

使用 Prometheus Adapter 实现自定义度量指标的前提是需要安装安装 Prometheus-Adapter。

安装好 Prometheus Adapter 后需要手动创建 **Yaml** 文件用于资源的监控实现扩容和缩容。如下示例最小副本数为 2，最大副本数为 5，当超过设置的吞吐量时就会进行扩容，Yaml 文件内容如下：

```
apiVersion: autoscaling/v2beta1
kind: HorizontalPodAutoscaler
metadata:
  name: tongwebhpa
spec:
  scaleTargetRef:
    apiVersion: extensions/v1beta1
    kind: Deployment
    name: tongweb
  minReplicas: 2
  maxReplicas: 5
  metrics:
  - type: Pods
    pods:
      metricName: http_requests_per_second
      targetAverageValue: 800m #800m 即 0.8 个/秒
```

附录 A license server 配置说明

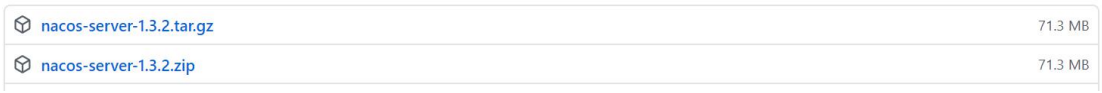
配置参数	说明	默认值
server.tongweb.license.type	license 文件类型，默认值为 file 表示读本地 license;值为 server 表示读 license server	file

server.tongweb.license.path	可选参数，默认去 TongWeb V7.0.6 目录下寻找 license.bat 文件，也可指定 license.bat 的具体路径	TongWeb V7.0.6 目录下
server.tongweb.license.master	License 节点地址，ip:端口的形式，节点默认端口是 8888	无

附录 B Nacos 安装说明

单机版 Nacos 安装：

1. 进入 Nacos 官网 https://github.com/alibaba/nacos/releases 下载 Nacos 压缩包



2. 解压：**tar -zxvf nacos-server-1.3.2.tar.gz**
3. 初始化 mysql 数据库：
   进入 Nacos：**cd nacos**
   执行 nacos/conf 目录下 **nacos-mysql.sql** 文件，进行数据库初始化。
4. 进入 nacos/conf 目录下，编辑 application.properties 文件：
   cd nacos/conf
   vim application.properties

```
spring.datasource.platform=mysql
db.num=1
db.url.0=jdbc:mysql://localhost:3306/nacos_config?characterEncoding=utf8&connectTimeout=1000&socketTimeout=3000&autoReconnect=true
db.user=xxxx
db.password=xxxx
```

5. 启动 Nacos：
   进入 nacos/bin 目录下执行**./startup.sh -m standalone** 进行启动（单机模式启动）。集群模式启动需要在 **cluster.conf** 中配置集群节点。

附录 C Prometheus 安装说明

Prometheus 安装

1. 从以下地址下载 Prometheus：
   https://prometheus.io/download/
2. 解压：
   tar xvfz prometheus-*.tar.gz
3. 查看 prometheus.yml，修改和添加以下配置：
   cd prometheus-*

```
scrape_configs:
  - job_name: 'node-admin-jobname'     #配置 k8s 的 master 和所有 node 的 name 和
```

```
ip:port，格式如下：

scrape_configs:
  - job_name: 'node-job-name'
   metrics_path: '/monitor/prometheus'
   static_configs:
   - targets: ['[tongweb.host]:[port]']
```

4. 启动 prometheus

   ./prometheus --config.file=prometheus.yml

附录 D CGI 使用

D.1 概述

CGI(Common Gateway Interface)为 web 服务器定义了一种与外部内容生成程序交互的方法，这些程序通常被称为 CGI 程序或 CGI 脚本。

当您使用 TongWeb V7.0.6 作为 HTTP 服务器并需要 CGI 支持时，可以添加 CGI 支持。

CGI 支持是使用 servlet 类 com.tongweb.catalina.servlets.CGIServlet 实现的。通常，这个 servlet 被映射到"**/cgi-bin/***"。

在 TongWeb V7.0.6 中默认禁用 CGI 支持。

D.2 启用

注：CGI 脚本用于执行 TongWeb V7.0.6 JVM 外部的程序。如果您正在使用 Java SecurityManager，这将绕过您在 tongweb.policy 中的安全策略配置。

启用 CGI 支持:

在 web 应用程序中启用 CGI 支持，请将示例 **CGI servlet** 复制到 web 应用程序的 **WEB-INF/web.xml** 文件中。部署后的应用在**%TW_HOME%/autodeploy/**目录下。

```
    <servlet>
    <servlet-name>cgi</servlet-name>
    <servlet-class>com.tongweb.catalina.servlets.CGIServlet</servlet-class>
    <init-param>
    <param-name>debug</param-name>
    <param-value>0</param-value>
    </init-param>
    <init-param>
    <param-name>cgiPathPrefix</param-name>
    <param-value>WEB-INF/cgi</param-value>
    </init-param>
    <load-on-startup>5</load-on-startup>
```

```
</servlet>
<!-- The mapping for the CGI Gateway servlet -->
<servlet-mapping>
<servlet-name>cgi</servlet-name>
<url-pattern>/cgi-bin/*</url-pattern>
</servlet-mapping>
```

D.3 配置项

CGI servlet 的 **init 参数**及说明:

cgiMethods

用逗号分隔的 HTTP 方法列表。默认值是 GET，POST。使用*作为脚本来处理所有的请求。

cgiPathPrefix

CGI 搜索路径将从 web 应用程序根目录+文件分隔符+这个前缀。默认情况下没有值，这将导致 web 应用程序根目录被用作搜索路径。推荐的值是 WEB-INF/cgi。

cmdLineArgumentsDecoded

如果命令行参数 emnts 被启用(通过 enableCmdLineArguments)，并且 TongWeb V7.0.6 在 Windows 上运行，那么每个解码的命令行参数都必须匹配此模式，否则请求将被拒绝。这是为了防止将命令行参数从 Java 传递到 Windows 时出现已知问题。这些问题可能导致远程代码执行。

cmdLineArgumentsEncoded

如果命令行参数 emnts 被启用(通过 enableCmdLineArguments)，则单个已编码的命令行参数必须匹配此模式，否则请求将被拒绝。默认值与 RFC3875 定义的允许值匹配，为 [a-zA-Z0-9\Q%;/?:@&,$-_.!~*'()\E]+。

enableCmdLineArguments

命令行参数是否根据 RFC3875 的 4.4 节从查询字符串生成，默认值为 false。

environment-variable

为 CGI 脚本的执行环境设置的环境。变量的名称取自参数名称。要配置名为 FOO 的环境变量，需要配置名为 environment-variable-FOO 的参数。参数值用作环境变量值。默认情况下没有环境变量。

Executable

用于运行脚本的可执行文件的名称。如果你的脚本本身是可执行的(例如，一个 exe 文件)，你可以显式地设置这个参数为空字符串。默认是 perl。

executable-arg-1, executable-arg-2 等等

可执行文件的附加参数。它们位于 CGI 脚本名称之前。默认情况下没有附加参数。

envHttpHeaders

一个正则表达式，用于选择作为环境变量传递给 CGI 进程的 HTTP 报头。注意，标题在匹配之前被转换为大写，并且整个标题名称必须与模式匹配。默认是：ACCEPT[-0-9A-Z]*|CACHE-CONTROL|COOKIE|HOST|IF-[-0-9A-Z]*|REFERER|USER-AGENT。

parameterEncoding

CGI servlet 中使用的参数编码的名称。默认是 System.getProperty("file.encoding","UTF-8")。这是系统默认编码，如果系统属性不可用，则为 UTF-8。

passShellEnvironment

是否应该将 TongWeb V7.0.6 进程的 shell 环境变量(如果有的话)传递给 CGI 脚本?默认是 false。

stderrTimeout

在结束 CGI 进程之前等待读取 stderr 完成的时间(以毫秒为单位)。默认是 2000。

D.4 配置 PHP 示例

1) 下载安装 PHP，此处略。
2) 复制 php.ini-development 到 php.ini，并编辑 php.ini
3) 去掉 extension_dir = "ext"的注释
4) 修改 cgi.force_redirect 的值为 0，并去掉注释
5) 配置应用中 WEB-INF/web.xml

```
<servlet>
    <servlet-name>php</servlet-name>
    <servlet-class>com.tongweb.catalina.servlets.CGIServlet</servlet-class>
    <init-param>
    <param-name>debug</param-name>
    <param-value>0</param-value>
    </init-param>
    <init-param>
    <param-name>cgiPathPrefix</param-name>
    <param-value>WEB-INF/cgi</param-value>
    </init-param>
<init-param>
<param-name>executable</param-name>
<param-value>F: \php\php-cgi.exe</param-value>
</init-param>
<load-on-startup>5</load-on-startup>
</servlet>
<servlet-mapping>
<servlet-name>php</servlet-name>
<url-pattern>/phpbin/*</url-pattern>
</servlet-mapping>
```

6) 将 index.php 等 php 文件放入应用的 WEB-INF/cgi 目录下（部署后的应用在%TW_HOME%/autodeploy/目录下）
7) 访问 http://<ip>:<port>/应用名/phpbin/index.php

附录 E WebDAV 使用

E.1 概述

WebDAV（Web-based Distributed Authoring and Versioning），它是一组 HTTP 协议的扩展，允许用户在远程 web 服务器上协作编辑和管理文件。当您使用 TongWeb V7.0.6 作为 HTTP 服务器并需要 WebDAV 支持时，可以添加 WebDAV 支持。

WebDAV 支持是使用 servlet 类 com.tongweb.catalina.servlets.WebdavServlet 实现的。

E.2 启用

注：WebdavServlet 不能作为默认 servlet 使用(即映射到'/')，因为它不能在这个配置中工作。

启用 WebDAV 需要修改应用程序的 **WEB-INF/web.xml** 文件。部署后的应用在**%TW_HOME%/autodeploy/**目录下:

```
<servlet>
<servlet-name>webdav</servlet-name>
<servlet-class>com.tongweb.catalina.servlets.WebdavServlet</servlet-class>
<init-param>
<param-name>debug</param-name>
<param-value>0</param-value>
</init-param>
<init-param>
<param-name>listings</param-name>
<param-value>true</param-value>
</init-param>
</servlet>
<servlet-mapping>
<servlet-name>webdav</servlet-name>
<url-pattern>/webdav/*</url-pattern>
</servlet-mapping>
```

以上配置将启用只读访问。若要启用读写访问，请添加:

```
<init-param>
<param-name>readonly</param-name>
<param-value>false</param-value>
</init-param>
```

要使内容可通过不同的 URL 编辑，使用以下映射:

```
<servlet-mapping>
<servlet-name>webdav</servlet-name>
<url-pattern>/webdavedit/*</url-pattern>
```

```
</servlet-mapping>
```

默认情况下，通过 WebDAV 无法访问/WEB-INF 和 META-INF。要访问这些 url，请使用:

```
<init-param>
<param-name>allowSpecialPaths</param-name>
<param-value>true</param-value>
</init-param>
```

附录 F 指令集和基础镜像关系

指令集	基础镜像
x86	223.71.97.99:51143/library/tongweb-base-x86
arm	223.71.97.99:51143/library/tongweb-base-arm64
ls	223.71.97.99:51143/library/tongweb-base-Loongson