架构总结

众安-董青松

目 录

1. CAS单点登录 3

1.1. WEB项目单点登录 3

1.2. Springboot项目配置 3

2. Docker学习 4

2.1. Docker查看镜像 4

2.2. Docker删除镜像 4

2.3. Docker构建容器 4

2.4. Docker运行容器 4

2.5. Docker查看运行容器 5

2.6. Docker停止容器 5

2.7. Docker删除容器 5

2.8. Docker停用并删除容器 6

3. Nginx学习总结 6

3.1. Nginx常用命令 6

3.1.1. Nginx启动 6

3.1.2. Nginx停止 6

3.2. Nginx配置详解 6

3.3. Nginx 6

4. 数据库总结 7

# CAS单点登录

## 基础概念

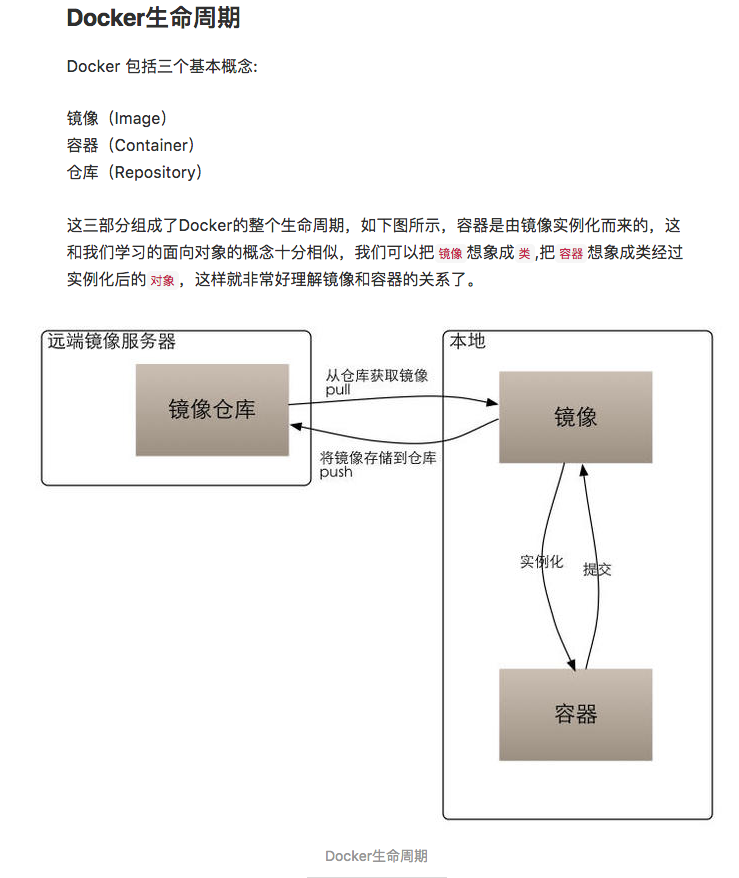
|  |
| --- |
| CAS-Ticket  TGT（Ticket Grangting Ticket）  TGT是CAS为用户签发的登录票据，拥有了TGT，用户就可以证明自己在CAS成功登录过  TGT封装了Cookie值以及此Cookie值对应的用户信息。  用户在CAS认证成功后，CAS生成cookie（叫TGC），写入浏览器，同时生成一个TGT对象，放入自己的缓存，TGT对象的ID就是cookie的值。  当HTTP再次请求到来时，如果传过来的有CAS生成的cookie，则CAS以此cookie值为key查询缓存中有无TGT ，如果有的话，则说明用户之前登录过，如果没有，则用户需要重新登录。  TGC （Ticket-granting cookie)  存放用户身份认证凭证的cookie  在浏览器和CAS Server间通讯时使用，并且只能基于安全通道传输（Https），是CAS Server用来明确用户身份的凭证。  ST（Service Ticket）  ST是CAS为用户签发的访问某一service的票据  用户访问service时，service发现用户没有ST，则要求用户去CAS获取ST  用户向CAS发出获取ST的请求，如果用户的请求中包含cookie，则CAS会以此cookie值为key查询缓存中有无TGT，如果存在TGT，则用此TGT签发一个ST，返回给用户。用户凭借ST去访问service，service拿ST去CAS验证，验证通过后，允许用户访问资源  ST是TGT签发的。用户在CAS上认证成功后，CAS生成TGT，用TGT签发一个ST，ST的ticketGrantingTicket属性值是TGT对象，然后把ST的值redirect到客户应用 |

## WEB项目单点登录

## Springboot项目配置

# Docker学习

--help 可以查看帮助文档



|  |
| --- |
| Docker 镜像、容器、仓库的概念Docker镜像 Docker镜像（Image）类似于虚拟机的镜像，可以将他理解为一个面向Docker引擎的只读模板，包含了文件系统。  例如：一个镜像可以完全包含了Ubuntu操作系统环境，可以把它称作一个Ubuntu镜像。镜像也可以安装了Apache应用程序（或其他软件），可以把它称为一个Apache镜像。  镜像是创建Docker容器的基础，通过版本管理和增量的文件系统，Docker提供了一套十分简单的机制来创建和更新现有的镜像。用户可以从网上下载一个已经做好的应用镜像，并通过命令直接使用。总之，应用运行是需要环境的，而镜像就是来提供这种环境。 Docker容器 Docker容器（Container）类似于一个轻量级的沙箱子（因为Docker是基于Linux内核的虚拟技术，所以消耗资源十分少），Docker利用容器来运行和隔离应用。  容器是从镜像创建的应用运行实例，可以将其启动、开始、停止、删除，而这些容器都是相互隔离、互不可见的。  可以吧每个容器看作一个简易版的Linux系统环境（包括了root用户权限、进程空间、用户空间和网络空间），以及与运行在其中的应用程序打包而成的应用盒子。  镜像自身是只读的。容器从镜像启动的时候，Docker会在镜像的最上层创建一个可写层，镜像本身将保持不变。就像用ISO装系统之后，ISO并没有什么变化一样。 Docker仓库 Docker仓库（Repository）类似与代码仓库，是Docker集中存放镜像文件的场所。  有时候会看到有资料将Docker仓库和注册服务器（Registry）混为一谈，并不严格区分。实际上，注册服务器是存放仓库的地方，其上往往存放着多个仓库。每个仓库集中存放某一类镜像，往往包括多个镜像文件，通过不同的标签（tag）来进行区分。例如存放Ubuntu操作系统镜像的仓库，称为Ubuntu仓库，其中可能包括14.04,12.04等不同版本的镜像。  根据存储的镜像公开分享与否，Docker仓库分为公开仓库（Public）和私有仓库（Private）两种形式。  目前，最大的公开仓库是Docker Hub，存放了数量庞大的镜像供用户下载。国内的公开仓库包括Docker Pool等，可以提供稳定的国内访问。如果用户不希望公开分享自己的镜像文件，Docker也支持用户在本地网络内创建一个只能自己访问的私有仓库。  当用户创建了自己的镜像之后就可以使用push将它上传到指定的公有或则私有仓库。这样用户下次在另一台机器上使用该镜像时，只需将其从仓库pull下来就可以了。  下面一篇文章介绍了有关上述概念的基本命令 |

## Docker镜像存放地址

|  |
| --- |
| sudo docker info 查看docker  docker 文件存放的路径：（Mac电脑）  /Users/用户名/Library/Containers/com.docker.docker/Data |

## docDocker查看镜像

|  |
| --- |
| docker images  docker images --help |

## Docker删除镜像

|  |
| --- |
| docker rmi +仓库的名称  **注意点：**  **1. 删除前需要保证容器是停止的  stop**  **2. 需要注意删除镜像和容器的命令不一样。 docker rmi ID  ,其中 容器(rm)  和 镜像(rmi)**  **3. 顺序需要先删除容器在删除镜像** |

## Docker构建容器

|  |
| --- |
| docker build -t 镜像名称 .  .表示基于当前路径下的Dockerfile文件  -f 可以指定dockerfile的路径 |

## Docker创建容器并运行

|  |
| --- |
| docker run –d –p 8080:8080 镜像名称  -p 9000:9000 :将容器的9000端口映射到主机的9000端口;  docker run --name mynginx -d nginx:latest（基于那个镜像）  --name 定义容器的名称为mynginx  -d 以后台的模式启动   * --name myphp-fpm :将容器命名为myphp-fpm * -v ~/nginx/www:/www :将主机中项目的目录www挂载到容器的/www |

## Docker查看运行容器

|  |
| --- |
| docker ps ---展示现在运行的容器  docker ps –a  docker ps –aq 查看所有容器ID |

## Docker停止容器

|  |
| --- |
| docker stop 容器ID 默认10s后停止容器  docker stop -t 1 容器ID 1s后立即停止容器   * 停用全部容器   docker stop $(docker ps -q)  此命令为级联命令   1. 查询所有容器的容器ID 2. docker stop 停止所有容器 |

## Docker删除容器

|  |
| --- |
| docker rm $(docker ps -aq)  此命令是级联命令:会执行两个操作：   1. docker ps –aq 查询所有容器 2. docker rm 删除容器 |

## Docker进入容器

Docker exec –it 容器名 /bin/bash

## Docker停用并删除容器

|  |
| --- |
| docker stop $(docker ps -q) & docker rm $(docker ps -aq) |
|  |

# Nginx学习总结

## Nginx常用命令

|  |
| --- |
| 查询服务器端口使用情况：netstat  查看nginx版本：nginx –v  查询服务器上的nginx相关的文件  find / -name "nginx"  默认的配置文件nginx.conf文件 find / -name "nginx.conf"  验证是否启动：localhost:8080  查询nginx进程：ps -ef | grep nginx  查看配置文件：cat /usr/local/etc/nginx/nginx.conf  停止NGINX: kill -QUIT 进程ID  快速停止:kill –INT 进程ID  快速停止:kill -TERM 进程ID  平滑重启：kill –HUP 进程ID  优雅的关闭进程,即等请求结束后再关闭:kill –QUIT 进程ID  改变配置文件,平滑的重读配置文件:kill -HUP进程ID  重读日志归档备份 kill -USER1 进程ID  平滑升级：kill –USER2 进程ID  优雅的关闭：kill –WINCH 进程ID  ngnix 的进程号可以用：nginx.pid  kill –HUP `cat nginx的pid文件`    重启cd /usr/local/sbin/ ./nginx -s reload  启动 cd /usr/local/sbin/ -> ./ nginx  检查 启动  netstat -lnt|grep 80  pgrep nginx  lsof -i:80  ps -aux|grep nginx  检查配置  /usr/local/nginx/sbin/nginx -t  检查启动效果  cur 127.0.0.1  Mac电脑：  kill -INT `cat /usr/local/var/run/nginx.pid`  /usr/local/bin sudo ./nginx 中启动  80端口 |

## Nginx配置文件详解

|  |
| --- |
| // 全局区 worker\_processes 1; // 有1个工作的子进程,可以自行修改,但太大无益,因为要争夺CPU,一般设置为 CPU数\*核数  Event {  // 一般是配置nginx连接的特性  // 如1个word能同时允许多少连接  worker\_connections 1024; // 这是指 一个子进程最大允许连1024个连接  }  http { //这是配置http服务器的主要段  Server1 { // 这是虚拟主机段    Location { //定位,把特殊的路径或文件再次定位 ,如image目录单独处理  } /// 如.php单独处理  }  Server2 {  }  } |

## Nginx跳转

|  |
| --- |
| nginx配置 proxy\_pass 第一种：proxy\_pass后缀不加斜杠  location /abc/ {  proxy\_pass http://172.16.1.38:8080;  }  访问url = <http://server/abc/test.jsp>，则被nginx代理后，请求路径会便问[http://proxy\_pass/abc/test.jsp](http://proxy_pass/abc/test.jsp" \t "_blank)，将test/ 作为根路径，请求test/路径下的资源  第二种：proxy\_pass后缀加斜杠  location /abc/ {  proxy\_pass http://172.16.1.38:8081/;  } |

# 工具总结

## IDEA配置启动参数

|  |
| --- |
| -Dserver.port=8090  -DDEPLOY\_ENV=test -Dserver.port=8090 |

## IDED常用命令-windows

## IDED常用命令-windows

### Nginx启动

# CAS单点登录

## 登录认证流程图

https://blog.csdn.net/u010475041/article/details/77886765

|  |
| --- |
| A系统第一次登录    A系统第二次登录    B系统第一次登录 |

### Nginx启动

### Nginx启动

### Nginx启动

|  |
| --- |
| Idea快捷键  设置  调整控制台输出字体大小  子主题 1  File->Settings->Editor->Colors & Fonts->Console Font  一行函数不折叠  File->Setting->Editor->General->Code Folding去掉  断点调试  F8  单步调试  F7  进入代码  Shift+F8  调到下一个断点/或者跳槽函数  F9  恢复程序  Alt +F8  查看debug的值  Ctrl + shift +F9  debug运行Java类  Ctrl + shift +F10  正常运行java  Alt +F10  显示执行断点  Alt+shift+F7  Force Step Into 这个是强制进入代码  Atl+F9  Run To Cursor 运行到光标处  注释  Ctrl +/  Ctrl +Shift +/  注释快捷键  Ctrl+ Shitf +N  主题更换  File-->setting-->Appearance&Behavior-->Appearance  Intellij:白底黑字  Darcula:黑底白字  常用  Shift +F6  重命名  Ａlt+Enter  快速修复  ALt+insert  生产get set方法  Ctrl+G  搜索  Ctrl +Ｄ  复制一行代码  Ctrl +Ｘ  删除一行代码  Ctrl+Alt+B  定位方法实现  Crtl + Shift +ENTER  自动不全if for function  Ctrl + Ｎ  查找类文件  Ctrl +Shift +N  查找文件  Crtl + Shift +L  格式化代码  Alt +F7  查看引用的  Ctrl+Z  撤销  Ctrl+Shift+Z  重做  Ctrl +F12  显示当前文件的结构 |

|  |
| --- |
|  |

# Git学习总结

## Git常用命令

### Git还原指定版本并同步到仓库

|  |
| --- |
| 还原到指定的版本：git reset –hard +提交的ID  同步到仓库：git push --force |

### Git还原指定版本并同步到仓库