## Netcore环境搭建

url(<https://docs.microsoft.com/zh-cn/dotnet/core/install/linux-centos>）

### 安装 sdk （开发环境）

如果安装 .NET Core SDK，则无需安装相应的运行时

sudo dnf install dotnet-sdk-3.1

### 安装运行时（运行环境 不支持开发）

sudo dnf install aspnetcore-runtime-3.1

作为 ASP.NET Core 运行时的一种替代方法，你可以安装不包含 ASP.NET Core 支持的 .NET Core 运行时：将上述命令中的 aspnetcore-runtime-3.1 替换为 dotnet-runtime-3.1。

sudo dnf install dotnet-runtime-3.1

## 防火墙设置

### 使用firewall

#### 开启防火墙

systemctl start firewalld

#### 查看状态

systemctl status firewalld

#### 禁止开机启动

systemctl disable firewalld

#### 停止运行

systemctl stop firewalld

#### 查看开放的端口

firewall-cmd --zone=public --list-ports

firewall-cmd --list-all

#### 更新防火墙规则

firewall-cmd --reload

#### 更新防火墙规则，重启服务

firewall-cmd --completely-reload

#### 以下都是指在public的zone下的操作，不同的Zone只要改变Zone后面的值就可以添加：

firewall-cmd --zone=public --add-port=80/tcp --permanent    （--permanent永久生效，没有此参数重启后失效）

### 使用iptables

url（<https://blog.csdn.net/XlxfyzsFdblj/article/details/78724434>）

#### 设置iptables

CentOS7默认使用的是firewall作为防火墙，这里改为iptables防火墙安装nginx

#### 关闭防火墙设置

systemctl stop firewalld.service  #停止firewall

systemctl disable firewalld.service  #禁止firewall开机启动

#### 安装iptables

yum install iptables-services

#### 修改文件配置允许80端口 vi /etc/sysconfig/iptables #编辑防火墙配置文件 按i进入文本编辑 在原文档中找到 -A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT 在该行下面添加 -A INPUT -m state --state NEW -m tcp -p tcp --dport 80 -j ACCEPT :wq! #保存退出

#### 重启防火墙使配置

systemctl restart iptables.service #最后重启防火墙使配置生效

#### 查看开放了那些端口

iptables -L -n

#### 设置防火墙开机启动

systemctl enable iptables.service #设置防火墙开机启动

## SELinux 设置

安全增强型 Linux（Security-Enhanced Linux）简称 SELinux，它是一个 Linux 内核模块，也是 Linux 的一个安全子系统。

### 查看selinux状态

getenforce

如果输出 enforcing 那就是开启了 selinux

### 临时关闭selinux

setenforce 0    ##设置SELinux 成为permissive模式

setenforce 1    ##设置SELinux 成为enforcing模式

### 永久关闭selinux

修改/etc/selinux/config 文件

将SELINUX=enforcing改为SELINUX=disabled

重启机器即可

## Nginx部署

url(<https://blog.csdn.net/doubleface999/article/details/105154595/>) url(<https://blog.csdn.net/XlxfyzsFdblj/article/details/78724434>)

### 添加 nginx 到 yum 源中（没有再添加）

sudo rpm -Uvh http://nginx.org/packages/centos/7/noarch/RPMS/nginx-release-centos-7-0.el7.ngx.noarch.rpm

### 安装nginx

yum install -y nginx

### 启动 nginx

systemctl start nginx.service

### 设置 nginx 开机自启动

systemctl enable nginx.service

### nginx 配置信息

#### 网站文件存放默认位置（Welcome to nginx 页面） /usr/share/nginx/html

#### 网站默认站点配置 /etc/nginx/conf.d/default.conf

#### 自定义 nginx 站点配置文件存放目录 /etc/nginx/conf.d/

#### nginx 全局配置文件 /etc/nginx/nginx.conf

### 启动 nginx

systemctl start nginx.service

### 关闭 nginx

systemctl stop nginx.service

### 重启 nginx

systemctl restart nginx.service

## 运行托管net

url(<https://docs.microsoft.com/zh-cn/aspnet/core/host-and-deploy/linux-nginx?view=aspnetcore-3.1>)

### 后台运行net

#### 打包net程序

dotnet publish --configuration Release

#### 运行（在打包好的项目目录中）

dotnet <项目.dll>

### 托管为后台进程并管理

#### systemd （通过systemd 管理）

systemd 是一个 init 系统，可以提供用于启动、停止和管理进程的许多强大的功能。

##### 创建服务文件

##### sudo nano /etc/systemd/system/kestrel-helloapp.service

以下是应用的一个示例服务文件：

ini复制

[Unit]

Description=Example .NET Web API App running on Ubuntu

[Service]

WorkingDirectory=/var/www/helloapp

ExecStart=/usr/bin/dotnet /var/www/helloapp/helloapp.dll

Restart=always

# Restart service after 10 seconds if the dotnet service crashes:

RestartSec=10

KillSignal=SIGINT

SyslogIdentifier=dotnet-example

User=www-data

Environment=ASPNETCORE\_ENVIRONMENT=Production

Environment=DOTNET\_PRINT\_TELEMETRY\_MESSAGE=false

[Install]

WantedBy=multi-user.target

在前面的示例中，管理服务的用户由 User 选项指定。 用户 (www-data) 必须存在并且拥有正确应用文件的所有权。

使用 TimeoutStopSec 配置在收到初始中断信号后等待应用程序关闭的持续时间。 如果应用程序在此时间段内未关闭，则将发出 SIGKILL 以终止该应用程序。 提供作为无单位秒数的值（例如，150）、时间跨度值（例如，2min 30s）或 infinity 以禁用超时。 TimeoutStopSec 默认为管理器配置文件（systemd-system.conf、system.conf.d、systemd-user.conf、user.conf.d）中 DefaultTimeoutStopSec 的值。 大多数分发版的默认超时时间为 90 秒。

# The default value is 90 seconds for most distributions.

TimeoutStopSec=90

Linux 具有区分大小写的文件系统。 将 ASPNETCORE\_ENVIRONMENT 设置为“生产”会导致搜索配置文件 appsettings.Production.json，而不是 appsettings.production.json。

必须转义某些值（例如，SQL 连接字符串）以供配置提供程序读取环境变量。 使用以下命令生成适当的转义值以供在配置文件中使用：

控制台复制

systemd-escape "<value-to-escape>"

环境变量名不支持冒号 (:) 分隔符。 使用双下划线 (\_\_) 代替冒号。 环境变量读入配置时，[环境变量配置提供程序](https://docs.microsoft.com/zh-cn/aspnet/core/fundamentals/configuration/?view=aspnetcore-3.1" \l "environment-variables)将双下划线转换为冒号。 以下示例中，连接字符串密钥 ConnectionStrings:DefaultConnection 以 ConnectionStrings\_\_DefaultConnection 形式设置到服务定义文件中：

复制

Environment=ConnectionStrings\_\_DefaultConnection={Connection String}

##### 保存该文件并启用该服务。

sudo systemctl enable kestrel-helloapp.service

##### 启用该服务，并确认它正在运行。

复制

sudo systemctl start kestrel-helloapp.service

sudo systemctl status kestrel-helloapp.service

在配置了反向代理并通过 systemd 管理 Kestrel 后，Web 应用现已完全配置，并能在本地计算机上的浏览器中从 http://localhost 进行访问。 也可以从远程计算机进行访问，同时限制可能进行阻止的任何防火墙。 检查响应标头，Server 标头显示由 Kestrel 所提供的 ASP.NET Core 应用。

### Supervisor（通过Supervisor管理）

(<https://www.cnblogs.com/chuankang/p/9473768.html>)

supervisord

负责管理进程的server端，配置文件是/etc/supervisor/supervisord.conf

supervisorctl

client端的命令行工具，管理子进程，配置文件在/etc/supervisor/supervisord.d/目录下

#### 安装supervisor

（<https://www.cnblogs.com/yjlch1016/p/10162918.html>）

（如果用的是阿里云的CentOS7会提示找不到supervisor，

则

yum install epel-release

先安装EPEL源）

yum install -y supervisor

##### 开机自启

systemctl enable supervisord

##### 启动supervisord

systemctl start supervisord

##### 查看状态

systemctl status supervisord

### 配置supervisord.conf

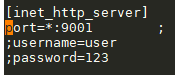
修改/etc/supervisord.conf

vim /etc/supervisord.conf

#### 1.配置启动文件目录

files = supervisord.d/\*.ini 改为 files = supervisord.d/\*.conf

#### 2.配置supervisord的展示页面



### 创建net启动文件

在/etc/supervisord.d文件夹下创建netcore启动文件testcore.conf

vim /etc/supervisord.d/testcore.conf

#### 编写信息

[program:testcore]

command=dotnet DHJ.Mes53.Web.Host.dll

directory=/root/core/apinetcore

environment=ASPNETCORE\_ENVIRONMENT=Production

user=root

stopsignal=INT

autostart=true

autorestart=true

startsecs=1

stderr\_logfile=/var/log/testcore.err.log

stdout\_logfile=/var/log/testcore.out.log

#### 介绍

command= 要执行的命令  
directory= 命令执行的目录  
environment= 环境变量  
user= 进程执行的用户身份  
stopsignal=INT  
autostart= 是否自动启动  
autorestart= 是否自动重启  
startsecs= 自动重启间隔  
stderr\_logfile= 标准错误日志  
stdout\_logfile= 标准输出日志

### 根据修改的配置文件重启

supervisorctl update （更新配置）

supervisord -c /etc/supervisord.conf（根据配置文件启动）

supervisorctl reload （重新加载）

systemctl restart supervisord.service（如果已经启动重启就可以了）

### 查看net是否启动

ps -ef | grep dotnet

## mysql安装

url（<https://www.jianshu.com/p/276d59cbc529>）

### 检查是否安装过mysql

rpm -qa | grep mysql

### 已经安装了**mysql-libs-5.1.73-5.el6\_6.x86\_64**，执行删除命令

rpm -e --nodeps mysql-libs-5.1.73-5.el6\_6.x86\_64

### 查询所有Mysql对应的文件夹

whereis mysql

### 删除相关目录或文件

rm -rf /usr/bin/mysql /usr/include/mysql /data/mysql /data/mysql/mysql

### 检查mysql用户组和用户是否存在，如果没有，则创建

cat /etc/group | grep mysql

cat /etc/passwd |grep mysql

groupadd mysql

useradd -r -g mysql mysql

### 从官网下载是用于Linux的Mysql安装包

wget https://dev.mysql.com/get/Downloads/MySQL-5.7/mysql-5.7.24-linux-glibc2.12-x86\_64.tar.gz

### 执行解压命令：

tar xzvf mysql-5.7.24-linux-glibc2.12-x86\_64.tar.gz

### 配置mysql文件路径

mv mysql-5.7.24-linux-glibc2.12-x86\_64 /usr/local/

cd /usr/local/

mv mysql-5.7.24-linux-glibc2.12-x86\_64 mysql

### 在**/usr/local/mysql**目录下创建data目录

mkdir /usr/local/mysql/data

### 更改mysql目录下所有的目录及文件夹所属的用户组和用户，以及权限

chown -R mysql:mysql /usr/local/mysql

chmod -R 755 /usr/local/mysql

### 编译安装并初始化mysql,**务必记住初始化输出日志末尾的密码（数据库管理员临时密码）**

cd /usr/local/mysql/bin

./mysqld --initialize --user=mysql --datadir=/usr/local/mysql/data --basedir=/usr/local/mysql

### 编辑配置文件my.cnf，添加配置如下

[root@localhost bin]# vi /etc/my.cnf

[mysqld]

datadir=/usr/local/mysql/data

port=3306

sql\_mode=NO\_ENGINE\_SUBSTITUTION,STRICT\_TRANS\_TABLES

symbolic-links=0

max\_connections=600

innodb\_file\_per\_table=1

lower\_case\_table\_names=1

### 测试启动mysql服务器

/usr/local/mysql/support-files/mysql.server start

### 添加软连接，并重启mysql服务

ln -s /usr/local/mysql/support-files/mysql.server /etc/init.d/mysql

ln -s /usr/local/mysql/bin/mysql /usr/bin/mysql

service mysql restart

### 登录mysql，修改密码(密码为步骤5生成的临时密码)

mysql -u root -p

set password for root@localhost = password('yourpass');

### 开放远程连接

mysql>use mysql;

msyql>update user set user.Host='%' where user.User='root';

mysql>flush privileges;

### 设置开机自动启动

1、将服务文件拷贝到init.d下，并重命名为mysql

[root@localhost /]# cp /usr/local/mysql/support-files/mysql.server /etc/init.d/mysqld

2、赋予可执行权限

[root@localhost /]# chmod +x /etc/init.d/mysqld

3、添加服务

[root@localhost /]# chkconfig --add mysqld

4、显示服务列表

[root@localhost /]# chkconfig –list

设置开机自启

/usr/lib/systemd/systemd-sysv-install enable mysql

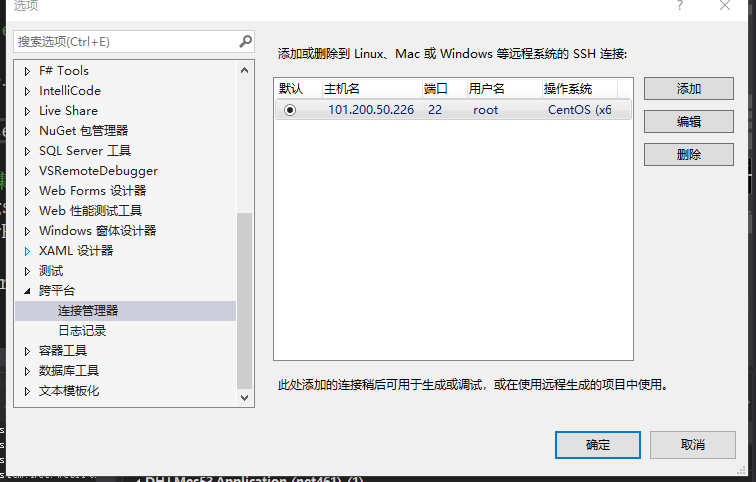
## Window向liunx传输文件夹

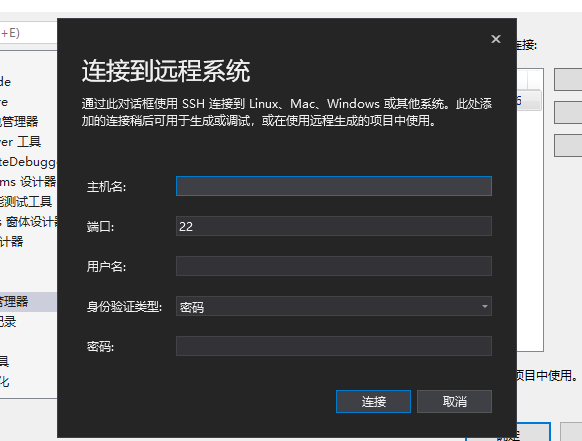
scp -r C:\Users\ZJH\Desktop\apinetcore\\* [root@192.168.43.88:/root/core/apinetcore](mailto:root@192.168.43.88:/root/core/apinetcore) （-r 是文件夹不带是文件）

## 通过vs附加到进程的方法调试liunx下的程序

### 配置vs上的附加进程ssh管理

选择配置-选项-跨平台-连接管理





配置连接

### 设置附加到进程查看liunx上的端口

查看liunx 上根据ip查看进程 netstat -nap | grep 22750

### 附加失败缺失vsdbg-linux-x64.tar.gz配置方法

#### 将压缩包放到一个文件服务器上

#### 修改/root/.vs-debugger/GetVsDbg.sh下的

url=[https://vsdebugger.azureedge.net/vsdbg-${target}/${vsdbgCompressedFile}](https://vsdebugger.azureedge.net/vsdbg-$%7btarget%7d/$%7bvsdbgCompressedFile%7d)

地址

为发布上去的地址

#### 重新运行脚本加载文件

/bin/sh /root/.vs-debugger/GetVsDbg.sh -v vs2019 -l /root/.vs-debugger/vs2019 -a /remote\_debugger

#### 重新附加到进程

## Xtrabackup使用

物理备份方式

分为增量备份（和前者要保持同步递增的方式备份）和全量备份（将全部数据库备份）