

## 43 | 如何进行Docker实验环境搭建?

2020-09-10 Chrono

《透视HTTP协议》

课程介绍 >



讲述: Chrono

时长 10:33 大小 9.67M



你好, 我是 Chrono。

《透视 HTTP 协议》这个专栏正式完结已经一年多了, 感谢你的支持与鼓励。

这一年的时间下来, 我发现专栏“实验环境的搭建”确实是一个比较严重的问题: 虽然我已经尽量把 Windows、macOS、Linux 里的搭建步骤写清楚了, 但因为具体的系统环境千差万别, 总会有各式各样奇怪的问题出现, 比如端口冲突、目录权限等等。

所以, 为了彻底解决这个麻烦, 我特意制作了一个 Docker 镜像, 里面是完整可用的 HTTP 实验环境, 下面我就来详细说一下该怎么用。

### 安装 Docker 环境

因为我不知道你对 Docker 是否了解, 所以第一步我还是先来简单介绍一下它。

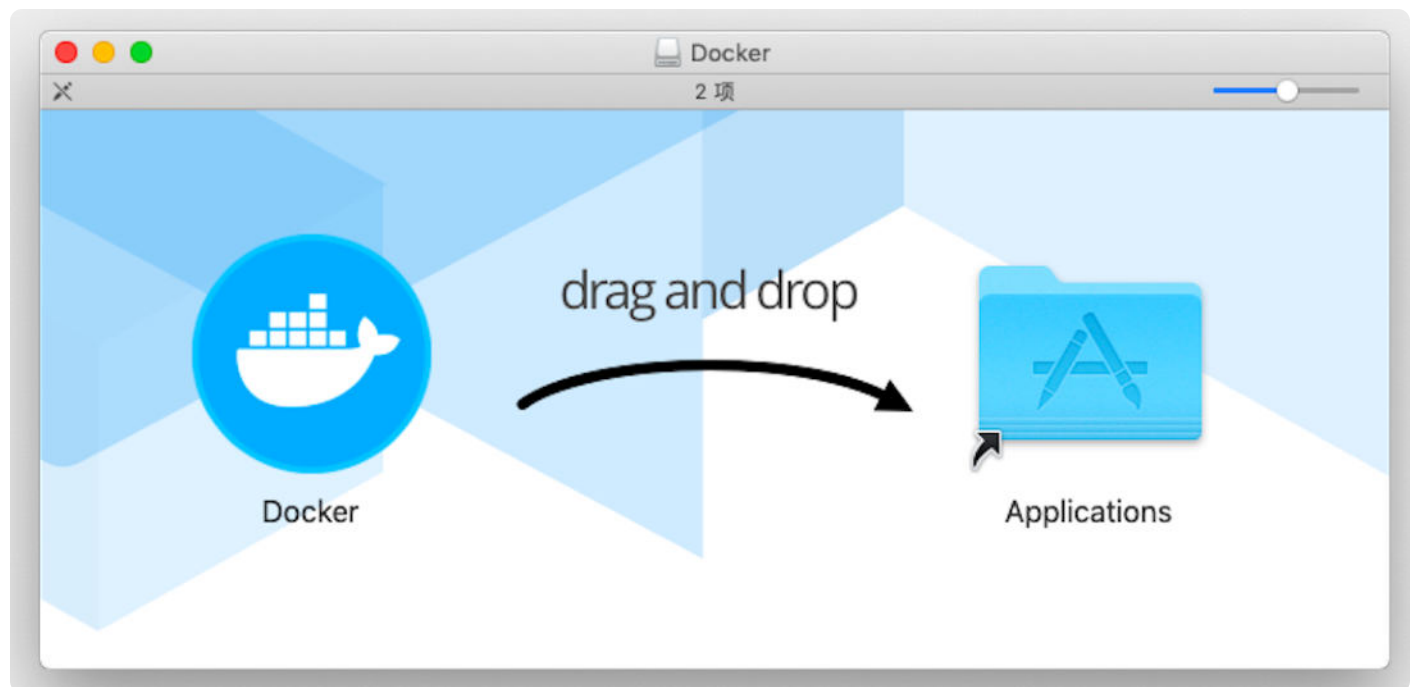
领资料



Docker 是一种虚拟化技术，基于 Linux 的容器机制（Linux Containers，简称 LXC），你可以把它近似地理解成是一个“轻量级的虚拟机”，只消耗较少的资源就能实现对进程的隔离保护。

使用 Docker 可以把应用程序和它相关的各种依赖（如底层库、组件等）“打包”在一起，这就是 Docker 镜像（Docker image）。Docker 镜像可以让应用程序不再顾虑环境的差异，在任意的系统中以容器的形式运行（当然必须要基于 Docker 环境），极大地增强了应用部署的灵活性和适应性。

Docker 是跨平台的，支持 Windows、macOS、Linux 等操作系统，在 Windows、macOS 上只要下载一个安装包，然后简单点几下鼠标就可以完成安装。



下面我以 Ubuntu 为例，说一下在 Linux 上的安装方法。

你可以在 Linux 上用 apt-get 或者 yum 安装 Docker，不过更好的方式是使用 Docker 官方提供的脚本，自动完成全套的安装步骤。

领资料

复制代码

```
1 curl -fsSL https://get.docker.com | bash -s docker --mirror Aliyun
```

因为 Docker 是国外网站，直接从官网安装速度可能比较慢。所以你还可以选择国内的镜像网站来加快速度，像这里我就使用“-mirror”选项指定了“某某云”。

Docker 是 C/S 架构，安装之后还需要再执行一条命令启动它的服务。

 复制代码

```
1 sudo service docker start    #Ubuntu启动docker服务
```

此外，操作 Docker 必须要有 sudo 权限，你可以用“usermod”命令把当前的用户加入“Docker”组里。如果图省事，也可以用 sudo 命令直接切换到 root 用户来操作。

 复制代码

```
1 sudo usermod -aG docker ${USER}  #当前用户加入Docker组
2 sudo su -                        #或者直接用root用户
```

这些做完后，你需要执行命令“docker version”“docker info”来检查是否安装成功。比如下面这张图，显示的就是我使用的 Docker 环境，版本是“18.06.3-ce”。

```
root@chrono-vb:~# docker version
Client:
 Version:           18.06.3-ce
 API version:       1.38
 Go version:        go1.10.3
 Git commit:        d7080c1
 Built:             Wed Feb 20 02:27:13 2019
 OS/Arch:           linux/amd64
 Experimental:      false

Server:
 Engine:
  Version:          18.06.3-ce
  API version:      1.38 (minimum version 1.12)
  Go version:       go1.10.3
  Git commit:       d7080c1
  Built:            Wed Feb 20 02:25:38 2019
  OS/Arch:          linux/amd64
  Experimental:     false
```

领资料

获取 Docker 镜像

如果你已经安装好了 Docker 运行环境，现在就可以从 Docker Hub 上获取课程相应的 Docker 镜像文件了，用的是“**docker pull**”命令。

 复制代码

```
1 docker pull chronolaw/http_study
```

这个镜像里打包了操作系统 Ubuntu 18.04 和最新的 Openresty 1.17.8.2，还有项目的全部示例代码。为了方便你学习，我还在里面加入了 Vim、Git、Telnet、Curl、Tcpdump 等实用工具。

由于镜像里的东西多，所以体积比较大，下载需要一些时间，你要有点耐心。镜像下载完成之后，你可以用“Docker images”来查看结果，列出目前本地的所有镜像文件。

```
root@chrono-vb:~# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
chronolaw/http_study latest             5f2f81314363       3 days ago         645MB
```

从图中你可以看到，这个镜像的名字是“chronolaw/http\_study”，大小是 645MB。

## 启动 Docker 容器

有了镜像文件，你就可以用“**docker run**”命令，从镜像启动一个容器了。

这里面就是我们完整的 HTTP 实验环境，不需要再操心这样、那样的问题了，做到了真正的“开箱即用”。

 复制代码

```
1 docker run -it --rm chronolaw/http_study
```

对于上面这条命令，我还要稍微解释一下：“-it”参数表示开启一个交互式的 Shell，默认使用的是 bash；“-rm”参数表示容器是“用完即扔”，不保存容器实例，一旦退出 Shell 就会自动删除容器（但不会删除镜像），免去了管理容器的麻烦。

“docker run”之后，你就会像虚拟机一样进入容器的运行环境，这里就是 Ubuntu 18.04，身份也自动变成了 root 用户，大概是下面这样的。

领资料





 复制代码

```
1 docker run -it --rm chronolaw/http_study
2
3 root@8932f62c972:/#
```

项目的源码我放在了 root 用户目录下，你可以直接进入“**http\_study/www**”目录，然后执行“**run.sh**”启动 OpenResty 服务（可参考[第 41 讲](#)）。

 复制代码

```
1 cd ~/http_study/www
2 ./run.sh start
```

不过因为 Docker 自身的限制，镜像里的 hosts 文件不能直接添加“[www.chrono.com](#)”等实验域名的解析。如果你想要在 URI 里使用域名，就必须在容器启动后手动修改 hosts 文件，用 Vim 或者 cat 都可以。

 复制代码

```
1 vim /etc/hosts #手动编辑hosts文件
2 cat ~/http_study/hosts >> /etc/hosts #cat追加到hosts末尾
```

另一种方式是在“docker run”的时候用“**--add-host**”参数，手动指定域名 /IP 的映射关系。

 复制代码

```
1 docker run -it --rm --add-host=www.chrono.com:127.0.0.1 chronolaw/http_study
```

保险起见，我建议你还是用第一种方式比较好。也就是启动容器后，用“cat”命令，把实验域名的解析追加到 hosts 文件里，然后再启动 OpenResty 服务。

 复制代码

```
1 docker run -it --rm chronolaw/http_study
2
3 cat ~/http_study/hosts >> /etc/hosts
4 cd ~/http_study/www
5 ./run.sh start
```

 领资料



## 在 Docker 容器里做实验

把上面的工作都做完之后，我们的实验环境就算是完美地运行起来了，现在你就可以在里面任意验证各节课里的示例了，我来举几个例子。

不过在开始之前，我要提醒你一点，因为这个 Docker 镜像是基于 Linux 的，没有图形界面，所以只能用命令行（比如 telnet、curl）来访问 HTTP 服务。当然你也可以查一下资料，让容器对外暴露 80 等端口（比如使用参数“`-net=host`”），在外部用浏览器来访问，这里我就不细说了。

先来看最简单的，[第 7 讲](#)里的测试实验环境，用 curl 来访问 localhost，会输出一个文本形式的 HTML 文件内容。

```
1 curl http://localhost      #访问本机的HTTP服务
```

 复制代码

```
root@89382f62c972:~/http_study/www# curl http://localhost
<!DOCTYPE html>
<html>
<head>
<title>Welcome to HTTP Study Page!</title> <style> body {
    width: 40em;
    margin: 0 auto;
    font-family: Tahoma, Verdana, Arial, sans-serif;
}
</style>
</head>
<body>

<h1>Welcome to HTTP Study Page!</h1>


<p>Now you are flying on OpenResty.</p>

</body>
</html>
```

领资料

然后我们来看[第 9 讲](#)，用 telnet 来访问 HTTP 服务，输入“`telnet 127.0.0.1 80`”，回车，就进入了 telnet 界面。

Linux 下的 telnet 操作要比 Windows 的容易一些，你可以直接把 HTTP 请求报文复制粘贴进去，再按两下回车就行了，结束 telnet 可以用“Ctrl+C”。

 复制代码

```
1 GET /09-1 HTTP/1.1
2 Host: www.chrono.com
```

```
root@89382f62c972:~/http_study/www# telnet 127.0.0.1 80
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
GET /09-1 HTTP/1.1
Host: www.chrono.com

HTTP/1.1 200 OK
Server: openresty/1.17.8.2
Date: Tue, 01 Sep 2020 06:29:58 GMT
Content-Type: text/plain
Connection: keep-alive
Content-Length: 90

host      =>www.chrono.com

raw header is :
GET /09-1 HTTP/1.1
Host: www.chrono.com
```

实验环境里测试 HTTPS 和 HTTP/2 也是毫无问题的，只要你按之前说的，正确修改了 hosts 域名解析，就可以用 curl 来访问，但要加上“-k”参数来忽略证书验证。

 复制代码

```
1 curl https://www.chrono.com/23-1 -vk
2 curl https://www.metroid.net:8443/30-1 -vk
```

 领资料



```
root@89382f62c972:~/http_study/www# curl https://www.chrono.com/23-1 -vk
* Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to www.chrono.com (127.0.0.1) port 443 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1
* successfully set certificate verify locations:
* CAfile: /etc/ssl/certs/ca-certificates.crt
* CApath: /etc/ssl/certs
* TLSv1.3 (OUT), TLS handshake, Client hello (1):
* TLSv1.3 (IN), TLS handshake, Server hello (2):
* TLSv1.3 (IN), TLS Unknown, Certificate Status (22):
* TLSv1.3 (IN), TLS handshake, Unknown (8):
* TLSv1.3 (IN), TLS Unknown, Certificate Status (22):
* TLSv1.3 (IN), TLS handshake, Certificate (11):
* TLSv1.3 (IN), TLS Unknown, Certificate Status (22):
* TLSv1.3 (IN), TLS handshake, CERT verify (15):
* TLSv1.3 (IN), TLS Unknown, Certificate Status (22):
* TLSv1.3 (IN), TLS handshake, Finished (20):
* TLSv1.3 (OUT), TLS change cipher, Client hello (1):
* TLSv1.3 (OUT), TLS Unknown, Certificate Status (22):
* TLSv1.3 (OUT), TLS handshake, Finished (20):
* SSL connection using TLSv1.3 / TLS_AES_256_GCM_SHA384
* ALPN, server accepted to use http/1.1
```

```
root@89382f62c972:~/http_study/www# curl https://www.metroid.net:8443/30-1 -k
scheme: https
tls: TLSv1.3
server suite: TLS_AES_256_GCM_SHA384
http version: 2
http/2 tag: h2
```

这里要注意一点，因为 Docker 镜像里的 Openresty 1.17.8.2 内置了 OpenSSL1.1.1g，默认使用的是 TLS1.3，所以如果你想要测试 TLS1.2 的话，需要使用参数“**-tlsv1.2**”。

 复制代码

```
1 curl https://www.chrono.com/30-1 -k --tlsv1.2
```

```
root@89382f62c972:~/http_study/www# curl https://www.chrono.com/30-1 -k --tlsv1.2
scheme: https
tls: TLSv1.2
server suite: ECDHE-RSA-AES256-GCM-SHA384
http version: 1.1
http/2 tag:
```

 领资料

## 在 Docker 容器里抓包

到这里，课程中的大部分示例都可以运行了。最后我再示范一下在 Docker 容器里 tcpdump 抓包的用法。



首先，你要指定抓取的协议、地址和端口号，再用“-w”指定存储位置，启动 tcpdump 后按“Ctrl+Z”让它在后台运行。比如为了测试 TLS1.3，就可以用下面的命令行，抓取 HTTPS 的 443 端口，存放到“/tmp”目录。

 复制代码

```
1 tcpdump tcp port 443 -i lo -w /tmp/a.pcap
```

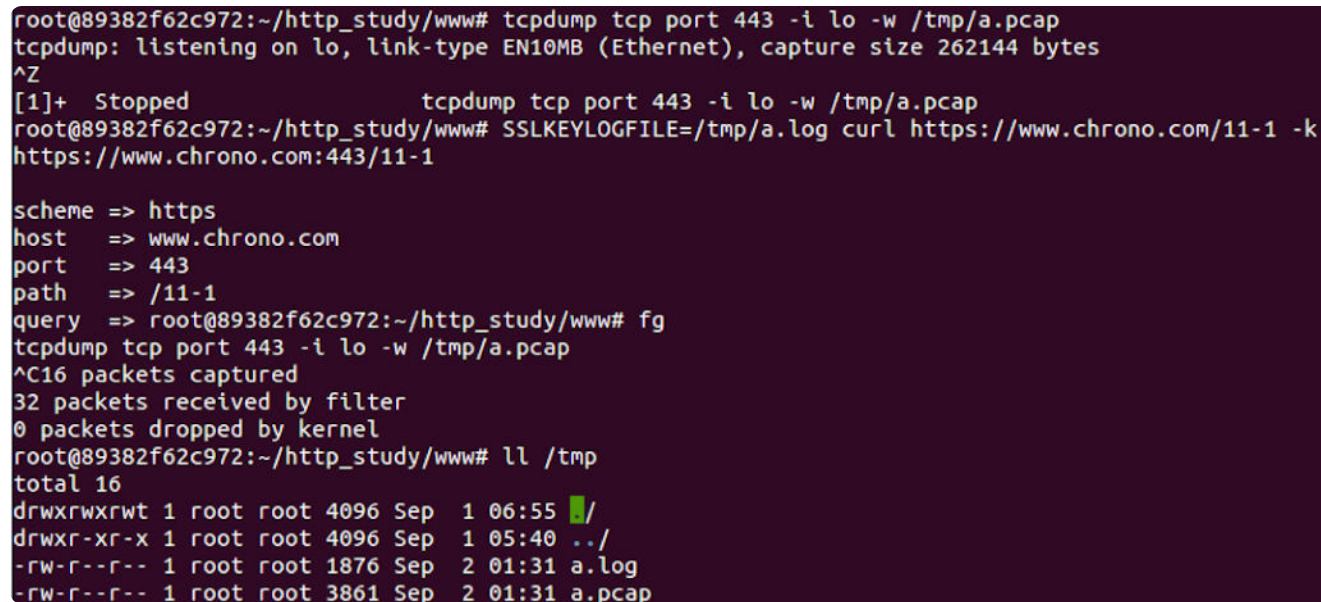
然后，我们执行任意的 telnet 或者 curl 命令，完成 HTTP 请求之后，输入“fg”恢复 tcpdump，再按“Ctrl+C”，这样抓包就结束了。

对于 HTTPS 需要导出密钥的情形，你必须在 curl 请求的同时指定环境变量“SSLKEYLOGFILE”，不然抓包获取的数据无法解密，你就只能看到乱码了。

 复制代码

```
1 SSLKEYLOGFILE=/tmp/a.log curl https://www.chrono.com/11-1 -k
```

我把完整的抓包过程截了个图，你可以参考一下。



```
root@89382f62c972:~/http_study/www# tcpdump tcp port 443 -i lo -w /tmp/a.pcap
tcpdump: listening on lo, link-type EN10MB (Ethernet), capture size 262144 bytes
^Z
[1]+  Stopped                  tcpdump tcp port 443 -i lo -w /tmp/a.pcap
root@89382f62c972:~/http_study/www# SSLKEYLOGFILE=/tmp/a.log curl https://www.chrono.com/11-1 -k
https://www.chrono.com:443/11-1

scheme => https
host   => www.chrono.com
port   => 443
path   => /11-1
query  => root@89382f62c972:~/http_study/www# fg
tcpdump tcp port 443 -i lo -w /tmp/a.pcap
^C16 packets captured
32 packets received by filter
0 packets dropped by kernel
root@89382f62c972:~/http_study/www# ll /tmp
total 16
drwxrwxrwt 1 root root 4096 Sep  1 06:55 ./
drwxr-xr-x 1 root root 4096 Sep  1 05:40 ../
-rw-r--r-- 1 root root 1876 Sep  2 01:31 a.log
-rw-r--r-- 1 root root 3861 Sep  2 01:31 a.pcap
```

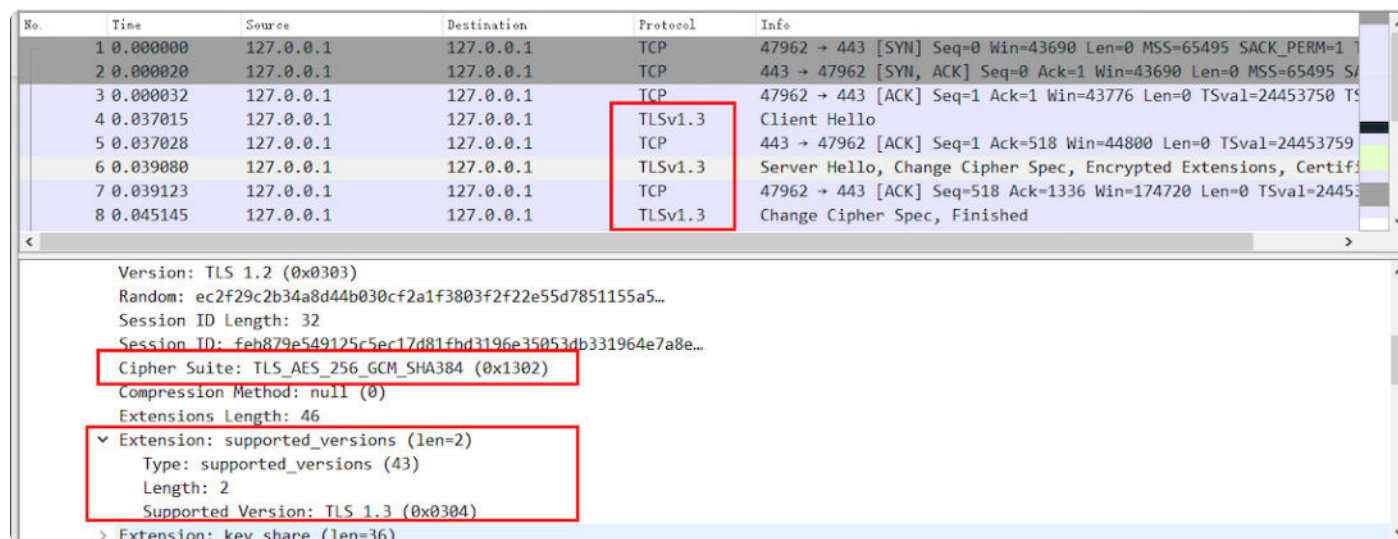
 领资料

抓包生成的文件在容器关闭后就会消失，所以还要用“docker cp”命令及时从容器里拷出来（指定容器的 ID，看提示符，或者用“docker ps -a”查看，也可以从 GitHub 仓库里获取 43-1.pcap/43-1.log）。

```
1 xxx: /tmp/a.pcap . #需要指定容器的ID
2 docker cp xxx: /tmp/a.log . #需要指定容器的ID
```

[复制代码](#)

现在有了 pcap 文件和 log 文件，我们就可以用 Wireshark 来看网络数据，细致地分析 HTTP/HTTPS 通信过程了（HTTPS 还需要设置一下 Wireshark，见 [第 26 讲](#)）。



在这个包里，你可以清楚地看到，通信时使用的是 TLS1.3 协议，服务器选择的密码套件是 TLS\_AES\_256\_GCM\_SHA384。

掌握了 tcpdump 的用法之后，你也可以再参考 [第 27 讲](#)，改改 Nginx 配置文件，自己抓包仔细研究 TLS1.3 协议的“supported\_versions”“key\_share”“server\_name”等各个扩展协议。

## 小结

今天讲了 Docker 实验环境的搭建，我再小结一下要点。

1. Docker 是一种非常流行的虚拟化技术，可以近似地把它理解成是一个“轻量级的虚拟机”；
2. 可以用“docker pull”命令从 Docker Hub 上获取课程相应的 Docker 镜像文件；
3. 可以用“docker run”命令从镜像启动一个容器，里面是完整的 HTTP 实验环境，支持 TLS1.3；
4. 可以在 Docker 容器里任意验证各节课里的示例，但需要使用命令行形式的 telnet 或者 curl；
5. 抓包需要使用 tcpdump，指定抓取的协议、地址和端口号；

[领资料](#)

6. 对于 HTTPS，需要指定环境变量“SSLKEYLOGFILE”导出密钥，再发送 curl 请求。

很高兴时隔一年后再次与你见面，今天就到这里吧，期待下次 HTTP/3 发布时的相会。



## == 课外小贴士 ==

- 01 在 Linux 里运行 OpenResty 经常会遇到文件访问权限的问题（403 Forbidden），Docker 镜像里简单的解决办法是使用指令“user root”，让 OpenResty 以 root 权限运行。
- 02 在 Windows 10 上虽然可以安装 Docker Desktop，但因为它使用了 Hyper-V，会与虚拟机软件 VirtualBox 产生冲突。
- 03 可以使用“docker exec”命令进入正在运行的容器，这就相当于开了一个新窗口，有的时候会很方便。

领资料



分享给需要的人，Ta订阅超级会员，你将得 50 元

Ta单独购买本课程，你将得 20 元

生成海报并分享

赞 14

提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 42 | DHE/ECDHE算法的原理

下一篇 44 | 先睹为快：HTTP/3实验版本长什么样子？

## 学习推荐

# JVM + NIO + Spring

各大厂面试题及知识点详解

限时免费



## 精选留言 (13)

写留言

领资料



Howard.Wundt

2020-09-10

首先祝老师教师节快乐！很期待着与老师的重逢。有个问题请教老师：轻量化虚拟机技术除了 Docker 外还有其他选择吗？Docker 现在的政治化让人很不舒服。

作者回复: 目前看来还没有同量级的对手，docker已经算是事实标准了，而且还有围绕它的很多生态，比如k8s，想要替代短时期内看不到可能性。





4

**Shanks-王冲**

2020-11-07

谢谢老师的Docker tutorial quick guide, 让我对Docker有了first touch; 前几天看Android开发的技术博客时, 不知怎么地就跳到Docker官网, 并瞧了瞧, 没敢下载来玩; 但今天又机缘巧合看到这篇文章, I just pulled, 感谢老师!

作者回复: docker挺好玩的, 我之前也是不太熟悉, 偶然一接触, 操作了几下就会用了, 然后就研究了下去。



2

**silence**

2021-09-13

老师请问我在测试tcpdump抓包的时候, 将您的命令粘贴进去

```
tcpdump tcp port 443 -i lo -w /tmp/a.pcap
```

报了以下错误怎么解决呢?

```
tcpdump: lo: SIOCETHTOOL(ETHTOOL_GET_TS_INFO) ioctl failed: Function not implemented
```

作者回复: 感觉像是网络的问题, 如果可能的话换个环境再做实验, 比如docker镜像。

共 2 条评论 >



1

**Geek\_78044b**

2020-09-20

老师确实很敬业, 最近刚买了课程, 一周多时间快速学习了一篇, 开始的破冰篇确实特别基础, 我还以为这门科是只针对的初学者的, 没什么难度, 后面讲http1.1, 2, 3, 安全篇等都还是比较有深度, 感谢老师的付出。

作者回复: 嗯, 这个也跟课程的总体设计有关, 一般都是要由浅入深循序渐进, 尽量全面覆盖知识点。如果基础好可以跳过前面的, 直接看后面感兴趣的部分。



1

领资料

**大土豆**

2020-09-10

HTTP/3会有个正式的发布会吗。。。我看现在快手和百度, HTTP3都已经在线上用了



作者回复: 现在只是草案, 虽然还没有正式发布, 但估计和最终版差距不会太大, 所以大家都提前做。

发布会肯定是不太会的, 不会那么夸张。



點點點, 点顛

2020-09-10

老师教师节快乐😊。感谢老师还记得我们

作者回复: 等以后http/3时再相聚。



dongge

2020-09-10

好敬业。

作者回复: thanks。



Harry

2020-09-10

期待再次与老师相会,  
谢谢老师!

作者回复: 感谢支持!



lesserror

2020-09-10

老师, 很棒👍!

作者回复: nice to meet you again.



领资料





**忧天小鸡**  
2021-12-22

又学了一点点docker，很棒的体验，认真负责

作者回复: docker很好玩，现在容器化是大趋势，早学有好处。



**dongge**  
2021-10-17

Chrono老师，是我在极客时间见过的最负责的老师。

作者回复: many thanks.



**zha.qiang**  
2020-10-19

Found a typo:

...

cu加l <https://www.chrono.com/30-1> -k --tlsv1.2

...

作者回复: 感谢指正，会联系编辑尽快修复。



**candy**  
2020-09-10

谢谢老师，节日快乐！还建立了镜像，学习更方便了

作者回复: 不客气，希望能帮到你。



领资料

