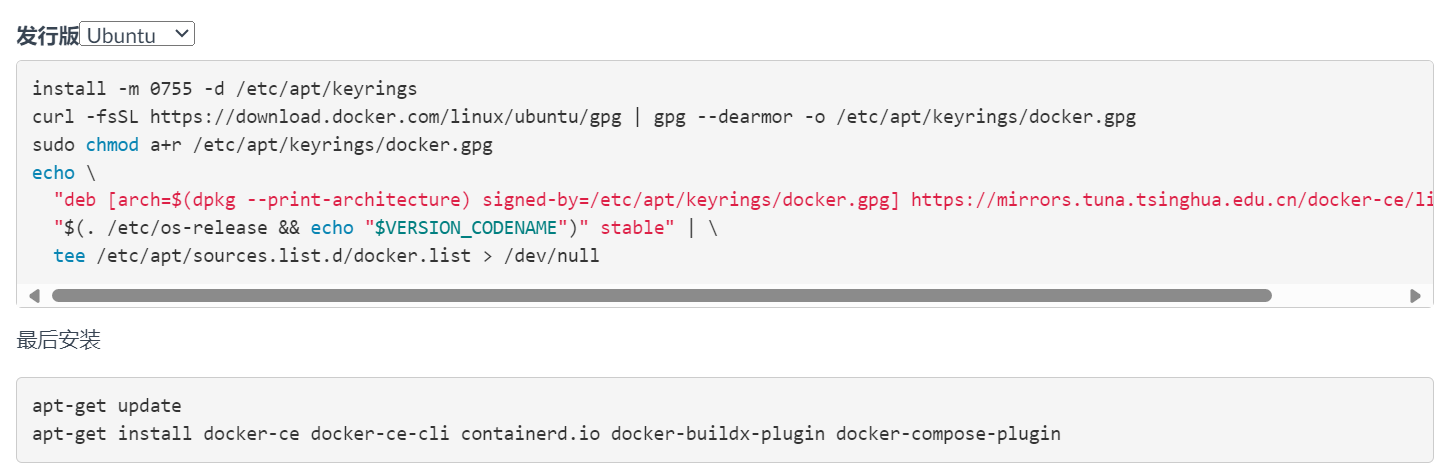
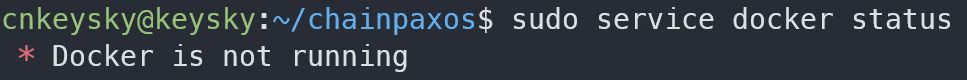
# 环境准备

## 1.1 安装 docker

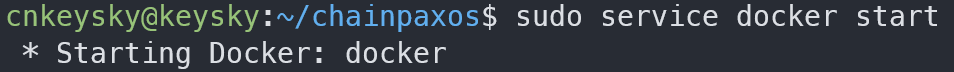
在Ubuntu22.04中安装docker，由于官方文档的不稳定性，现给出官方安装网址https://docs.docker.com/engine/install/ubuntu/，可以根据该网址去安装。由于网络不稳定，可以使用清华源，现在提供对应的网址https://mirrors.tuna.tsinghua.edu.cn/help/docker-ce/，有需要可以根据网址的提示安装，需要注意的是要选择Ubuntu。如果感觉麻烦，可以使用我们提供的install\_docker.sh脚本，该脚本默认使用的是官方源，如果需要清华源，需要自行修改。下图是清华源的安装，具体指令可以根据上面的网址自行复制并执行。



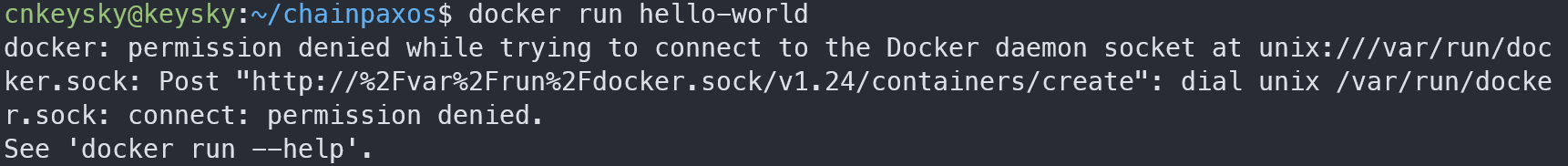
查看 docker 服务状态



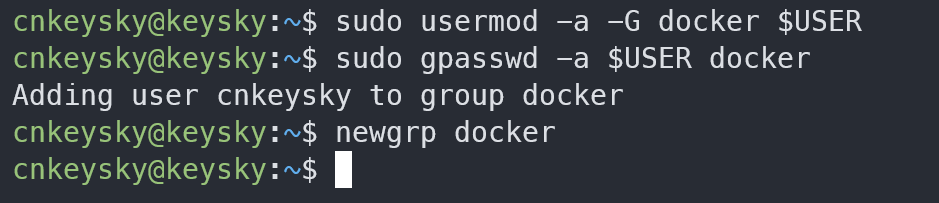
启动 docker 服务（可以使用stop停止docker服务）

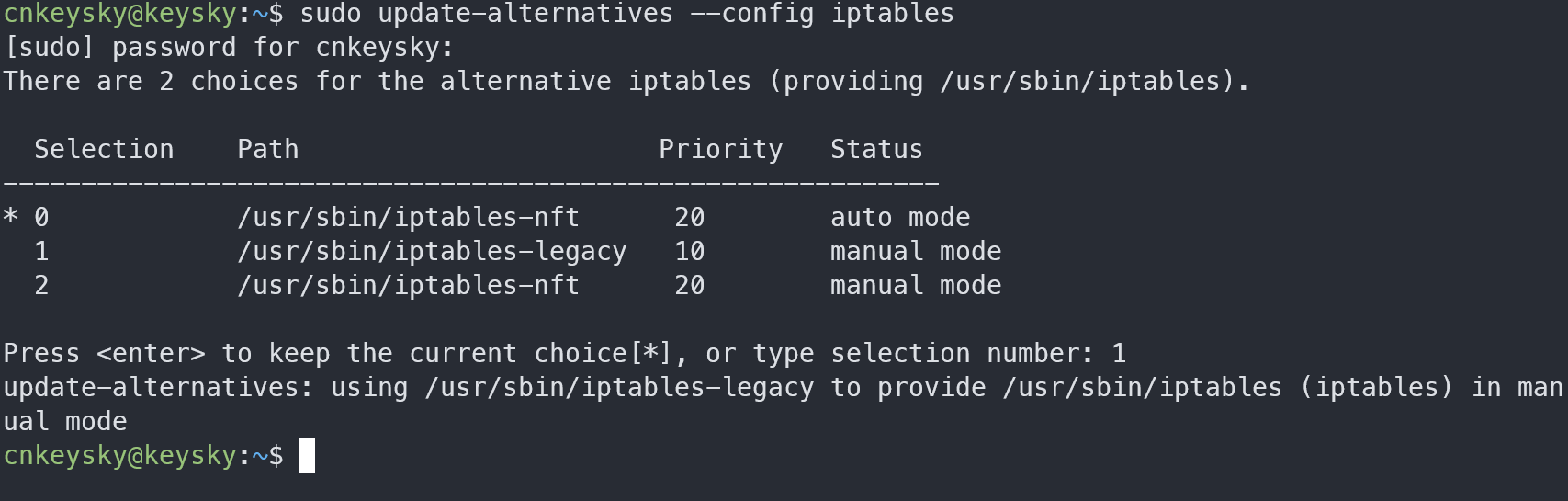


Docker服务启动之后，可以通过安装hello-world项目来确定docker服务没有问题。该项目会先从远程仓库拉取。

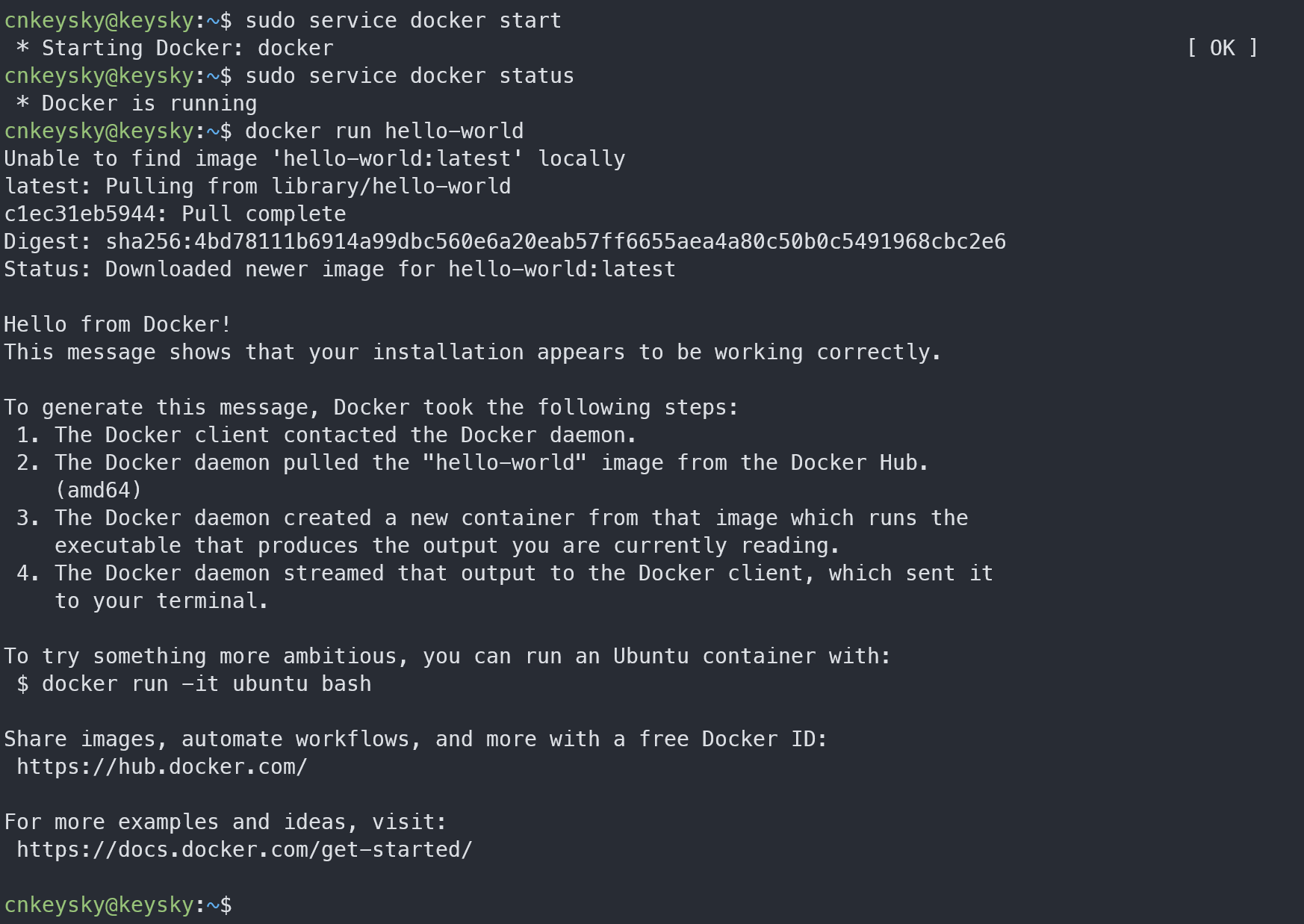


第一次运行会出现上图中的错误，主要是docker服务启动失败，使用如下方式解决：





Docker 运行日志是/var/log/docker.log，出现其他无法启动的问题，可以查看具体情况。解决上述问题之后重新启动docker服务，并运行hello-world项目，结果如下：



至此，docker服务安装并完成测试。

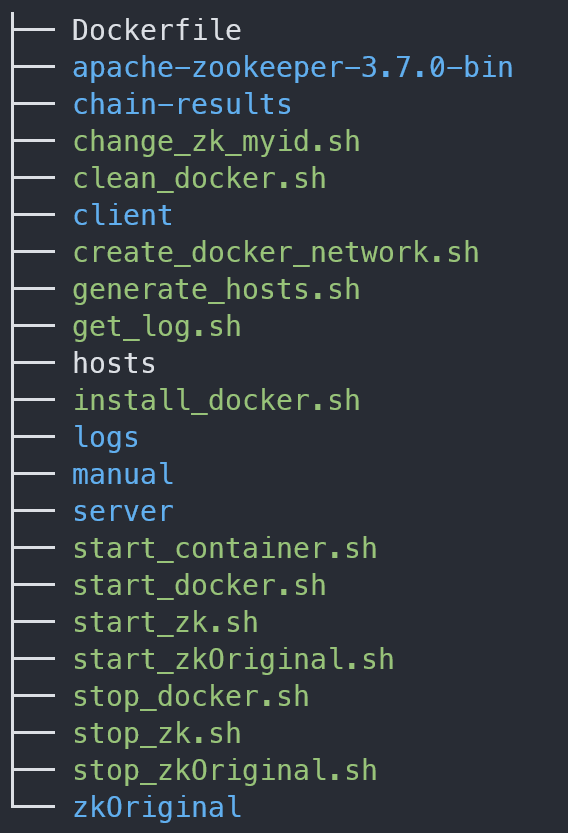
## 1.2 安装sshpass

在宿主服务器中安装sshpass，该工具包提供无需确认密码的功能，方便脚本化。



# 项目结构

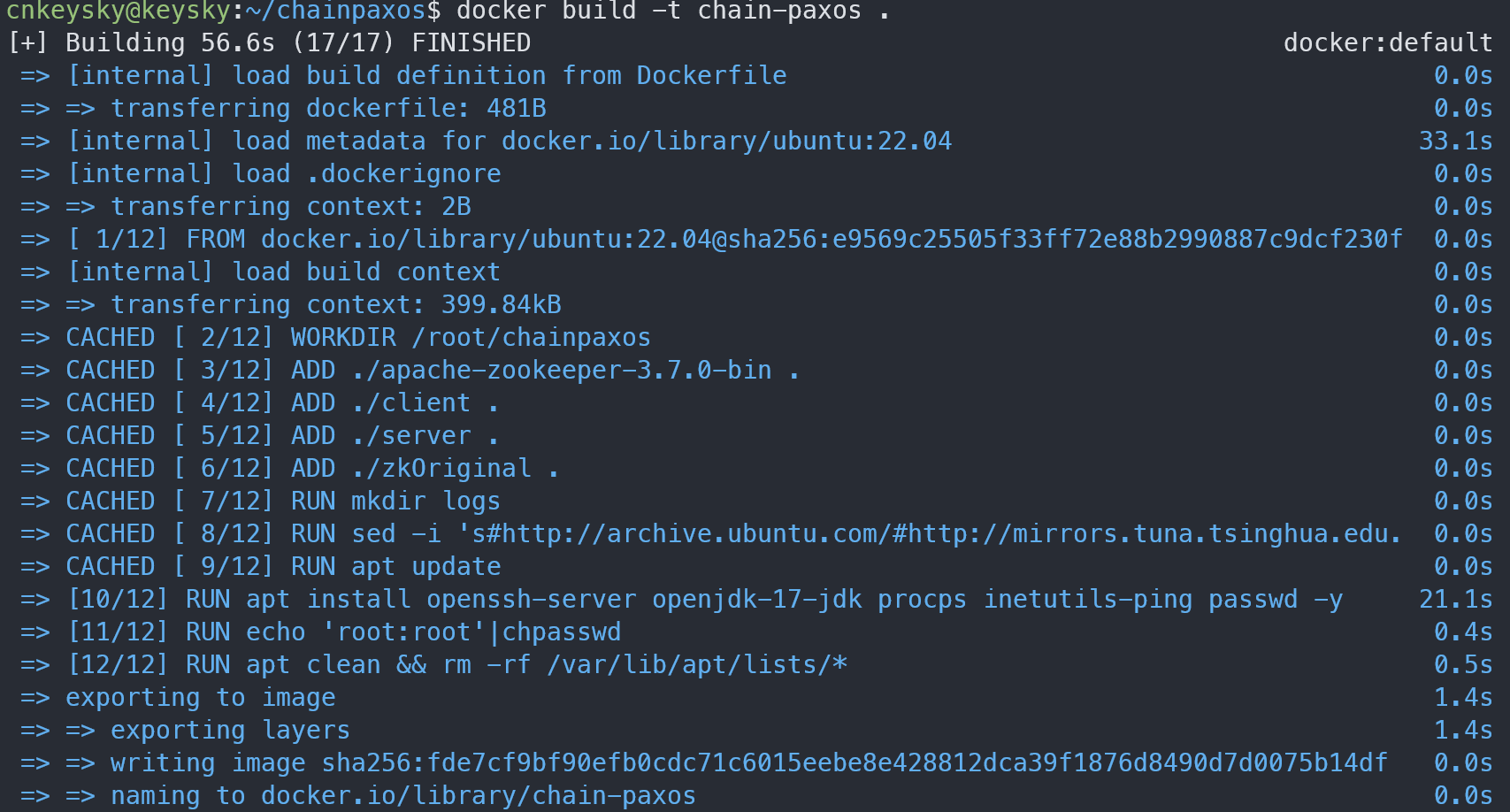
下面的图片主要展示项目的结构，其中蓝色表示目录，绿色是文件。这些文件和目录存放在$HOME/chainpaxos目录中。



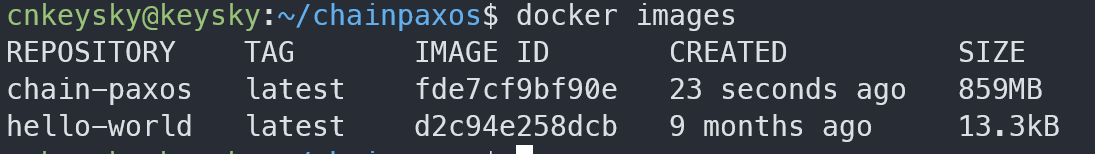
* Dockfile文件用于构建容器。
* apache-zookeeper-3.7.0-bin是使用了chainpaxos算法的项目。
* chain-results用于可视化结果。
* change\_zk\_myid.sh用于修改zookeeper集群中的myid值。
* clean\_docker.sh用于删除docker容器，容器一旦删除，如果还想要再次运行，那么此时需要运行一次start\_docker.sh脚本。
* client是chainpaxos客户端。
* create\_docker\_network.sh用于创建容器网络段，该脚本通常情况下不需要自行调用，除非你需要自行定义容器IP地址，推荐使用默认设置。
* generate\_hosts.sh用于生成容器IP地址。
* get\_log.sh把容器中的log文件拷贝到当前目录中，这些log文件用于结果的可视化生成。
* hosts容器IP地址，使用genenrate\_hosts.sh生成。
* install\_docker.sh用于构建docker环境。
* logs目录保存了容器中的log文件，这些日志文件是按照算法名称存储。
* manual目录中的脚本用于测试环境使用。
* server是chainpaxos的服务端。
* start\_container.sh脚本用于启动docker容器。
* start\_docker.sh脚本同样用于启动docker容器，与start\_container.sh脚本不同的是该脚本只能运行一次，多次运行会造成容器名称冲突，因此通常在第一次生成容器时运行，之后每次运行只需要运行start\_container.sh脚本即可。
* start\_zk.sh脚本用于启动apache-zookeeper-3.7.0-bin项目。
* start\_zkOriginal.sh脚本用于启动原始zookeeper项目。
* stop\_docker.sh脚本用于停止docker容器。
* stop\_zk.sh脚本用于停止apache-zookeeper-3.7.0-bin项目。
* stop\_zkOriginal.sh脚本用于停止原始zookeeper项目。
* zkOriginal是原始zookeeper项目。

# 容器和脚本说明

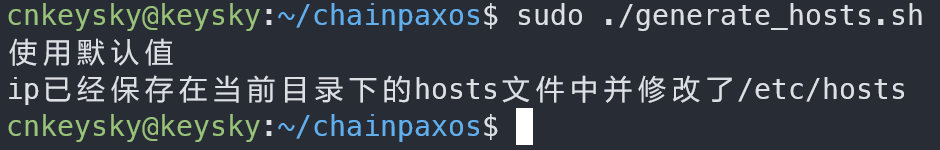
构建docker镜像，镜像名称设置为chain-paxos。



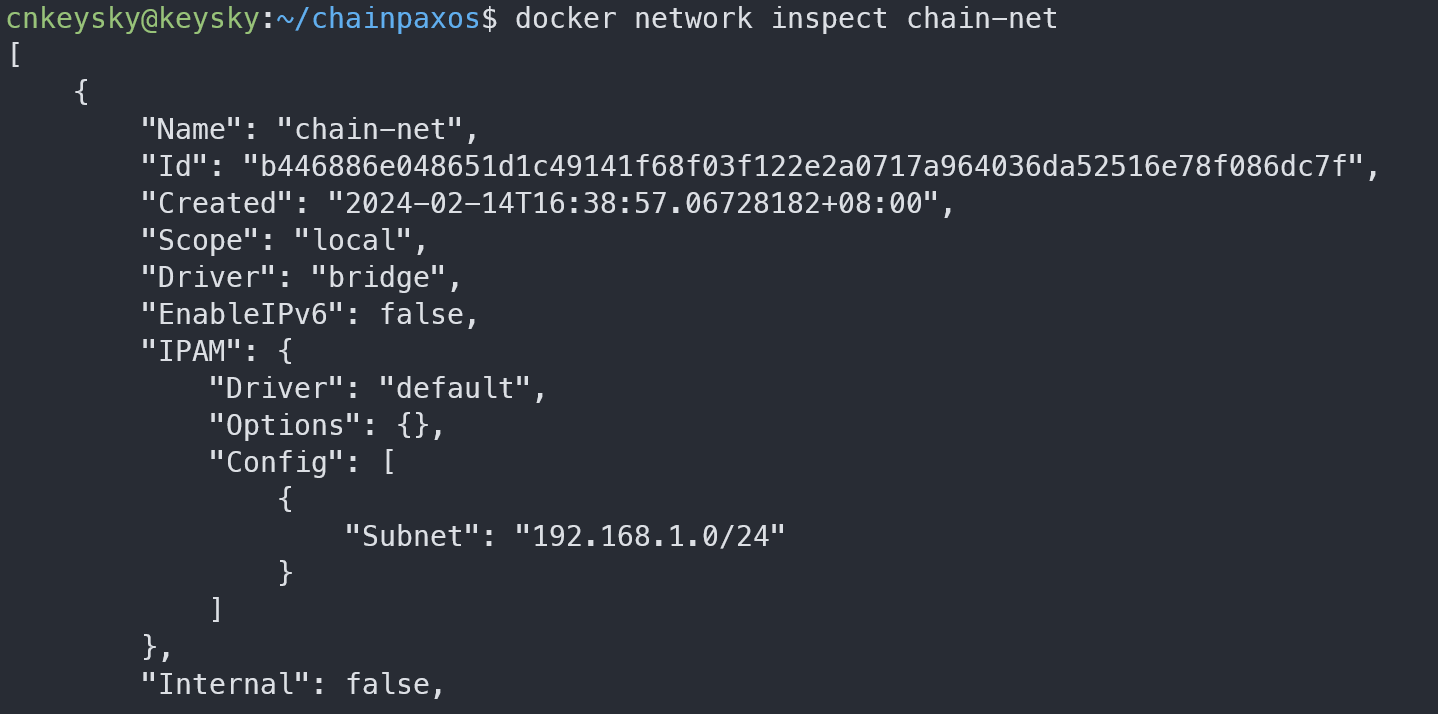
镜像构建完成之后，使用docker相关指令进一步查看，确认没有问题。



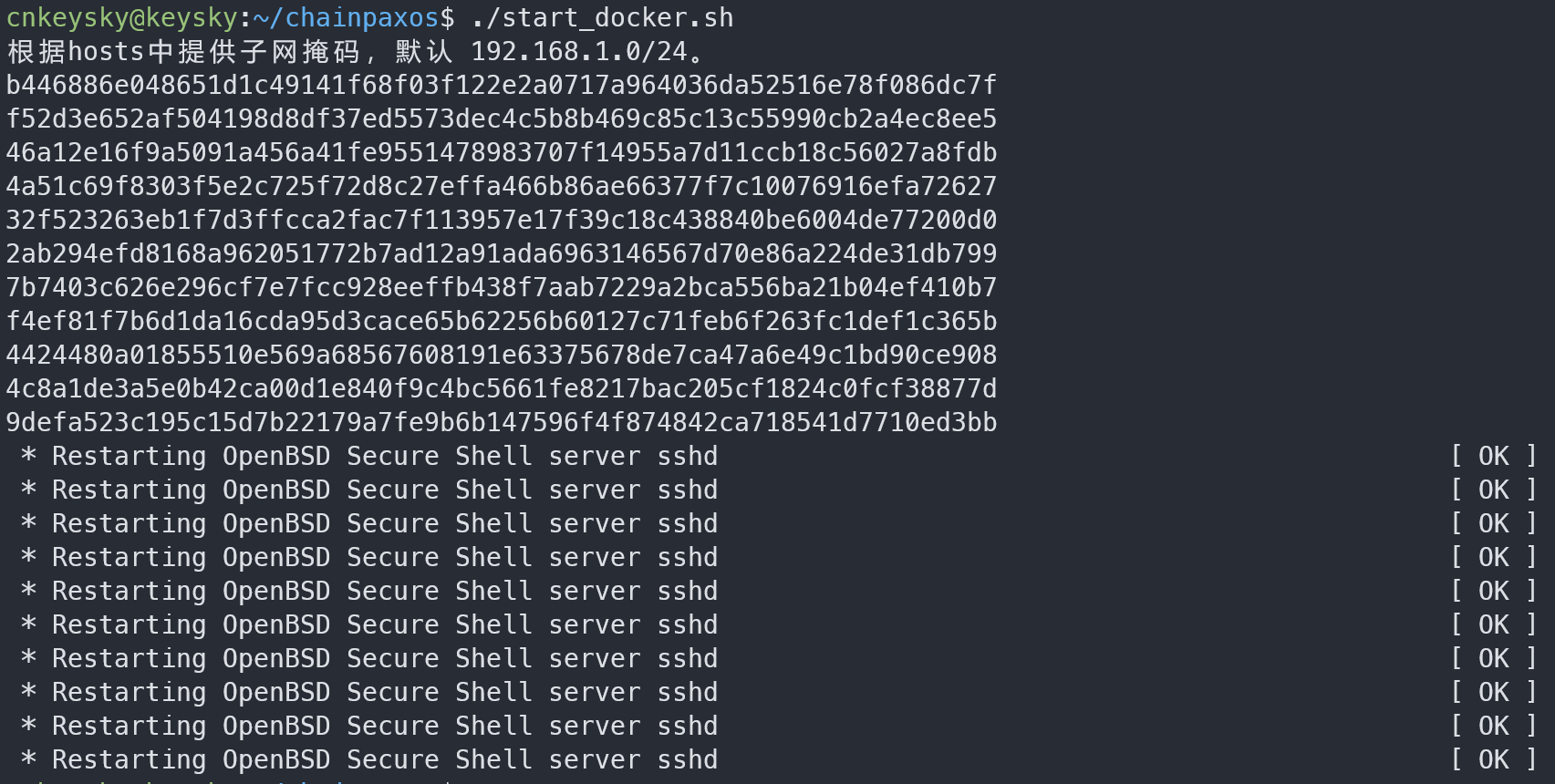
使用generate\_hosts.sh脚本生成容器ip地址，默认生成10个IP地址，从192.168.1.1开始，脚本支持自定义，但建议使用默认。由于该脚本需要修改/etc/hosts文件，所以需要使用sudo方式运行该脚本。



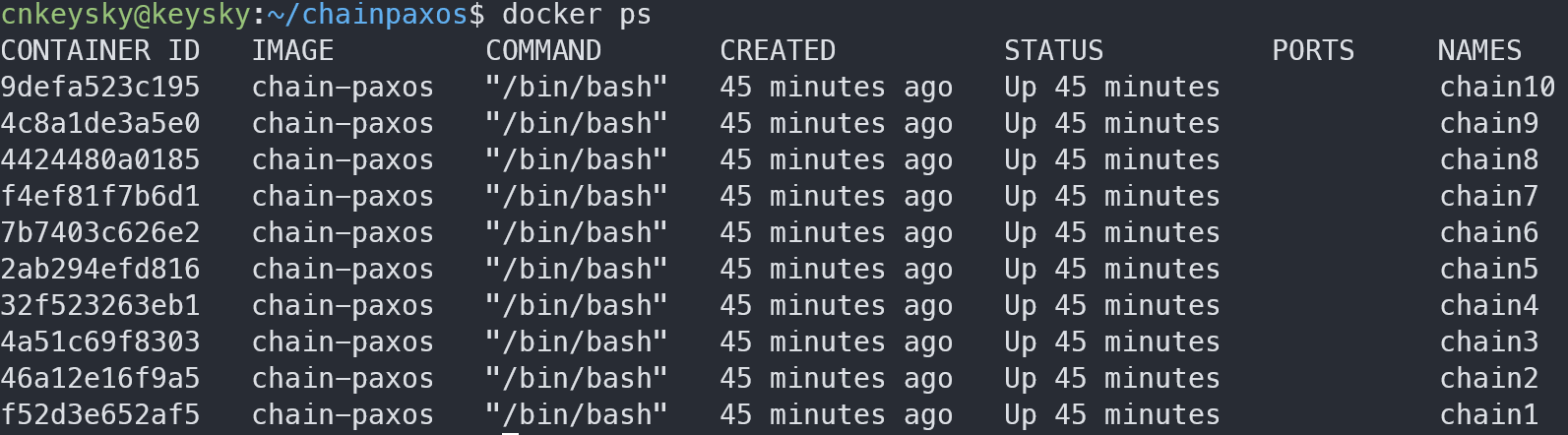
如果使用自定义IP地址，需要自行调用create\_docker\_network.sh脚本，以生成对应的容器网络段，具体指令可以自行查看该脚本中的内容。方便起见，这里我们使用默认设置，在使用start\_docker.sh脚本生成容器时，该脚本会自动帮我们调用create\_docker\_network.sh脚本，生成默认的容器网络段。



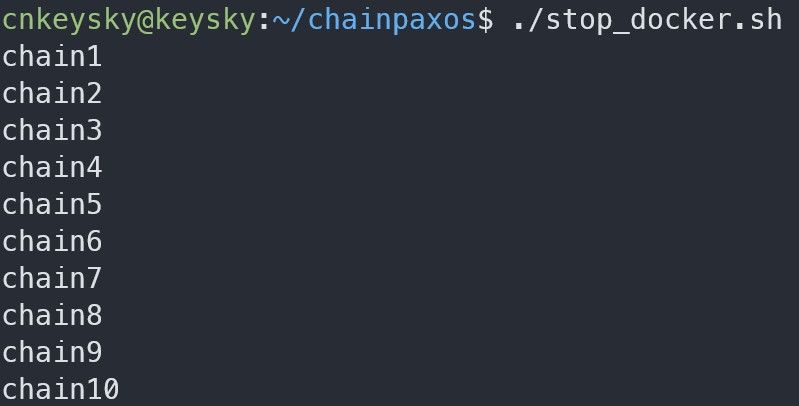
接着，我们使用start\_docker.sh脚本去生成相应的容器，容器的IP地址是根据hosts文件中的IP依次赋予，这些容器都存在同一个网段内。该脚本同时还负责启动ssh服务，以便之后通过ssh连接这些容器执行测试任务。下面我们将执行该脚本：



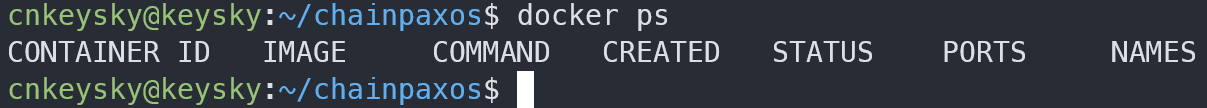
此时docker容器和ssh服务已经完成启动，接着我们可以使用docker命令查看容器运行情况：



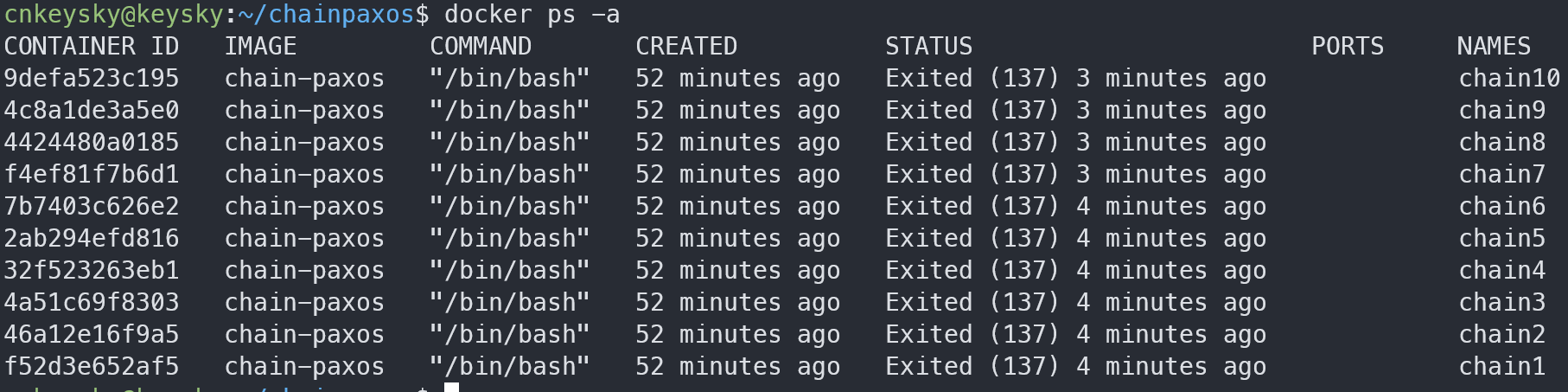
如果需要停止容器运行，可以运行stop\_docker.sh脚本：



此时我们再次查看容器运行情况，可以发现当前已经没有容器在运行：



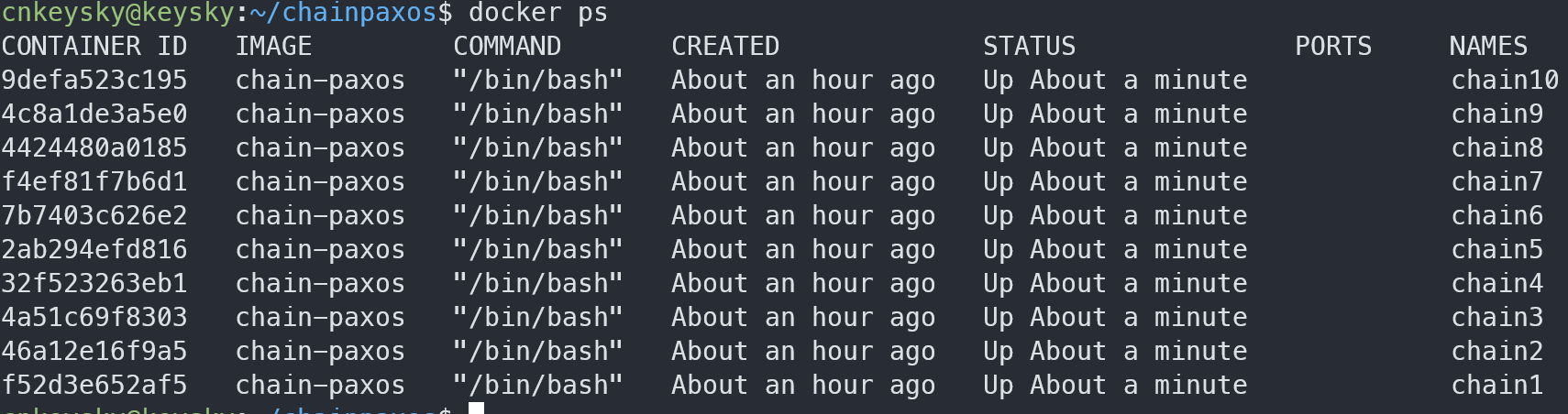
由于我们运行了start\_docker.sh脚本并且没有删除容器，所以容器还是存在，只是不在运行态：



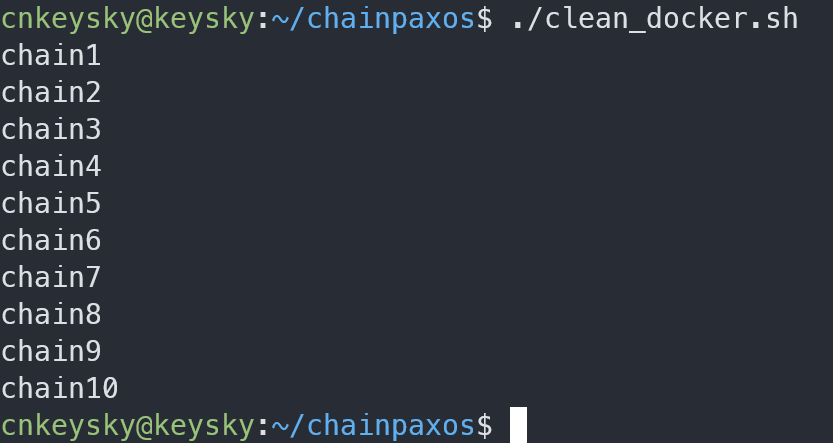
由于创建的容器还是存在，所以我们再次运行容器时就不能再次运行start\_docker.sh脚本，此时我们使用start\_container.sh脚本进行替代，需要注意的时，该脚本的运行是建立在start\_docker.sh脚本运行过的情况下，现在我们使用start\_docker.sh重新运行容器：



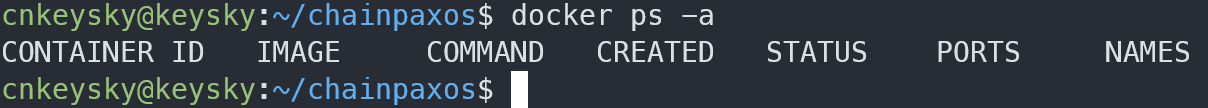
此时容器已经在运行态，我们使用docker命令来再次查看一下：



如果不再需要容器，我们可以运行clean\_docker.sh脚本：



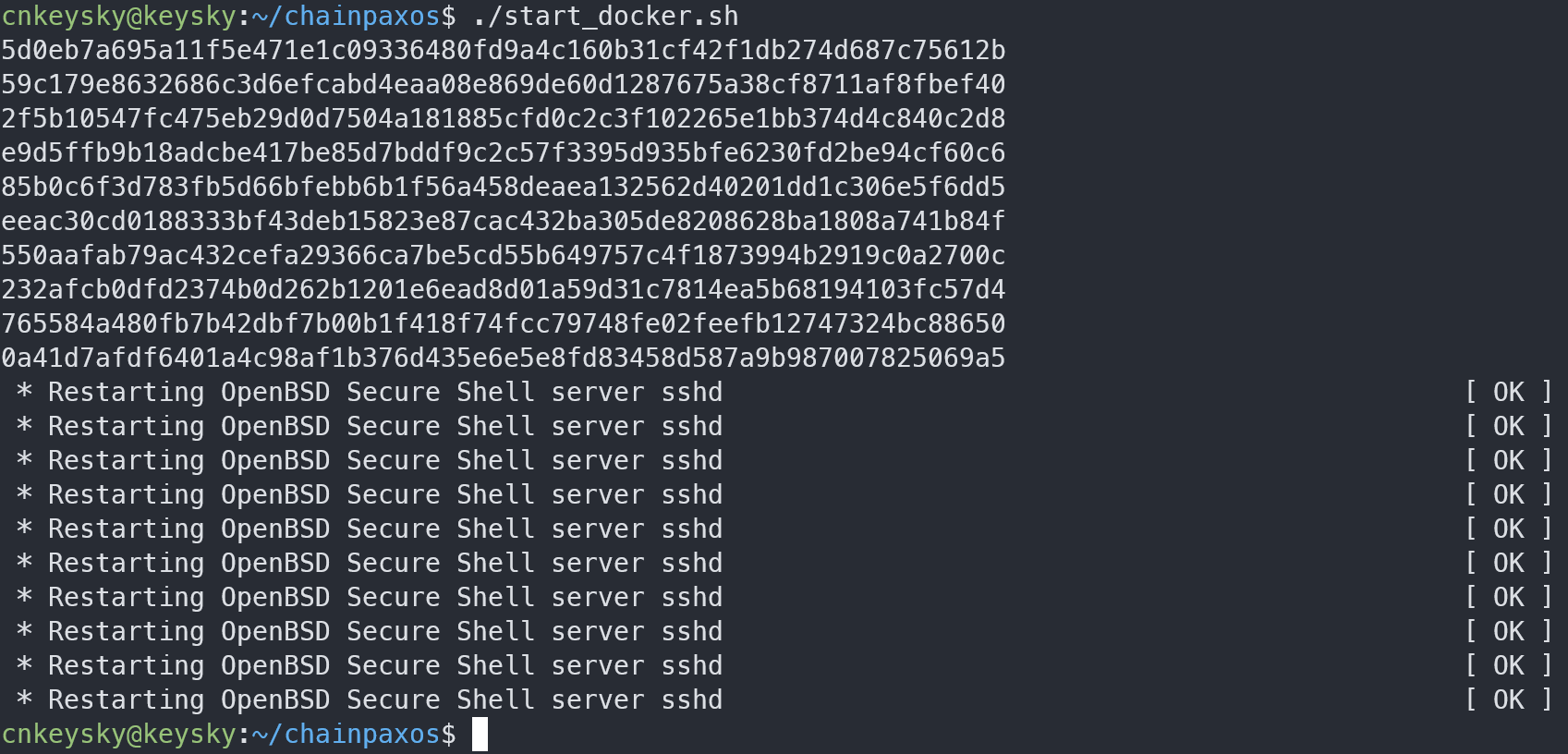
此时让我们通过docker命令来确认容器已经被删除：



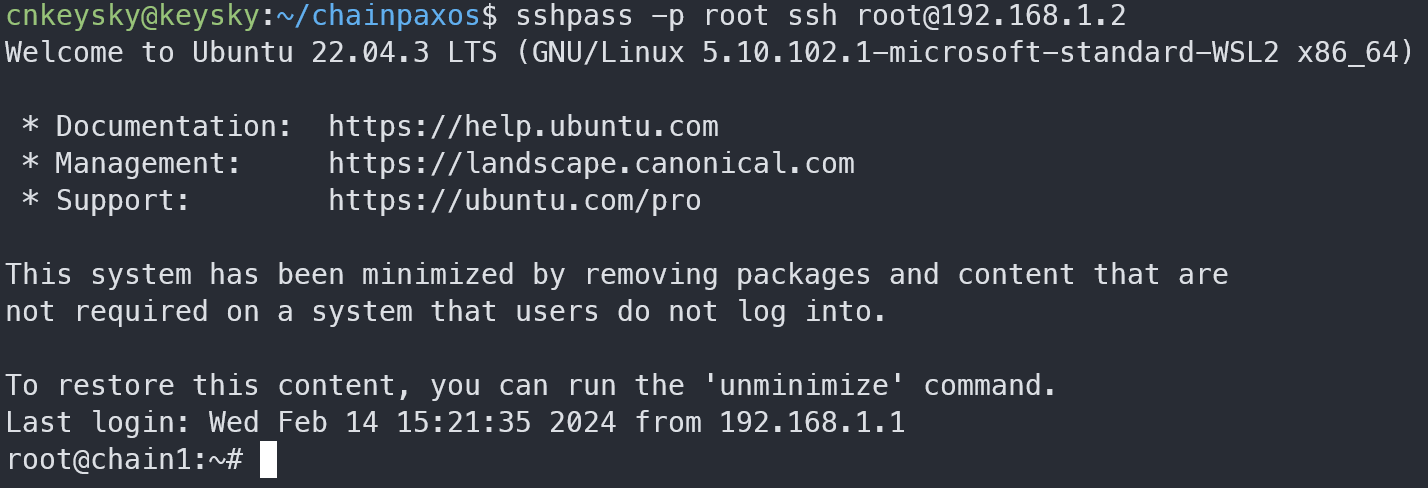
目前为止，主要的脚本都已经介绍，其他的脚本的使用将在其他章节进行介绍，下一章节我们将正式开始算法环境的运行。

# 环境运行

在开始本章节之前，请确保你已经详细阅读并掌握了之前的章节。确保已经运行过generate\_hosts.sh脚本（一定要使用sudo方式运行），然后，我们再次运行start\_docker.sh脚本：



docker容器和ssh服务已经启动，此时我们使用ssh登录容器测试一下：



如果上述图片中没有该执行结果，则需要直接使用ssh命令连接目标主机，然后依次接受ssh授权即可。

# 实验

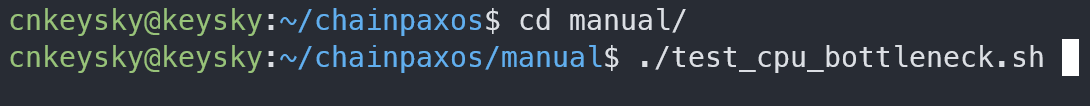
在我们启动docker容器之后，就可以进行测试了。

## 测试CPU瓶颈

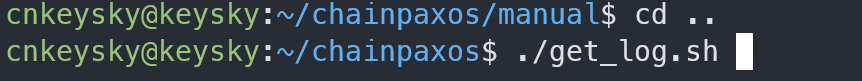
首先，执行chainpaxos目录下的start\_zkOriginal.sh脚本。



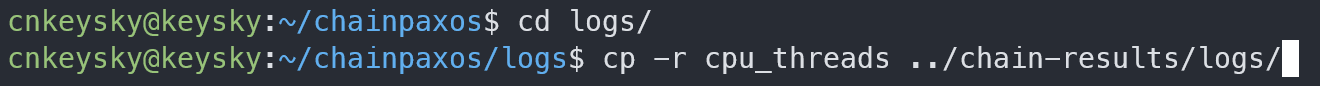
在该脚本启动完成之后，我们需要进入到manual目录中并执行test\_cpu\_bottleneck.sh脚本。



该脚本执行完成之后，需要回到chainpaxos目录并执行get\_log.sh脚本，该脚本会将容器中的结果拷贝到chainpaxos/log目录下。

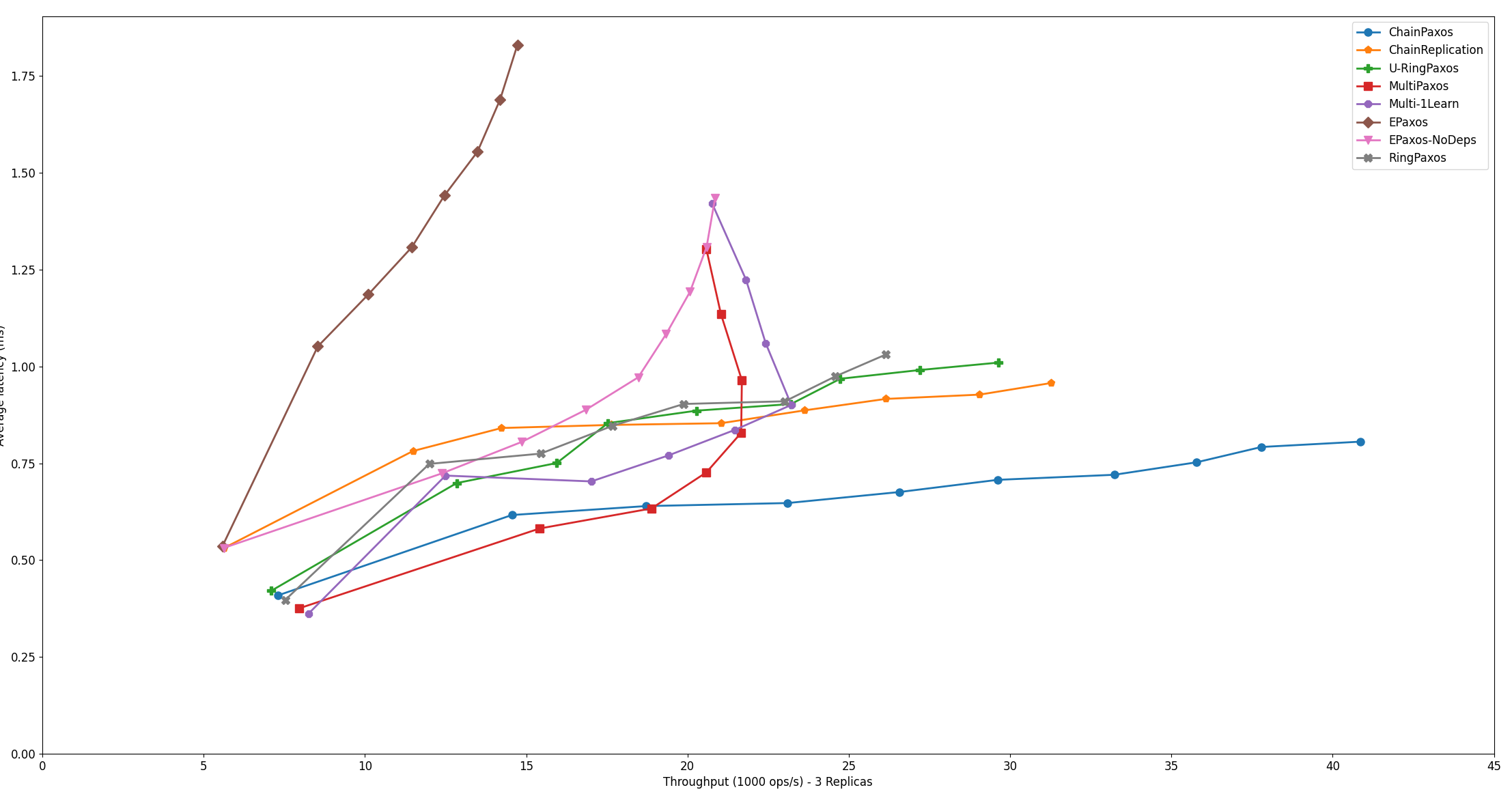


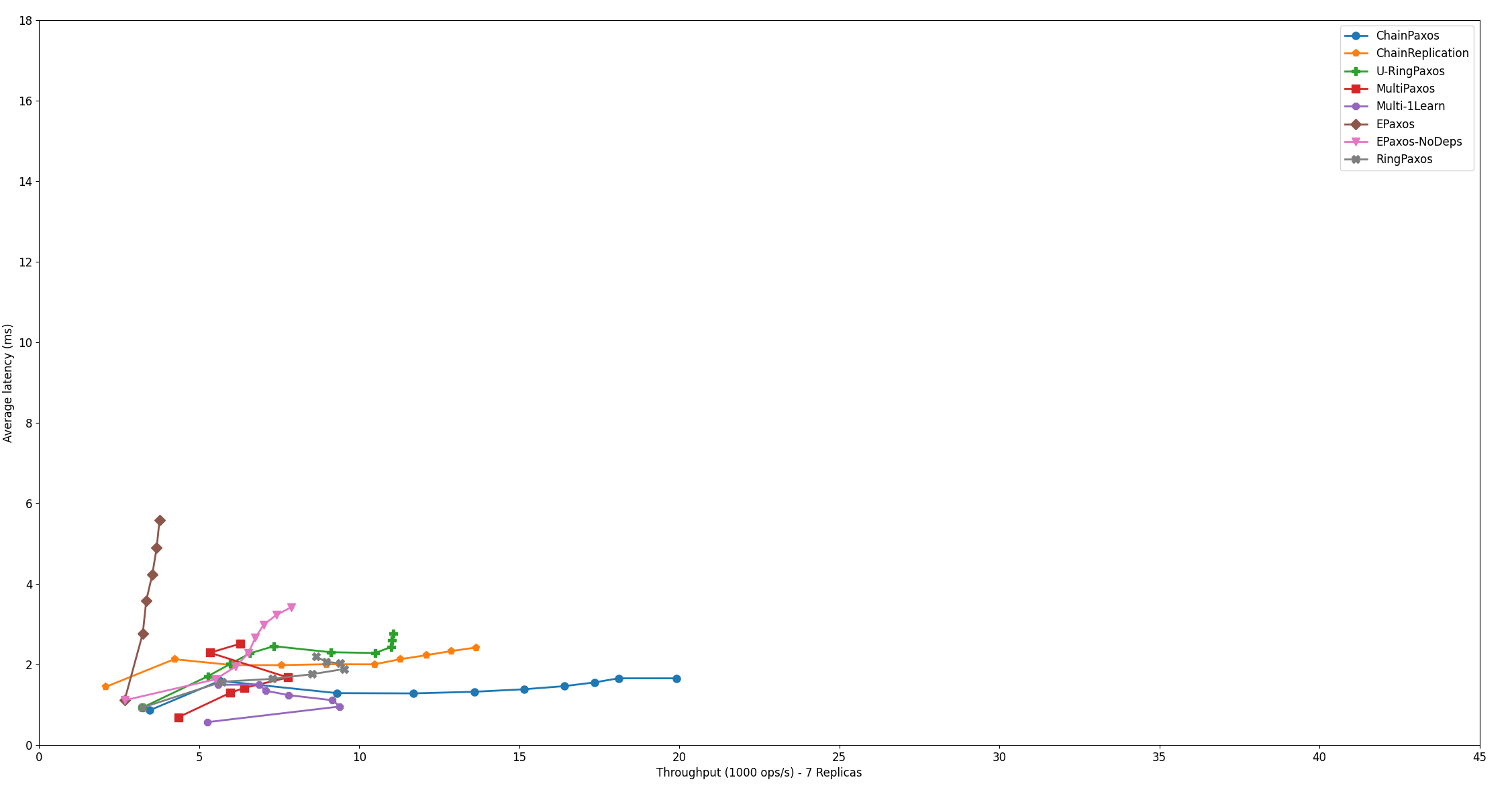
然后，进入到该目录并将cpu\_threads目录拷贝到chainpaxos/chain-results/logs目录下。



这些操作执行完成之后，我们进入到chainpaxos/chain-results目录中，通过python3运行perf\_alt.py脚本，最终将得到如下结果。

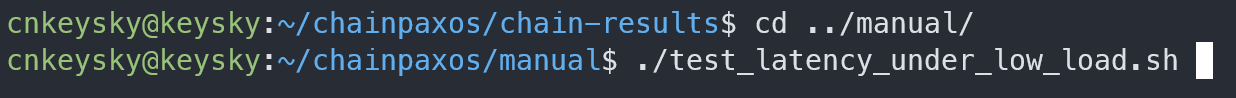




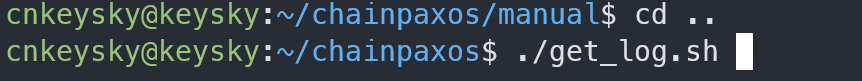


## 测试低负载下的延迟

现在，让我们再次回到chainpaxos/manual目录下，并执行test\_latency\_under\_low\_load.sh脚本。



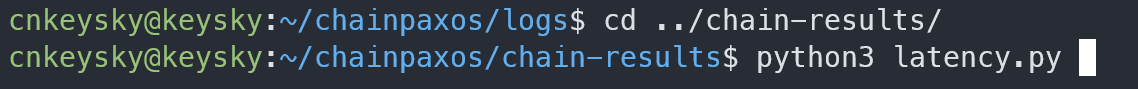
该脚本执行完成之后，需要回到chainpaxos目录并执行get\_log.sh脚本，该脚本会将容器中的结果拷贝到chainpaxos/log目录下。

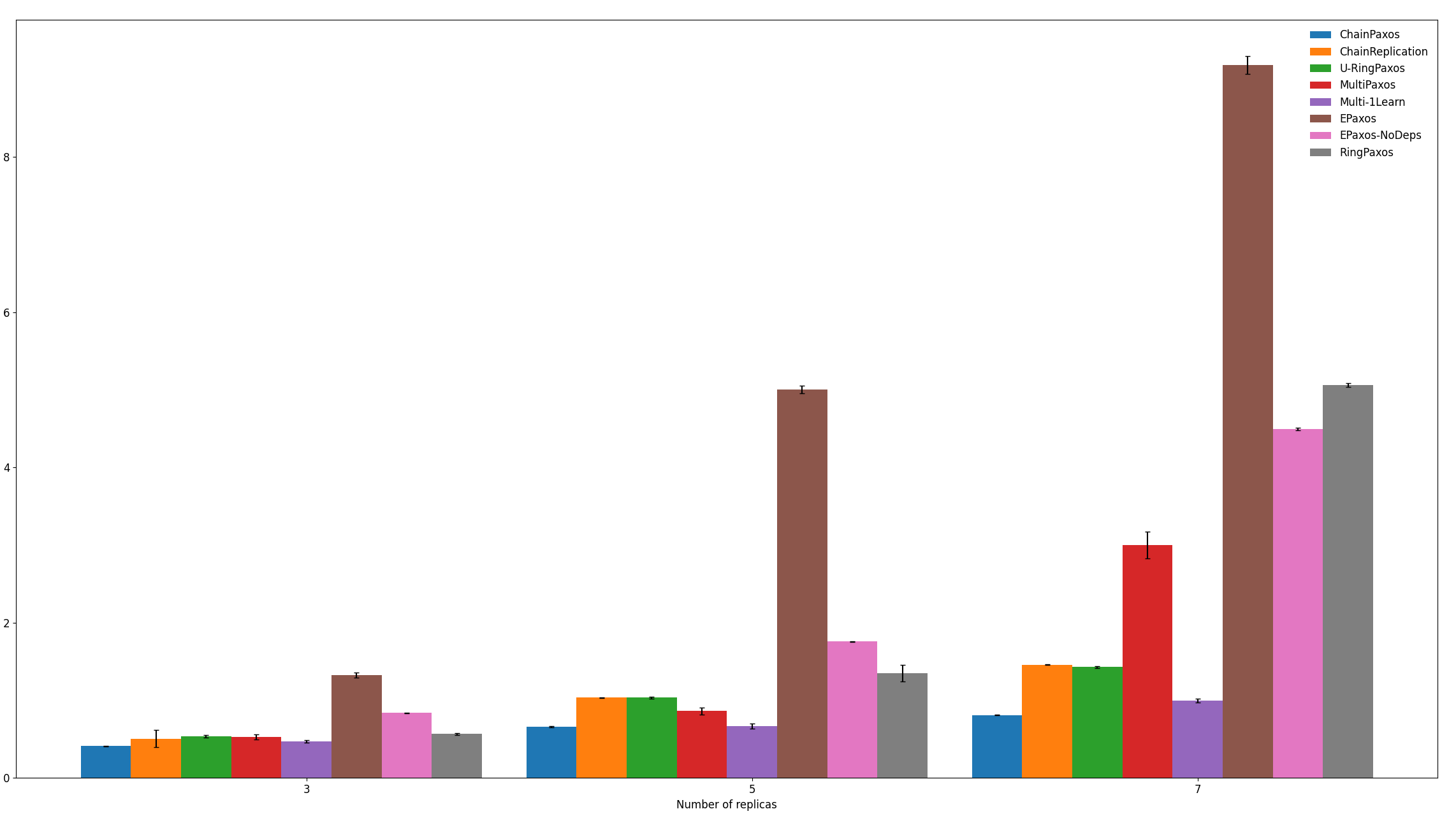


然后，进入到该目录并将latency目录拷贝到chainpaxos/chain-results/logs目录下。



这些操作执行完成之后，我们进入到chainpaxos/chain-results目录中，通过python3运行latency.py脚本，最终将得到如下结果。



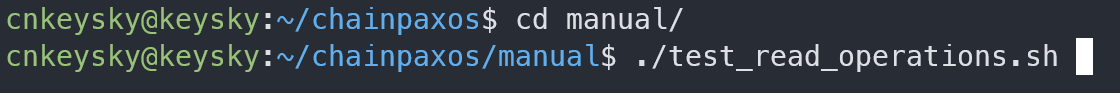


## 测试读操作

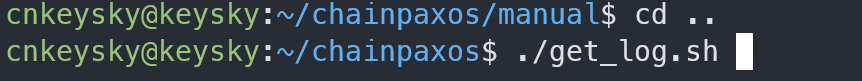
依次执行完5.1和5.2之后，我们需要进入chainpaxos目录下，执行stop\_zkOriginal.sh脚本，这将停止zookeeper的运行。



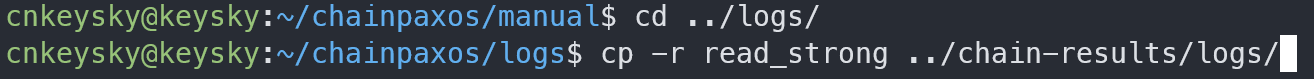
现在，让我们再次回到chainpaxos/manual目录下，并执行test\_read\_operations.sh脚本。



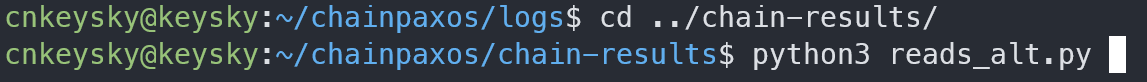
该脚本执行完成之后，需要回到chainpaxos目录并执行get\_log.sh脚本，该脚本会将容器中的结果拷贝到chainpaxos/log目录下。

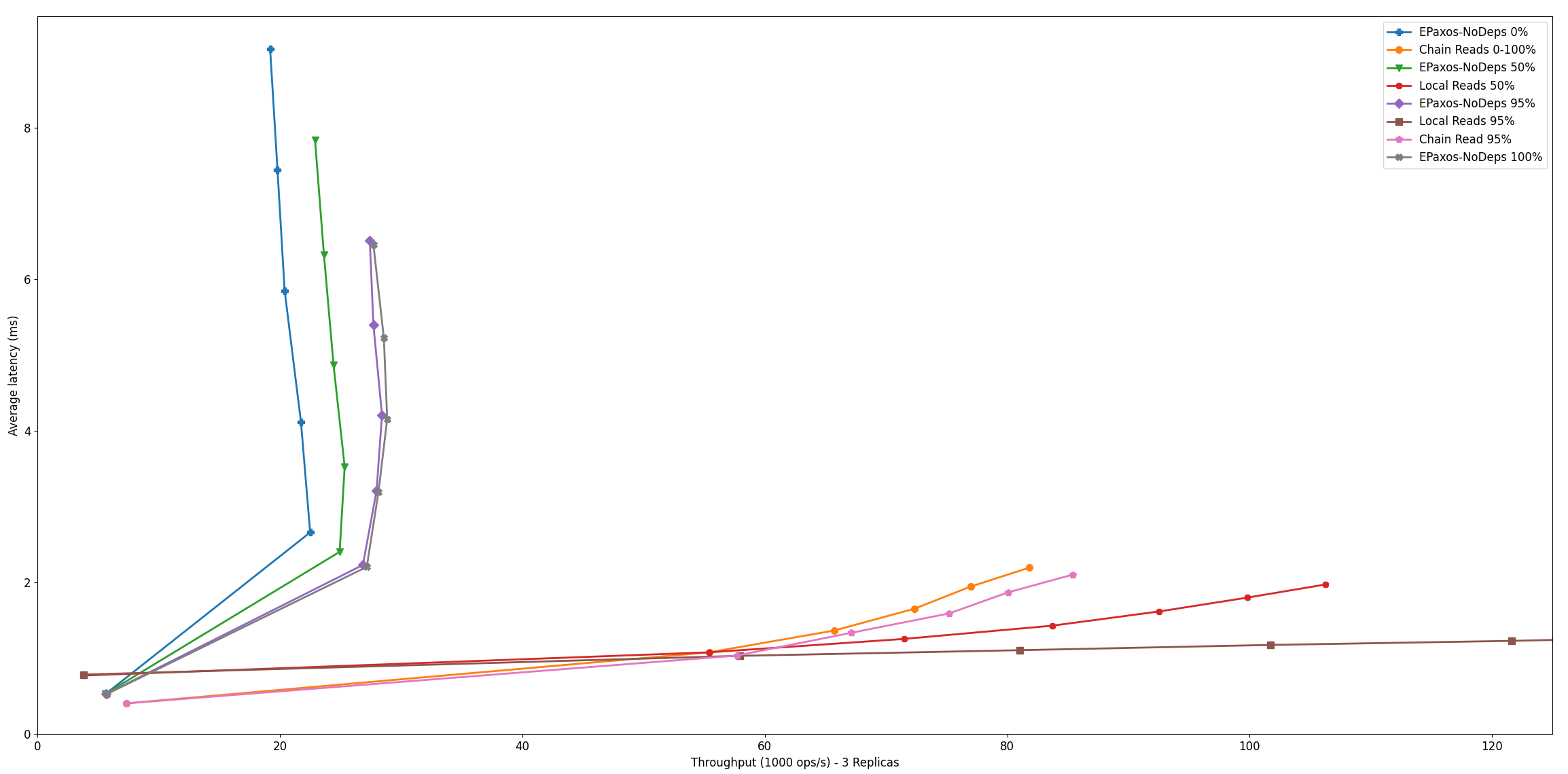


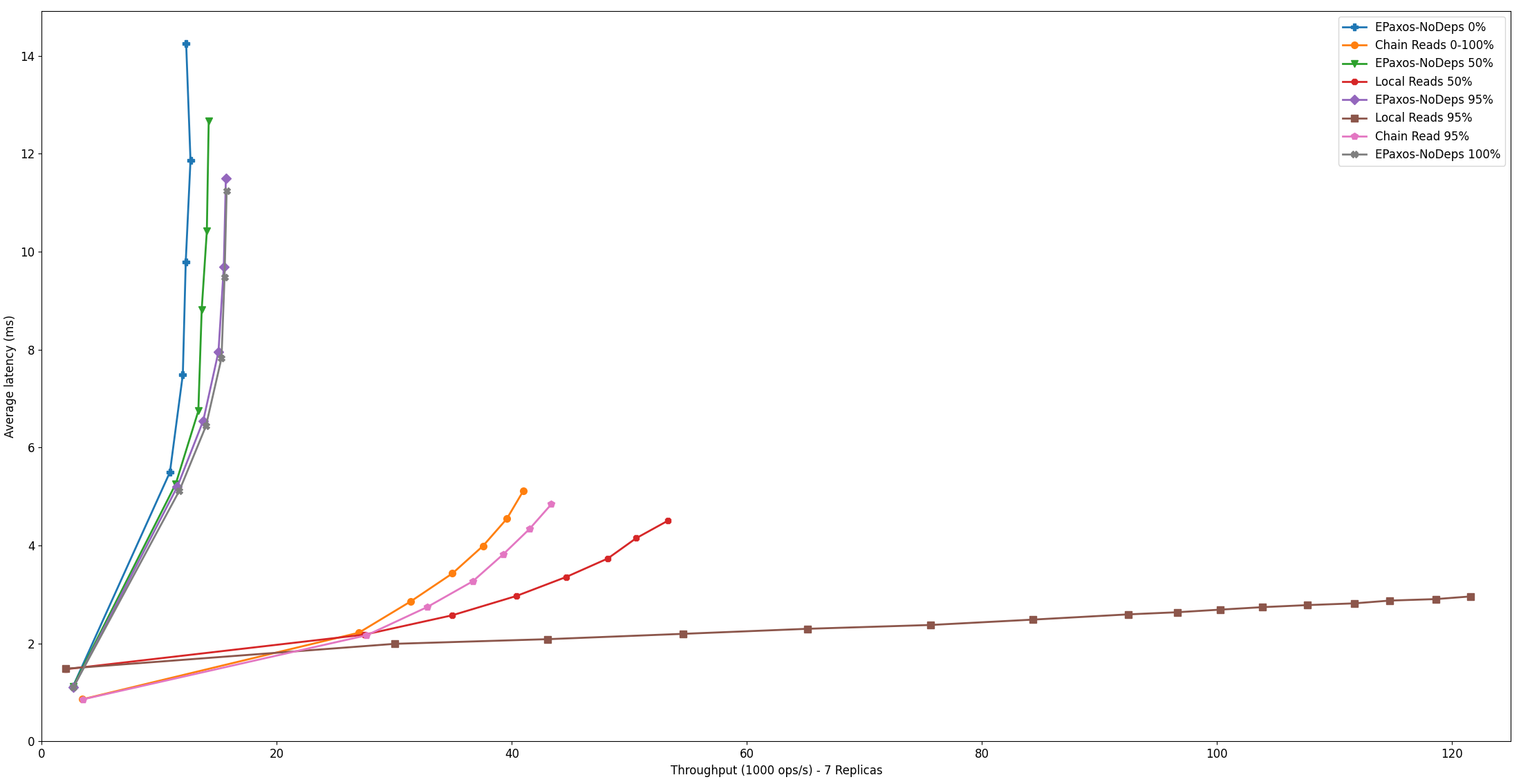
然后，进入到该目录并将read\_strong目录拷贝到chainpaxos/chain-results/logs目录下。



这些操作执行完成之后，我们进入到chainpaxos/chain-results目录中，通过python3运行reads\_alt.py脚本，最终将得到如下结果。

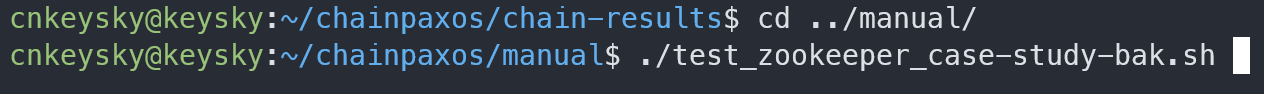




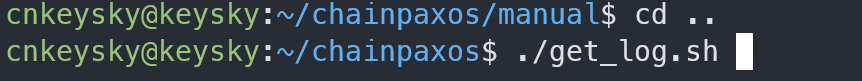


## ZooKeeper案例研究

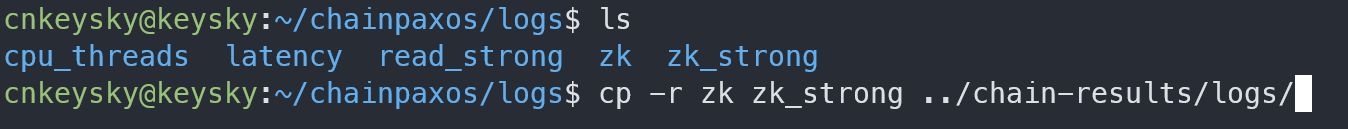
现在，让我们回到chainpaxos/manual目录下，并执行test\_zookeeper\_case-study.sh脚本。



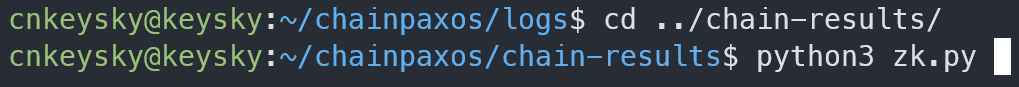
该脚本执行完成之后，需要回到chainpaxos目录并执行get\_log.sh脚本，该脚本会将容器中的结果拷贝到chainpaxos/log目录下。

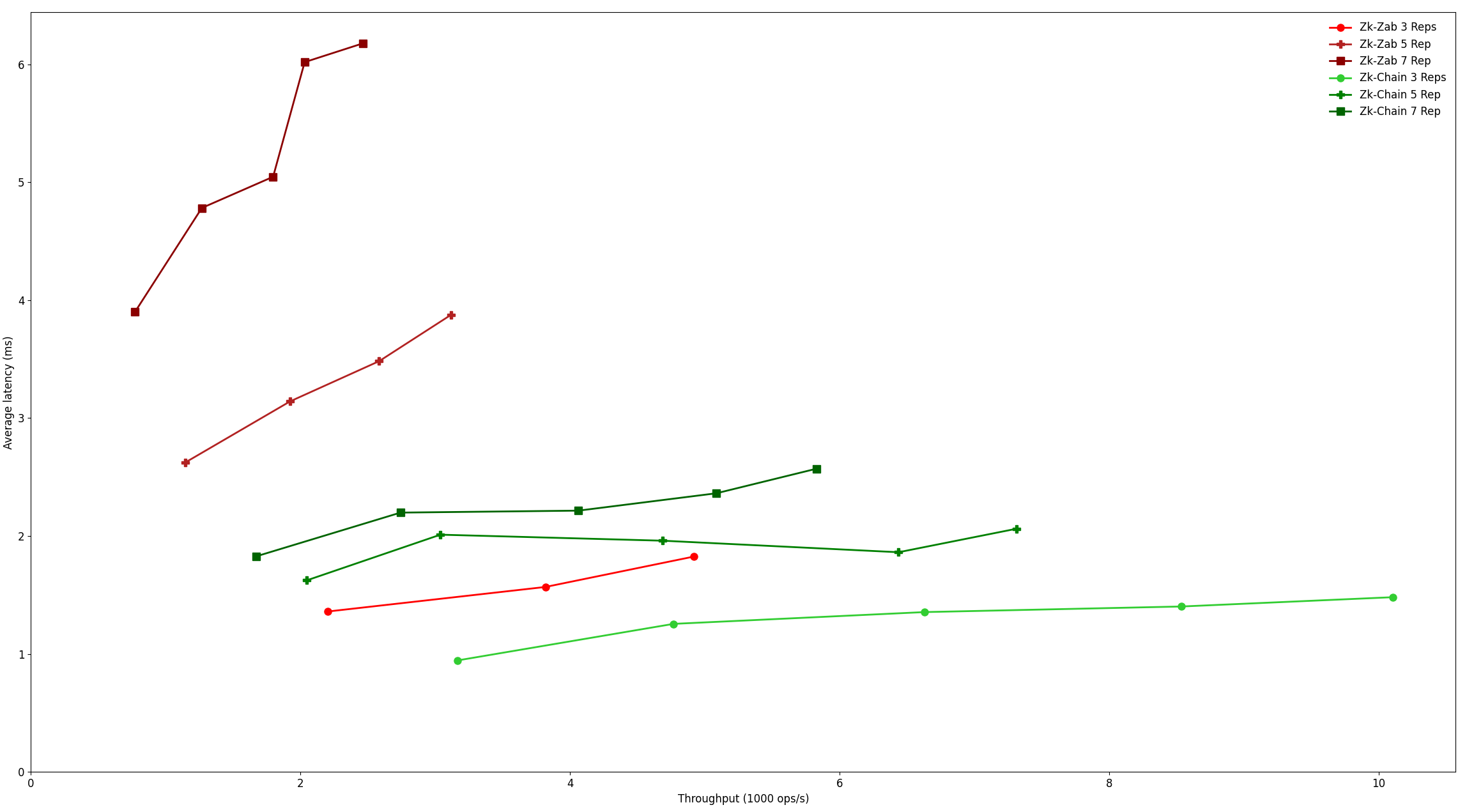


然后，进入到该目录并将zk和zk\_strong目录拷贝到chainpaxos/chain-results/logs目录下。



这些操作执行完成之后，我们进入到chainpaxos/chain-results目录中。首先，通过python3运行zk.py脚本，将得到如下结果。





然后，再使用python3运行zk\_replica.py脚本，将得到如下结果。

