Министерство высшего образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ

УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Кафедра безопасности информационных систем (БИС)

LINUX, DOCKER, BASH, ОКРУЖЕНИЕ ДЛЯ РАЗРАБОТКИ

Отчет по лабораторной работе №1

по дисциплине «Системное программирование»

Студентка гр.737-1

\_\_\_\_\_\_ Агеева В.С.

\_\_.\_\_.2021г

Принял

Руководитель

доцент кафедры БИС

\_\_\_\_\_\_\_ Романов А.С.

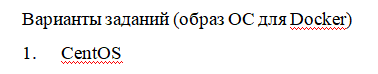
\_\_.\_\_.2021г

Томск 2021

1 Введение

Целью лабораторной работы является ознакомление с операционной системой и основными командами ОС Linux, возможностями Docker для контейнеризации окружения программного обеспечения и его зависимостей, средств разработки и сборки ПО. Необходимо подготовить Dockerfile, в котором осуществляется виртуализация операционной системы и устанавливаются необходимые пакеты.

В ходе лабораторной работы требуется выполнить задание и разработать скрипт, соответствующее 1 варианту (рисунок 1.1).



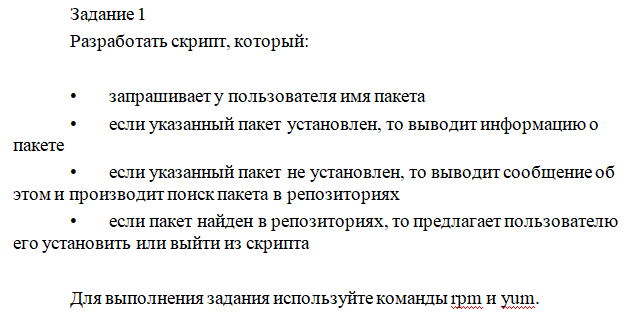


Рисунок 1.1 – Вариант и задание на лабораторную работу

После выполнения задания, необходимо загрузить код готовой программы и Dockerfile на github.com.

2 Ход работы

Для начала лабораторной работы рекомендуется реализовывать установку Docker на операционной системе Ubuntu, запустить которую можно на виртуальной машине (рисунок 2.1).

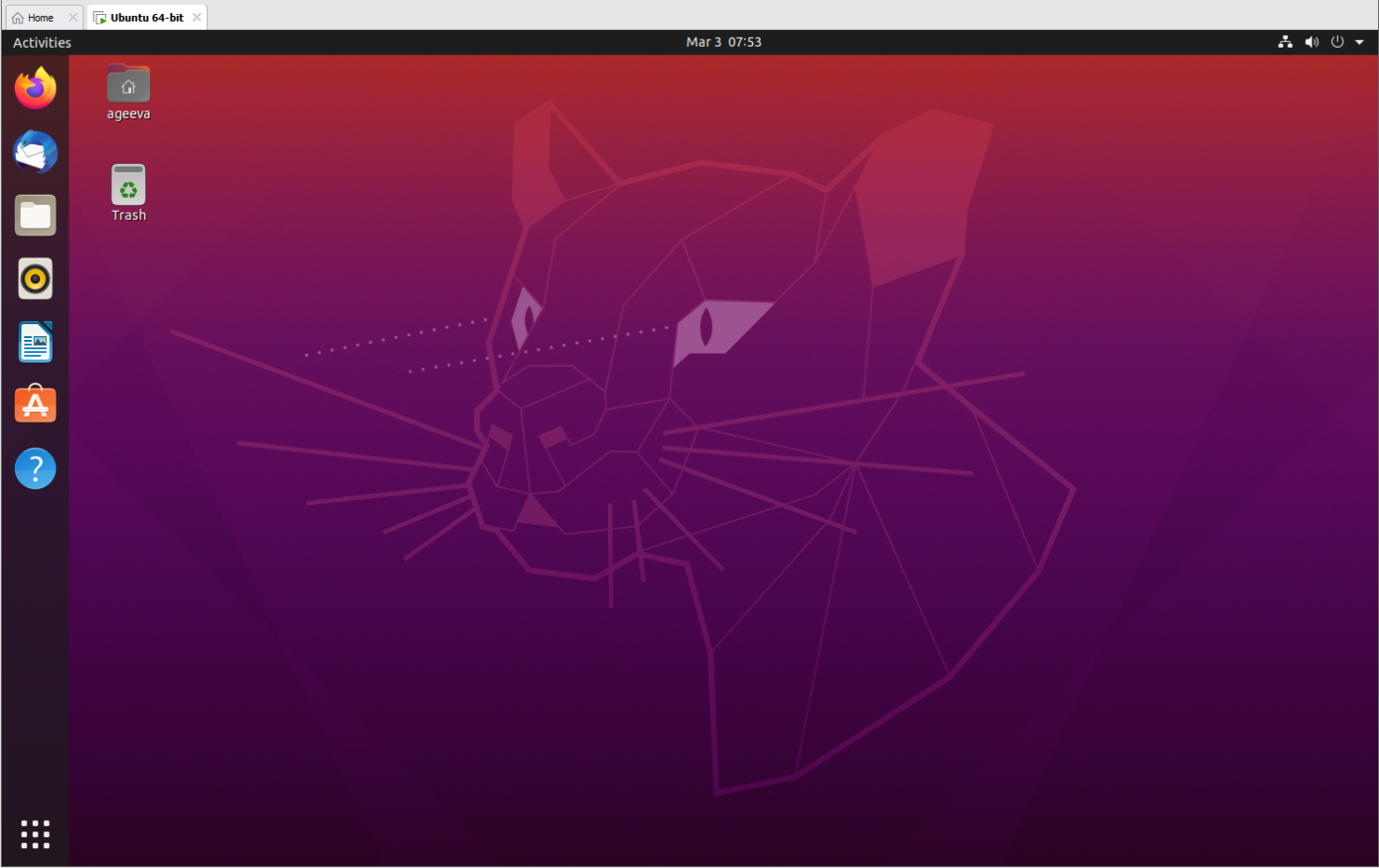


Рисунок 2.1 – Ubuntu 20.04.2.0, реализованная на VMware

После приступим к работе с терминалом. Для вызова терминала нужно нажать сочетание клавиш Ctrl+Alt+T. Запустив его, можно увидеть следующее окно (рисунок 2.2).

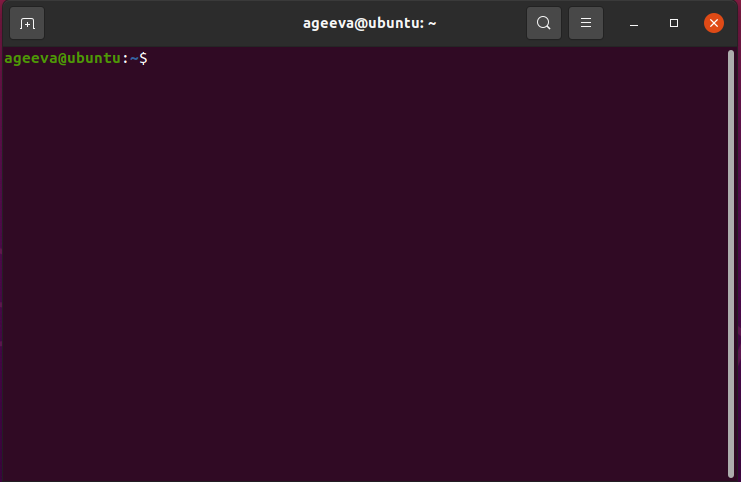


Рисунок 2.2 – Терминал

Для того чтобы подготовить систему к установке Dockera, для начала нужно обновить индекс пакетов (загружаем списки пакетов из репозиториев и "обновляем" их, чтобы получить информацию о новейших версиях пакетов и их зависимостях) (рисунок 2.3).

Команда: sudo apt-get update

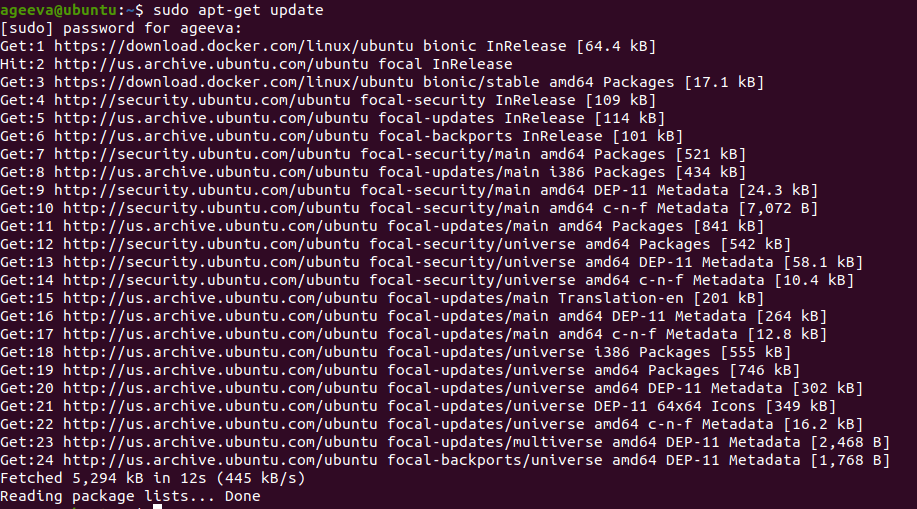


Рисунок 2.3 - Обновление индекс пакетов

Затем установим необходимые пакеты, которые позволяют apt использовать пакеты по HTTPS (рисунок 2.4).

Команда: sudo apt install apt-transport-https ca-certificates curl software-properties-common

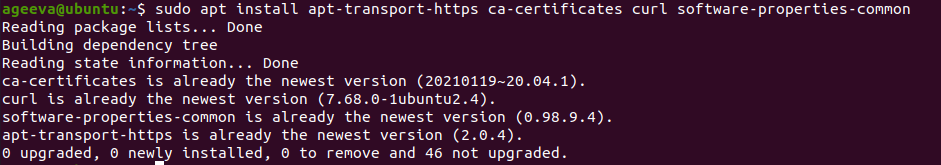


Рисунок 2.4 – Установка необходимых пакетов

Затем добавим в свою систему ключ GPG официального репозитория Docker (рисунок 2.5).

Команда: curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -



Рисунок 2.5 – Добавление в систему ключа GPG

Добавим репозиторий Docker в список источников пакетов APT (рисунок 2.6).

Команда: sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"

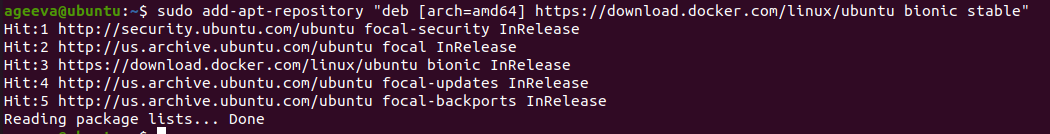


Рисунок 2.6 – Добавление репозитория Docker в список источников пакетов APT

Затем обновим базу данных пакетов информацией о пакетах Docker из вновь добавленного репозитория (рисунок 2.7).

Команда: sudo apt-get update

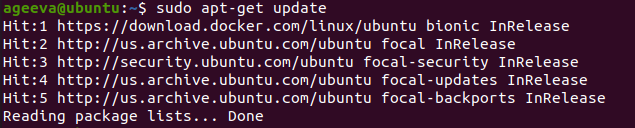


Рисунок 2.7 – Обновление базы данных пакетов

Следует убедиться, что устанавливаем Docker из репозитория Docker, а не из репозитория по умолчанию Ubuntu.

Команда: apt-cache policy docker-ce

Вывод получится приблизительно следующий (рисунок 2.8).

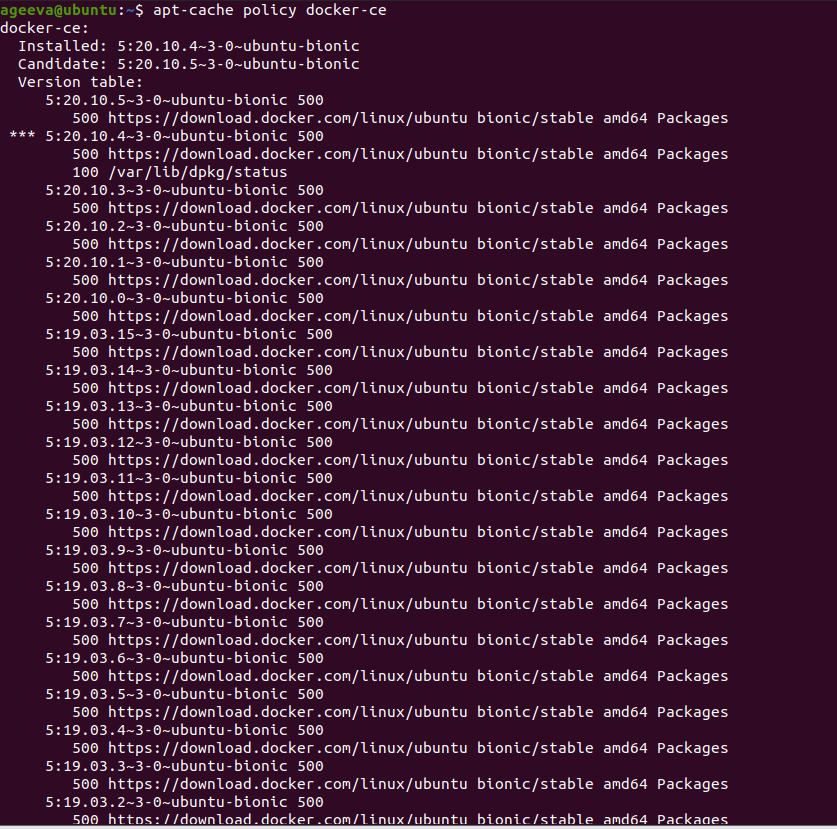


Рисунок 2.8 – Проверка установки

Далее установим Docker (рисунок 2.9).

Команда: sudo apt install docker-ce

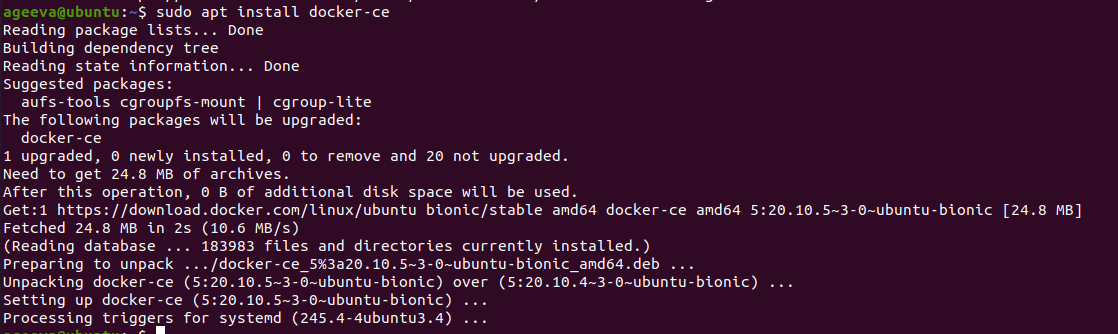


Рисунок 2.9 – Установка docker

Теперь Docker установлен, демон запущен, и процесс будет запускаться при загрузке системы.  Убедимся, что процесс запущен (рисунок 2.10).

Команда: sudo systemctl status docker

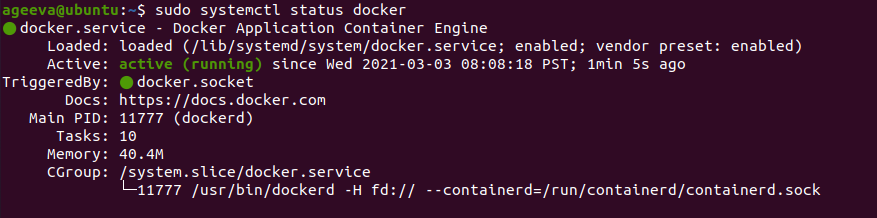


Рисунок 2.10 – Проверка запущен ли процесс

Вывод должен быть похож на представленный выше, сервис должен быть запущен и активен (рисунок 2.10).

Чтобы не вводить sudo каждый раз при запуске команды docker, добавим имя своего пользователя в группу docker (рисунок 2.11).

Команда: sudo usermod -aG docker ${USER}



Рисунок 2.11 – Добавление пользователя в группу docker

Для применения этих изменений в составе группы необходимо разлогиниться и снова залогиниться на сервере или задать следующую команду (рисунок 2.12).

Команда: su - ${USER}



Рисунок 2.12 – Применение изменений через команду

Для продолжения работы необходимо ввести пароль пользователя.

Убедимся, что пользователь добавлен в группу docker можно следующим образом (рисунок 2.13).

Команда: id –nG



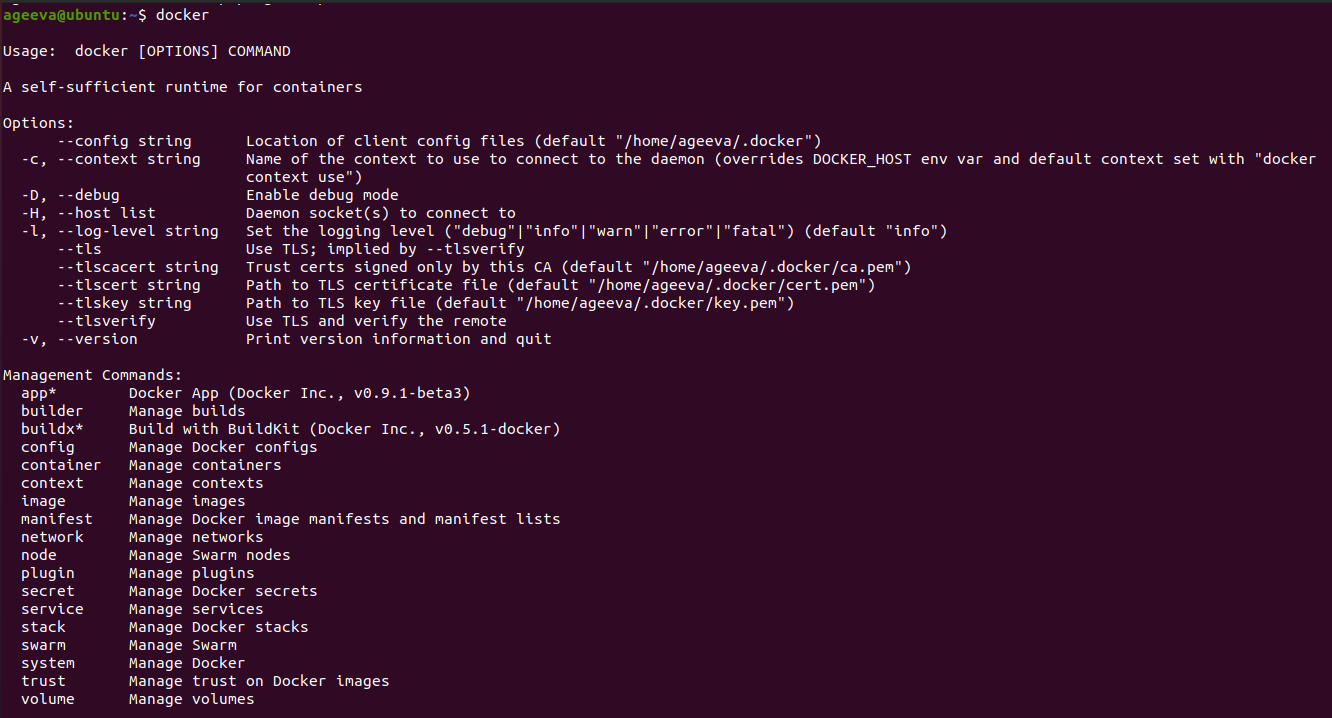
Рисунок 2.13 – Проверка добавления пользователя

Команда docker позволяет использовать различные опции, команды с аргументами. Синтаксис выглядит следующим образом:

docker [option] [command] [arguments]

Для просмотра всех доступных подкоманд введем следующую команду (рисунок 2.14).

Команда: docker



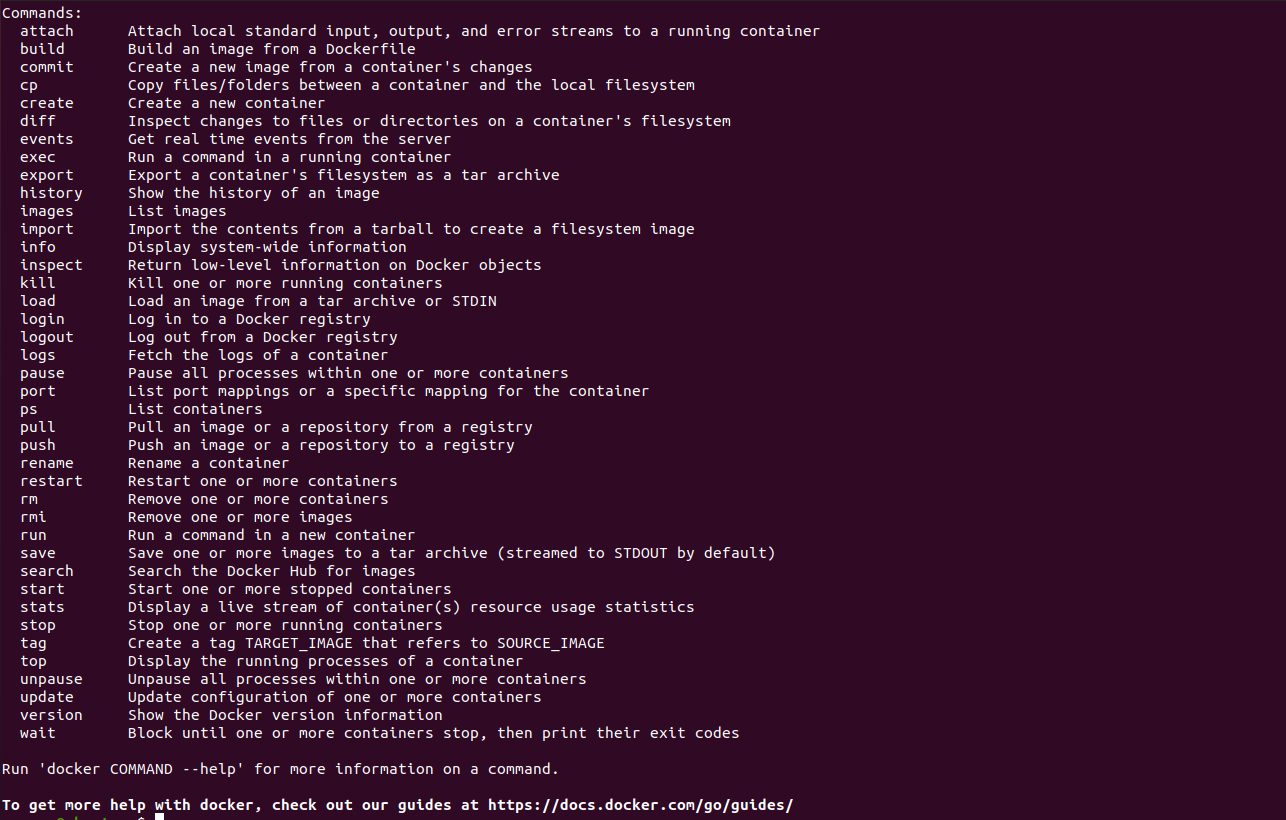
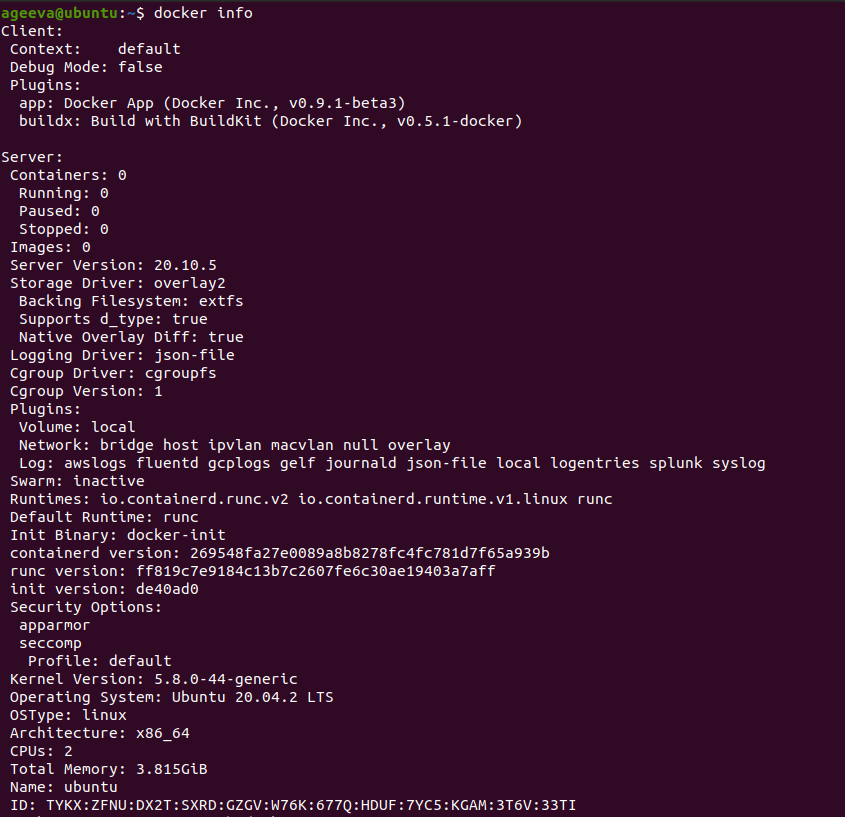


Рисунок 2.14 – Подкоманды Docker

Для просмотра всей информации о Docker используется следующая команда (рисунок 2.15).

Команда: docker info



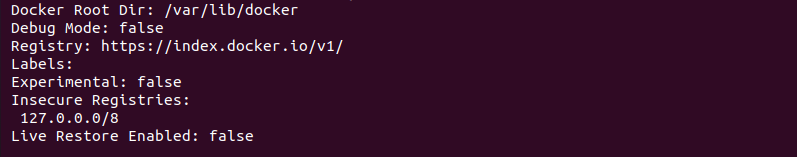


Рисунок 2.15 – Информация о docker

Для поиска необходимых образов на Docker Hub (рисунок 2.16) используется команда docker и подкоманда search. Чтобы найти образ Ubuntu, нужно ввести:

Команда: docker search ubuntu

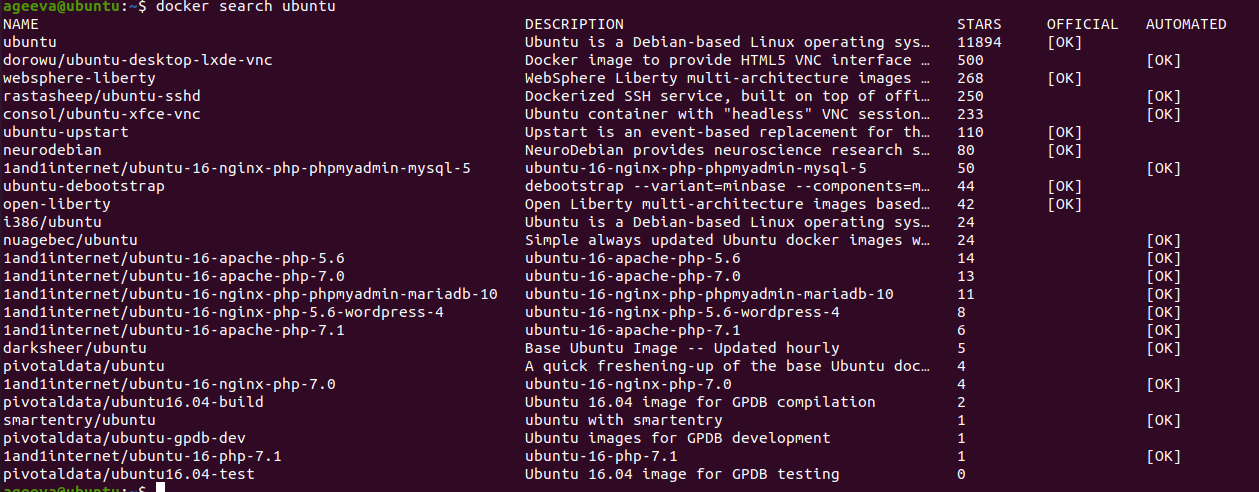


Рисунок 2.16 – Поиск в Docker Hub

Скрипт просматривает Docker Hub и возвращает список всех образов, имена которых подходят под заданный поиск.

Проще говоря, с помощью команды search был организован поиск, по ключевому слову Ubuntu, образов в Docker Hub.

В столбце OFFICIAL строка OK показывает, что образ построен и поддерживается компанией, которая занимается разработкой этого проекта.

После чего с помощью команды pull нужно выбрать необходимую версию и приступить к скачиванию (рисунок 2.17).

Команда: docker pull ubuntu

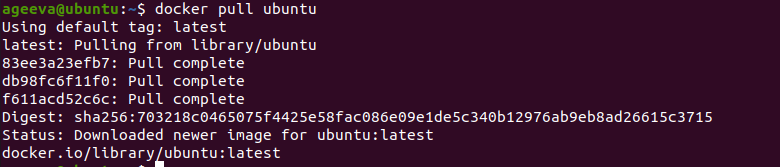


Рисунок 2.17 – Скачивание образа

Для просмотра загруженных на компьютер образов нужно ввести следующую команду (рисунок 2.18).

Команда: docker images

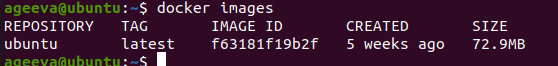


Рисунок 2.18 – список загруженных на компьютер образов

После загрузки образа можно запустить контейнер с загруженным образом с помощью подкоманды run (рисунок 2.19).

В качестве примера запустим контейнер с помощью последней версии образа Ubuntu. Комбинация параметров -i и -t обеспечивает интерактивный доступ к командному процессору контейнера.

Команда: docker run -it ubuntu



Рисунок 2.19 – Запуск контейнера через команду run

Далее обновим индекс пакетов (рисунок 2.20).

Команда: sudo apt-get update

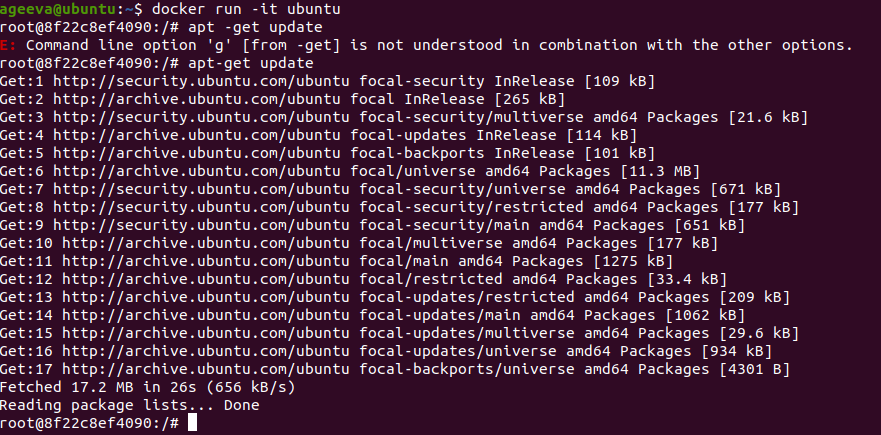


Рисунок 2.20 – Обновление индекса пакетов для гостевой ОС

Для выполнения работы необходимо установить текстовый редактор Nano (рисунок 2.21).

Nano - текстовый редактор для консоли. Используется в Unix-подобных ОС, распространяется свободно.

Команда: apt install nano

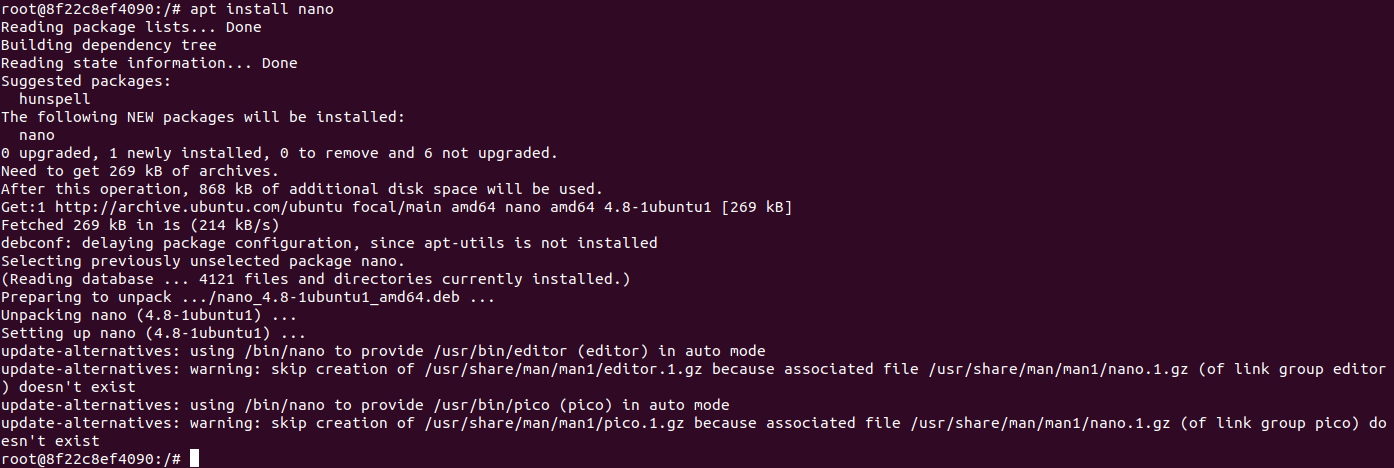


Рисунок 2.21 - Установка текстового редактора

Для создания текстового файла используем следующую команду (рисунок 2.22).

Команда: nano «название файла».sh

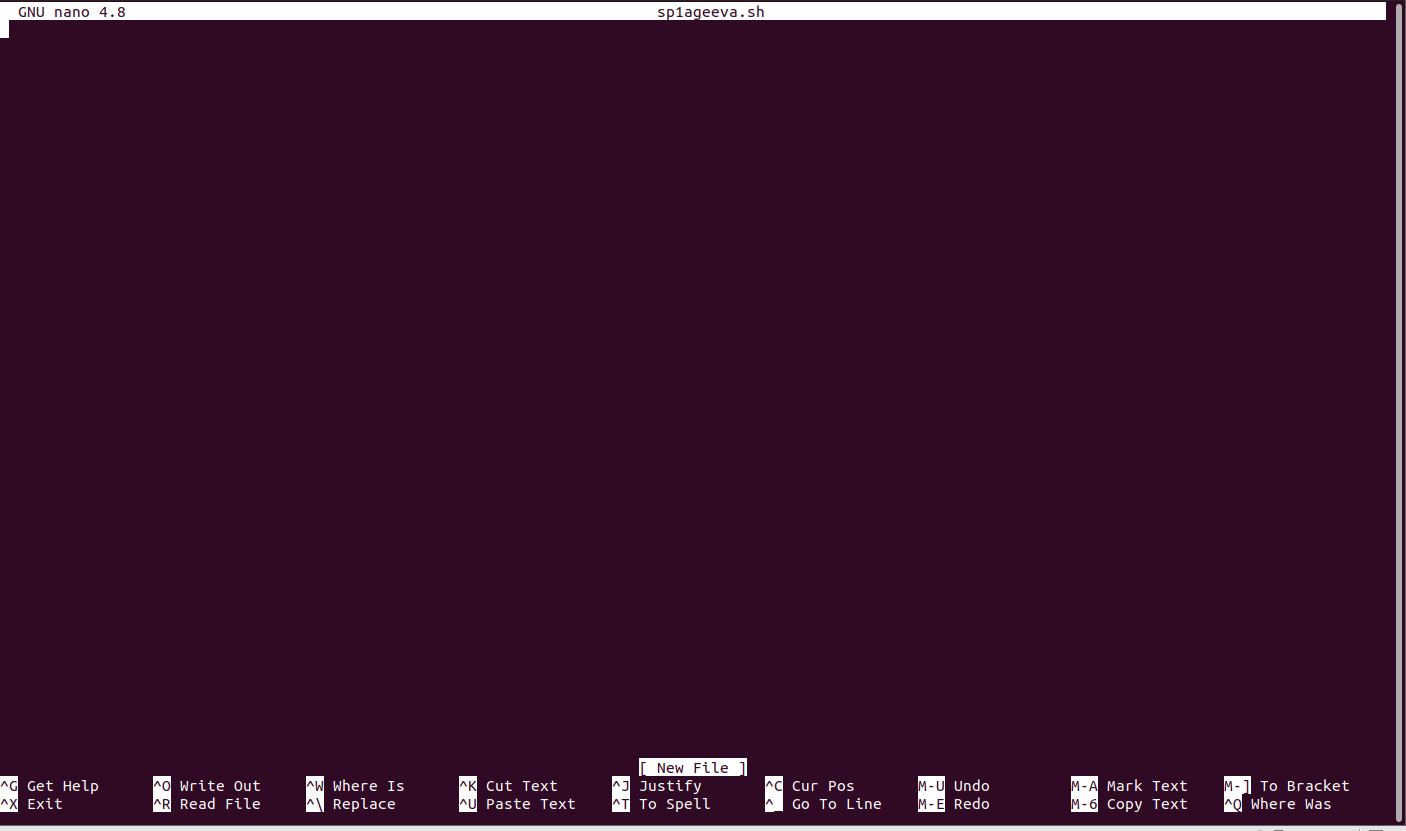


Рисунок 2.22 – Создание файла

Проверим работу текстовых файлов с помощью команды echo (рисунок 2.23).



Рисунок 2.23 – Вывод Hello World

Далее нужно проверить созданный файл с помощью команды ls (рисунок 2.24).

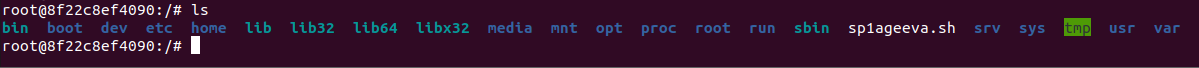


Рисунок 2.24 – Проверка созданного файла

Необходимо сделать данный файл исполняемым, для этого используем команду chmod (рисунок 2.25).

Команда: chmod ugo+x “название файла”.sh

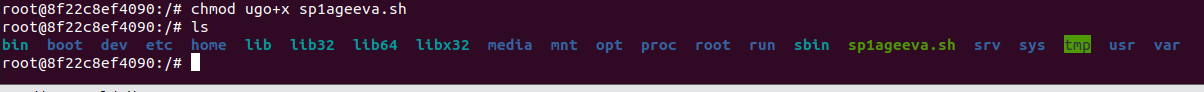


Рисунок 2.25 – Установка прав на файл

Проверка как работает пример скрипта (рисунок 2.26).

Команда: sh “название файла”.sh

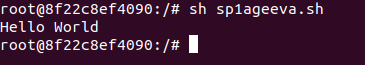


Рисунок 2.26 – Проверка работы скрипта

Чтобы выйти и остановить контейнер, введем команду exit (рисунок 2.27).

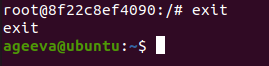


Рисунок 2.27 – Выход их контейнера

Чтобы посмотреть список всех контейнеров и их coin id (для подключения и запуска), нужно ввести следующую команду (рисунок 2.28).

Команда: docker ps –a

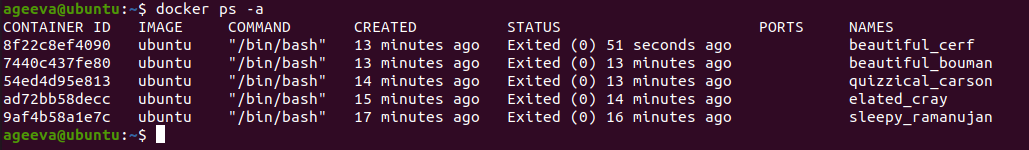


Рисунок 2.28 – Список всех контейнеров

Для запуска остановленного контейнера, нужно ввести следующую команду (рисунок 2.29):

Команда: sudo docker start “coin id”

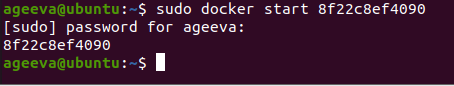


Рисунок 2.29 – Запуск остановленного контейнера

Подключение к существующему контейнеру осуществляется через следующую команду (рисунок 2.30)

Команда: sudo docker attach “coin id”



Рисунок 2.30 – Подключение к существующему контейнеру

Далее создадим Dockerfile с помощью текстового редактора nano (рисунок 2.31).

Команда: nano dockerfile

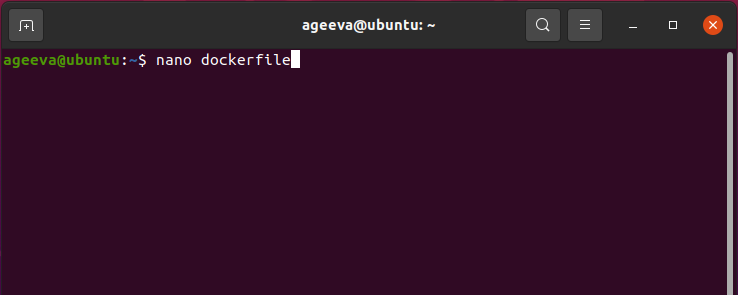


Рисунок 2.31 - Создание Dockerfile

После создания dockerfile откроется окно, где мы можем приступать к написанию кода (рисунок 2.32).

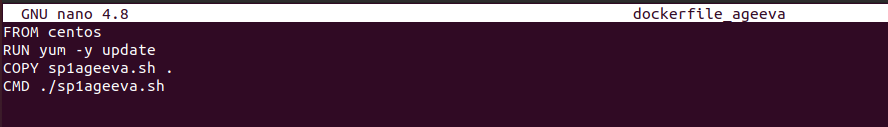


Рисунок 2.32 – Dockerfile

После чего стоит собрать образ (рисунок 2.33).

Команда: docker build -t container\_ageeva -f dockerfile\_ageeva .

Данная команда означает, что мы строим изображение контейнера (незапущенный контейнер) с названием container\_ageeva по инструкции из файла dockerfile\_ageeva ., рабочий каталог текущий (т.к. точка).

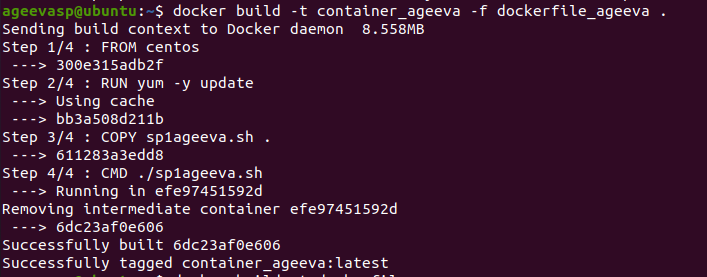


Рисунок 2.33 – Сборка образа

Команда: docker run –it «имя файла» (рисунок 2.34).

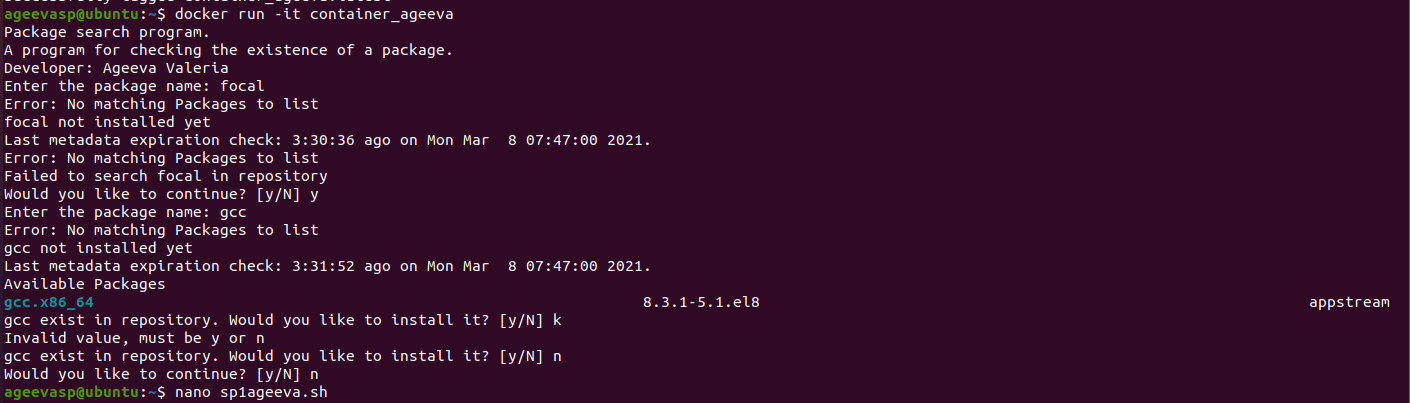
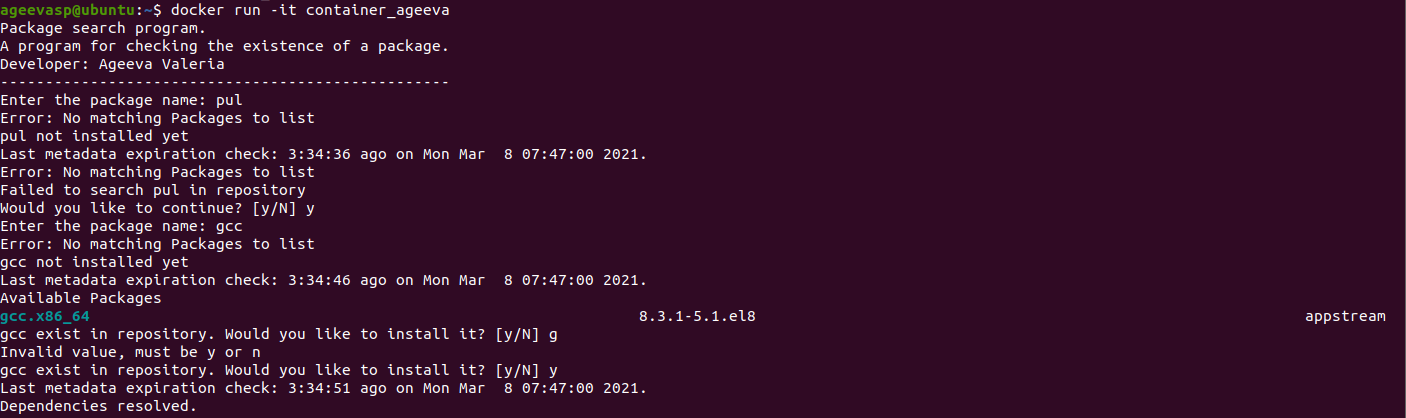
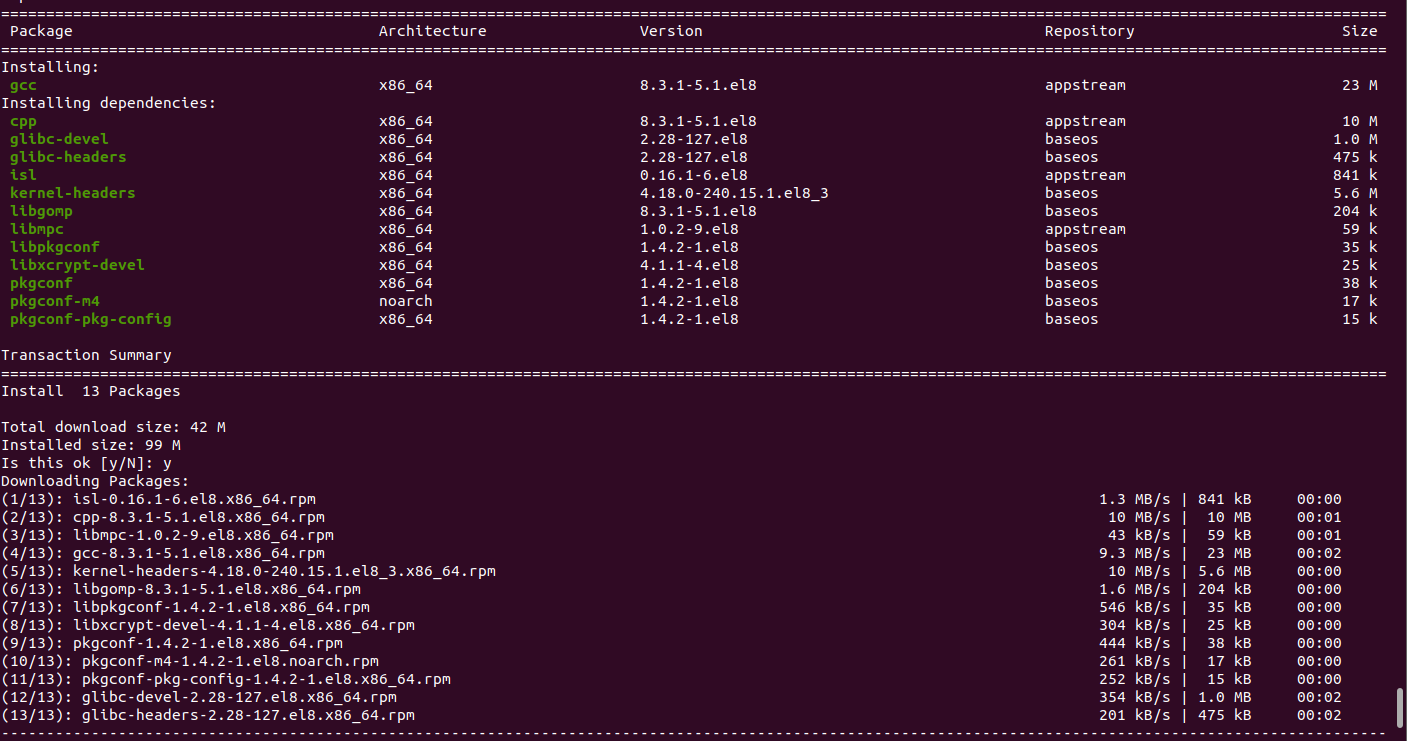
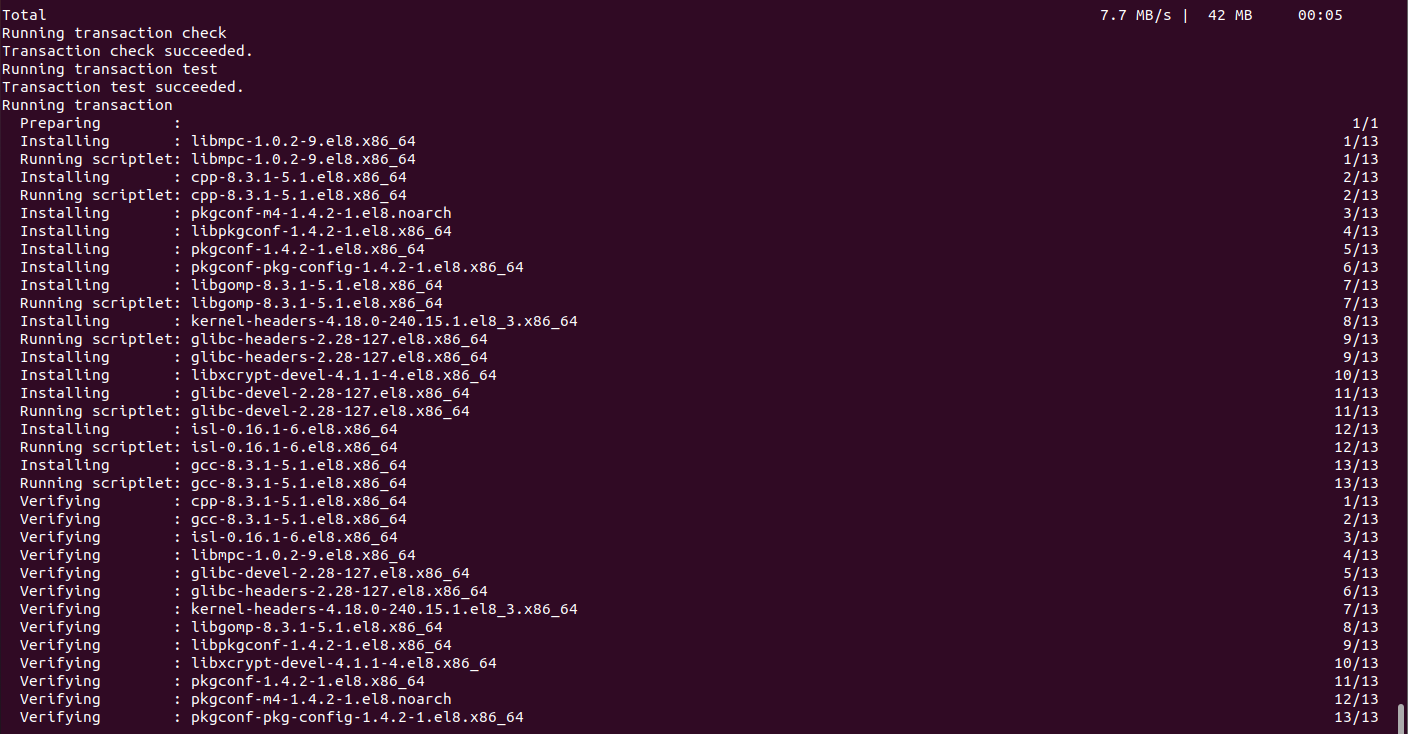


Рисунок 2.34 – Запуск контейнера и тестирование его работы







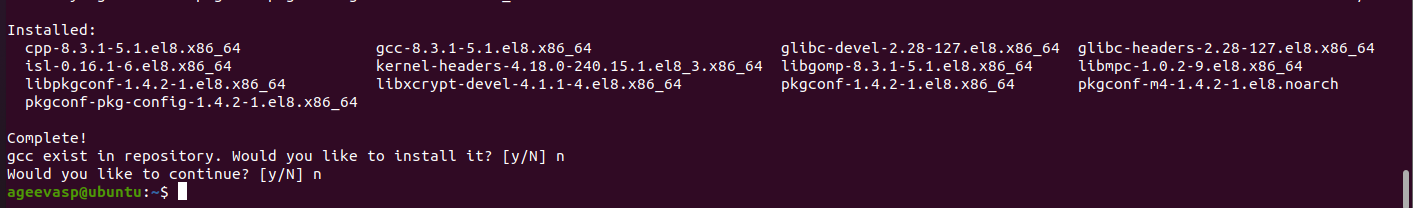


Рисунок 2.35 – Запуск контейнера и тестирование установки пакета

Рабочий скрипт и Dockerfile загружены на github (ссылка на репозиторий: https://github.com/7371avs/SP\_Lab1), а также приложены в архиве к отчету.

3 Заключение

В ходе лабораторной работы были получение навыки работы со средой контейнеризации ПО Docker и с командным интерпретатором bash. В ходе работы был написан скрипт и Dockerfile для воссоздания контейнера согласно варианту.