# Домашнее задание к занятию "5.4. Практические навыки работы с Docker"

## Задача 1

В данном задании вы научитесь изменять существующие Dockerfile, адаптируя их под нужный инфраструктурный стек.

Измените базовый образ предложенного Dockerfile на Arch Linux c сохранением его функциональности.

FROM ubuntu:latest

RUN apt-get update && \

apt-get install -y software-properties-common && \

add-apt-repository ppa:vincent-c/ponysay && \

apt-get update

RUN apt-get install -y ponysay

ENTRYPOINT ["/usr/bin/ponysay"]

CMD ["Hey, netology”]

**Ответ:   
ссылка на поняшу https://hub.docker.com/layers/151849744/dmitrykraska/ponysay-alpine/latest/images/sha256-15c6496647caf5f36c3eea8638e52a46d754fbbd200ac1233792b283fe7539ad?context=explore  
test@ubuntu:~/Docker$ docker build -t ponysayq1 -f dockerfile9 .**

Dockerfile

FROM alpine:latest

RUN apk update && \

apk add openssl python3 texinfo && \

wget -O ponysay.zip http://github.com/erkin/ponysay/archive/master.zip && \

unzip ponysay.zip && \

cd ponysay-master && \

./setup.py install --freedom=partial && \

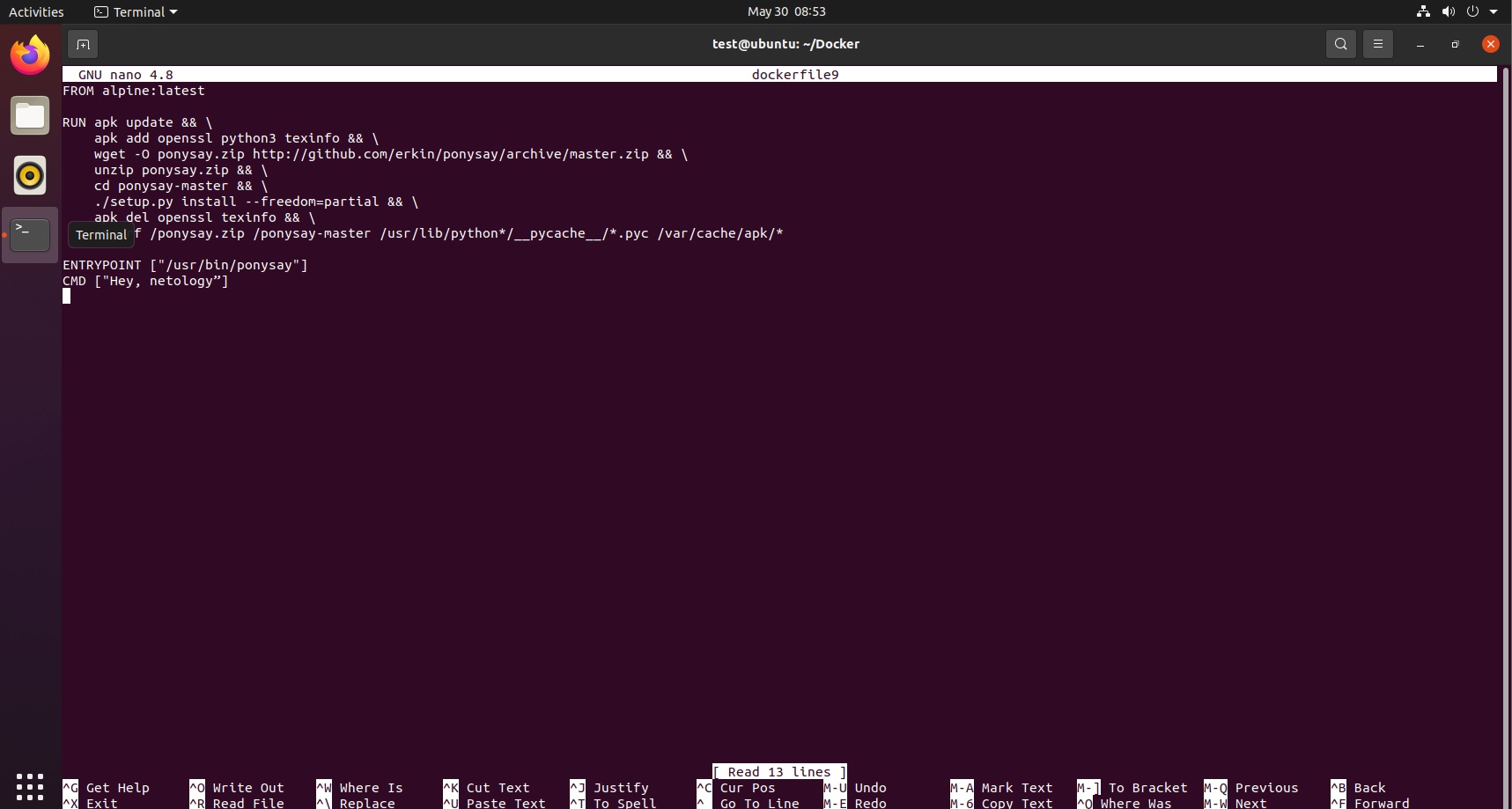
apk del openssl texinfo && \

rm -rf /ponysay.zip /ponysay-master /usr/lib/python\*/\_\_pycache\_\_/\*.pyc /var/cache/apk/\*

ENTRYPOINT ["/usr/bin/ponysay"]

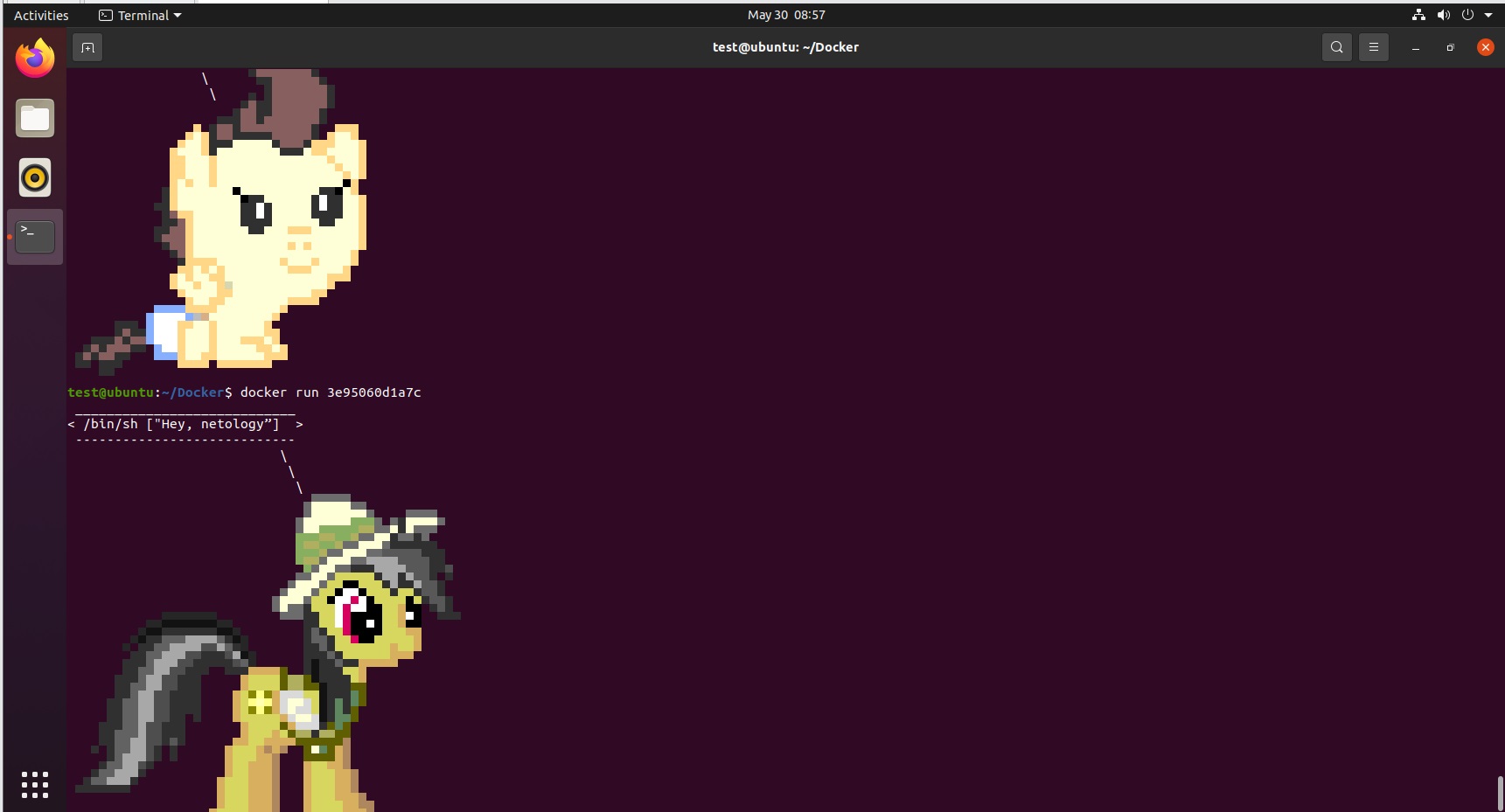
CMD ["Hey, netology”]

См. скрин: ponysay dockerfile “Домашнее задание к занятию 5.4. Практические навыки работы с Docker.jpg”



**test@ubuntu:~/Docker$ docker run 3e95060d1a7c**

**см. скрин “docker run ponysay Домашнее задание к занятию 5.4. Практические навыки работы с Docker.jpg”**



**Задача 2**

В данной задаче вы составите несколько разных Dockerfile для проекта Jenkins, опубликуем образ в dockerhub.io и посмотрим логи этих контейнеров.

* Составьте 2 Dockerfile:
  + Общие моменты:
    - Образ должен запускать [Jenkins server](https://www.jenkins.io/download/)
  + Спецификация первого образа:
    - Базовый образ - [amazoncorreto](https://hub.docker.com/_/amazoncorretto)
    - Присвоить образу тэг ver1
  + Спецификация второго образа:
    - Базовый образ - [ubuntu:latest](https://hub.docker.com/_/ubuntu)
    - Присвоить образу тэг ver2
* Соберите 2 образа по полученным Dockerfile
* Запустите и проверьте их работоспособность
* Опубликуйте образы в своём dockerhub.io хранилище

Ссылка на мой amazoncoretto

https://hub.docker.com/layers/dmitrykraska/amazoncorreto/v1/images/sha256-b13e99a7276bb09557ac32d26c0de2e48ef28a93e2bf2c5faa016a22510d7274?context=explore  
Запускаем [amazoncorretto](https://hub.docker.com/_/amazoncorretto).  
  
взят пример с офф сайта: см. ссылку https://www.jenkins.io/doc/book/installing/linux/#debianubuntu  
  
test@ubuntu:~/Docker$ sudo nano dockerfile1

FROM amazoncorretto

RUN yum -y update

RUN yum install java-1.8.0 -y

RUN yum update -y

RUN yum install wget -y

RUN yum install -y initscripts

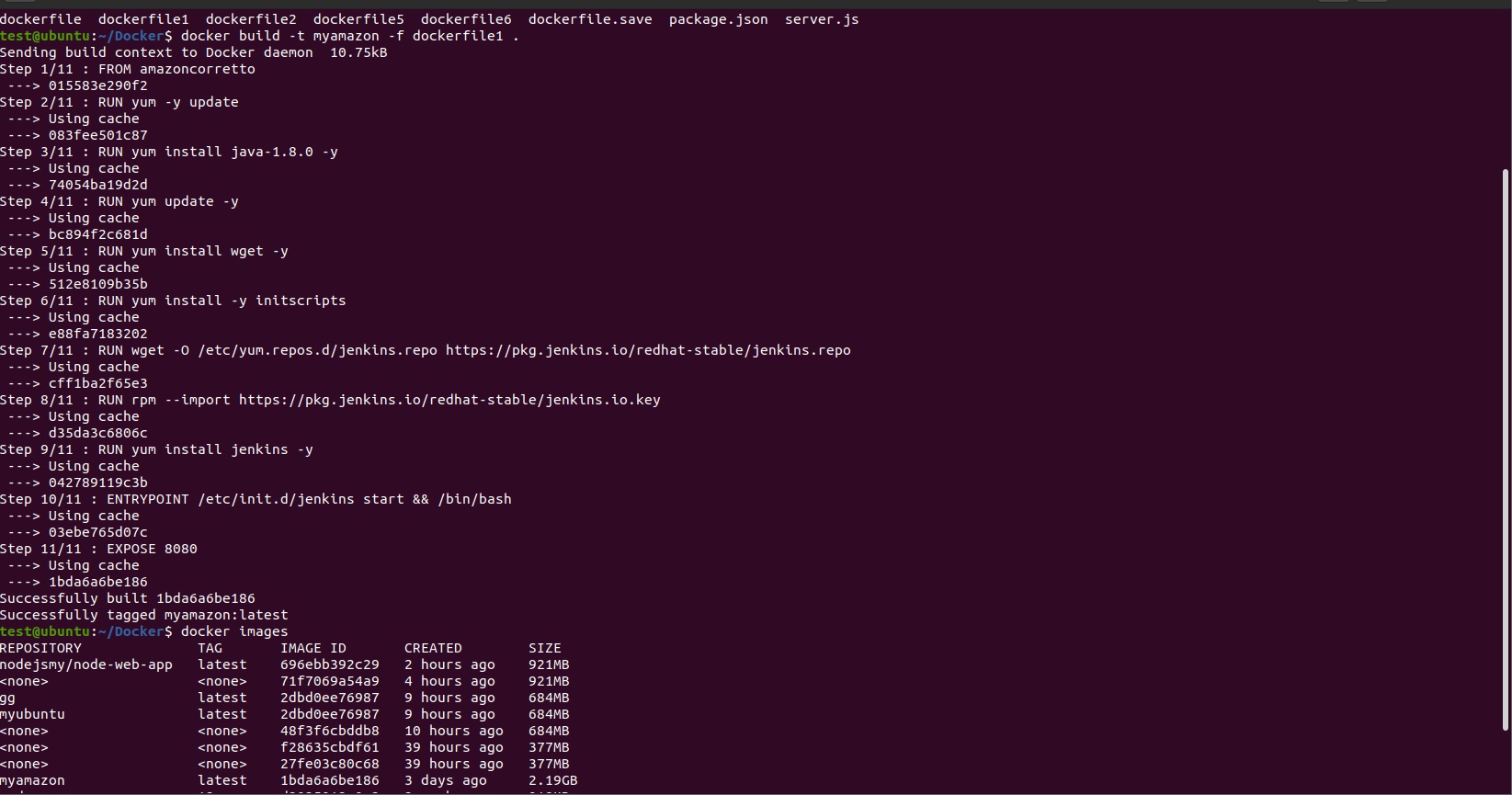
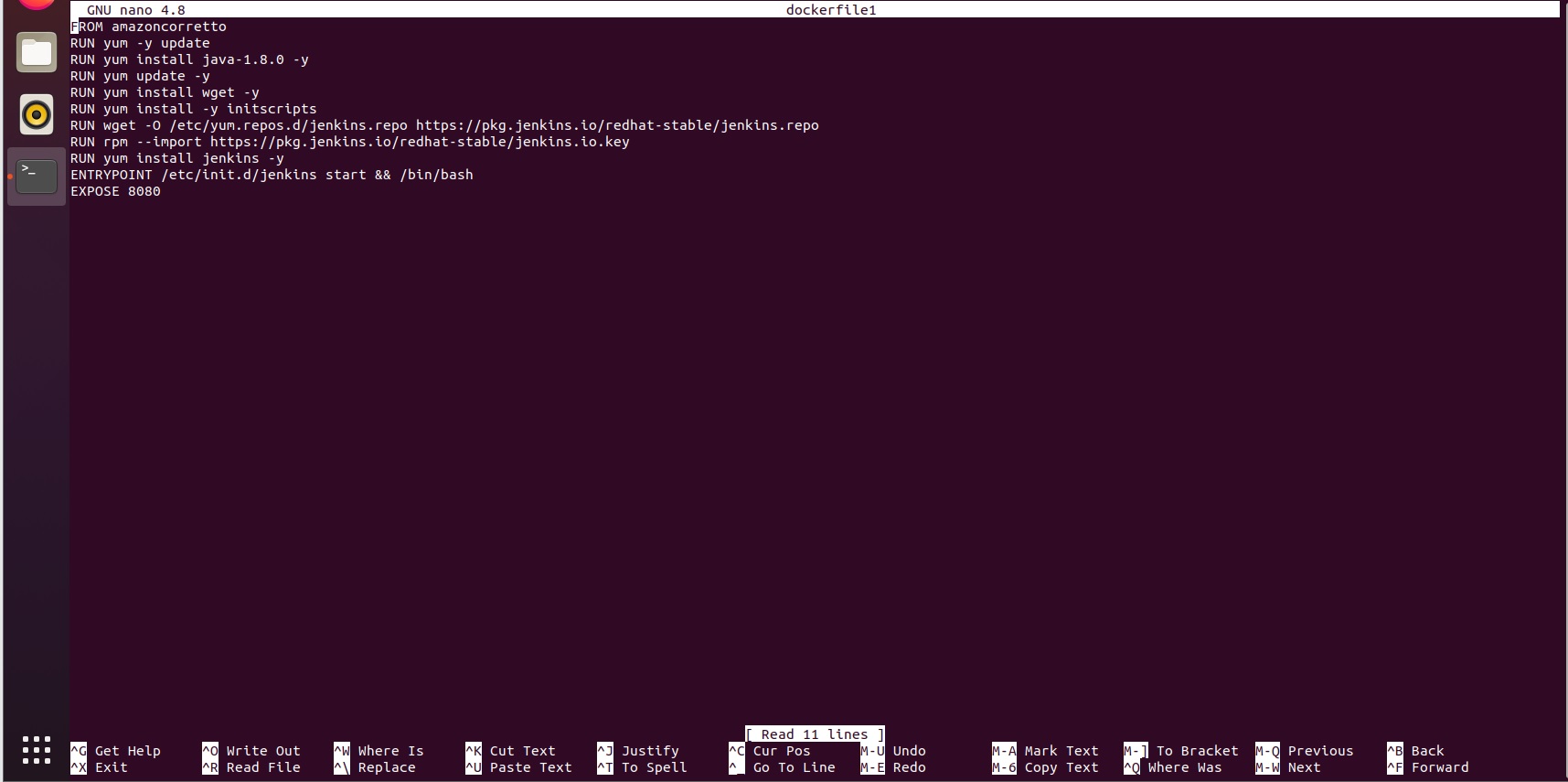
RUN wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/>

RUN rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key

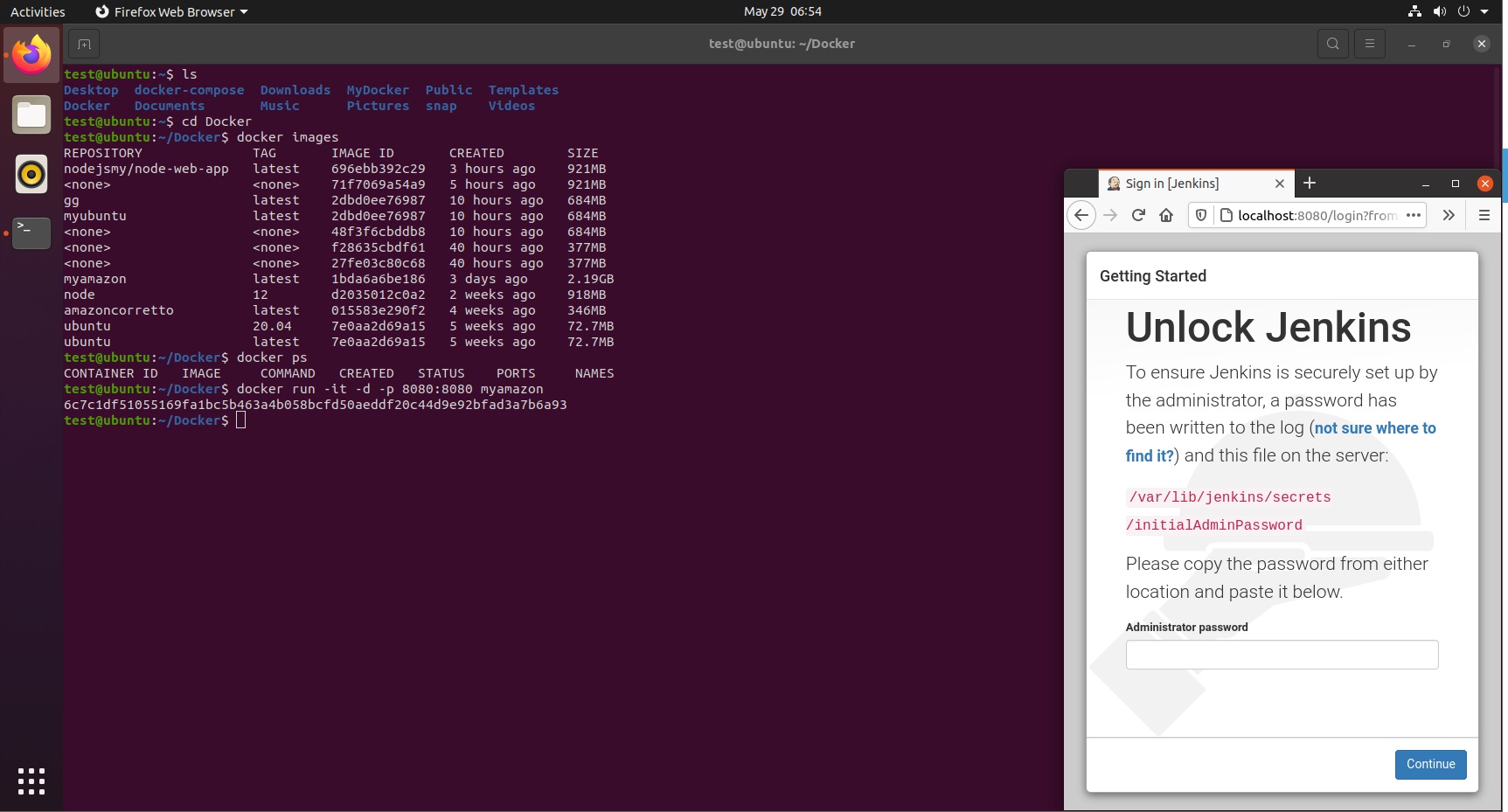
RUN yum install jenkins -y

ENTRYPOINT /etc/init.d/jenkins start && /bin/bash

EXPOSE 8080  
см. Скрин. “dockerfile amazoncoretto Домашнее задание к занятию 5.4. Практические навыки работы с Docker.jpg”  
  
test@ubuntu:~/Docker$ docker build -t myamazon -f dockerfile1 .  
см. скрин “amazon docker build Домашнее задание к занятию 5.4. Практические навыки работы с Docker”



test@ubuntu:~/Docker$ docker run -it -d -p 8080:8080



см. скрин “amazon run Домашнее задание к занятию 5.4. Практические навыки работы с Docker.jgp”

ссылка на образ убунты  
https://hub.docker.com/layers/dmitrykraska/ubuntu/v1/images/sha256-4b5063d5f603b8b7db75911d05162ae9ca698545ad9c451017c606ee17fa81f2?context=explore

Запускаем Ubuntu.  
  
test@ubuntu:~/Docker$ sudo nano dockerfile5  
  
FROM ubuntu:20.04

RUN apt-get -y update && apt-get install -y gnupg2

RUN apt-get update && apt-get install openjdk-8-jdk openjdk-8-jre -y

RUN apt-get install wget -y

RUN wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | apt-key add -

RUN sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'

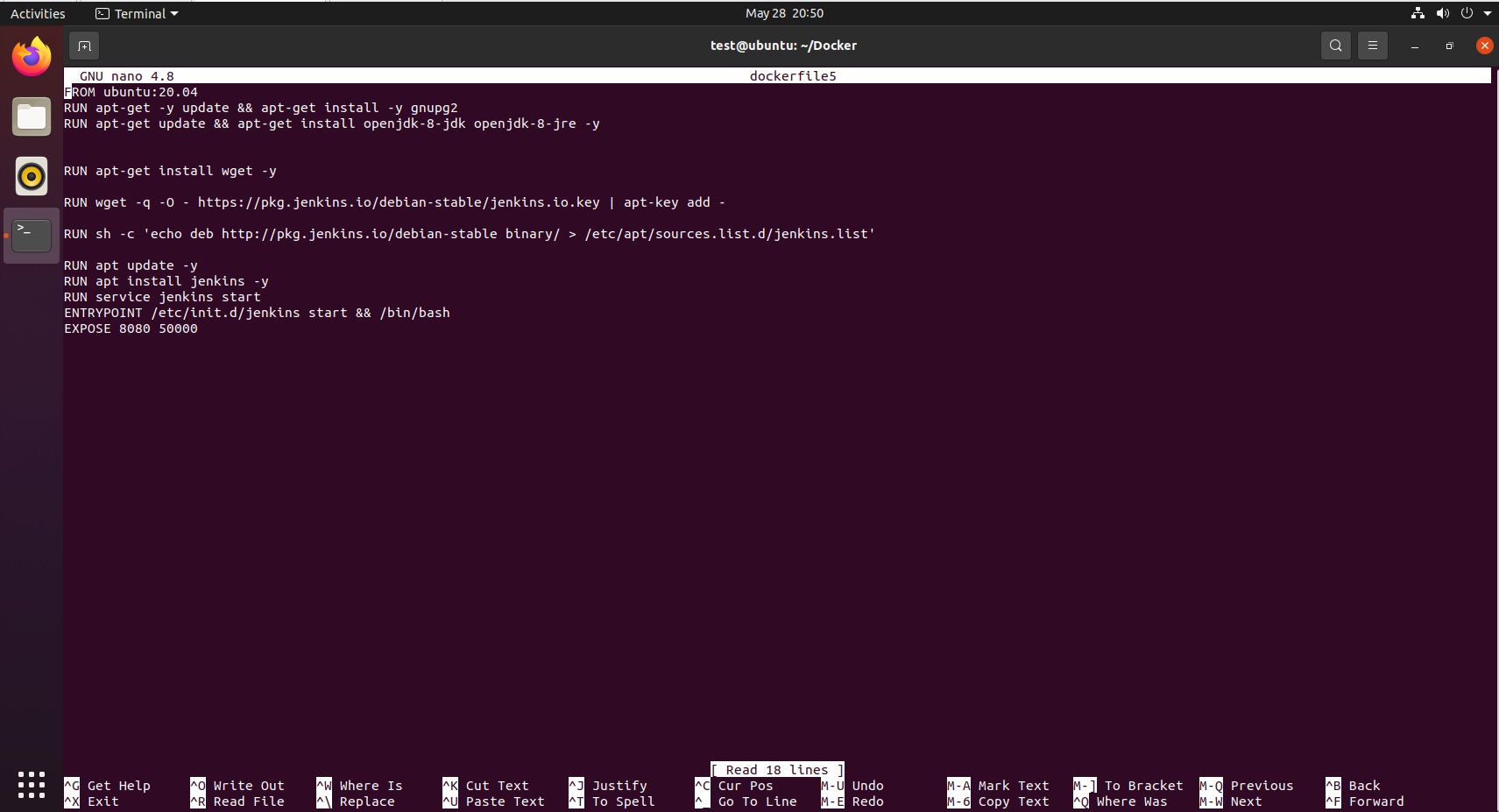
RUN apt update -y

RUN apt install jenkins -y

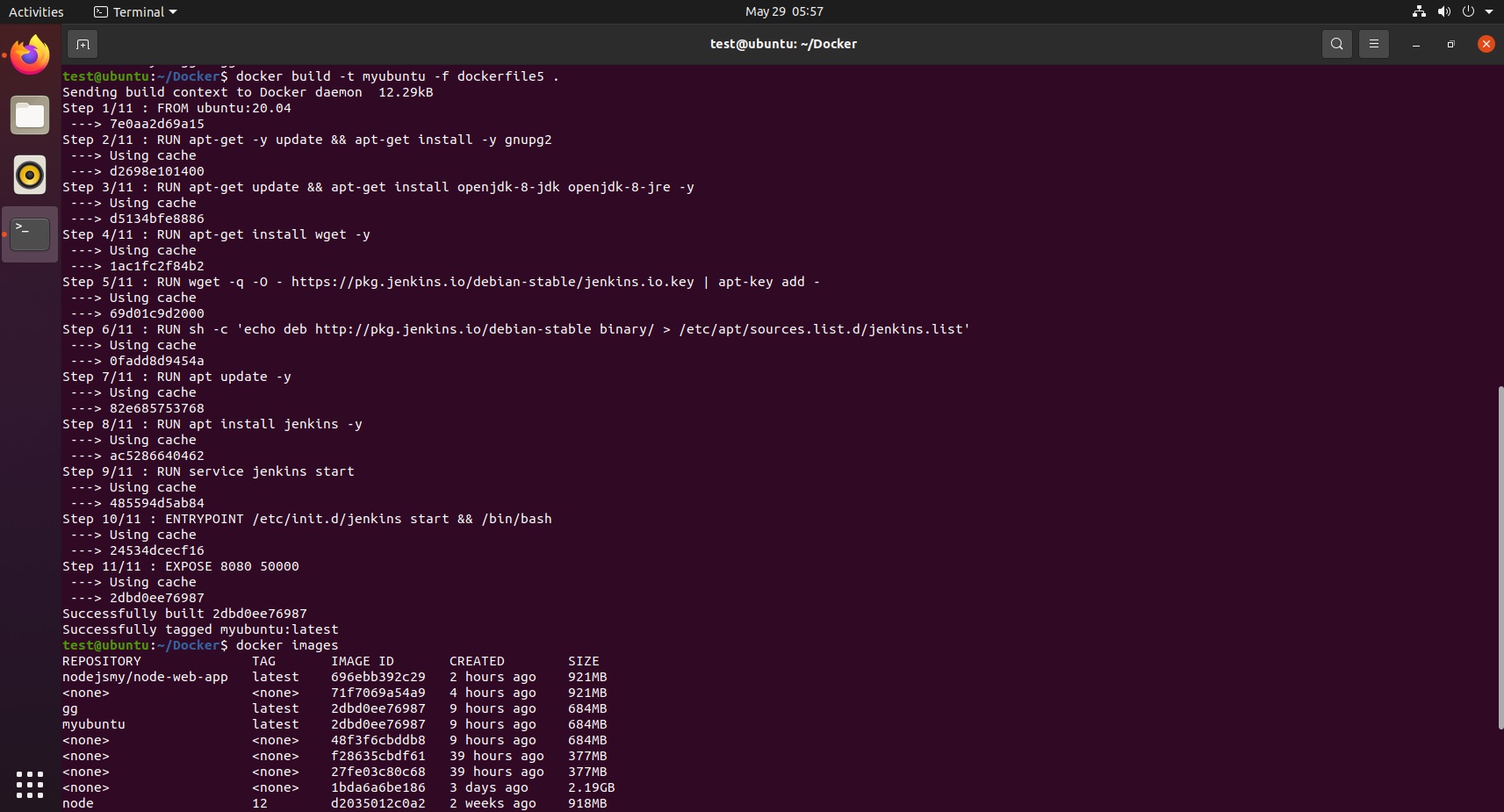
RUN service jenkins start

ENTRYPOINT /etc/init.d/jenkins start && /bin/bash

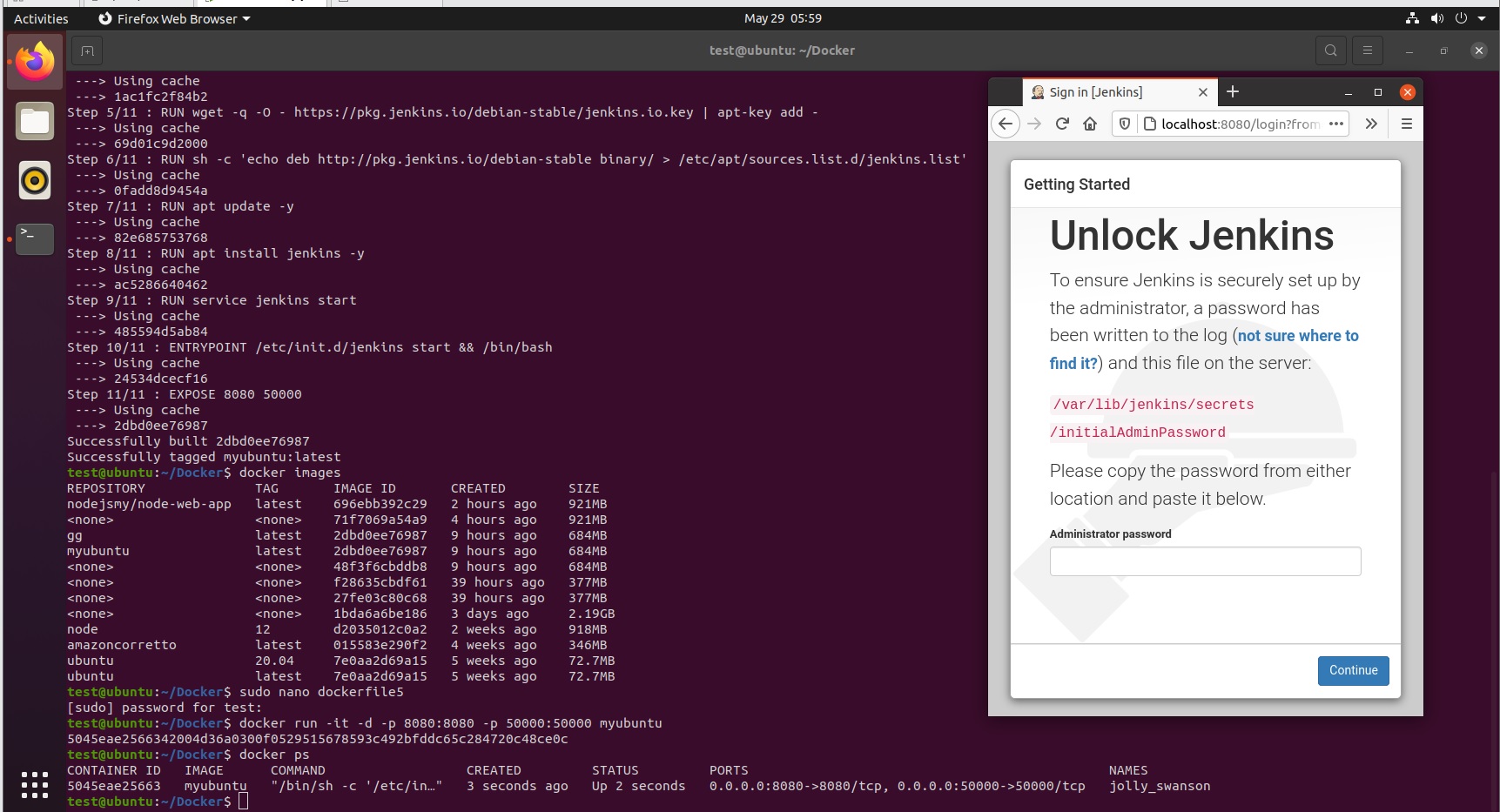
EXPOSE 8080 50000  
  
см. скрин “Ubuntu Домашнее задание к занятию 5.4. Практические навыки работы с Docker.jpg”



**test@ubuntu:~/Docker$ docker build -t myubuntu -f dockerfile5 .  
см. Скрин “dockerbuild ubuntu Домашнее задание к занятию 5.4. Практические навыки работы с Docker.jpg”  
  
test@ubuntu:~/Docker$ docker run -it -d -p 8080:8080 -p 50000:50000 myubuntu**



**см. скрин “Ubuntu Jenkins up Домашнее задание к занятию 5.4. Практические навыки работы с Docker.jpg”  
  
Ссылка на hub.docker.com ubuntu**[**https://hub.docker.com/r/dmitrykraska/ubuntu/tags?page=1&ordering=last\_updated**](https://hub.docker.com/r/dmitrykraska/ubuntu/tags?page=1&ordering=last_updated) **Ссылка на hub.docker.com amazoncorreto  
https://hub.docker.com/r/dmitrykraska/amazoncorreto/tags?page=1&ordering=last\_updated  
  
  
Задача 3**



В данном задании вы научитесь:

объединять контейнеры в единую сеть

исполнять команды "изнутри" контейнера

Для выполнения задания вам нужно:

Написать Dockerfile:

Использовать образ <https://hub.docker.com/_/node> как базовый

Установить необходимые зависимые библиотеки для запуска npm приложения <https://github.com/simplicitesoftware/nodejs-demo>

Выставить у приложения (и контейнера) порт 3000 для прослушки входящих запросов

Соберите образ и запустите контейнер в фоновом режиме с публикацией порта

Запустить второй контейнер из образа ubuntu:latest

Создайть docker network и добавьте в нее оба запущенных контейнера

Используя docker exec запустить командную строку контейнера ubuntu в интерактивном режиме

Используя утилиту curl вызвать путь / контейнера с npm приложением

**Ответ:**Делаем ДЗ по официальному гайду **https://nodejs.org/ru/docs/guides/nodejs-docker-webapp/**

sudo nano package.json #   
  
  
{

"name": "docker\_web\_app",

"version": "1.0.0",

"description": "node.js on docker",

"author": "first last <first.last@example.com>",

"main": "server.js",

"scripts": {

"start": "node server.js"

},

"dependencies": {

"express": "^4.16.1"

}

}

sudo nano server.js # создаем файл server.js, который определяет веб-приложение на основе фреймворка Express.js

Примечание

const port = 3000;

поменяли port с 8080 на 3000

'use strict';

const express = require('express');

// константы

const port = 3000;

const host = '0.0.0.0';

// приложение

const app = express();

app.get('/', (req, res) => {

res.send('Hello World');

});

app.listen(port, host);

console.log(`running on http://${host}:${port}`);

docker build . -t nodejsmy/node-web-app # билдим образ

см. скрин “3 задание dockerfile nodejsДомашнее задание к занятию 5.4. Практические навыки работы с Docker.jpg”

dockerfile  
FROM node:12

# создание директории приложения

WORKDIR /usr/src/app

# установка зависимостей

# символ астериск ("\*") используется для того чтобы по возможности

# скопировать оба файла: package.json и package-lock.json

COPY package\*.json ./

RUN npm install

# Если вы создаете сборку для продакшн

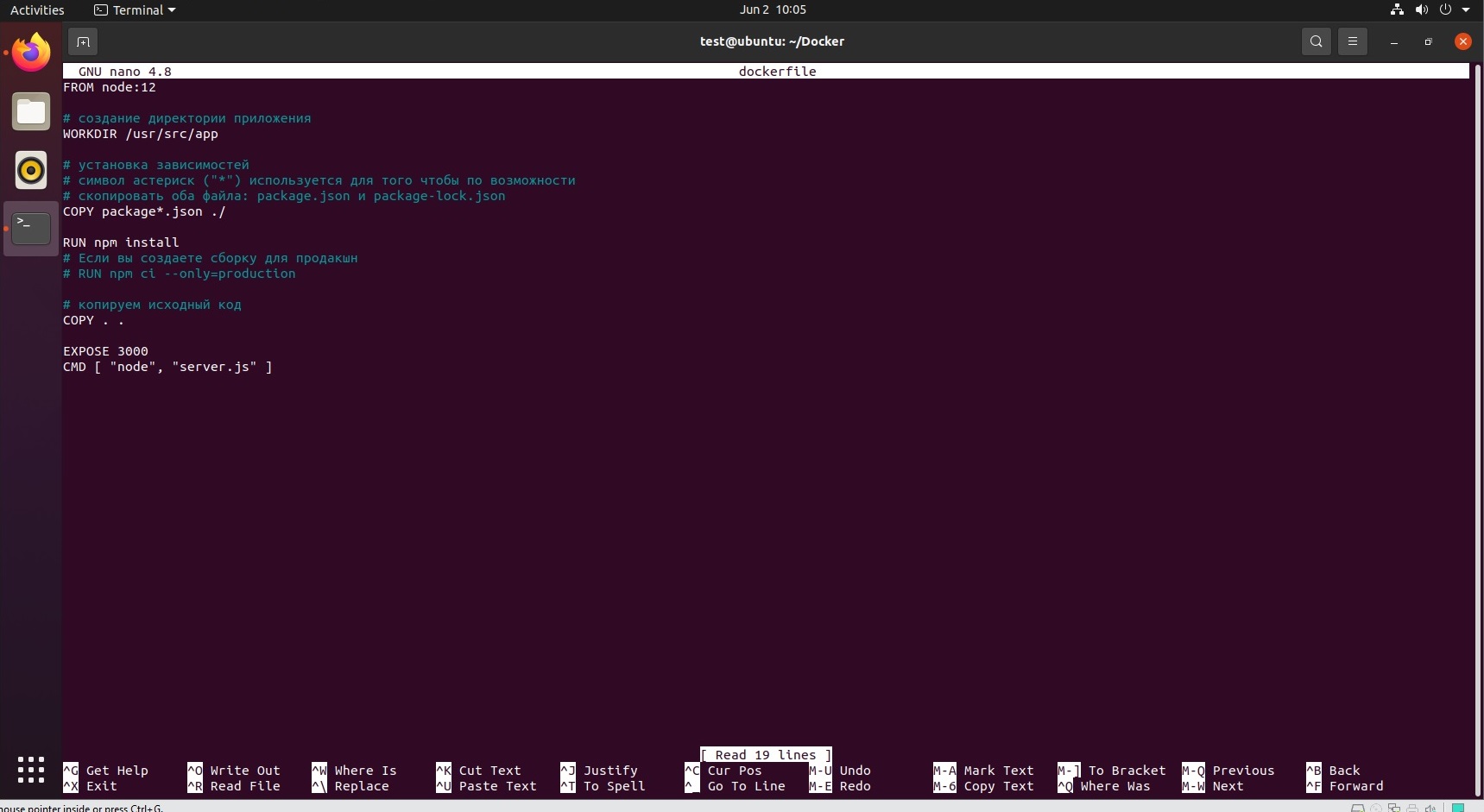
# RUN npm ci --only=production

# копируем исходный код

COPY . .

EXPOSE 3000

CMD [ "node", "server.js" ]



test@ubuntu:~/Docker$ sudo nano .dockerignore

node\_modules

npm-debug.log

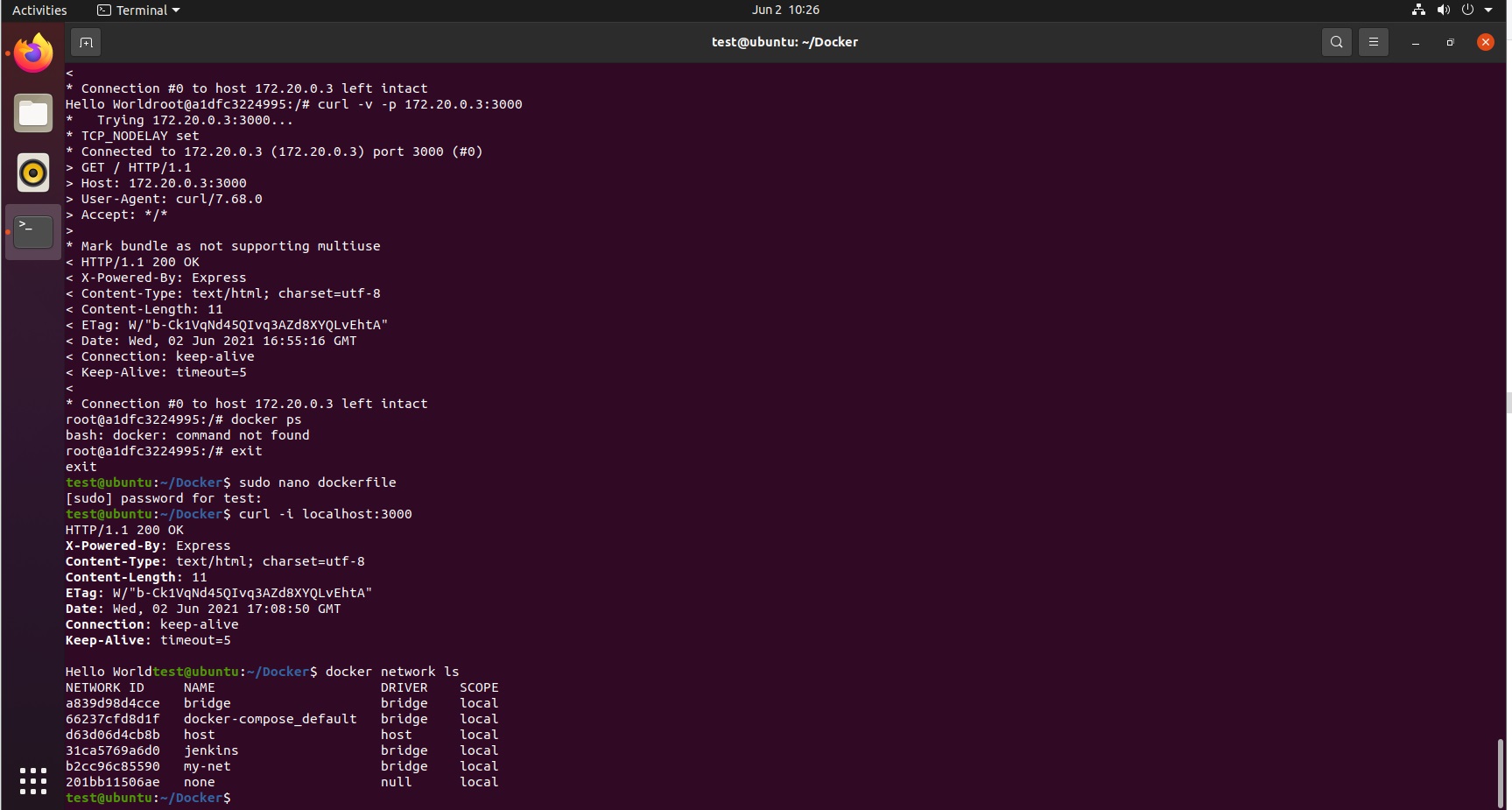
test@ubuntu:~/Docker$ docker build -t mynodejs -f dockerfile .  
test@ubuntu:~/Docker$ docker run -d -ti -p 3000:3000 75017866ac81 /bin/bash #поднимаем контейнер с node.js  
test@ubuntu:~/Docker$ docker run -d -ti ubuntu  
# поднимаем контейнер с ubuntu  
test@ubuntu:~/Docker$ docker exec -ti 0840b7af5d98 /bin/bash

test@ubuntu:~/Docker$ docker exec -ti 0840b7af5d98 /bin/bash

test@ubuntu:~/Docker$ curl -i localhost:3000 # смотрим работаем ли node.js

test@ubuntu:~/Docker$ docker network create my-net # создаем сетевой мост с названием my-net

test@ubuntu:~/Docker$ docker network ls # смотрим сетевые мосты  
см. скрин “curl общий Домашнее задание к занятию 5.4. Практические навыки работы с Docker.jpg”   
  
  
test@ubuntu:~/Docker$ docker build . -t nodejsmy/node-web-app # билдим образ  
  
test@ubuntu:~/Docker$ docker run -p 3000:3000 -d --network b2cc96c85590 nodejsmy/node-web-app #запускаем контейнер nodejsmy/node-web-app все как по ссылке можно сделать –network сейчас или позже docker connect  
  
ссылка оффициальная <https://nodejs.org/ru/docs/guides/nodejs-docker-webapp/>  
  
  
test@ubuntu:~/Docker$ curl -i localhost:3000 # тестим работает ли node.js



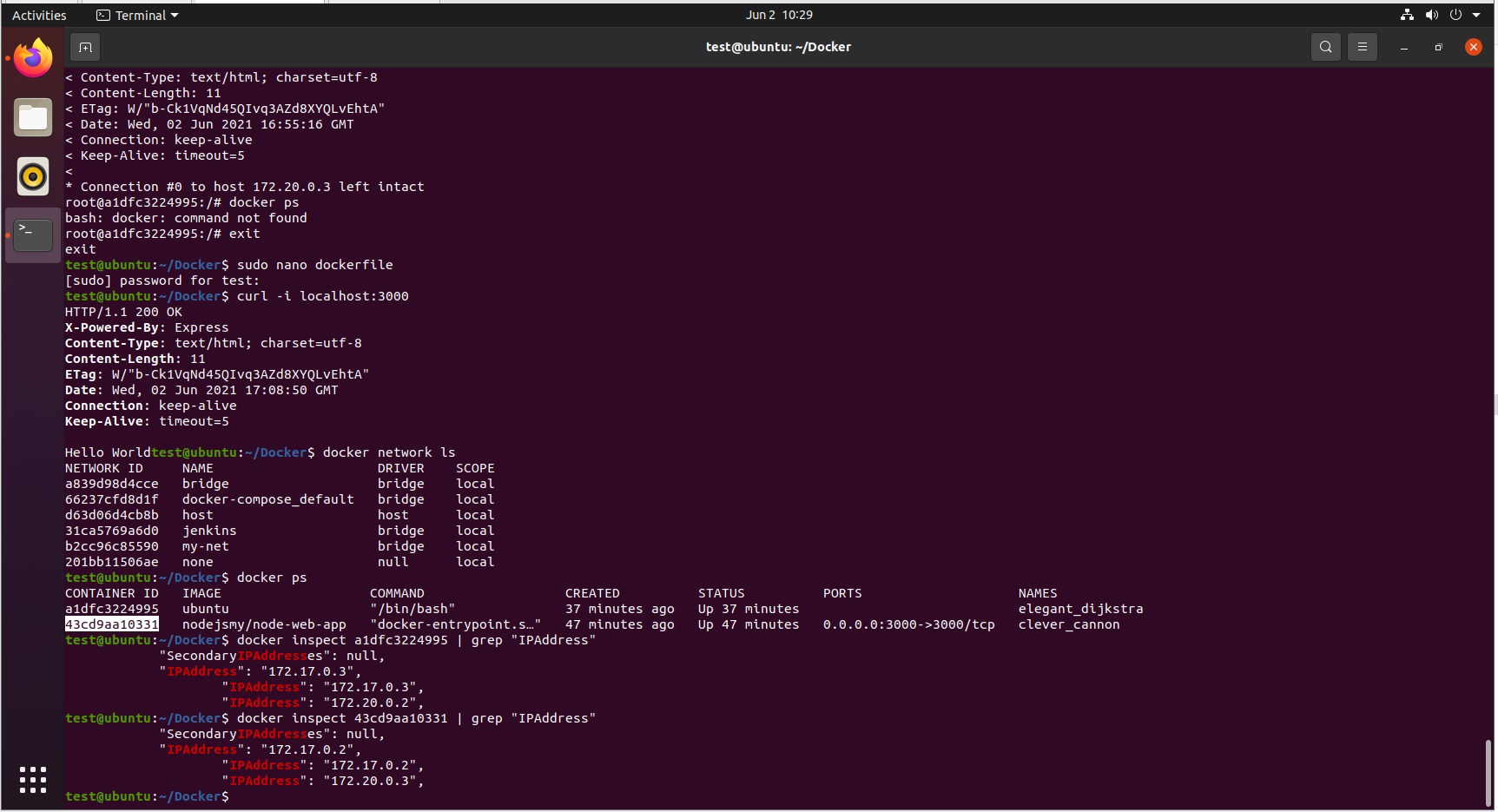
test@ubuntu:~/Docker$ docker run -d –ti ubuntu # запускаем контейнер с убунтой  
  
docker network disconnect id\_container # если хотим отключить от лишних сетей  
test@ubuntu:~/Docker$ docker network connect my-net 8d32c725d7fd # подключаем контейнер к сети my-net

test@ubuntu:~/Docker$ docker network connect my-net c0e57534a371 # подключаем контейнер к сети my-net

test@ubuntu:~/Docker$ docker inspect c0e57534a371 | grep "IPAddress" # узнаем IP контейнера

test@ubuntu:~/Docker$ docker inspect 8d32c725d7fd | grep "IPAddress" # узнаем IP контейнера

см. скрин “ip containers Домашнее задание к занятию 5.4. Практические навыки работы с Docker.jpg”



test@ubuntu:~/Docker$ docker exec -ti c0e57534a371 /bin/bash # залетаем в контейнер

root@c0e57534a371:/# apt-get update; apt-get install curl -y# устанавливаем curl  
  
root@c0e57534a371:/# curl -v -p 172.20.0.3:3000 # проверяем пинг в другой контейнер, где -v подробно (открыли по заданию в другом контейнере).

См. скрин “docker nodejs final curl Домашнее задание к занятию 5.4. Практические навыки работы с Docker.jpg”