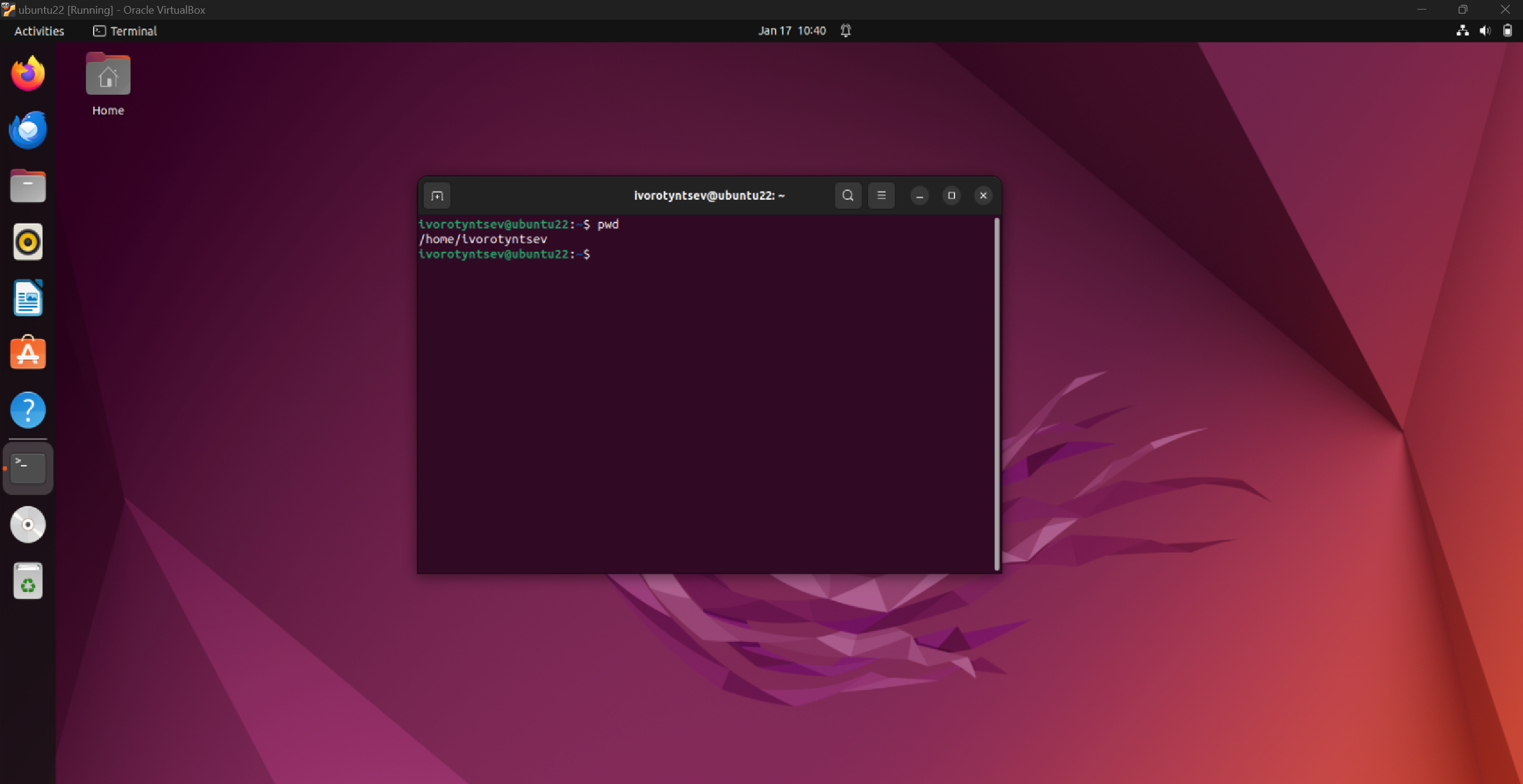
**DevOps Final Project**

**Ilya Vorotyntsev 324755734 (0584050797)**

**Aleksandr Pitkin 333879013**

**Lecture 1:**

Screenshot from installed Ubuntu on VirtualBox:



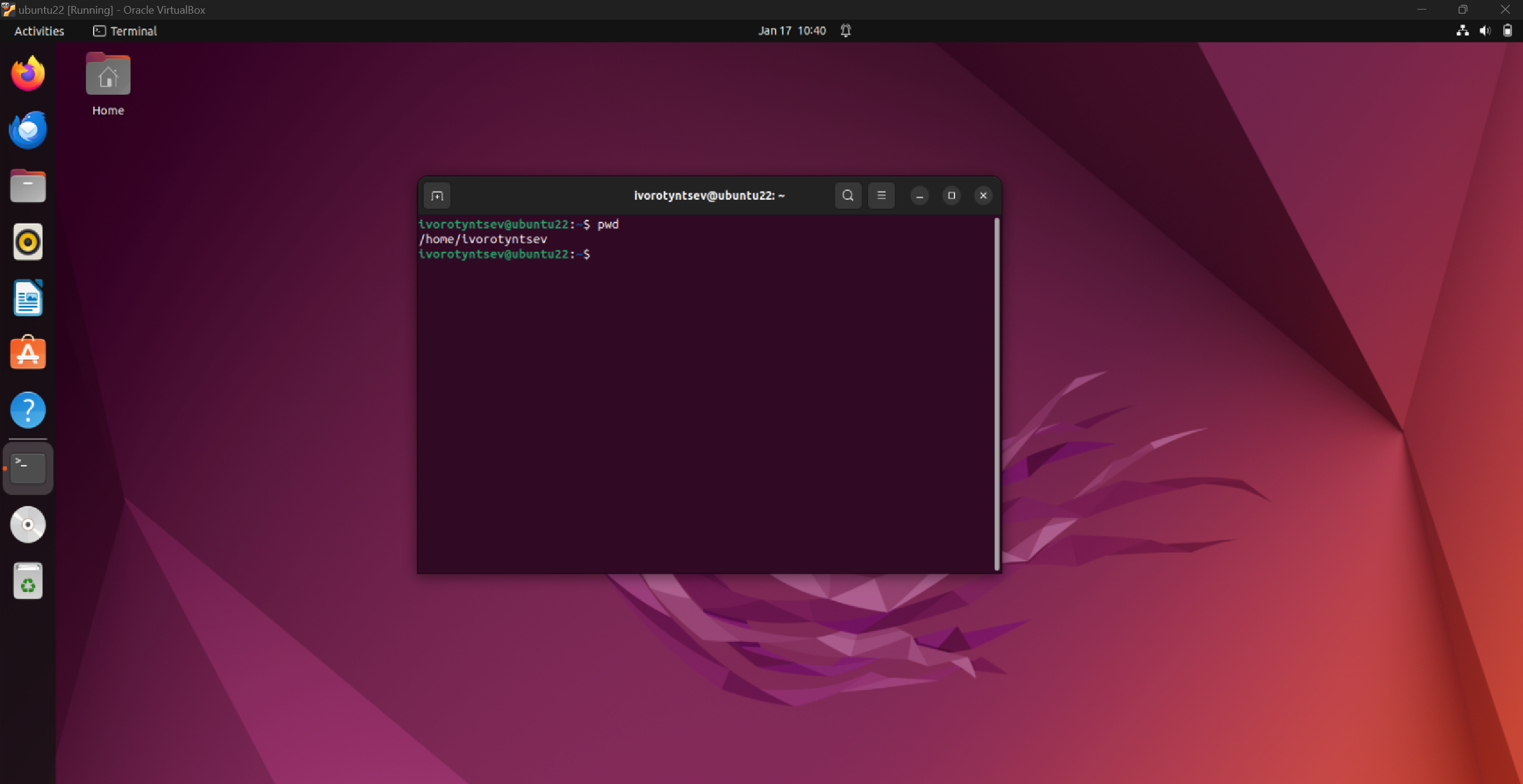
**Lecture 2:**

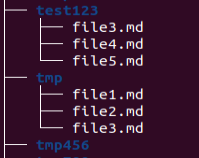
Questions:

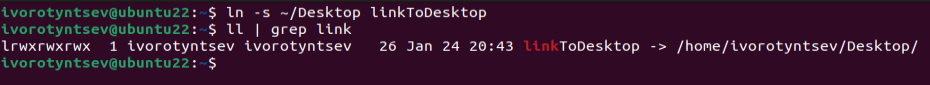
1. what is your username?
2. what is your machine name?
3. what is your current directory?

Answers:

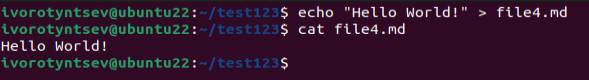
1. ivorotyntsev
2. ubuntu22
3. /home/ivorotyntsev



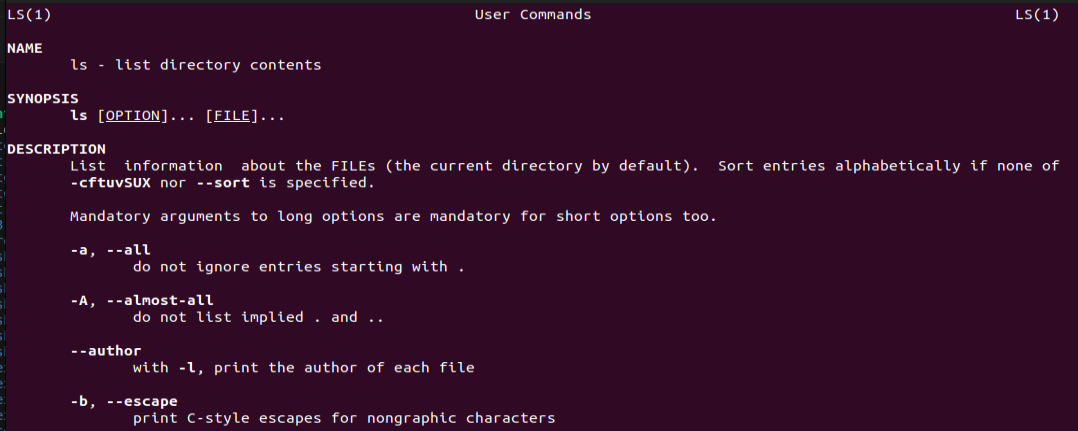
Use *mkdir + touch* commands for creating files and directories:  
  


Creating soft link using *ln* command:  


Append text to file and print it using *cat*:

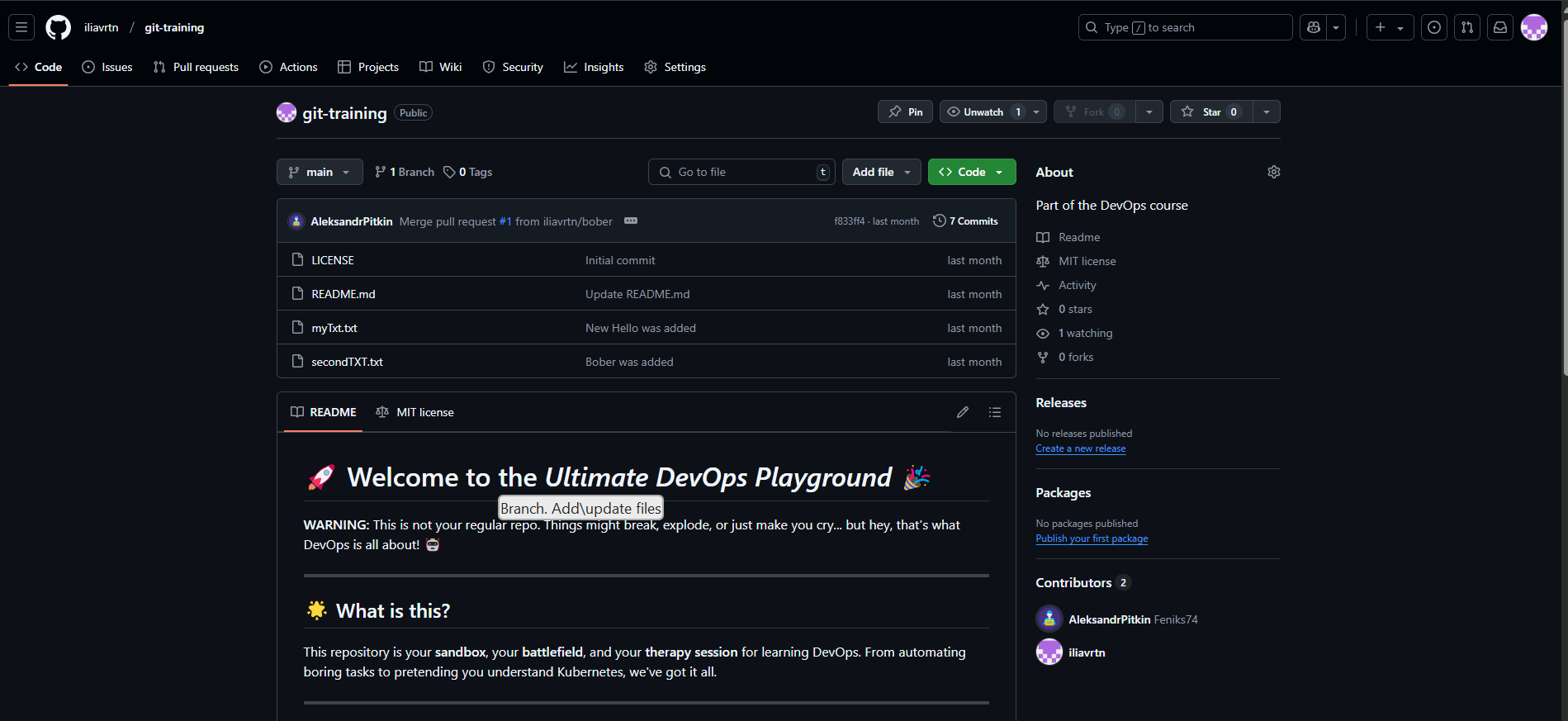


Read manual for *ls* command using *man*:



**Lecture 3:**

Created repository called git-training with some files pushed after commit. For addition, added couple of contributors for the project.



Pull request was created by one of the contributors, and the other one approved the request.

A screenshot of a computer

Description automatically generated

Changes that was done in the file:

A screenshot of a computer

Description automatically generated

**Why We Need Dependabot?**

We need Dependabot because it automates the process of keeping our project dependencies up-to-date. This saves us from the tedious task of manually checking for updates and helps prevent the use of outdated or vulnerable packages.

**When to Use Dependabot?**

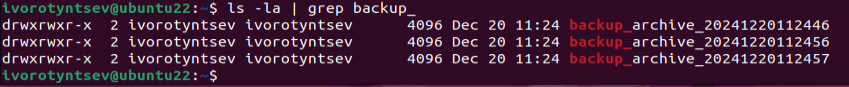
We can integrate Dependabot into our GitHub repositories to continuously monitor and update dependencies. It is especially useful when new versions or security patches are released, ensuring our projects stay current and secure.

**Why It Is So Important to Have Dependabot?**

Dependabot is important because it enhances project security by promptly addressing vulnerabilities. It ensures compatibility with the latest software standards and maintains the overall health of the codebase, allowing developers to focus more on building features rather than managing dependencies.

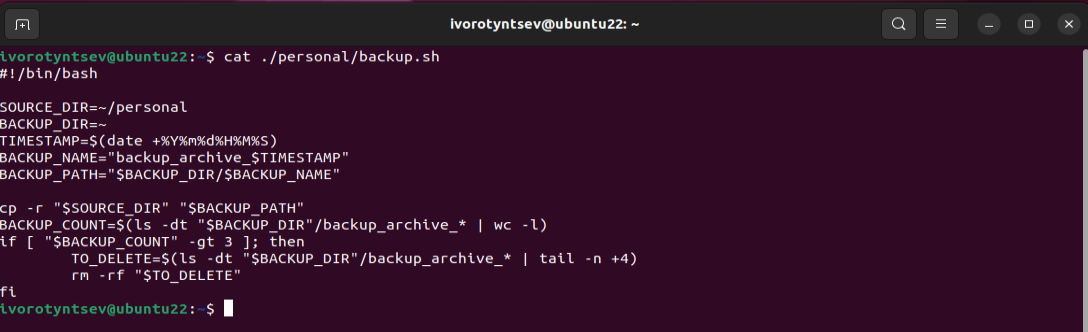
**Lecture 4:**

The result after running *backup.sh* script:



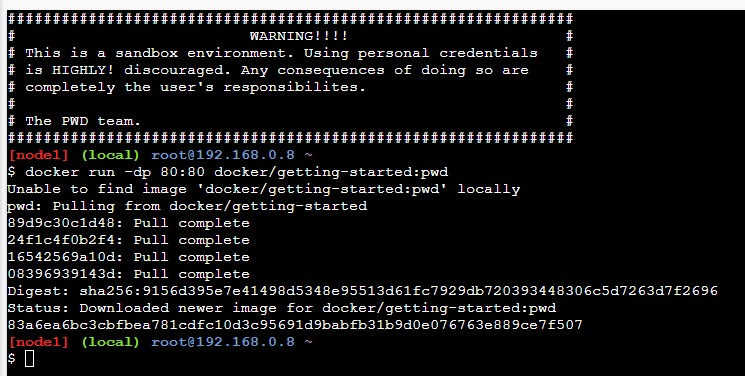
The script works as follows:

1. Tells the system which shell interpreter to use (*/bin/bash)*.
2. Defines the folder to back up *(~/personal*).
3. Defines the main location where backups should be stored (~).
4. Creates a unique time-based label (*YYYYMMDDHHMMSS*).
5. Combines a prefix and the time stamp to form the backup folder name.
6. Builds the full path to the new backup folder.
7. Recursively copies the chosen folder into the new backup location.
8. Counts how many backups currently exist by listing them and measuring the total lines.
9. Checks if there are more than three backups—if yes, deletes all the extras beyond the third newest.



**Lecture 5:**

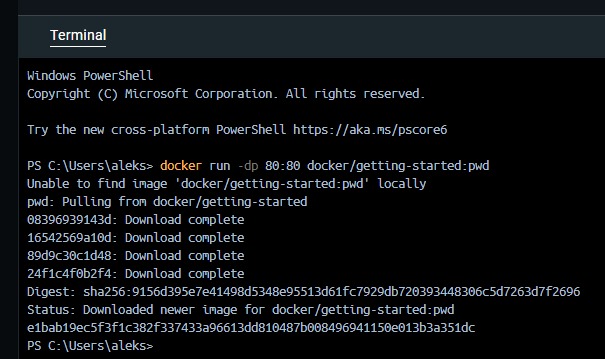
Docker playground:



After running a container:  
A screenshot of a computer

Description automatically generated

Docker Desktop:



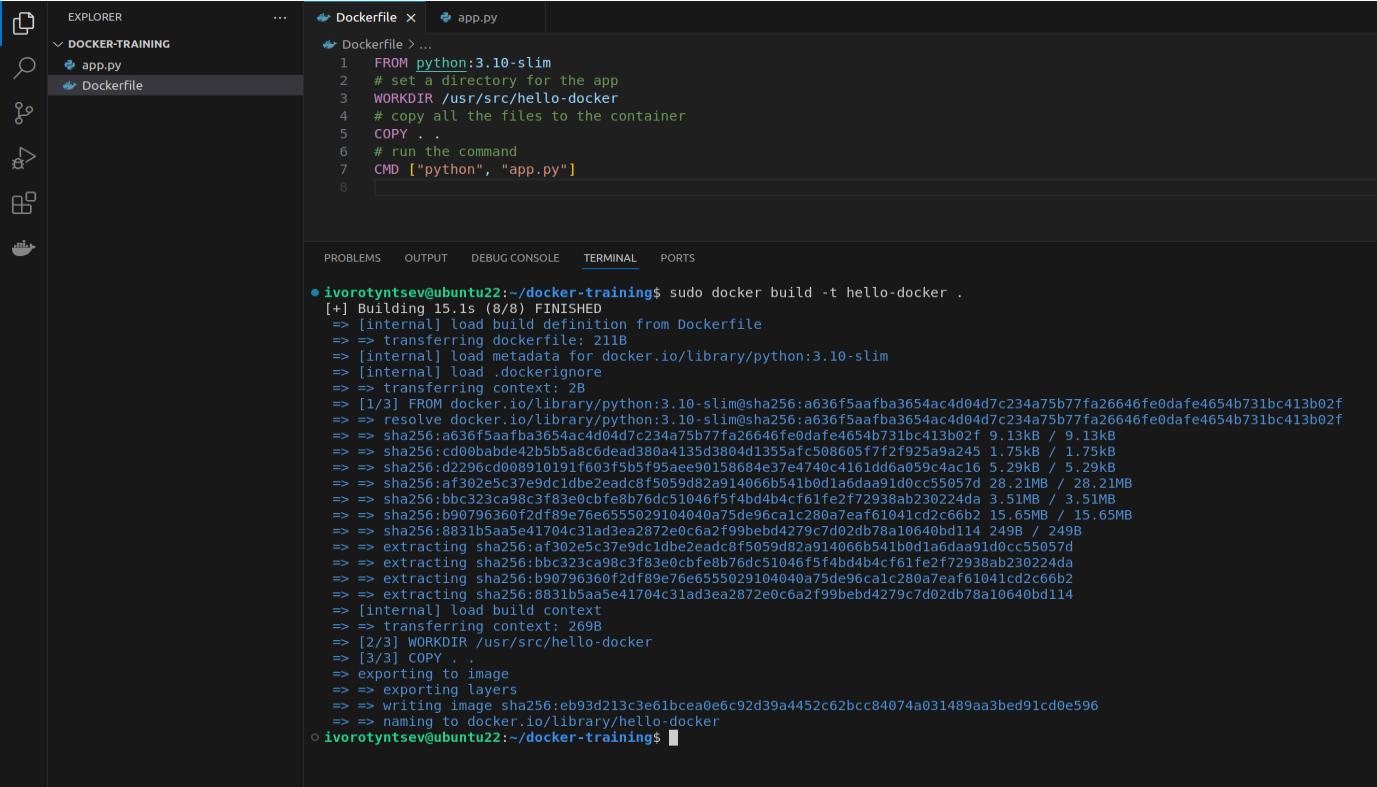
After running the container:

A screenshot of a computer

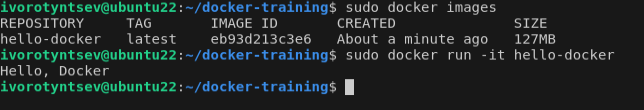
Description automatically generated

Simple Python script running in container:

1. Build the image



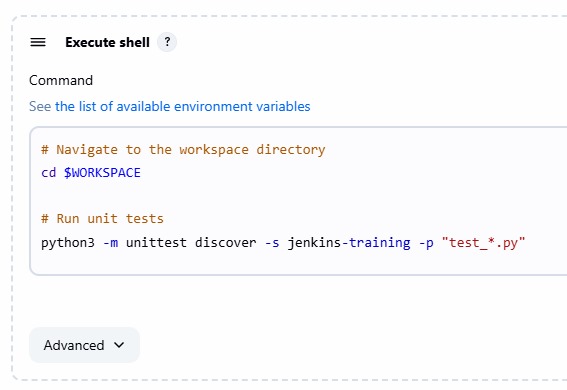
1. Run the container:



**Lecture 6:**

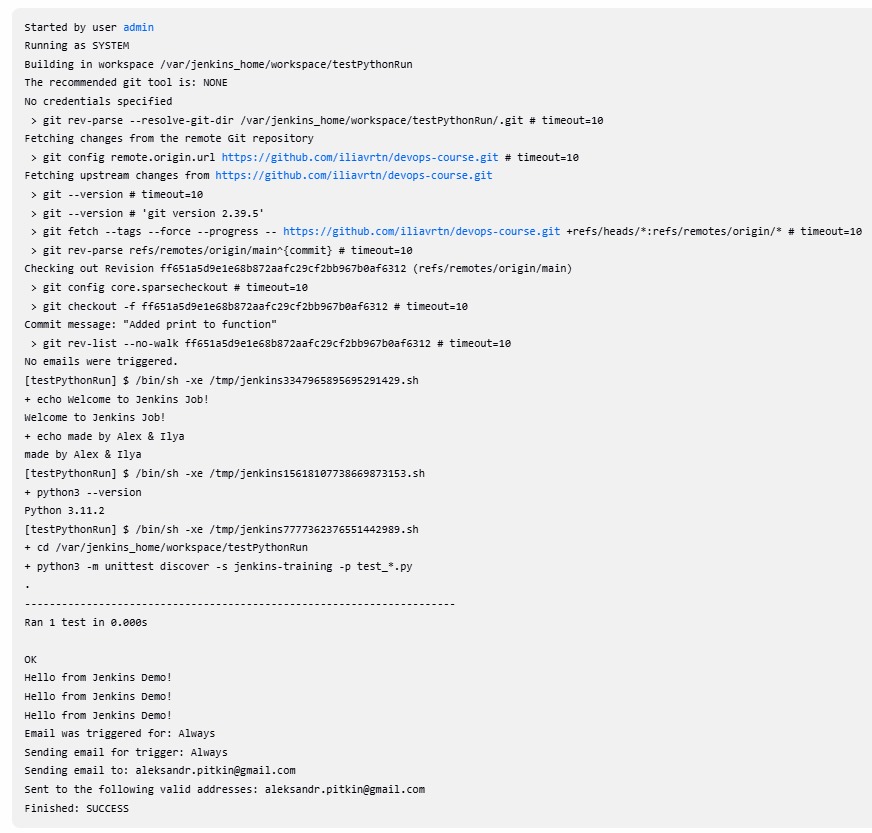
Ran Jenkins docker container and configured Jenkins job:A screenshot of a chat

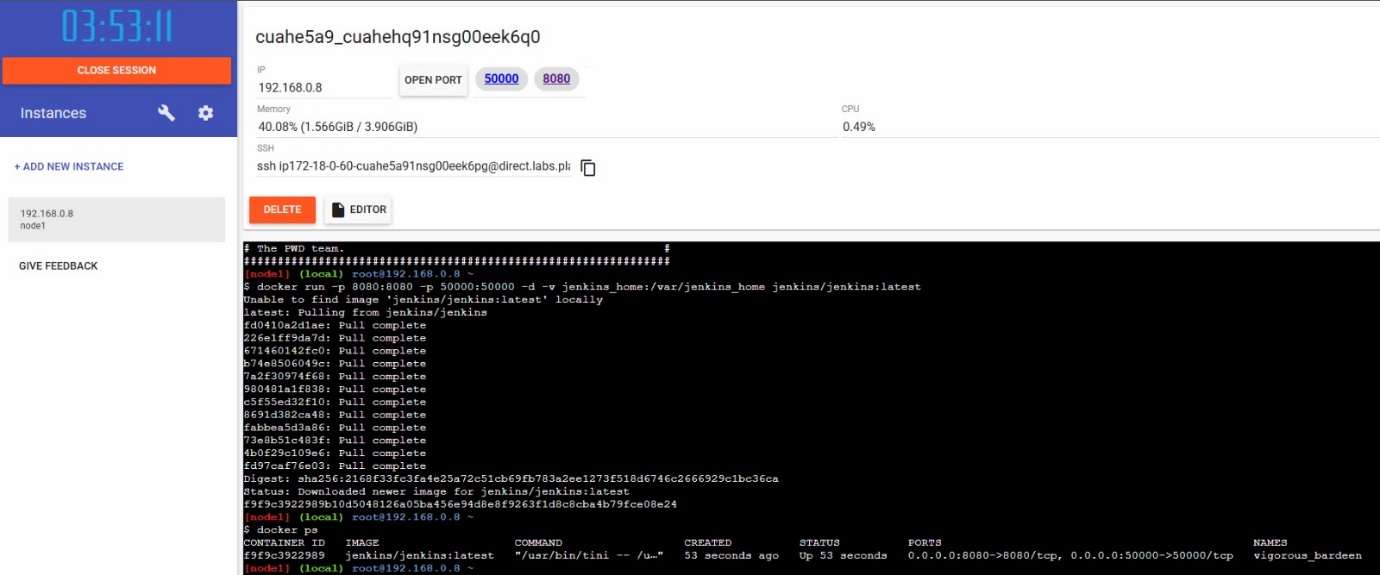
Description automatically generated



A screenshot of a email notification

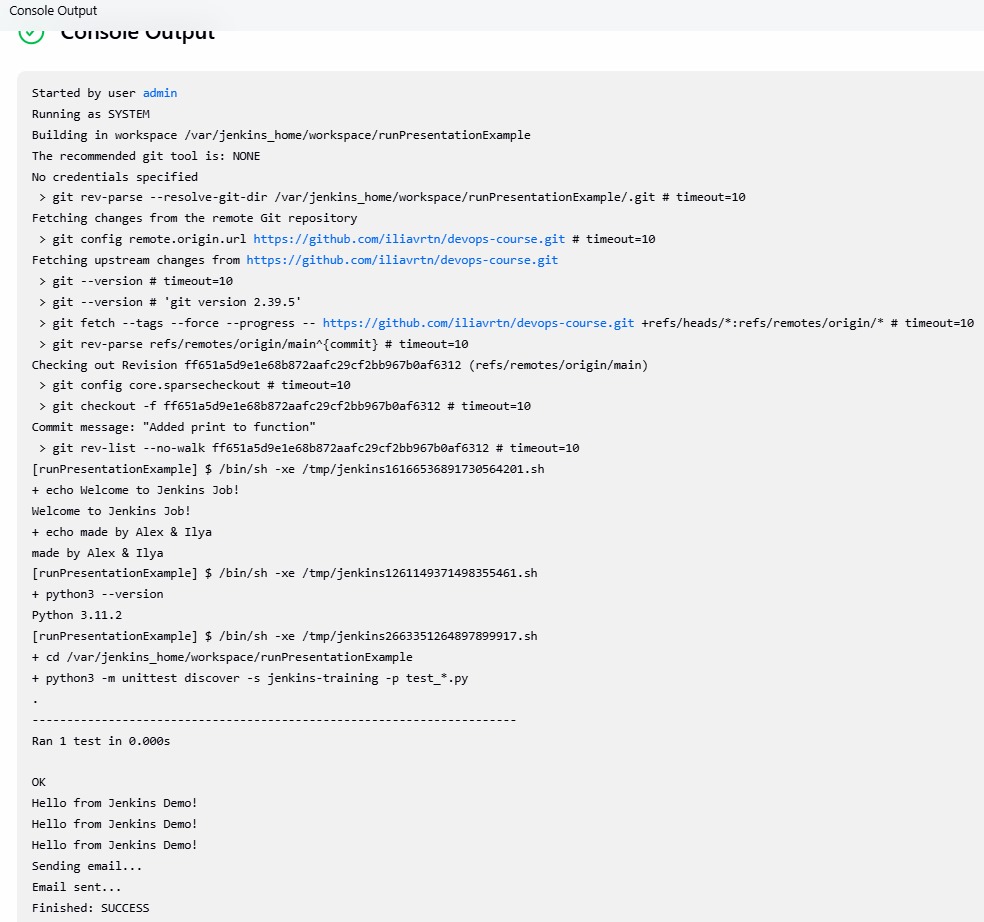
Description automatically generated



Ran Jenkins docker container on docker playground and built job from presentation:  


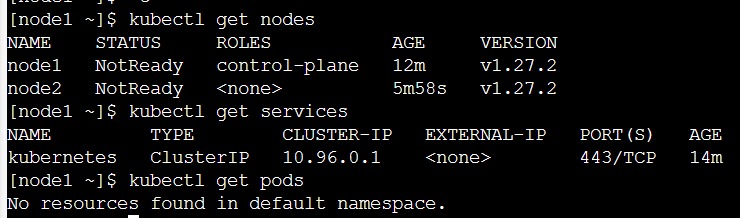
A screenshot of a computer

Description automatically generated

Ran class exercise on docker playground:  


**Lecture 7:**

Class example with k8s playground, created master node + 2 worker nodes:

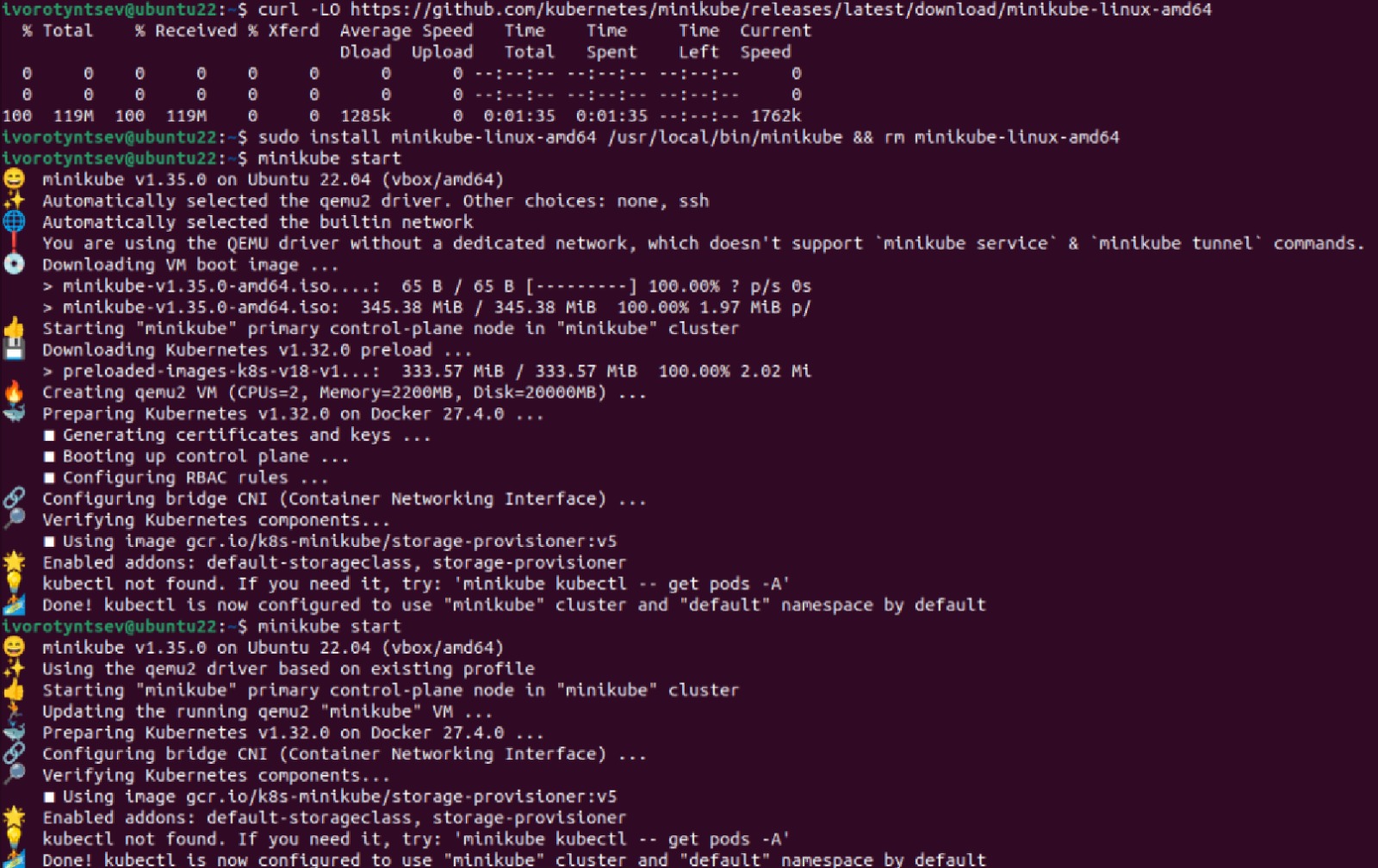


A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Installing Kubernetes using minikube and testing with Postman:  


A screenshot of a computer

Description automatically generated

A computer screen with white text

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A computer screen with white text

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A computer screen with white text

Description automatically generated

A screen shot of a computer code

Description automatically generated