A Project Report

On

Creating my OWN CLOUD Server

Submitted in partial fulfillment of the
requirements for the award of the degree of

MASTER OF COMPUTER APPLICATIONS

SUBMITTED BY

PENMATSA JAGATH KALYAN RAJU

(21A91F0048)

Under the Esteemed Guidance of

Ms. A. SRUTHI SPANDHANA, MCA,

Assistant Professor

DEPARTMENT OF MCA

ADITYA ENGINEERING COLLEGE

An Autonomous Institution

(Approved by AICTE, Affiliated to JNTUK & Accredited by NBA, NAAC with ‘A’ Grade)

Recognized by UGC under the sections 2(f) and 12(B) of the UGC act 1956

SURAMPALEM- 533 437, E.G.Dt, ANDHRA PRADESH

2021-2023

ADITYA ENGINEERING COLLEGE

An Autonomous Institution

(Approved by AICTE, Affiliated to JNTUK & Accredited by NBA, NAAC with 'A' Grade)

Recognized by UGC under the sections 2(f) and 12(B) of the UGC act 1956

SURAMPALEM- 533 437, E.G.Dt, ANDHRA PRADESH

DEPARTMENT OF MCA

CERTIFICATE

This is to certify that the project work entitled, **"Creating my OWN CLOUD Server",** is a bonafide work carried out by **PENMATSA JAGATH KALYAN RAJU** bearing Regd.No:**21A91F0048** submitted to the requirements for the award of the Computer Applications in partial fulfilment of the requirements for the award **o**f degree of **MASTER OF COMPUTER APPLICATIONS** from **Aditya Engineering College**, Surampalem during the academic year **2021-2023**.

Project Guide	**Head of the Department**
Ms. A. SRUTHI SPANDHANA, MCA.	**Mrs. D. Beulah, M.Tech, (Ph.D).**
Associate Professor,	Associate Professor,
Department of MCA,	Department of MCA,
Aditya Engineering College,	Aditya Engineering College,
Surampalem-533437.	Surampalem-533437.

EXTERNAL EXAMINER

DECLARATION

I here declare that the mini project work being present in this report entitled **“Creating my OWN CLOUD Server”** submitted to **ADITYA ENGINEERING COLLEGE, Surampalem** has been carried out by me alone under the guidance of **Ms. A. SRUTHI SPANDHANA, MCA.**

Place: Surampalem

Date: **PENMATSA JAGATH KALYAN RAJU**

(21A91F0048)

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of people who made it possible, whose constant guidance and encouragement crowned our efforts with success. It is a pleasant aspect that I have now the opportunity to express my gratitude for all of them.

The first person I would like to thank is my guide **Ms. A. SRUTHI SPANDHANA, MCA, Assistant Professor, Department of MCA**, **Aditya Engineering College, Surampalem.** Her wide knowledge and logical way of thinking have made a deep impression on me. Her understanding, encouragement and personal guidance have provided the basis for this project. She is a source of inspiration for innovative ideas and her kind support is well known to all her students and colleagues.

I wish to thank **Mrs. D. Beulah, M.Tech, (Ph.D.) Head of the Department, Aditya Engineering College, Surampalem,** who has extended her support for the success of this project.

I also wish to thank **Dr. M. Sreenivasa Reddy, M.Tech, Ph.D., Principal, Aditya Engineering College, Surampalem,** who has extended his support for the success of this project.

I would like to thank all the **Management & Technical Supporting Staff** for their timely help throughout the project.

ABSTRACT

ABSTRACT

ownCloud is an on-premise file sync and store solution that allows users to create their own cloud, manage services, permissions, and even provide hosting to other users. ownCloud is a cloud service that is secure, private and owned by the user himself. ownCloud solves the problem of remote access to confidential files, by providing Universal File Access across devices running on almost all operating systems and containers.

Most Cloud Platform Providers like Google Cloud Platform, Amazon Web Services, and the Microsoft Azure, provide paid enterprise versions that are packed with highly complex interfaces and features.

Private Cloud file storage system. Keeps file sync between all of your devices Provides remote/ mobile access to all internal server files. Allows easy sharing of file with workers and co-workers. Open-Source and free software. Cross-platform, provides integration of devices running on a variety of operating systems.

.

CONTENTS

CONTENTS

CHAPTER-1
INTRODUCTION

1. INTRODUCTION

1.1 Brief Information about the Project

My project is all about creating ownCloud server even though there are various cloud providers are there in the market we can use these free tiers with some limitations they set for use if we have a own cloud we can maintain our cloud our self in this way one can self-reliant on them self.

Coming to the various cloud providers in the market all of them will charge a heavy amount of money if we cross the limit but in here as a individual or a family who own the cloud can create the cloud keeping the requirements in mind it is less expensive when it is compared with those clouds.

This is more secured when it was compared with the other clouds because the devices which are to access the cloud has to be in the same wi-fi connection with cloud server and has to enter the login id and password of the cloud server both in this way it is way more secured than the other clouds because these clouds can be accessed by anyone who has the login id and password from anywhere so this will limit the unauthorized access.

1.2 Motivation and Contribution of the Project

After using various cloud platforms to store and access my data i felt of creating ownCloud Sever because an opensource server application that allows you to access your files from a limited range in a secure way. The files are stored on a server running ubuntu server. You can access your files via your desktop or mobile device like you might know it from, any device that was connected to your wi-fi network. The difference with ownCloud is that you stay in control of your data as you can install cloud server in your ubuntu sever.

1.3 Objective of the Project

ownCloud is the perfect solution for a self-hosted Google Drive or Dropbox alternative. Using this we can host whatever the files we are hosting in google cloud and other cloud providers we can do that locally using this in this way it is less expensive when it was compared with these providers. We can maintain continuity in all the devices in which are connected to same wi-fi network. By using the docker here we run number of files at a time.

1.4 organization of the Project

- **Chapter 2: System Analysis:** This chapter consists of description of current system, proposed system, algorithms and requirement specifications.
- **Chapter 3: System Design:** This chapter mainly consists of modul description and unified modeling language diagrams: use case diagrams, class diagrams, sequence diagrams, collaboration diagrams, and activity diagrams.
- **Chapter 4: Technology Description:** This chapter mainly consists oftechnology used in this project.
- **Chapter 5: Sample code:** This chapter mainly consists of sample code forfew modules.
- **Chapter 6: Testing:** This chapter mainly consists of testing techniques andtest cases for modules.
- **Chapter 7: Screenshots:** This chapter mainly consists of output screens ofthis project.
- **Conclusion:** Main conclusion of this project.

CHAPTER-2
SYSTEM ANALYSIS

2. SYSTEM ANALYSIS

2.1 Existing System

Downtime is often cited as one of the biggest cloud computing disadvantages. since cloud computing systems are internet-based, service outages are always an unfortunate possibility and can occur for any reason.

Although cloud service providers implement the best security standards and industry certifications, storing data and important files on external service providers always opens up risks. Any discussion involving data must address security and privacy, especially when it comes to managing sensitive data. Going on with the drawbacks of cloud computing, another one concerns vulnerability.in cloud computing, every component is online, which exposes potential vulnerabilities. Even the best teams suffer severe attacks and security breaches from time to time. Since cloud computing is built as a public service, it's easy to run before you learn to walk.

Disadvantages

- Security and privacy
- Vulnerability to attacks
- Limited control and flexibility
- Cost concerns
- Down time

Laptop
Network
Smartphone
Cloud Computing
Tablet computer
Server
Mobile
Database

2.2 Proposed System

Creating ownCloud server even though there are various cloud providers are there in the market we can use these free tiers with some limitations they set for use if we have a own cloud we can maintain our cloud our self in this way one can self- reliant on them self.

Coming to the various cloud providers in the market all of them will charge a heavy amount of money if we cross the limit but in here as a individual or a family who own the cloud can create the cloud keeping the requirements in mind it is less expensive when it is compared with those clouds.

Advantages

- Simple platform
- Less expensive
- More secured
- Continuity
- Easily customizable
- Scalable
- No hidden fees

2.3 Feasibility Study:

An important outcome of preliminary investigation is the determination that the system request is feasible. This is possible only if it is feasible within limited resource and time.

Feasibility is the study of impact, which happens in the organization by the development of a system. The impact can be either positive or negative. When the positives nominate the negatives, then the system is considered feasible. The different feasibilities that have to be analyzed are

- Operational Feasibility
- Economic Feasibility
- Technical Feasibility

Operational Feasibility

Operational Feasibility deals with the study of prospects of the system to be developed. This system operationally eliminates all the tensions of the admin and helps him in effectively tracking the project progress. This kind of automation will surely reduce the time and energy, which previously consumed in manual work. Based on the study, the system is proved to be operationally feasible.

Economic Feasibility

Economic Feasibility or Cost-benefit is an assessment of the economic justification for a computer-based project. The organization needed not spend much more for development of the system already available. The only thing is to be done is making an environment for the development with an effective supervision. If we are doing so, we can attain the maximum usability of the corresponding resources. Even after the development, the organization will not be in a condition to invest more in the organization. Therefore, the system is economically feasible.

Technical Feasibility

According to Roger S. Pressman, Technical Feasibility is the assessment of the technical resources of the organization. The system is technically feasible for development and can be developed with the existing facility. In my case here the data the ubuntu server is technically supports both docker and cloud server, So, in the case of my project it is technically feasible.

2.4 Functional Requirements:

- Mainly for this project the most important requirement is ubuntu operating system in the server version 22.0.
- Docker has to be used to run several devices at a time.
- An wi-fi connection is must and should because here in any cloud the data is can only be accessed with an active internet and here specifically the data is transferred, stored and accessed by using this wi-fi as a medium.
- An ethernet cable is needed to connect the server to the wi-fi.

2.5 Non-Functional Requirements:

Efficiency/Performance

- Here we can connect various devices to this cloud server based on the capacity of the ram of the system.
- We can run various applications on this system in this way it will reduce the load on the normal device as well.

Usability:

- The user interface of the system should be very user friendly.
- It should not take more than 120 seconds for the other device to connect to this Server.
- It should not take more than 90 seconds to access the data from this server.

2.6 Requirement Specifications

2.6.1 Software Requirements

Operating system	:	Ubuntu OS
Database	:	Cloud sever
Framework	:	Docker

2.6.2 Minimum Hardware Requirements

Processor	:	Intel Core i3
Speed	:	3.60GHz
Hard Disk	:	500 GB.
RAM	:	4 GB

CHAPTER-3
SYSTEM DESIGN

3. SYSTEM DESIGN

3.1 INTRODUCTION

Ubuntu Server Cloud is an Ubuntu derivative server optimized for cloud deployments. It includes a suite of tools specifically designed to ease the deployment, management and scaling of operations in the cloud. The platform is underpinned by Canonical's Trusty Tahr Linux distribution and comes with the same level of support, resources and stability as Ubuntu Server.

Ubuntu Server Cloud can be provisioned and managed through the Canonical System Manager. It offers the same level of customization and management features as Ubuntu Server, including the ability to deploy custom applications and manage a fleet of devices through Canonical's management tool Ubuntu Device Management.

3.2 System Architecture:

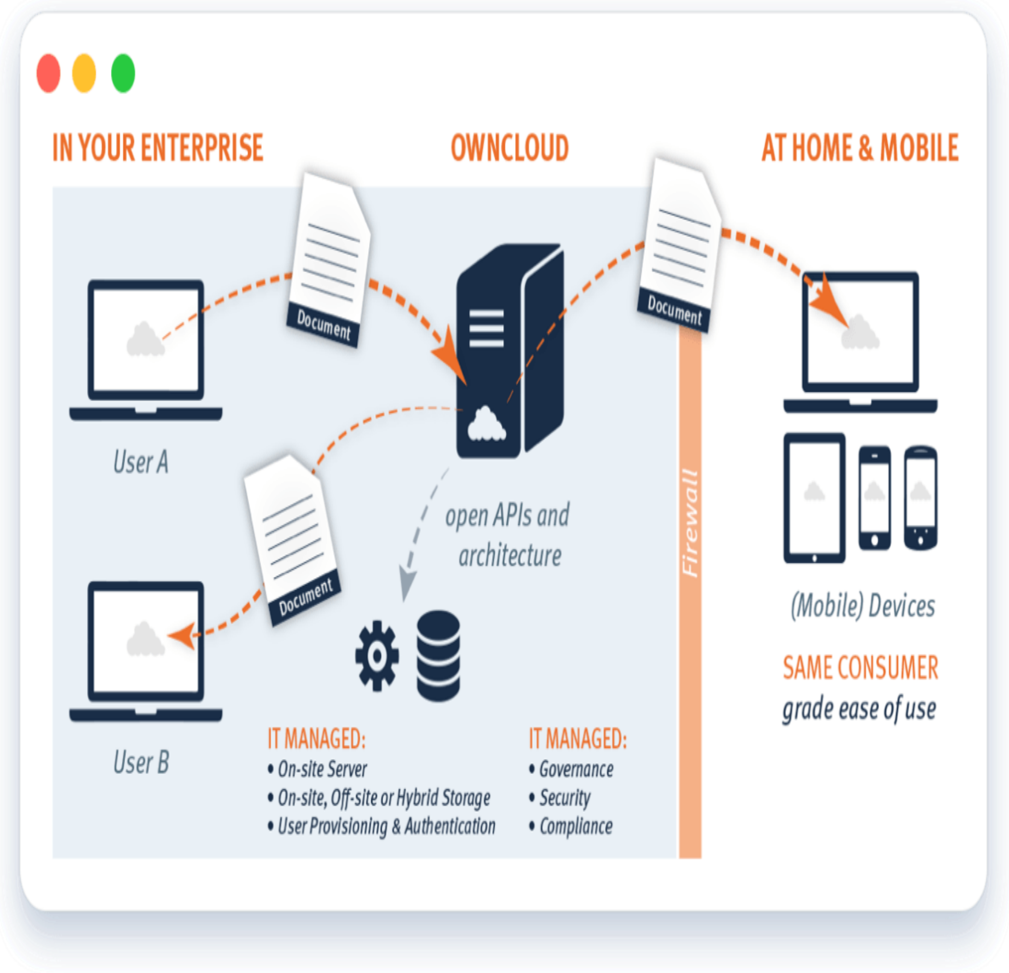


Fig: 3.2.1: System architecture

Architecture Description:

- Once you have created your Ubuntu cloud server, you can access it using a secure shell (SSH) connection using the IP address or hostname of the server andyour login credentials.
- You can use the 'sudo' command to perform administrative tasks and installsoftware on the server.
- The Ubuntu server includes a firewall called 'ufw' which can be used tocontrol incoming and outgoing network traffic.
- You can use the 'apt' package manager to update and upgrade the packages onthe server.
- Ubuntu server comes with a built-in monitoring tool called 'Systemd' whichcan be used to check the status of services and processes.

- Ubuntu server also includes the 'netstat' command which can be used to check network connections and ports.
- To manage the user accounts and permissions, you can use the 'adduser' and 'usermod' commands.
- You can use the 'df -h' command to check the disk space usage on the server.
- You can use the 'top' command Regenerate response e and performance.
- You can use the 'nano' or 'vi' text corto con sole server.
- To backup and restore files, you can use the 'tar' command.
- You can use the 'ps aux' command to check running processes on the server.
- To configure your server's network settings you can use the 'ifconfig' and'route' commands.
- To schedule tasks and automate processes, you can use the 'cron' service
- To ensure server security, you should keep your software updated, configure your firewall and use strong passwords.

3.3 Modules Description

- Admin
- User

Admin module:

- In this module, the Admin has to login by using valid user name and password.
- After login successful he can manage activities of storage and access to this cloud.

User:

Here the one who is accessing the cloud is called client user. He can share the resources and edit the existing files also but at first to enter in to the they had to be on the same wi-fi network and then they has to use the user password which was providedby the admin.

3.4 Data Dictionary

1. Ubuntu server cloud data dictionary is a reference library that describes the data that is stored in the Ubuntu server cloud.
2. The dictionary provides information on objects and fields in the data, and can help you find the data you are looking for.

You can use the dictionary to:

varied information on the data, including: object class, field name, object data type, value data type, and more

Column Name	Data Type	Constraints	Description
Username	VARCHAR2(45)	NOT NULL	Specifies the username
Password	VARCHAR2(45)	NOT NULL	Specifies the Password

3.4.1 Data table for User Login

Column Name	Data Type	Constraints	Description
Id	INTEGER	NOT NULL	Specifies the Id
Username	VARCHAR2(45)	NOT NULL	Specifies the username
keyword	VARCHAR2(45)	NOT NULL	Specifies the keyword
dt	VARCHAR2(45)	NOT NULL	Specifies the dt

3.4.2 Data table for User Access

3.5 UML Diagrams

- The unified modeling language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules.
- A UML system is represented using five different views that describe the system from distinctly different perspective. UML is specifically constructed through two different domains.
- UML Analysis modeling, this focuses on the user model and structural model views of the system.
- UML design modeling, which focuses on the behavioral modeling, implementation modeling and environmental model views.

These are divided into the following types.

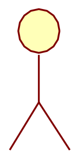
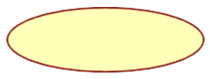
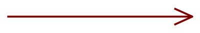
- Use case diagram
- Class diagram
- Sequence diagram

3.5.1 Use case Diagram

Use Case diagrams identify the functionality provided by the system (use cases), the users who interact with the system (actors), and the association between the users and the functionality. Use Cases are used in the Analysis phase of software development to articulate the high-level requirements of the system. The primary goals of Use Case diagrams include:

- Providing a high-level view of what the system does.
- Identifying the users ("actors") of the system.
- Determining areas needing human-computer interfaces.

Graphical Notation: The basic components of Use Case diagrams are the Admin, the Use Case, and the Association.

Admin	An Admin, as mentioned, is a user of the system, and is depicted using a stick figure. The role of the user is written beneath the icon. Admin are not limited to humans. If a system communicates with another application, and expects input or delivers output, then that application can also be considered an actor.	
Use Case	A Use Case is functionality provided by the system, Use Cases are depicted with an ellipse. The name of the use case is written within the ellipse.	
Directed Association	These Associations are used to link Admin with Use Cases, and indicate that an Admin participates in the UseCase in some form.	

Behind each Use Case is a series of actions to achieve the proper functionality, as well as alternate paths for instances where validation fails, or errors occur. These actions can be further defined in a Use Case description. Because this is not addressed in UML, there are no standards for Use Case descriptions. However, there are some common templates can follow, and whole books on the subject writing of Use Case description.

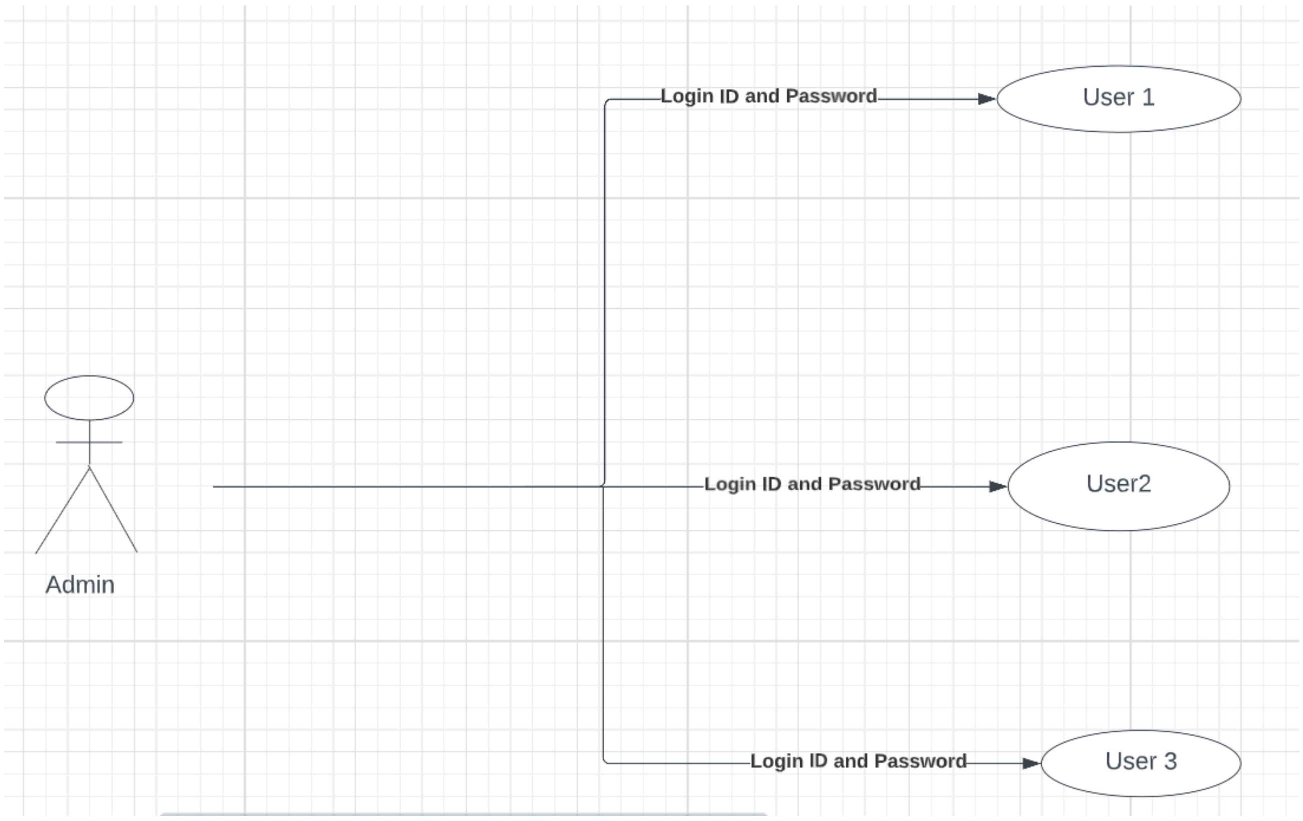


Fig 3.5.1.1: Use case diagram for Admin

Description:

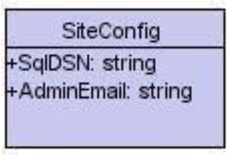
The above diagram represents the use case diagram of the how an ubuntu cloud server Admin and user relation is discribed. The above use case diagram contains Admin development through login and password. The admin performs the functionalities in this use case diagram.

3.5.2 Class Diagram

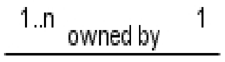
Class diagrams identify the class structure of a system, including the properties and methods of each class. Also depicted are the various relationships that can exist between classes, such as an inheritance relationship. Part of the popularity of Class diagrams stems from the fact that many CASE tools, such as Rational XDE, will auto-generate code in a variety of languages, these tools can synchronize models and code, reducing the workload, and can also generate Class diagrams from object- oriented code.

Graphical Notation: The elements on a Class diagram are classes and the relationships between them.

Class: Classes are building blocks in object-oriented programming. A class is depicted using a rectangle divided into three section. The top section is name of class, the middle section defines the properties of class. The bottom section list the methods of the class.



Association: An Association is a generic relationship between two classes, and is modeled by a line connecting the two classes. This line can be qualified with the type of relationship, and can also feature multiplicity rule (e.g. one-to-one, one-to-many, many-to-many) for the relationship.



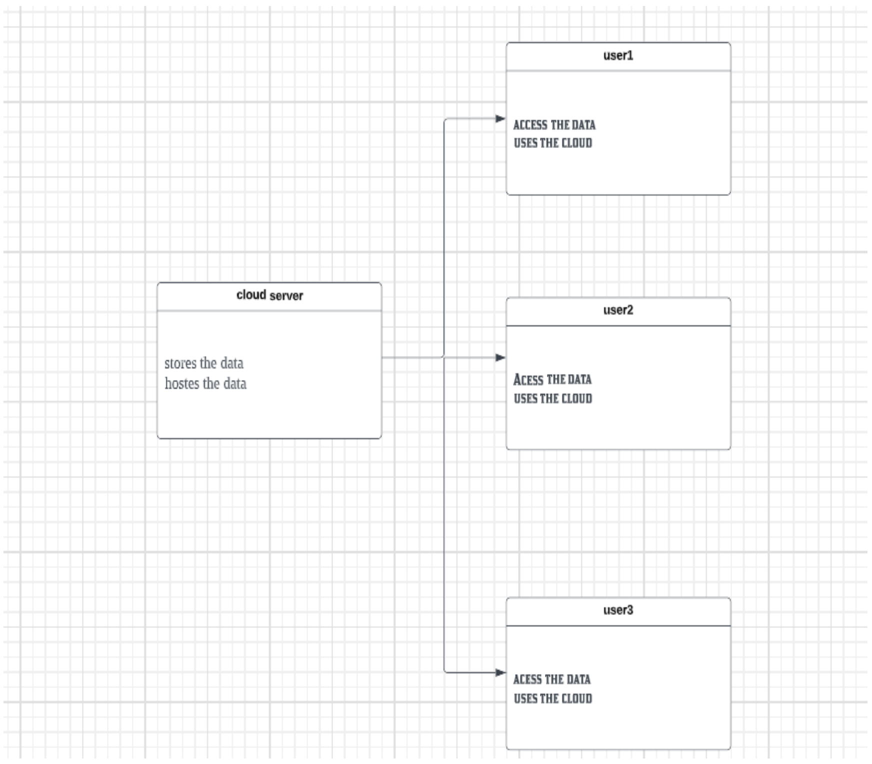


Fig 3.5.2.1: Class diagram for System

Description:

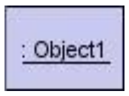
The above diagram represents the Class diagram of the project creating ownCloud server. The above class diagram contains cloud server and various users namely user1, user2, user3.

3.5.3 Sequence diagram

Sequence diagrams document the interactions between classes to achieve a result, such as a use case. Because UML is designed for object-oriented programming, these communications between classes are known as messages. The Sequence diagram lists objects horizontally, and time vertically, and models these messages over time.

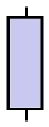
Graphical Notation: In a Sequence diagram, classes and actors are listed as columns, with vertical lifelines indicating the lifetime of the object over time.

Object: Objects are instances of classes, and are arranged horizontally. The pictorial representation for an Object is a class (a rectangle) with the name prefixed by the object name (optional) and a semi-colon.



Lifeline: The Lifeline identifies the existence of the object over time. The notation 2for a Lifeline is a vertical dotted line extending from an object.

Activation: Activations, modeled as rectangular boxes on the lifeline, indicate when the object is performing an action.



Message: Messages, modeled as horizontal arrows between Activations, indicate the communications between objects

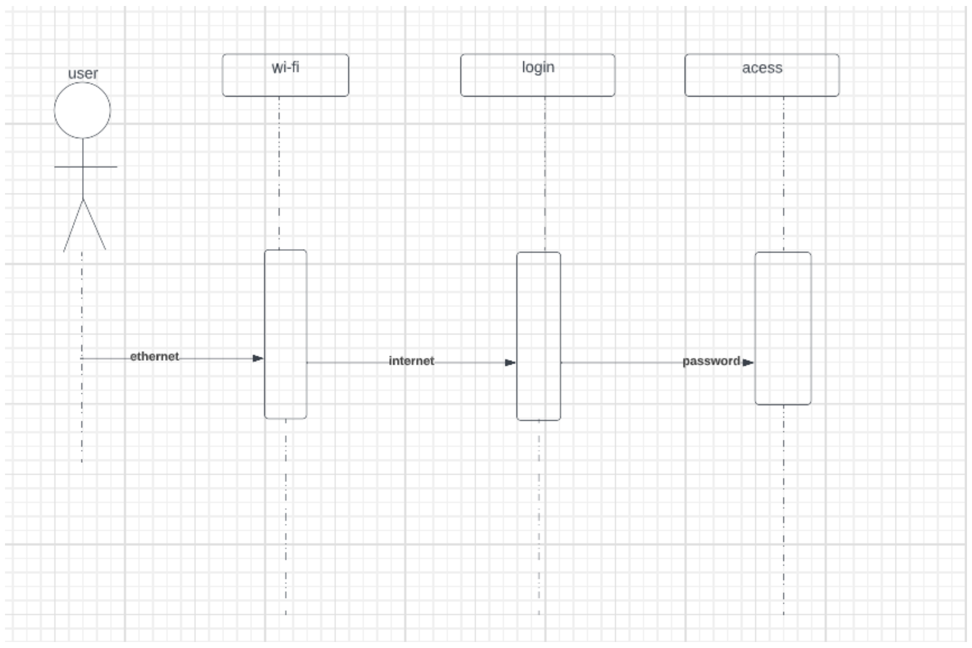


Fig 3.5.3.1: Sequence diagram for System

Description:

The above diagram represents the sequence diagram of the project, ownCloud server and how it works.

CHAPTER-4

TECHNOLOGY DESCRIPTION

4.TECHNOLOGY DESCRIPTION

4.1 Introduction to Ubuntu server Os:

As you should probably know, Linux powers the majority of the web we see today. This is mainly because Linux systems are inherently more secure and stable than other systems. There are several types of Linux distributions for powering servers. Some notable ones include Ubuntu, Red Hat, Debian, and CentOS. Ubuntu, in particular, has been enjoyinga surge in popularity as a server distro in recent times. In this guide, our editors have outlinedwhy the Linux Ubuntu server is outgrowing many of its competitions. Stay with us throughout this guide to learn why Ubuntu shines as a server distro.

What is the Ubuntu Server?

It is an operating system developed by Canonical and a large number of open source developers across the world. It is meant to power modern-day servers that serve static and dynamic web pages, applications, files, containers, and many more. The ability to run this on a wide range of platforms and architecture make this a suitable choice for enterprises as well as hobbyists.

Difference Between Ubuntu Server and Desktop?

If you are a predominantly desktop user, you might wonder what’s the difference between your everyday Ubuntu desktop solution and its server counterpart. Before we answer this question, let us discuss the dissimilarities between a desktop and a server first.

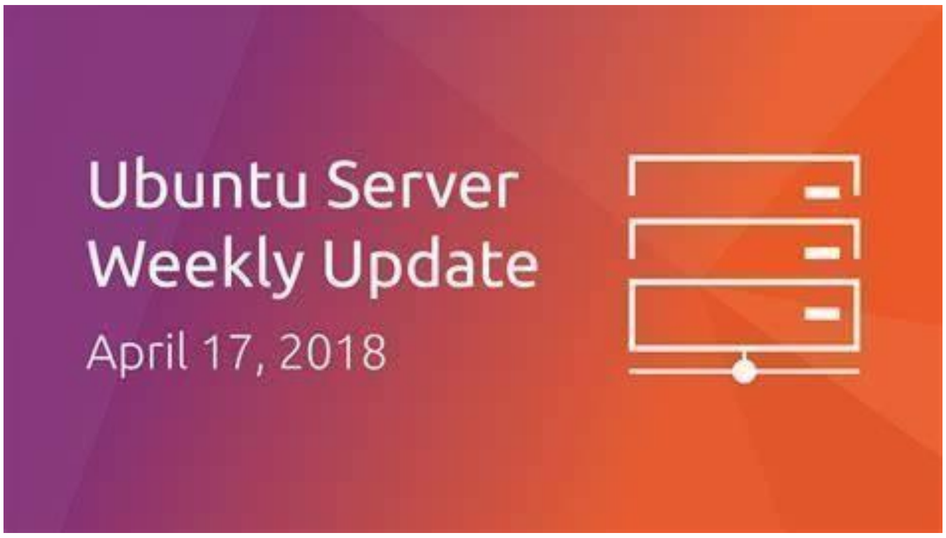
When talking about desktops, we refer to personal computers that we use for everyday tasks like productivity, gaming, and office works. These systems are equipped with peripheral devices such as a keyboard, mouse, and modems for obvious reasons. They are also powered by reasonable hardware resources. We usually use Linux desktop environments on these devices. The Ubuntu desktop is arguably among the best Linux distributions for such systems.

Servers, on the other hand, are much beefier in terms of CPU resources. This is because they are designed to be more powerful, stable, and secure for long term usage. Since they are often managed remotely, most servers do not include common peripheral devices.

This is known as a headless setup, obtaining the jargon from the omission of I/O devices. Traditionally, servers come in two form factors, either rackmount or tower.

How Popular is the Ubuntu Server?

Ubuntu has been gaining steady popularity as a server distribution for some time now. Many corporations are switching to Ubuntu from other server distros due to its solid ecosystem and reliable support. Among the 37% of global websites powered by various Linux distributions, Ubuntu accounts for a whopping 45% share. At the same time, competitors like the Red Hat Enterprise Linux (RHEL) has only a mere 2% share. Debian and CentOS, each has around 18% share in this regard.

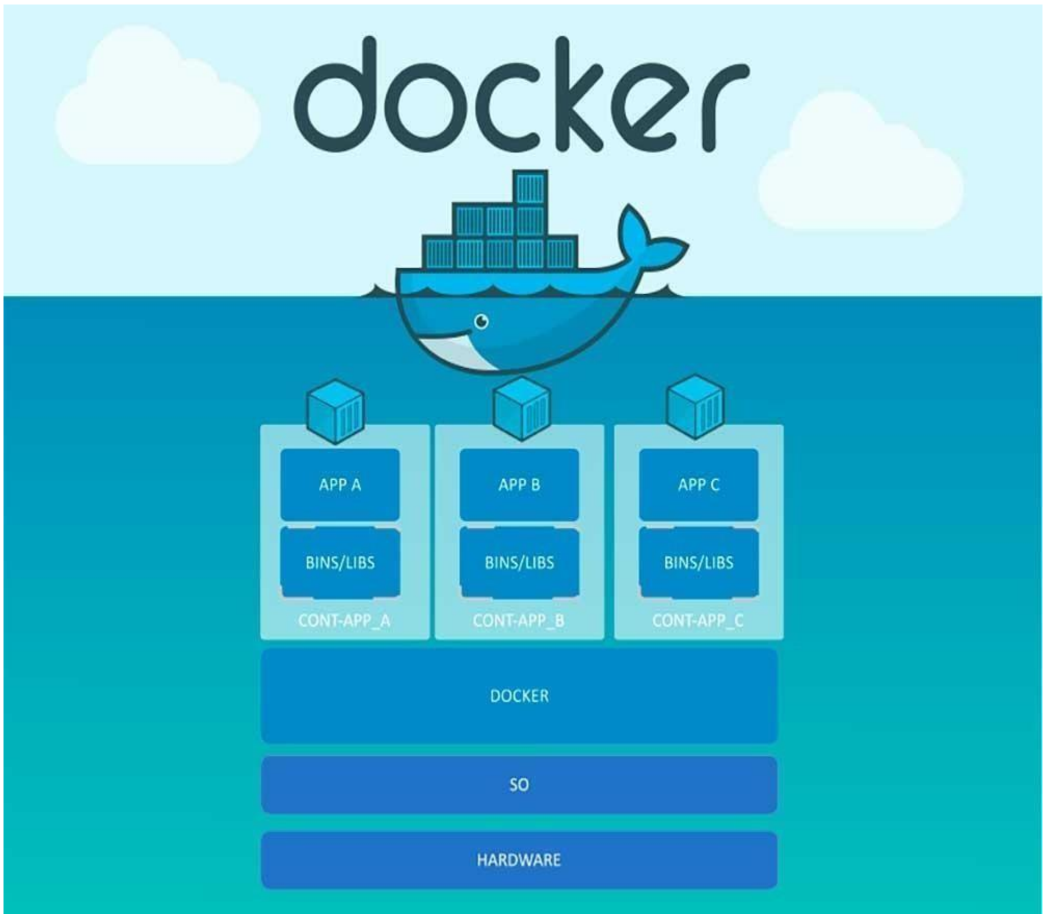


4.2 Introduction to Tasksel:

With Tasksel, you can use a simple interface to configure your system for any task. This program is used during installation, but users also have access at any time through the Ubuntu package managers like Sudo or APT-GET. You might think that there are few differences between tasks available in this menu versus meta-packages since they both provide similar capabilities.

4.3 Introduction to Docker:

Docker is an application that simplifies the process of managing application processes in Containers. Containers let you run your applications in resource-isolated processes. They’re similar to virtual machines, but containers are more portable, more resource-friendly, and more dependent on the host operating system.



CHAPTER-5

SAMPLE CODE

Sample code:

To install ubuntu server:

To install the ubuntu sever operating system in your laptop us this link given below.

https://ubuntu.com/download/server

Installing Tasksel on Ubuntu 22.04:

First, update your system to ensure all existing packages are up to date to avoid any conflicts during the installation.

```
sudo apt update && sudo apt upgrade -y
```

By default, TaskSel is available in Ubuntu 22.04 default repository making the installation quick and straightforward. In your terminal, use the following command

```
sudo apt install tasksel -y
```

To launch TaskSel, in your terminal, use the following command.

```
sudo tasksel
```

Installing Docker:

First, update your existing list of packages:

```
sudo apt update
```

Next, install a few prerequisite packages which let apt use packages over HTTPS:

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

Then add the GPG key for the official Docker repository to your system:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
```

Add the Docker repository to APT sources:

```
echo "deb [arch=$(dpkg --print-architecture) signed-
by=/usr/share/keyrings/docker-archive-keyring.gpg]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" |
sudotee /etc/apt/sources.list.d/docker.list > /dev/null
```

Update your existing list of packages again for the addition to be recognized:

```
sudo apt update
```

Make sure you are about to install from the Docker repo instead of the default Ubuntu repo:

```
apt-cache policy docker-ce
```

install Docker:

```
sudo apt install docker-ce
```

Docker should now be installed, the daemon started, and the process enabled to start on boot. Check that it’s running:

```
sudo systemctl status docker
```

IP Configuration:

This command is used to configurate the ip address

```
Ifconfig
```

This command is used to edit the local port number

```
sudo vi /etc/ssh/sshd_config
```

This command is used to restart the ssh

```
Sudo systemctl restart ssh
```

CHAPTER-6
TESTING

TESTING

6.1 Introduction

In general, software engineers distinguish software faults from software failures. In case of a failure, the software does not do what the user expects. A fault is a programming error that may or may not actually manifest as a failure. A fault can also be described as an error in the correctness of the semantic of a computer program. A fault will become a failure if the exact computation conditions are met, one of them being that the faulty portion of computer software executes on the CPU. A fault can also turn into a failure when the software is ported to a different hardware platform or a different compiler, or when the software gets extended. Software testing is the technical investigation of the product under test to provide stakeholders with quality related information.

System Testing and Implementation

The purpose is to exercise the different parts of the module code to detect coding errors. After this the modules are gradually integrated into subsystems, which are then integrated themselves too eventually forming the entire system. During integration of module integration testing is performed. The goal of this is to detect designing errors, while focusing the interconnection between modules. After the system was put together, system testing is performed. Here the system is tested against the system requirements to see if all requirements were met and the system performs as specified by the requirements. Finally accepting testing is performed to demonstrate to the client for the operation of the system.

For the testing to be successful, proper selection of the test case is essential. There are two different approaches for selecting test case. The software or the module to be tested is treated as a black box, and the test cases are decided based on the specifications of the system or module. For this reason, this form of testing is also called “black box testing”.

The focus here is on testing the external behavior of the system. In structural testing the test cases are decided based on the logic of the module to be tested. A common approach here is to achieve some type of coverage of the statements in the code. The two forms of testing are complementary: one tests the external behavior, the other tests the internal structure.

Testing is an extremely critical and time-consuming activity. It requires proper planning of the overall testing process. Frequently the testing process starts with the test plan. This plan identifies all testing related activities that must be performed and specifies the schedule, allocates the resources, and specifies guidelines for testing. The test plan specifies conditions that should be tested; different units to be tested, and the manner in which the module will be integrated together. Then for different test unit, a test case specification document is produced, which lists all the different test cases, together with the expected outputs, that will be used for testing. During the testing of the unit the specified test cases are executed and the actual results are compared with the expected outputs. The final output of the testing phase is the testing report and the error report, or a set of such reports. Each test report contains a set of test cases and the result of executing the code with the test cases. The error report describes the errors encountered and the action taken to remove the error.

Testing Techniques

Testing is a process, which reveals errors in the program. It is the major quality measure employed during software development. During testing, the program is executed with a set of conditions known as test cases and the output is evaluated to determine whether the program is performing as expected. In order to make sure that the system does not have errors, the different levels of testing strategies that are applied at differing phases of software development are:

Black Box Testing

- In this strategy some test cases are generated as input conditions that fully execute all functional requirements for the program. This testing has been uses to find errors in the following categories:
- Incorrect or missing functions
- Interface errors
- Errors in data structure or external database access
- Performance errors
- Initialization and termination errors.

- In this testing only the output is checked for correctness. The logical flow of the data is not checked.

White Box Testing

In this testing, the test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases. It has been uses to generate the test cases in the following cases:

- Guarantee that all independent paths have been executed.
- Execute all logical decisions on their true and false sides.
- Execute all loops at their boundaries and within their operational.
- Execute internal data structures to ensure their validity.

Testing Strategies

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

This System consists of 3 modules. Those are Reputation module, route discovery module, audit module. Each module is taken as unit and tested. Identified errors are corrected and executable unit are obtained.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

System Testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

Functional Testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.
Systems/Procedures	: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes

.

6.2 SAMPLE TEST CASES SPECIFICATION

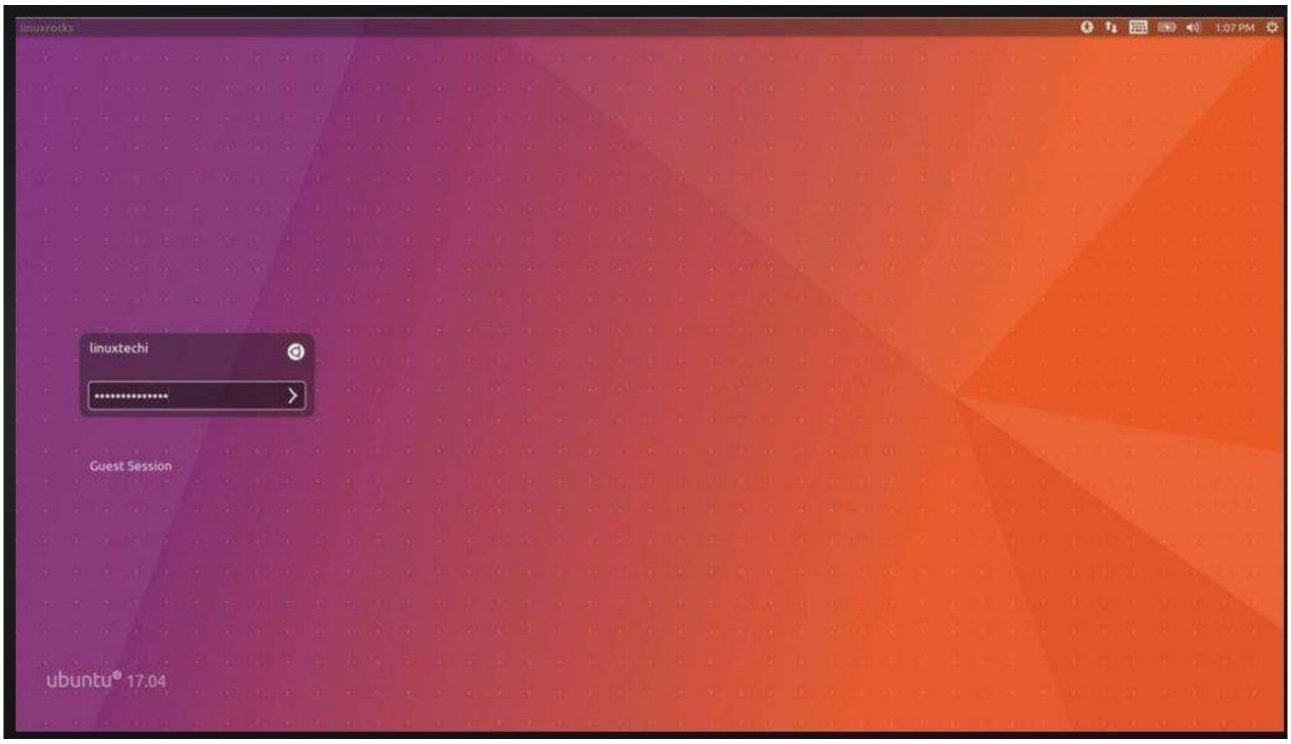
Test case id	Test case name	Input	Expected Output	Observed Output	Result
T1	User login	Enter valid user name and password	User login should be done successful.	User login is successful.	Pass
T2	User login	Enter invalid user name and password.	User login should be unsuccessful.	User login is unsuccessful	Pass
T3	Server login	Enter valid Server name and password	Server login should be done successful.	Server login is successful.	Pass
T4	Server login	Enter invalid Server name and password.	Server login should be unsuccessful.	Server login is unsuccessful	Pass

Table 6.2.1: Sample Test Cases Specification

CHAPTER-7
SCREENSHOTS

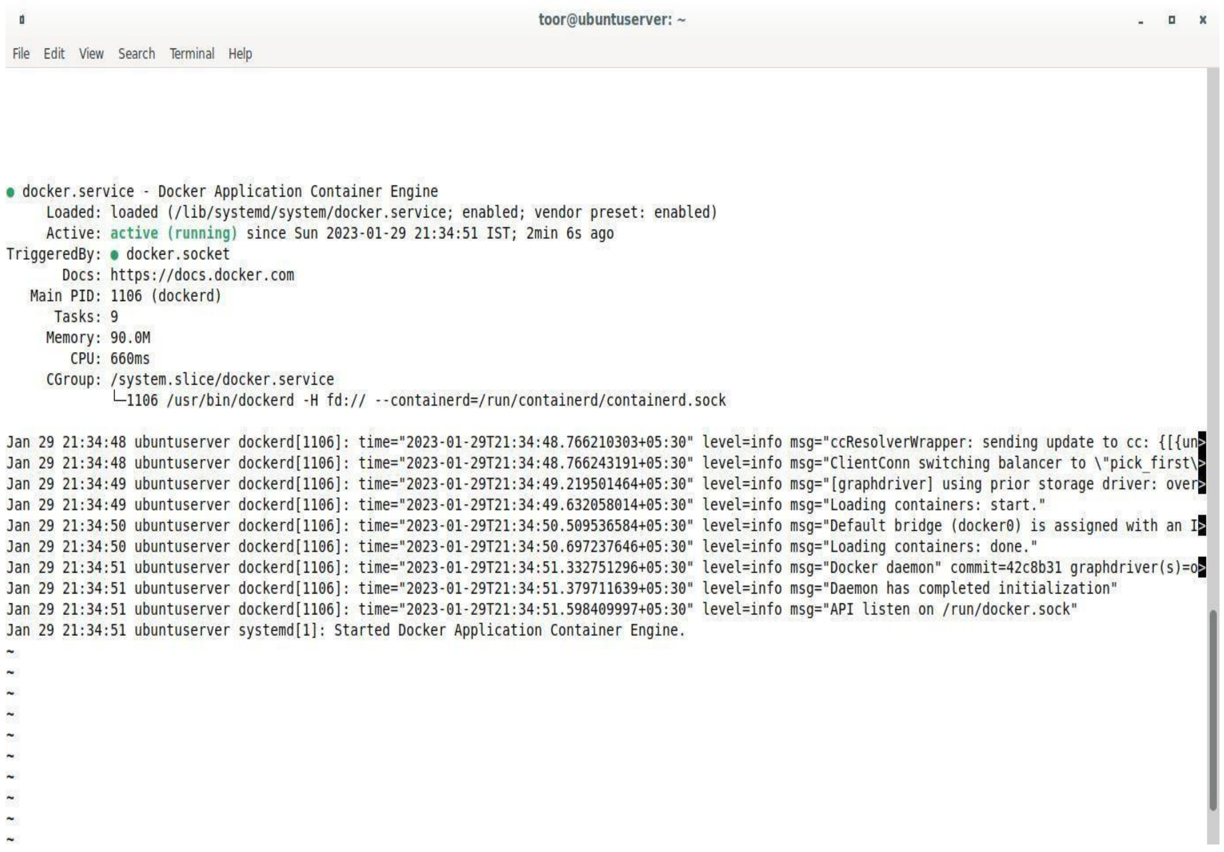
SCREENSHOTS

7.1 Login Page



Description: The above screenshot displays Login Page consists of Username and Password options to login. Here the user will enter his username and password to login. To enter in to the ubuntu server.

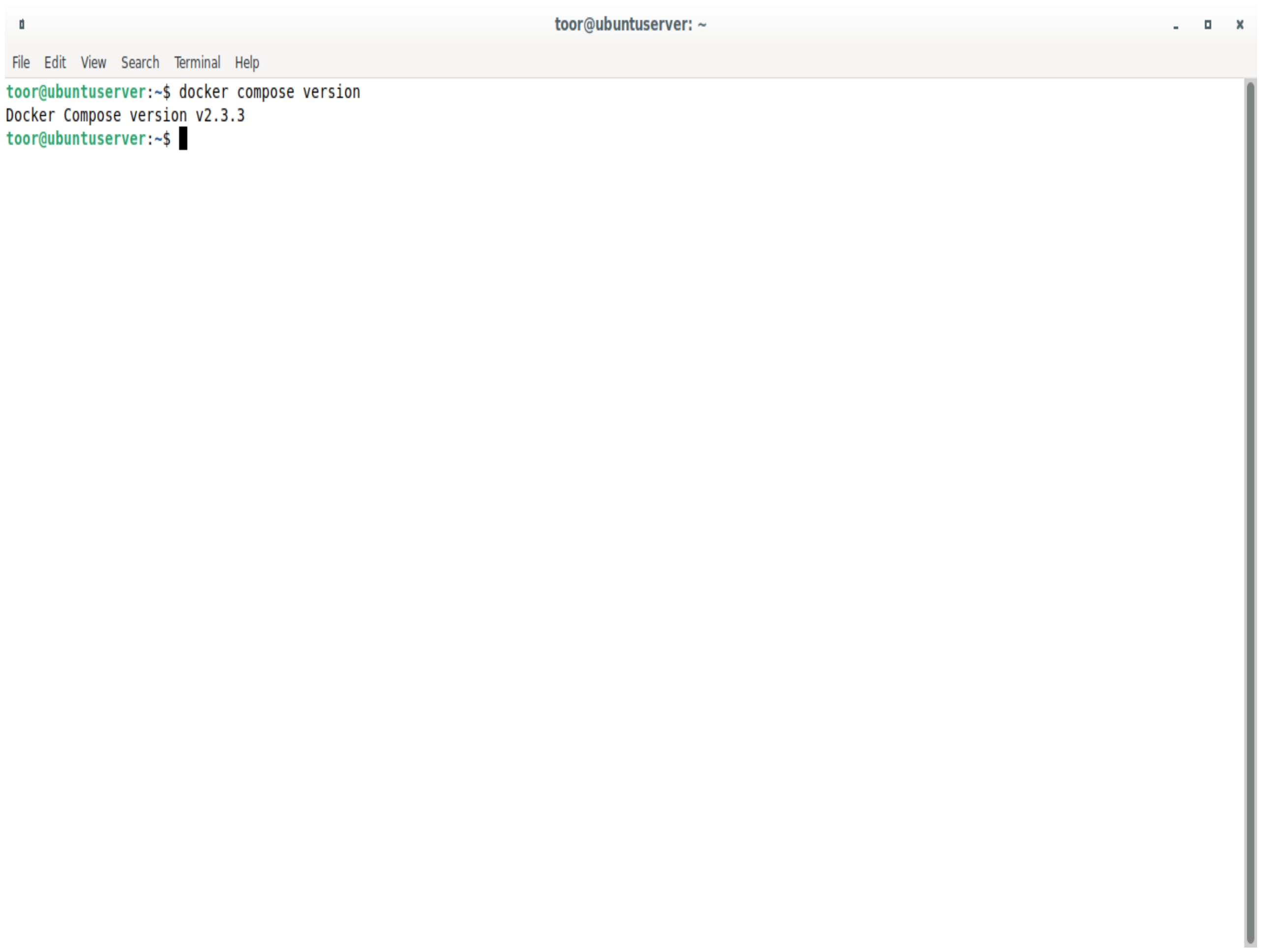
7.2 Docker Active (running)



```
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2023-01-29 21:34:51 IST; 2min 6s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 1106 (dockerd)
      Tasks: 9
     Memory: 90.0M
        CPU: 660ms
     CGroup: /system.slice/docker.service
             └─1106 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Jan 29 21:34:48 ubuntuserver dockerd[1106]: time="2023-01-29T21:34:48.766210303+05:30" level=info msg="ccResolverWrapper: sending update to cc: {[{un>
Jan 29 21:34:48 ubuntuserver dockerd[1106]: time="2023-01-29T21:34:48.766243191+05:30" level=info msg="ClientConn switching balancer to \"pick_first\>
Jan 29 21:34:49 ubuntuserver dockerd[1106]: time="2023-01-29T21:34:49.219501464+05:30" level=info msg="[graphdriver] using prior storage driver: over>
Jan 29 21:34:49 ubuntuserver dockerd[1106]: time="2023-01-29T21:34:49.632058014+05:30" level=info msg="Loading containers: start."
Jan 29 21:34:50 ubuntuserver dockerd[1106]: time="2023-01-29T21:34:50.509536584+05:30" level=info msg="Default bridge (docker0) is assigned with an I>
Jan 29 21:34:50 ubuntuserver dockerd[1106]: time="2023-01-29T21:34:50.697237646+05:30" level=info msg="Loading containers: done."
Jan 29 21:34:51 ubuntuserver dockerd[1106]: time="2023-01-29T21:34:51.332751296+05:30" level=info msg="Docker daemon" commit=42c8b31 graphdriver(s)=o>
Jan 29 21:34:51 ubuntuserver dockerd[1106]: time="2023-01-29T21:34:51.379711639+05:30" level=info msg="Daemon has completed initialization"
Jan 29 21:34:51 ubuntuserver dockerd[1106]: time="2023-01-29T21:34:51.598409997+05:30" level=info msg="API listen on /run/docker.sock"
Jan 29 21:34:51 ubuntuserver systemd[1]: Started Docker Application Container Engine.
~
~
~
~
~
~
~
~
~
~
```

Description: The above screenshot displays Docker is Active and running.

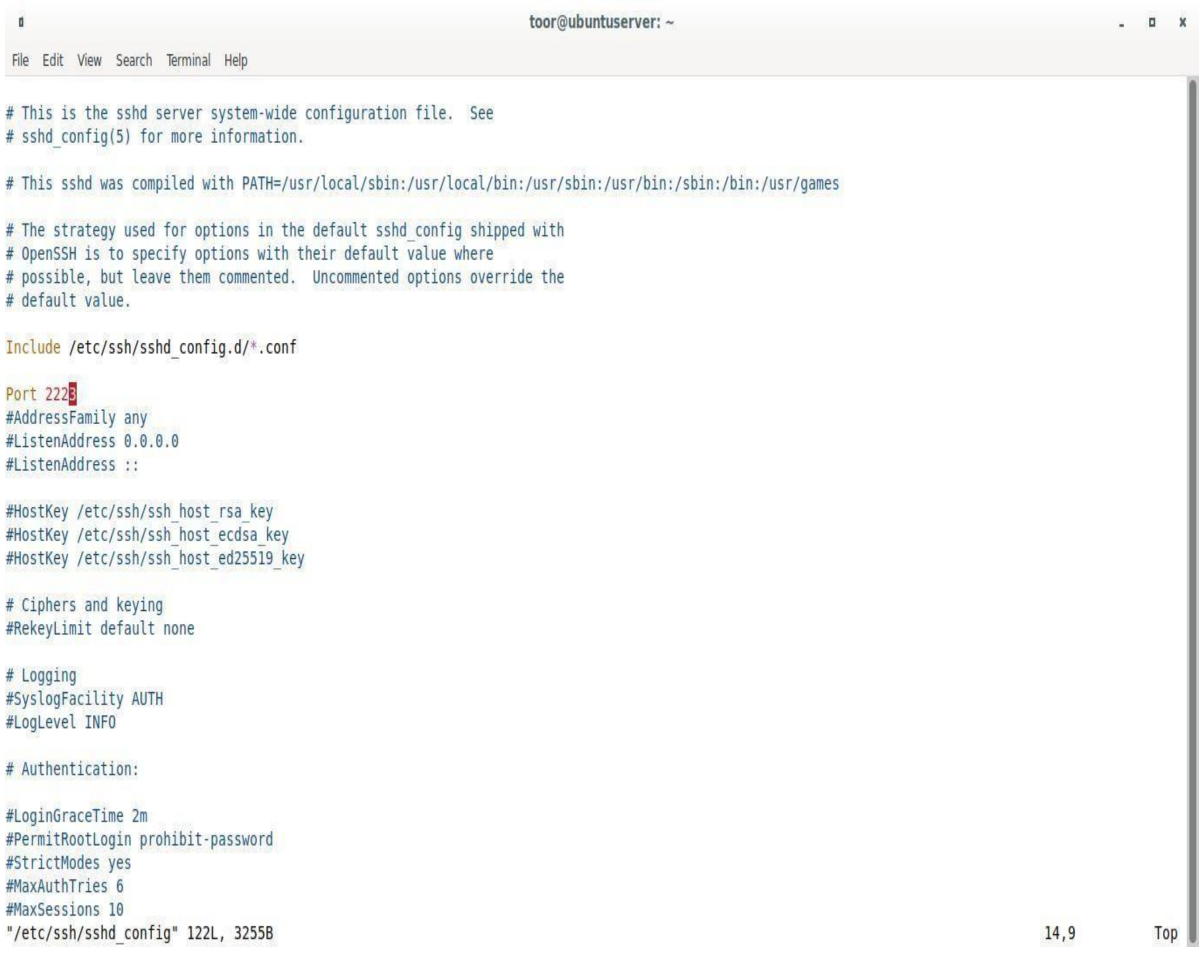
7.3 Docker version



```
toor@ubuntuserver:~$ docker compose version
Docker Compose version v2.3.3
toor@ubuntuserver:~$ 
```

Description: The above screenshot displays the Docker version.

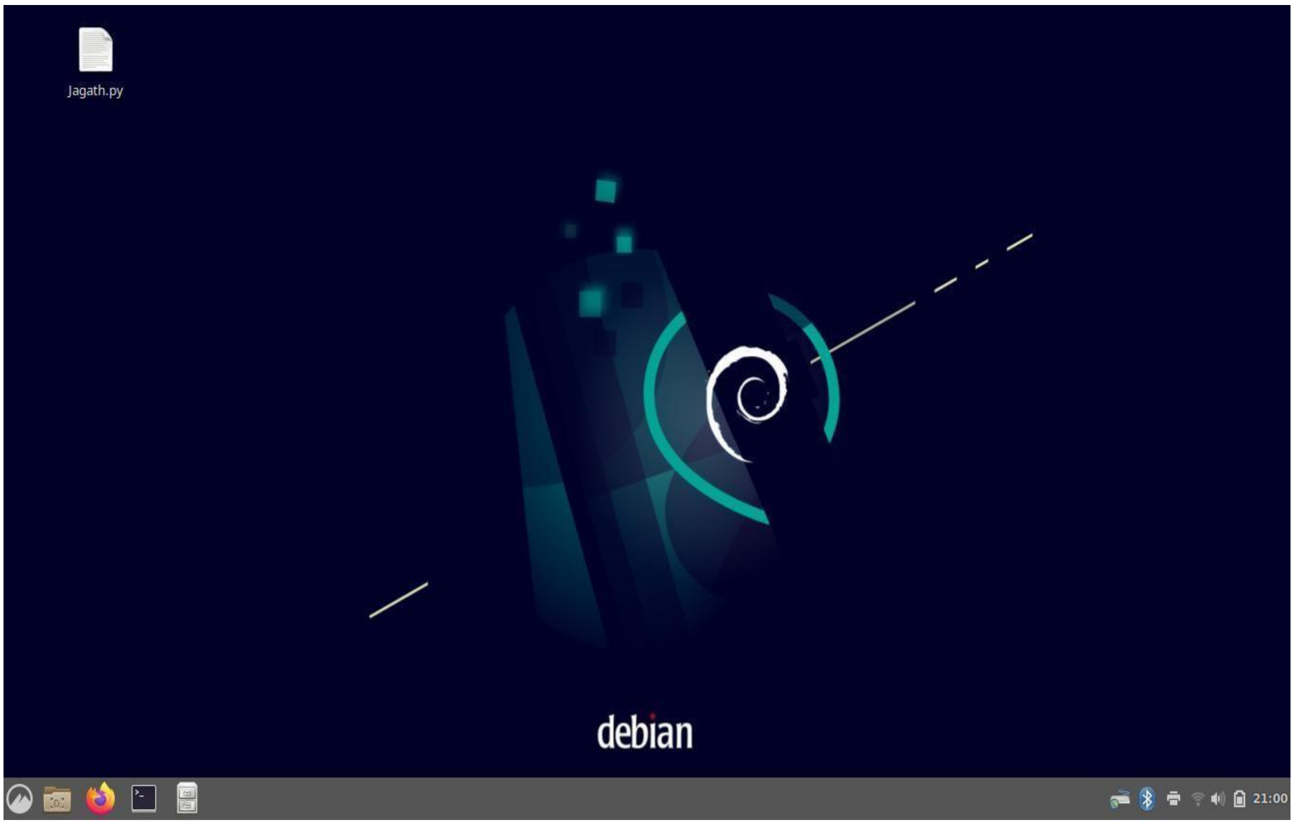
7.4 Local port number



```
# This is the sshd server system-wide configuration file.  See
# sshd_config(5) for more information.

# This sshd was compiled with PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games

# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented.  Uncommented options override the
# default value.

Include /etc/ssh/sshd_config.d/*.conf

Port 2223
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key
#HostKey /etc/ssh/ssh_host_ed25519_key

# Ciphers and keying
#RekeyLimit default none

# Logging
#SyslogFacility AUTH
#LogLevel INFO

# Authentication:

#LoginGraceTime 2m
#PermitRootLogin prohibit-password
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10
"/etc/ssh/sshd_config" 122L, 3255B                                                    14,9          Top
```

Description: The above screenshot displays local port number edit option this is opened by using command of **Include /etc/ssh/sshd_config.d/*.conf**

7.5 File that was created on the desktop



Description: The above screenshot displays the file **Jagath.py .** Which was created by the user by using the other system in this way the data can be stored in the cloud.

CONCLUSION

CONCLUSION

In this system, that enables the users to store and access the data in a secured way through the internet. It also hosts the data which will decrease the work load on the system. This also maintain the continuity of data in all the local systems. Only the devices in the wi-fi network can access this so this minimizes the risk of attackers as well.in these ways this local cloud is very useful for small needs.

BIBLIOGRAPHY/REFERENCE

BIBLIOGRAPHY

WEB SITES REFERRED:

1. http://www.guru99.com
2. http://www.w3school.com
3. http://www.firstsiteguide.com
4. http://www.edureka.co