# An Exploration of Turing Pi Based Edge Cloud

**Sdmay24-03**

**Owen Perrin, Kale Kester, Nick Bergan, Andrew Phelps, Owen Hening, Cooper Caruso**

**Advisor: Akhilesh Tyagi**

# Introduction

*Problem*: While consumer cloud offerings like Amazon Web Services, Azure, or Google Cloud offer services which abstract resource consumption but provide a medium to deploy applications, some use cases necessitate a locally administered private cloud

*Solution:* Our team will create a proof of concept which explores fundamental cloud principles, to develop a private cloud using the Turing Pi 2.

# Design Approach

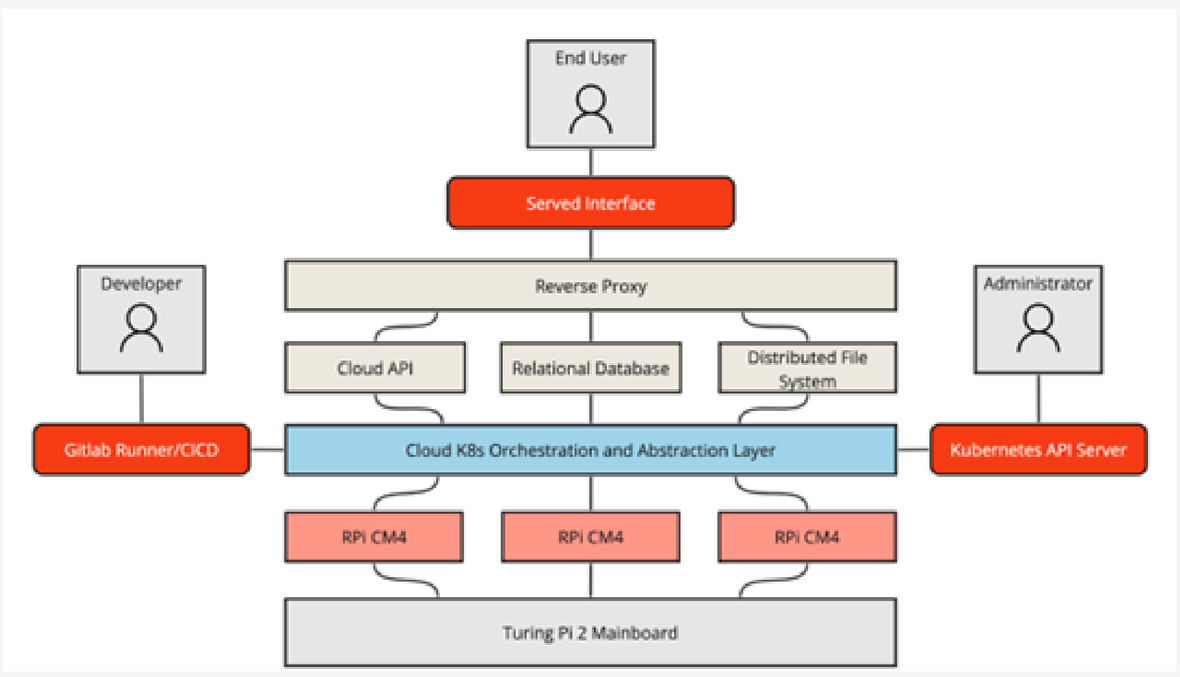



*Fig. 1* High-level design diagram of the cloud system. Higher levels of abstraction devolve into specific hardware as the diagram moves from top to bottom. Each main user of the cloud is also shown.

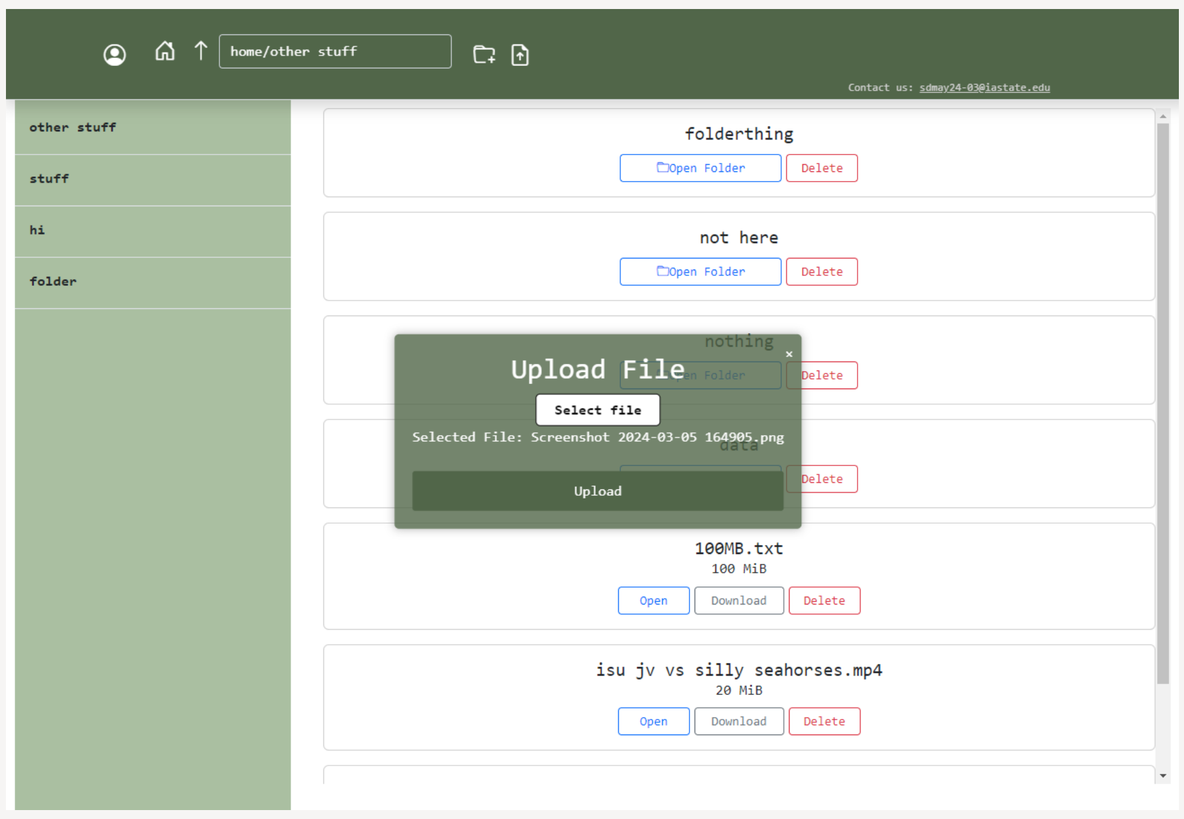



*Fig. 2* The User Interface implementation. The UI displays the files that have been uploaded along with options to download and delete them. Uploading files is also an option through the UI.

# Context

## *Intended Users and Use Cases:*

A data-sensitive user would also take advantage of this in-house, locally hosted server and workflow platform as it replicates the advantages of cloud-based scalability with the addition of having absolute control over the implementation in terms of security and accessibility.

Small business owners could make use of a similar platform and design to either run a handful of low resource-intensive programs or test out the concept of a private cloud based on scalable containers.

# Tests and Results

*Subsystems to test:*

- Private Cloud Stack
- Implemented Software Application
- DFS
- Hardware connection to Internet
- Overall management capabilities

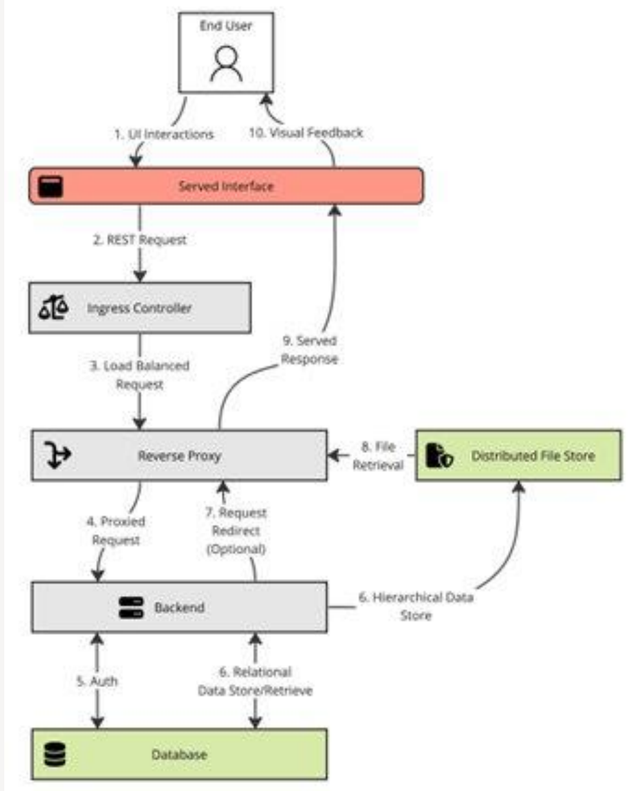
*Testing Method:*

- Unit tests, and White Box Testing
- Interface, Integration, and partial or whole system

*Maximum Tested Performance:*

- 41 MBPS transfer speed within the DFS storage
- 3.5 MBPS transfer speed between the DFS to either local storage or ethernet, pinned to a single node
- 4.1 MBPS from UI to DFS, through the entire system

# Technical Details




*Technology Used:*

- *Turing Pi 2 with 3 compute modules*
- *Kubernetes – Cloud abstraction layer*
- *Django – API*
- *Kubgres – Rational Database*
- *Ceph – DFS*
- *NGINX – Reverse Proxy Server*
- *React – User Interface*

# Design Requirements

## *Hardware Requirements:*

- The private cloud will be deployed on a Turing Pi 2
- The system will be able to support between 1-4 compute modules at any time

## *Cloud and Containerization Requirements:*

- Scale containerized applications across all clusters according to their resource needs
- Web API to deploy scalable containerized applications to the private cloud
- Expose API endpoints which support blob storage
- Robust monitoring via its interface which reports functional status (e.g. nominal, process failures) and resource utilization
- Support a containerized video streaming application
- Performance improvement for scalable containerized applications as more compute clusters are added

## *Constraints:*

- The blob storage will support at least 10 users concurrently downloading and/or uploading
- The containerized video streaming application, when deployed, will support 3 simultaneous streams with a 3500 Kbps bitrate (1080p30)
- The blob storage will have an effective throughput of at least 15 Mbps upload and download
- Files of up to 32Gb will be supported for upload and download by the blob storage