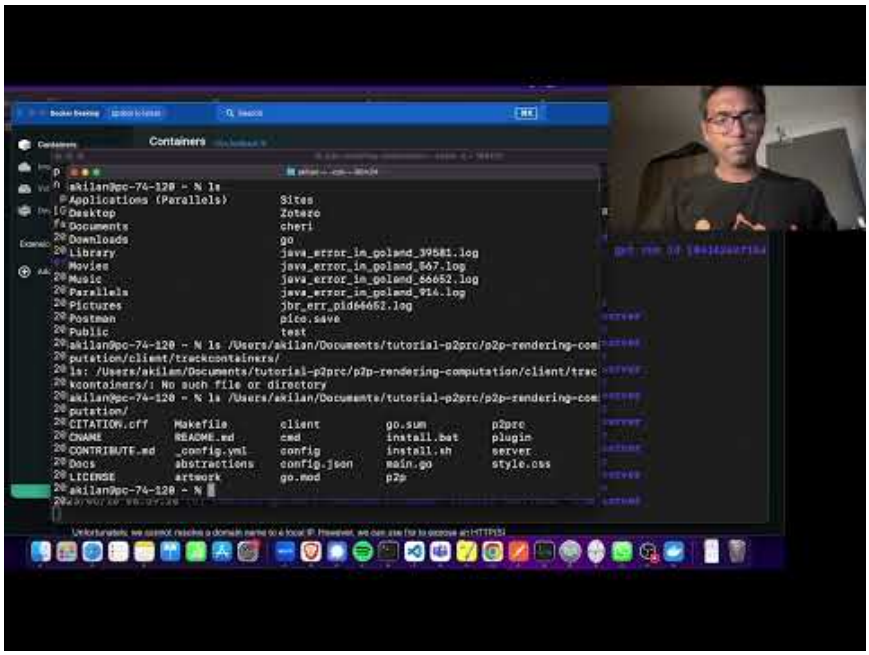
Contents

]]

The main aim of this project was to create a custom peer to peer network. The user acting as the client has total flexibility on how to batch the tasks and the user acting as the server has complete flexibility on tracking the container's usages and killing the containers at any point of time.

0.1 Latest tutorial



1 Table of contents in the Docs folder

1. Introduction
2. Installation
3. Abstractions
4. Implementation
   (a) Client Module
   (b) P2P Module
   (c) Server Module
   (d) Config Module

(e) Cli Module

(f) Plugin Module

5. Language bindings

   (a) Haskell

1.1 Introduction

This project aims to create a peer to peer (p2p) network, where a user can use the p2p network to act as a client (i.e sending tasks) or the server (i.e executing the tasks). A prototype application will be developed, which comes bundled with a p2p module and possible to execute docker containers or virtual environments across selected nodes.

1.1.1 Objectives

- Background review on peer to peer network, virtual environments, decentralized rendering tools and tools to batch any sort of tasks.
- Creating p2p network
- Server to create a containerised environment
- The client node to run tasks on Server containerised node

Read more on the introduction

1.2 Extend your application with P2PRC

```
package main

import (
    "fmt"
    "github.com/Akilan1999/p2p-rendering-computation/abstractions"
)

func main() {
    _, err := abstractions.Init(nil)
    if err != nil {
        fmt.Println(err)
        return
```

```
    }

    // start p2prc
    _, err = abstractions.Start()
    if err != nil {
        fmt.Println(err)
        return
    }

    // Run server till termination
    for {

    }
}
```

1.2.1 Export once this is added export P2PRC as environment paths

```
export P2PRC=<PROJECT PATH>
export PATH=<PROJECT PATH>:${PATH}
```

Read more ...

1.3 Installation from source

1. Ensure the Go compiler is installed

   ```
   go version
   ```

2. Ensure docker is installed (Should run without sudo)

   ```
   docker ps
   ```

3. Clone this repository

   ```
   git clone https://github.com/Akilan1999/p2p-rendering-computation
   ```

4. Install and build the project

   ```
   make install
   ```

5. If you look closely you will get outputs such as: `// Add them to your .bashrc file export P2PRC=/<path>/p2p-rendering-computation export PATH=/<path>/p2p-rendering-computation:${PATH}`

5. Test if it works

   ```
   p2prc -h
   ```

   or

   ```
   ./p2prc -h
   ```

   Read more on the installation and usage

1.4 Design Architecture

The design architecture was inspired and based on the linux kernel design. The project is segmented into various modules. Each module is responsible for certain tasks in the project. The modules are highly dependent on each other hence the entire codebase can be considered as a huge monolithic chuck which acts as its own library

Read more on the Design Architecture

1.5 Implementation

The programming language used for this project was Golang. The reason Go lang was chosen was because it is a compiled language. The entire codebase is just a single binary file. When distributing to other linux distributing the only requirement would be the binary file to run the code. It is easy to write independant modules and be monolithic at the sametime using Go. Using Go.mod makes it very easy to handle external libraries and modularise code. The go.mod name for the project is git.sr.ht/~akilan1999/p2p-rendering-computation.

Read more on the Implementation

1.6 Find out more

As we are working on the open source project p2prc (i.e p2p network designed for computation).If you are interested in participating as a contributor or just providing feedback on new features to build or even just curious about new features added to the project. We have decided to create a discord group.