RANCANG BANGUN PENYIMPANAN TERDISTRIBUSI MENGGUNAKAN BLOCKCHAIN DENGAN PROTOKOL INTERPLANETARY FILESYSTEM (IPFS)

SKRIPSI

Diajukan sebagai salah satu syarat untuk menyelesaikan Pendidikan Diploma Empat (D-4) Program Studi Teknik Komputer dan Jaringan Jurusan Teknik Elektro Politeknik Negeri Ujung Pandang

Muhammad Fadhil Abrar Lasawedi

42518017

PROGRAM STUDI TEKNIK KOMPUTER DAN JARINGAN

POLITEKNIK NEGERI UJUNG PANDANG

MAKASSAR

2022

HALAMAN PENGESAHAN

Skripsi dengan judul **RANCANG BANGUN PENYIMPANAN TERDISTRIBUSI MENGGUNAKAN BLOCKCHAIN DENGAN PROTOKOL INTERPLANETARY FILESYSTEM (IPFS)** oleh Muhammad Fadhil Abrar Lasawedi dengan NIM 425 18 017 dinyatakan layak untuk diujikan.

Makassar, 2022

Mengesahkan,

Pembimbing I,	Pembimbing II,
Meylanie Olivya,S.T.,M.T.	Ir.Dahlia,M.T.
NIP. 198205032014042002	NIP. 196412311991032003

Mengetahui,

Koordinator Program Studi

Teknik Komputer dan Jaringan

Eddy Tarmadi, S.T.,M.T.

NIP.197908232010121001

HALAMAN PENERIMAAN

Pada hari ini, Kamis tanggal 22 September 2022 Tim Penguji Ujian Sidang Skripsi telah menerima dengan baik skripsi oleh mahasiswa: **Muhammad Fadhil Abrar Lasawedi** nomor induk mahasiswa **42518017** dengan judul **"Rancang Bangun Penyimpanan Terdistribusi Menggunakan Blockchain Dengan Protokol Interplanetary Filesystem (IPFS)"**.

Makassar, 22 September 2022

Tim Penguji Ujian Sidang Skripsi:

1. Irfan Syamsuddin,S.T. M.Com.ISM.,Ph.D.	Ketua	(.....)
2. Muhammad Nur Yasir Utomo, S.ST., M.Eng.	Sekretaris	(.....)
3. Irmawati,S.T.,M.T.	Anggota	(.....)
4. Eddy Tungadi,S.T., M.T.	Anggota	(.....)
5. Meylanie Olivya,S.T.,M.T.	Anggota	(.....)
6. Ir.Dahlia,M.T.	Anggota	(.....)

KATA PENGANTAR

Puji dan syukur penulis panjatkan kehadirat Tuhan Yang Maha Esa yang telah melimpahkan rahmat dan karunia-Nya, sehingga penulisan skripsi ini yang berjudul "Rancang Bangun Penyimpanan Terdistribusi Menggunakan Blockchain Dengan Protokol Interplanetary Filesystem (Ipfs)" dapat diselesaikan dengan baik. Skripsi ini disusun guna memenuhi salah satu syarat untuk menyelesaikan studi serta dalam rangka memperoleh gelar Diploma IV (D-4/S1 Terapan) pada Program Studi Teknik Komputer dan Jaringan Jurusan Teknik Elektro Politeknik Negeri Ujung Pandang.

Penulis menyadari bahwa keberhasilan penyusunan skripsi ini tidak lepas dari bantuan berbagai pihak baik secara langsung maupun tidak langsung. Oleh karena itu, dengan rendah hati penulis mengucapkan terima kasih kepada:

1. Kedua orang tua penulis yakni Darmawan Lasawedi dan Eliaumra, Saudara penulis yakni Muhammad Fayi Afkar Lasawedi dan Muhammad Fayyadh Arhab Lasawedi serta keluarga besar penulis yang telah memberikan semangat, motivasi, dukungan, bimbingan, dan doa kepada penulis.
2. Bapak Ahmad Rizal Sultan, S.T., M.T., Ph.D. selaku Ketua Jurusan Teknik Elektro Politeknik Negeri Ujung Pandang.
3. Bapak Eddy Tungadi, S.T., M.T. selaku Koordinator Program Studi Teknik Komputer dan Jaringan.
4. Ibu Meylanie Olivya,S.T.,M.T. selaku pembimbing I dan Ibu Ir.Dahlia,M.T. selaku pembimbing II yang telah mencurahkan waktu dan kesempatannya untuk mengarahkan penulis dalam menyelesaikan skripsi ini.
5. Seluruh dosen dan Staf Jurusan Teknik Elektro, Khususnya Program Studi D4 Teknik Komputer dan Jaringan.
6. Teman-teman seperjuangan di Program Studi D-4 Teknik Komputer dan Jaringan angkatan 2018 yang mempunyai peranan besar dalam membantu menyusun skripsi ini dan mengajarkan banyak hal kepada penulis baik dari segi akademik maupun non akademik.

7. Semua pihak yang telah memberikan bantuan moril maupun materil yang tidak dapat disebutkan satu per satu.

Penulis menyadari bahwa skripsi ini masih kurang dari kata sempurna, sehingga penulis mengharapkan kritik dan saran yang sifatnya membangun untuk perbaikan di masa mendatang. Semoga tulisan ini bermanfaat.

Makassar,22 September 2022

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

SURAT PERNYATAAN

Saya bertanda tangan di bawah ini :

Nama : Muhammad Fadhil Abrar Lasawedi

NIM : 425 18 017

Menyatakan dengan sebenar-benarnya bahwa segala pernyataan dalam skripsi ini yang berjudul "**RANCANG BANGUN PENYIMPANAN TERDISTRIBUSI MENGGUNAKAN BLOCKCHAIN DENGAN PROTOKOL INTERPLANETARY FILESYSTEM (IPFS)**" merupakan gagasan dan hasil karya sendiri dengan arahan komisi pembimbing, dan belum pernah diajukan dalam bentuk apa pun pada perguruan tinggi dan instansi mana pun.

Semua data dan informasi yang digunakan telah dinyatakan secara jelas dan dapat diperiksa kebenarannya. Sumber informasi yang berasal atau dikutip dari karya yang diterbitkan dari penulis lain telah disebutkan dalam naskah dan dicantumkan dalam skripsi ini.

Jika pernyataan saya tersebut di atas tidak benar, saya siap menanggung risiko yang ditetapkan oleh Politeknik Negeri Ujung Pandang.

Makassar, 20 September 2022

Muhammad Fadhil Abrar Lasawedi

NIM. 42518017

RANCANG BANGUN PENYIMPANAN TERDISTRIBUSI MENGGUNAKAN BLOCKCHAIN DENGAN PROTOKOL INTERPLANETARY FILE SYSTEM (IPFS)

RINGKASAN

Beberapa tahun belakangan ini, internet sudah memasuki segala aspek kehidupan manusia. Salah satu teknologi yang digunakan untuk menunjang segala kebutuhan layanan internet adalah layanan penyimpanan data. Salah satu kelemahan teknologi penyimpanan data saat ini ialah sifatnya yang terpusat. Dimana semua data yang disimpan pada satu entitas dan dapat disalahgunakan oleh pihak tersebut. Oleh karena itu, diperlukan sebuah sistem penyimpanan yang terdistribusi dan memiliki validasi yang kuat. Salah satu teknologi yang memliki karaterstik tersebut ialah *Blockchain* dengan menggunakan protokol IPFS atau *Interplanetary File System.* Penelitian ini bertujuan untuk merancang dan membangun sebuah sistem IPFS yang bisa diterapkan di berbagai bidang, terutama di bidang yang membutuhkan penyimpanan data yang aman serta memiliki validasi data yang kuat. Penelitian ini berhasil membangun sebuah sistem penyimpanan berbasis IPFS dan semua fungsi dari sistem berjalan dengan baik. Adapun hasil pengujian performa pada sistem penyimpanan terdistribusi berbasis IPFS, rata-rata indeks yang didapatkan adalah 4 dengan kategori sangat baik. Nilai rata-rata throughput 6,8 Mbps termasuk dalam kategori sangat baik dengan indeks 4, begitupun *packet loss* dengan nilai rata-rata 0,068% termasuk dalam kategori sangat baik.

Kata Kunci: Penyimpanan Terdistribusi, IPFS, *Blockchain*

BAB I PENDAHULUAN

1.1 Latar Belakang

Beberapa tahun belakangan ini, internet sudah memasuki segala aspek kehidupan manusia. internet telah menjadi pusat pendistribusian segala jenis informasi, mulai dari berita, perdagangan, multimedia, dan lain sebagainya. Salah satu teknologi yang digunakan untuk menunjang segala kebutuhan layanan internet adalah layanan penyimpanan data.

Saat ini, teknologi penyimpanan yang umum digunakan ialah penyimpanan tersentralisasi. Salah satu kelemahan penyimpanan tersentralisasi ialah sifatnya yang terpusat. Dimana semua data yang disimpan pada satu entitas dapat disalahgunakan oleh pihak tersebut. Selain itu, kurangnya validasi data membuat penyimpanan tersentralisasi kurang efektif untuk diterapkan pada bidang perizinan, arsip, kesehatan, dan bidang-bidang lain yang membutuhkan validasi data yang kuat. Oleh karena itu, diperlukan sebuah sistem penyimpanan yang terdistribusi dan memiliki validasi yang kuat. Salah satu teknologi yang memliki karaterstik tersebut ialah *Blockchain* dengan menggunakan protokol IPFS atau *Interplanetary File System.*

Blockchain adalah database terdistribusi yang terdiri dari blok data yang saling berhubungan yang dilindungi dengan menggunakan konsep kriptografi. Konsensus node dalam jaringan *blockchain* menetapkan aturan untuk menggunakan dan memperbarui blockchain. Ini adalah kesepakatan antar *node* dalam jaringan *blockchain* yang sama misalnya Bitcoin atau Ethereum. Setiap *node* mungkin memiliki salinan *blockchain* yang biasa disebut *full node* atau bergantung pada *node* lain untuk informasi blockchain yang biasa disebut *lightweight node*(Sanka, 2021).

IPFS adalah sistem file terdesentralisasi yang memungkinkan distribusi konten dalam desentralisasi jaringan seperti beberapa sistem berbagi Peer-to-Peer. Ini juga mendukung hubungan yang aman di antara konten tersebut, serta memungkinkan penggunaan data yang kompleks struktur seperti yang digunakan di git atau blockchain (Tenorio-Fornés, 2017).

Penelitian ini bertujuan untuk merancang dan membangun sebuah sistem IPFS yang bisa diterapkan di berbagai bidang, terutama di bidang yang membutuhkan penyimpanan data yang aman serta memiliki validasi data yang kuat, contohnya pada bidang kesehatan, arsip, atau bahkan pada *Internet of Things*. Diharapkan penelitian ini bisa menjadi acuan untuk pengembangan aplikasi berbasis IPFS dimasa yang akan datang.

1.2 Rumusan Masalah

Bagaimana cara merancang dan membangun penyimpanan terdistribusi menggunakan IPFS (Interplanetary File System)?

1.3 Ruang Lingkup Penelitian

Penelitian ini hanya menbahas cara merancang dan membangun sistem penyimpanan terdistribusi menggunakan IPFS.

1.4 Tujuan Penelitian

Penelitian ini bertujuan untuk merancang dan membangun sistem penyimpanan terdistribusi menggunakan IPFS, yang nantinya akan diterapkan untuk menyelesaikan masalah-masalah pada sistem penyimpanan konvensional.

1.5 Manfaat Penelitian

Penelitian ini diharapkan dapat memberi manfaat secara akademik dan berdampak kepada instansi terkait khususnya yang membutuhkan sistem penyimpanan file berupa arsip, foto, dan sejenisnya.

1.5.1 Penulis

Berikut adalah manfaat penelitian kepada penulis:

1. Memberi wawasan mengenai cara merancang dan membangun sistem penyimpanan terdistribusi dengan menggunakan IPFS
2. Sebagai bahan referensi dalam pengembangan teknologi IPFS, khususnya IPFS yang terintegrasi dengan jaringan *blockchain*.

1.5.2 Masyarakat

Berikut adalah manfaat penelitian kepada masyarakat:

1. Sebagai salah satu sarana penyimpanan *file* secara online.
2. Sebagai salah satu literatur pengetahuan apabila instansi hendak mengimplementasikan teknologi IPFS dalam penyimpanan file.
3. Dengan sistem ini penyimpanan file menjadi lebih aman, karena file tersimpan di beberapa node dan juga telah terenkripsi.

BAB II TINJAUAN PUSTAKA

2.1 Studi Terkait

Beberapa studi telah mengkaji penerapan IPFS (*Inter Planetary File System*) di berbagai bidang. (Aziz, 2019) melakukan penelitian berjudul "Analisis dan Implementasi Komunikasi Antar Node IPFS Pada Smart Contract Ethereum". Penelitian ini bertujuan untuk membangun aplikasi berbasis IPFS dan mengukur QoS (*Quality Of Service*) dan integirtas data dari IPFS. Hasilnya, sistem yang telah dibangun berjalan dengan baik, efisien dan efektif, serta mampu menjaga integritas data dengan baik. Adapun hasil pengukuran QoS dengan parameter *throughput*, *packet loss*, dan *delay*, memperoleh rata-rata indeks 3 dengan kategori "Memuaskan".

(Fajar. 2020) telah melakukan penelitian berjudul "Quality Of Service Ethereum Blockchain Berbasis Ipfs Untuk Validasi Ijazah Sekolah". Penelitian ini bertujuan untuk menganalisis tingkat efisiensi dari sistem berbasis IPFS untuk validasi ijazah. Hasilnya, sistem berbasis IPFS lebih baik secara signifikan apabila dibandingkan dengan sistem yang tidak berbasis IPFS. Adapun hasil pengukuran QoS dari sistem tersebut ialah 4 dengan kategori "Baik".

(Nizamuddin, 2018) telah melakukan penelitian berjudul "IPFS-Blockchain-based Authenticity of Online Publications". Penelitian ini bertujuan untuk membuat sebuah sistem validasi keaslian dan buku-buku yang di-*publish* secara *online*. Hasilnya sistem yang dibuat berjalan dengan sangat baik.

(Sun, 2020) telah mengkaji penerapan sistem penyimpanan berbasis IPFS pada bidang kesehatan melalui penelitiannya yang berjudul "Blockchain-Based Secure Storage and Access Scheme for Electronic Medical Records in IPFS". Penilitian ini membahas tentang penerapan IPFS untuk menyimpan data *Electronic Medical Records*. Hasilnya sistem yang dibuat berjalan dengan sangat baik, namun masih terdapat kekurangan menurut penulis, karena sistem penyimpanan *medical records* yang telah dibuat tidak memperhitungkan pengguna yang sudah tidak memliki *medical records.*

Dari semua penelitian tersebut, dapat disimpulkan bahwa sistem atau aplikasi penyimpanan berbasis IPFS, dapat berjalan secara efisien dan efektif dibanding sistem penyimpanan lainnya, serta mampu menjaga integritas data yang dikelola.

2.2 Sistem Terdistribusi

Sistem terdistribusi adalah suatu kesatuan dari elemen-elemen yang saling berinteraksi secara sistematis untuk mendistribusikan data, informasi, proses, objek, dan layanan dari dan kepada pengguna yang terkait di dalamnya (Budi,2006).

Adapun infrastruktur utama dari aplikasi sistem terdistribusi (Setyoadi, 2012), meliputi:

1. Jaringan komputer baik dalam skala lokal (LAN), metropolitan (MAN), skala luas (WAN), maupun skala global (Internet).
2. Beragam perangkat keras dan lunak, serta pengguna yang berada dan saling terkait dalam sistem jaringan yang membentuknya.

Penerapan sistem terdistribusi merupakan bentuk usaha untuk memanfaatkan secara optimal sistem jaringan komputer yang dibangun di dalam perusahaan. Sistem terdistribusi dibangun dengan tujuan:

1. Mengatasi *bottleneck*.
   Tumpukan pekerjaan pada suatu terminal dapat didistribusikan ke terminal-terminal lain.
2. Mendukung layanan yang tersebar.
   Misalnya layanan penjualan dengan menggunakan terminal terminal yang tersebar di berbagai tempat.
3. Mendukung sistem kerja jarak jauh.
   Misalnya sistem kerja *small office home office* yang memungkinkan karyawan untuk bekerja dari rumah sehingga tidak harus datang ke kantor.
4. Memudahkan kerja kelompok.
   Dengan memudahkan *sharing* data dan tetap memungkinkan kerjasama walaupun letak anggota kelompok berjauhan.

Adapun beberapa jenis arsitektur sistem terdistribusi ialah sebagai berikut:

1. *Client Server*

   Client menghubungi *server* untuk mendapatkan data, yang kemudian memformat dan menampilakan pada pengguna

2. *Tightly Coupled* (*Clustured*)

   Mesin-mesin terintegrasi yang menjalankan proses yang sama secara bersamaan dengan membagi tugas ke dalam beberapa bagian yang dijalankan masing-masing mesin. Apabila proses telah selesai, hasil pengerjaan masing-masing mesin digabungkan menjadi satu.

3. *Peer-to-Peer*

   Arsitektur dimana tidak ada mesin yang menyediakan layanan atau mengelola sumber daya jaringan sehingga segala tanggung jawab dibagikan diantara seluruh mesin.

2.3 Blockchain

Blockchain adalah *database* dengan sistem terdistribusi yang bisa menyimpan data yang terus diperbarui. Sistem yang dijalankan oleh *blockchain* adalah salah satu contoh penerapan dari sistem terdistribusi dengan status bersama, dimana node didistribusikan secara geografis dan terhubung melalui *peer to peer* jaringan (Lone & Mir, 2019). Setiap data yang didistribusikan divalidasi oleh semua *node* yang bergabung (Yli-Huumo , 2016). Sehingga apabila ada perbedaan salinan data diantara salah satu *peer, peer* yang lain bisa mencegah data tersebut diproses. *Blockchain* mememiliki beberapa macam diantaranya (Laurance, 2017).

a. *Blockchain* Publik

   Blockchain ini berupa sebuah jaringan terdistribusi besar yang dijalankan melalui *native token,* contohnya *bitcoin*. Jaringan ini terbuka untuk siapapun yang ingin mengembangkan sistem didalamnya. Serta *source code*-nya dipelihara secara *open source* oleh komunitas yang terlibat.

b. *Blockchain* Permisif

   Blockchain Permisif adalah *Blockchain* yang memberikan syarat bagi pengembang yang ingin mengembangkan sistem di dalam jaringan Blockchain

tersebut. Layaknya *blockchain* publik, *blockchain* permisif ini juga dijalankan melalui *native token*. *Source code* yang disediakan dapat bersifat *open source* atau tidak.

c. *Blockchain* Privat

Blockchain privat adalah *blockchain* berskala kecil dan tidak memerlukan *native token* dalam pengoperasiannya. *Blockchain* ini membatasi anggota yang ingin berpartisipasi kepada pihak tertentu saja, sehingga *blockchain* jenis ini sering digunakan oleh konsorsium dengan anggota tepercaya dan mentransaksikan informasi rahasia.

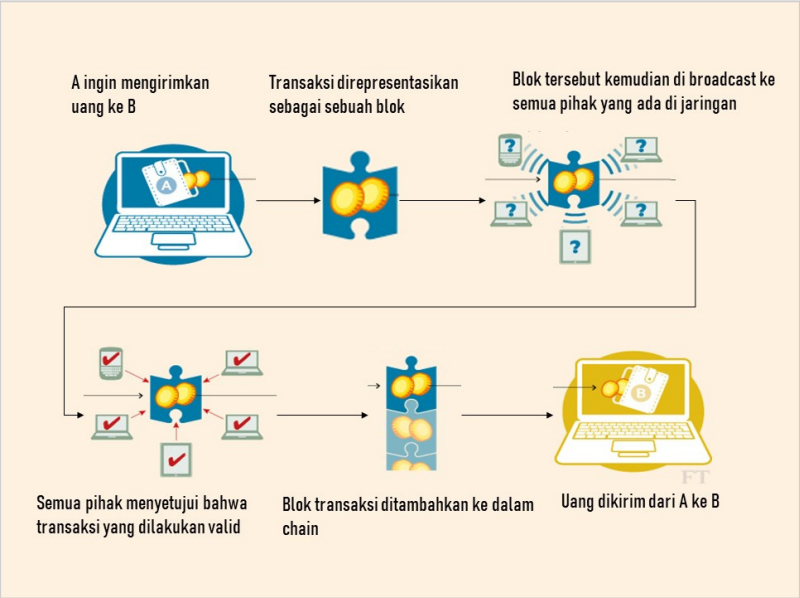
Menurut (Sultan, 2018), Blockchain memiliki beberapa karakteristik yang menjadi keunggulan dari teknologi ini. Berikut ini beberapa karakteristik Blockchain:

a. *Immutable*, adalah sifat dimana sebuah data atau variabel tidak dapat diubah lagi setelah nilainya diberikan. Salah satu cara untuk membuat sebuah data *immutable* adalah dengan menuliskannya pada media sekali tulis seperti pada CD atau DVD, cara lainnya adalah menggunakan metode kriptografi yang menghasilkan *immutability* semu, semu karena secara fisik data tetap dapat diubah namun secara logika dapat diperiksa dengan menggunakan perhitungan matematis. Teknologi *Blockchain* menggunakan metode kriptografi untuk mengamankan data transaksi dari adanya perubahan tanpa izin.
b. *Decentralized*, suatu data pada jaringan *blockchain,* dapat diakses dan disalin oleh semua pihak yang berpartipasi di dalam jaringan. Hal ini menciptakan desentralisasi data, dimana semua pihak memiliki salinan data yang terdapat di dalam *blockchain*.
c. *Consensus Driven*, setiap *block* pada jaringan *blockchain* divalidasi melalui sebuah *Consensus Model* yang menyediakan syarat untuk memvalidasi sebuah *block*. Proses ini berlangsung tanpa adanya otoritas pusat atau pihak ketiga yang melakukan proses validasi.
d. Transparan, semua pihak di jaringan *Blockchain* mempunyai akses ke seluruh database dan riwayat transaksi secara lengkap. Tidak ada satu pihak pun yang memegang kendali atas data atau informasi yang terdapat pada jaringan

blockchain. Setiap pihak dapat memverifikasi catatan mitra transaksinya secara langsung, tanpa perantara.

Gambaran mengenai cara kerja *blockchain* dapat diperoleh dengan mengetahui bagaimana jaringan *blockchain* bekerja. Jaringan tersebut adalah kumpulan *node* yang beroperasi pada Blockchain yang sama (Arief, 2017). Suatu *node*, secara umum, dapat berfungsi sebagai titik masuk untuk pengguna yang berbeda-beda pada *blockchain*, tetapi untuk mempermudah, setiap pengguna dianggap bertransaksi pada *blockchain* melalui *node* mereka sendiri. Berikut adalah gambaran mengenai cara kerja *blockchain*:

1. Pengguna berinteraksi dengan Blockchain melalui sepasang *public* dan *private key*. Mereka menggunakan *private key* untuk menandai transaksi mereka sendiri, dan alamat mereka dapat ditelusuri melalui *public key* mereka yang tersedia di jaringan. Penggunaan kriptografi asimetris membawa integritas, otentikasi, dan nonrepudiation ke dalam jaringan. Setiap transaksi yang ditandatangani disiarkan melalui *node* pengguna ke *peer* yang bertetangga
2. *Peer* yang bertetangga memastikan bahwa transaksi ini valid sebelum me-relay lebih jauh. Transaksi yang tidak valid akan diabaikan. Pada akhirnya, transaksi akan disebarkan ke seluruh jaringan.
3. Transaksi yang telah dihimpun dan divalidasi oleh jaringan menggunakan proses di atas dalam rentang waktu yang disepakati, diurut dan dipaketkan pada kandidat block yang diberi timestamp. Proses ini disebut dengan mining. Node mining akan menyebarkan kembali blok ini ke dalam jaringan.
4. Node-node lain akan memverifikasi bahwa blok yang disarankan mengandung transaksi yang valid, dan merujuk lewat hash blok sebelumnya dari rantai yang tepat. Apabila terjadi demikian, blok tersebut akan ditambahkan ke dalam rantai. Apabila sebaliknya, blok tersebut akan diabaikan. Ini menandai akhir dari suatu siklus.



Gambar 2.1 Cara Kerja *Blockchain* yang Digambarkan dengan Transaksi Uang

(Sumber: https://www.itworks.id/)

Pada gambar 2.1 dapat dilihat cara kerja *blockchain* yang digambarkan dengan transaksi uang, proses ini berlangsung secara terus-menerus. Pada dasarnya, *blockchain* merupakan kumpulan beberapa *node* yang tidak mempercayai satu sama lain yang berbagi database tanpa adanya pihak perantara yang dipercaya.

2.4 IPFS (*Interplanetary File System*)

IPFS adalah protokol *hypermedia peer to peer* terdistribusi yang dirancang untuk menyimpan konten digital dengan integritas tinggi dan aksesibilitas global (Nizamuddin, 2018). File yang sudah tersimpan di IPFS menghasilkan hash file dan disimpan pada blockchain Ethereum. Kemudian hash file yang sudah disimpan pada blockchain dapat dipanggil kembali pada file yang disimpan pada IPFS untuk bisa mengaksesnya kembali (Rajalakshmi, 2018).

InterPlanetary File System (IPFS) menyediakan *platform* baru untuk mengembangkan aplikasi, dan mendistribusikan data yang besar. IPFS berjalan secara *peer-to-peer*. *Node* IPFS menyimpan objek IPFS di penyimpanan lokal. Semua *node* terhubung satu sama lain dan melakukan transfer objek. Objek-objek ini mewakili file dan struktur data sejenisnya. (Benet, 2014).

Saat file diunggah ke IPFS, file tersebut akan dibagi menjadi beberapa bagian, masing-masing berisi paling banyak 256 kb data atau *link* ke potongan lainnya. Setiap potongan diidentifikasi oleh *hash* kriptografik, yang bernama *content identifier* (CID), yang dihitung dari isi. *Link* tersebut juga berisi CID, sehingga membentuk *Directed Acyclic Graph* Merkle (Merkle DAG) yang menggambarkan file secara keseluruhan dan dapat digunakan untuk merekonstruksi file apa pun dari potongannya. Karena Merkle DAG, seluruh file dapat diidentifikasi hanya dengan menggunakan hash root. Ketika *node* telah membagi file menjadi beberapa bagian, dan Merkle DAG telah terbentuk, *node* tersebut mendaftarkan dirinya sebagai *provider* melalui DHT (*Distributed Hash Table*).

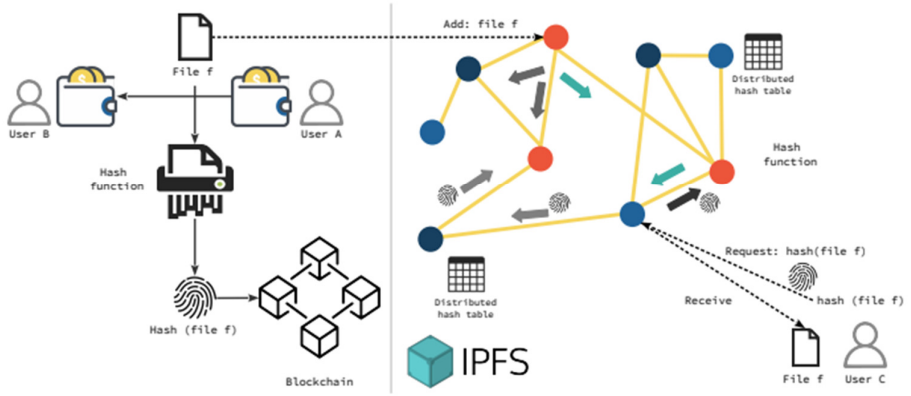
DHT pada dasarnya adalah penyimpanan *key-value* terdistribusi (Steichen, 2018). DHT menggunakan *node identifier* dan *key*, keduanya harus memiliki panjang yang sama, bersama dengan jarak metrik untuk dengan mudah menyimpan dan mengambil informasi. Saat mencari sebuah data atau nilai, sebuah *node* berusaha mencari *node identifier* yang menyerupai *key* dan mengambil data dari mereka. Itu dilakukan dengan menggunakan *bucket* untuk melacak *node* dalam jaringan. *Bucket* tersebut diatur sedemikian rupa sehingga setiap *node* dalam jaringan memiliki informasi yang tepat.

DHT menyimpan dua jenis informasi. Pertama, setiap kali file diunggah melalui *node*, *node* tersebut mendaftarkan dirinya sebagai penyedia dari potongan *file*. Kedua, DHT berisi informasi tentang cara membuat koneksi ke *node* dengan id tertentu, contohnya dengan memberikan alamat IP. Dengan demikian, *node* IPFS dapat meminta alamat *node* melalui DHT dan kemudian menyambungkannya ke *node* tersebut untuk mengambil file.

Karena penggunaan *content identifier* oleh IPFS untuk mengidentifikasi, memverifikasi dan mentransfer file, IPFS ini sangat cocok untuk digunakan dengan *blockchain*. Faktanya, *hash root* dari *file* dapat dikirim ke blockchain melalui sebuah transaksi, serta tidak ada informasi selain *hash* yang diperlukan untuk mengambil *file* dari IPFS. Selain itu, *file* berbeda dengan hash yang sama tidak dapat dengan mudah dibuat, jadi kecil kemungkinan terjadinya duplikasi *hash* yang dapat mengakibatkan ketidakcocokan data. Dari karakteristik-

karakteristik yang telah dijelaskan dapat disimpulkan bahwa, *file* yang disimpan di IPFS dapat dengan mudah diverifikasi dan kecil kemungkinan untuk menyimpan file yang berbeda dengan *identifier* yang sama.

Pada jaringan IPFS, node menyimpan kumpulan objek (*hash file*). penyimpanan lokal, dan mereka terhubung satu sama lain untuk mentransfer objek. *Node*, dalam hal ini pengguna, tidak diharuskan untuk menyimpan semua data yang diterbitkan dalam jaringan. Alih-alih, mereka dapat memilih data mana yang ingin simpan. Pengguna yang ingin mengambil salah satu dari file tersebut mengakses lapisan abstraksi di mana mereka cukup memanggil hash dari file yang mereka inginkan. IPFS kemudian, setelah mencari secara seksama melalui node, mencari *node* terdekat yang memiliki apa yang mereka butuhkan dan mengirimkan pengguna file yang diinginkan. *File* yang diakses di-*cache* secara lokal pada node IPFS. Hal ini membuat *file* tersebut tersedia untuk diunggah ke node lain dan dengan demikian membantu dengan distribusi beban untuk konten populer. (Politou, 2020)



Gambar 2.2 Gambaran cara kerja IPFS

(Sumber : Politou, 2020)

Pada gambar 2.2 dipaparkan cara kerja sistem IPFS dalam menyimpan data atau file. Pada bagan sebelah kiri, digambarkan operasi ketika data disimpan dalam IPFS atau DFS lainnya dan hanya hash dan metadatanya yang disimpan di blockchain. Di bagan sebelah kanan, digambarkan secara rinci tentang penyimpanan dan pengambilan data IPFS. Pertama, pengguna menyimpan file f di IPFS. Selanjutnya, untuk mengambil data tersebut, pengguna lain melakukan

permintaan untuk file f menggunakan nilai hash yang sesuai. Dikarenakan node mengetahui lokasi file tersebut, mereka dapat melakukannya secara efisien mengambil data dari node yang menyimpan file yang diminta (node oranye) untuk mengirimkannya ke pengguna akhirnya.

Berbeda dengan pengalamatan lokasi tradisional yang digunakan oleh HTTP di mana satu server menghosting banyak file dan informasi harus diambil dengan mengakses server tersebut, IPFS menggunakan metode pengalamatan konten, yaitu mencari konten dengan hash kriptografisnya, memastikan keaslian konten di mana pun itu berada. Implikasi dari metode ini luar biasa karena IPFS dapat mengubah Internet dari berbasis lokasi, menjadi jaringan file terdistribusi berbasis konten. Pertama dan terpenting, IPFS menghilangkan masalah tautan HTTP rusak karena alamat yang diberikan akan selalu mengarah ke konten yang sama yang ditambahkan jaringan IPFS karena bahkan sedikit perubahan akan menghasilkan alamat yang berbeda. Seperti yang telah disebutkan, kelebihan lain dari IPFS adalah ketahanan sensornya karena konten web tidak lagi bergantung pada satu entitas. Sifat bebas sensor telah dimanfaatkan dalam banyak kesempatan untuk melewati batasan kebijakan web dan untuk memungkinkan kebebasan berbicara dan hak untuk mengakses informasi. Dalam hal ini, dikatakan bahwa IPFS dapat mengembangkan web dan bahkan mengganti "sistem file terdistribusi" paling sukses yang pernah ada, yakni HTTP.

2.5 Node.js

Karena menggunakan blockchain pribadi atau lokal yang berjalan, kami perlu mengonfigurasi aplikasi untuk mengembangkan smart contract (Bhosale, 2019). Untuk itu, akan memerlukan Node Package Manager atau NPM yang mencakup Node Js.

2.6 Truffle

Truffle merupakan tool penting untuk mengembangkan sebuah smart contract yang nantinya diunggah ke dalam jaringan blockchain Ethereum. Karena

di dalam framework Ini menggunakan Bahasa pemrograman Solidity untuk mengembangkan smart contract.

2.7 Metamask

Metamask adalah salah satu aplikasi desentralisasi yang tertua dan banyak digunakan pada jaringan Ethereum. MetaMask merupakan ekstension dari chrome, berfungsi sebagai tempat penyimpanan Ether.

2.8 Ganache

Ganache adalah jaringan *blockchain* pribadi, yang merupakan *blockchain* pengembangan *server* lokal yang dapat digunakan untuk bertindak seperti *blockchain* publik (Bhosale, 2019). Ganache digunakan untuk *deploy smart contract* dan untuk menjalankan tes. Ganache menyediakan 10 akun dengan 100 Ethereum untuk menguji *smart contract* yang ada di *blockchain* local.

2.9 *Quality of Service*

Quality of Service (QoS) digunakan untuk mengukur seberapa baik jaringan dan upaya untuk menentukan karakteristik dan sifat dari sebuah layanan (Wulandari, 2016). Pengukuran kualitas jaringan dikategorikan berdasarkan standar THIPON. Parameter quality of service terdiri dari :

1. Throughput

Throughput yaitu kecepatan transfer data efektif, yang diukur dalam bps (*bit per second*). Throughput adalah jumlah total kedatangan paket yang sukses yang diamati pada tujuan selama interval waktu tertentu dibagi oleh durasi interval waktu tersebut. Kategori *Throughput* diperlihatkan di tabel 2.1

Tabel 2.1 Kategori *Throughput* Standar THIPON

Kategori	***Throughput* (bps)**	**Indeks**
Sangat Bagus	100	4
Bagus	75	3
Sedang	50	2
Jelek	<25	1

Persamaan perhitungan *Throughput*:

$$Throughput = \frac{Paket\ data\ diterima}{Lama\ Pengamatan} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.1)$$

2. Packet Loss

Packet Loss merupakan suatu parameter yang menggambarkan suatu kondisi yang menunjukkan jumlah total paket yang hilang dapat terjadi karena *collision* dan *congestion* pada jaringan. Indeks dan kategori *packet loss* ditunjukkan pada Tabel 2.2.

Tabel 2.2 Kategori *Packet Loss* Standar THIPON

Kategori	***Packet Loss* (%)**	**Indeks**
Sangat Bagus	0	4
Bagus	3	3
Sedang	15	2
Jelek	25	1

Persamaan perhitungan *Packet Loss*:

$$Packet\ Loss = \frac{Paket\ data\ dikirim - Paket\ data\ diterima) \times 100\%}{Paket\ data\ yang\ dikirim} (2.2)$$

BAB III METODE PENELITIAN

3.1 Tempat Penelitian

Tempat penelitian dilaksanakan di Laboratorium Tugas Akhir Kampus 1 Politeknik Negeri Ujung Pandang, Tamalanrea Indah, Tamalanrea, Tamanlanrea Indah, Kec. Tamalanrea, Kota Makassar, Sulawesi Selatan 90245. Dimulai dari bulan Februari 2022 sampai dengan bulan Agustus 2022.

3.2 Perangkat Penelitian

Berdasarkan latar belakang terkait pengembangan teknologi sistem penyimpanan terdistribusi berbasis blockchain menggunakan protokol *Interplanetary File System*, Berikut rincian kebutuhan pengembangan penelitian dan instrumen perangkat keras yang digunakan.

3.2.1 Perangkat Keras

Tabel 3.1 Perangkat Keras

No	Perangkat Keras	Spesifikasi	Keterangan
1.	1 Unit Laptop	Processor: AMD A8-6410 RAM: 16 GB Storage: 480 GB SSD	Sebagai perangkat yang digunakan untuk membuat *node* virtual dan membangun aplikasi client
2.	3 *Virtual Node Server*	Processor : 2 Cores RAM : 2 GB Storage : 60 GB	Sebagai perangkat yang digunakan untuk *node* IPFS

3.2.2 Perangkat Lunak

Tabel 3.2 Perangkat Lunak

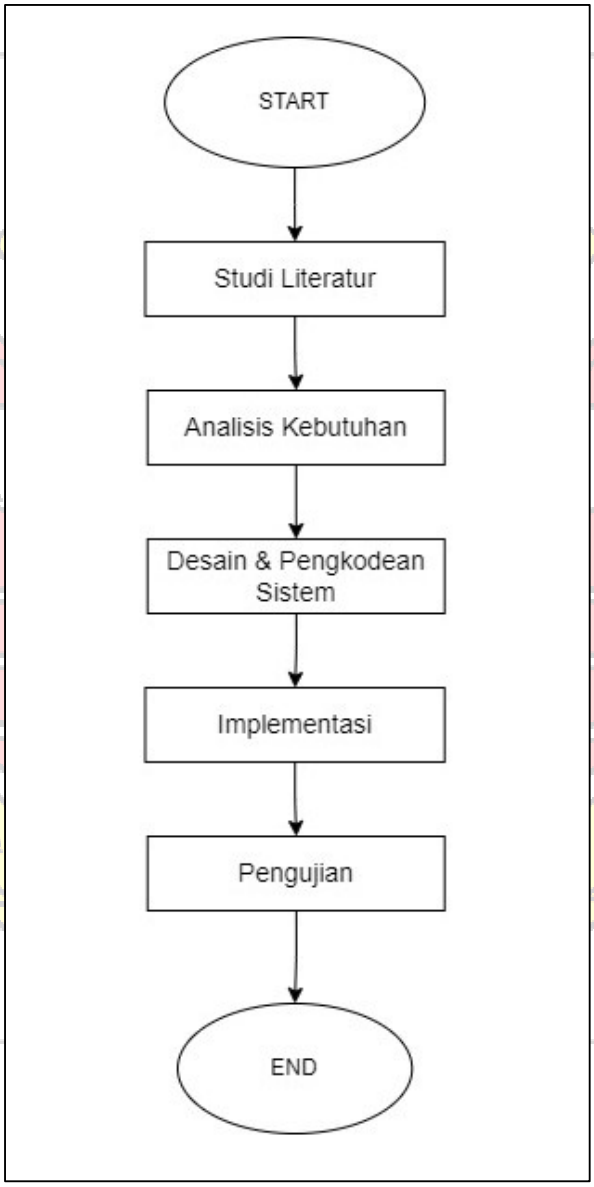
No	Perangkat Lunak	Spesifikasi	Keterangan
1.	Vmware Workstation	Versi 16.0	Sebagai aplikasi yang digunakan untuk membuat *node* IPFS virtual

Tabel 3.2 Perangkat Lunak

2.	Ubuntu Server	Versi 18.04	Sistem operasi *node* IPFS
3.	Visual Studio Code	Versi 1.7.11	Aplikasi yang digunakan untuk pengkodean sistem *user interface*
4.	Node Js	Versi 18.6.0	Package yang diperlukan untuk pengkodean sistem
5.	React Js	Versi 17.0.2	*Framework* yang digunakan untuk membangun *user interface*
5.	Truffle	Versi 5.3.13	*Package* yang diperlukan untuk pengkodean sistem
6.	Metamask	Versi 10.18.4	Aplikasi yang digunakan untuk menghubungkan *user interface* ke *blockchain*
7.	Ganache	Versi 2.5.4	Sebagai *blockchain* lokal
8.	Google Chrome	Versi 105.0	Sebagai *browser* ujicoba *user interface*
9.	Wireshark	Versi 4.0.0	Sebagai aplikasi yang digunakan pada pengujian sistem

3.3 Metode Penelitian

Agar penelitian yang dilakukan dapat berjalan dengan baik dan terstruktur diperlukan sebuah prosedur penelitian sehingga hasil yang diperoleh sesuai dengan tujuan penelitian. Penjelasan mengenai tahapan proses penelitian yang dilakukan digambarkan pada Gambar 3.1.



Gambar 3.1 Tahapan Proses Penelitian

3.3.1 Studi Literatur

Tahapan pertama adalah melakukan *literature review* untuk memahami apa saja yang dibutuhkan untuk implementasi blockchain dalam sistem penyimpanan. Selain itu juga untuk menelusuri lebih dalam dari permasalahan yang terjadi saat ini. Sumber untuk materi didapatkan dari *e-book*, jurnal dan lainnya.

3.3.2 Analisis Kebutuhan

Di dalam proses ini melakukan analisis yang kebutuhan calon pengguna aplikasi untuk memahami aplikasi apa saja yang dibutuhkan oleh user dan kebutuhan apa saja yang diperlukan untuk mengembangkan aplikasi tersebut. Untuk menjalankan aplikasi ini user harus menggunakan browser yang sudah mendukung metamask.

3.3.3 Desain & Pengkodean Sistem

Tahapan ini berfokus pada desain pembuatan aplikasi tersebut termasuk arsitektur aplikasi, *user interface*, dan proses pengkodean. Pada tahapan ini menerangkan apa saja yang dibutuhkan aplikasi dari tahap menganalisis kebutuhan sampai tahap desain untuk bisa digunakan sebagai aplikasi pada alur *implementation*.

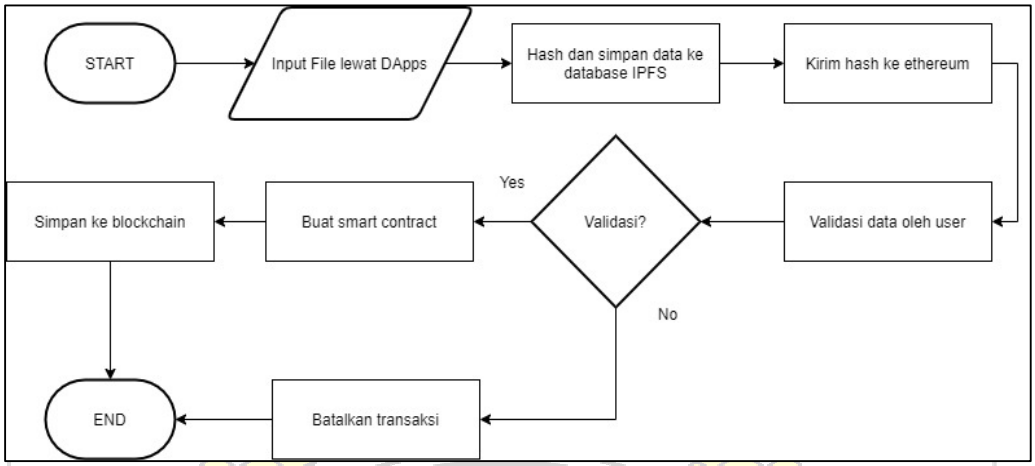
3.3.4 Implementasi

Pada tahap ini, perancangan perangkat lunak diimplementasikan sebagai sebuah program yang nyata.

3.3.5 Pengujian

Program diuji untuk memastikan apakah sesuai dengan kebutuhan atau tidak. Setelah pengujian, perangkat lunak dapat dikirimkan ke *user*.

3.4 Gambaran Sistem



Gambar 3.2 Flowchart Sistem

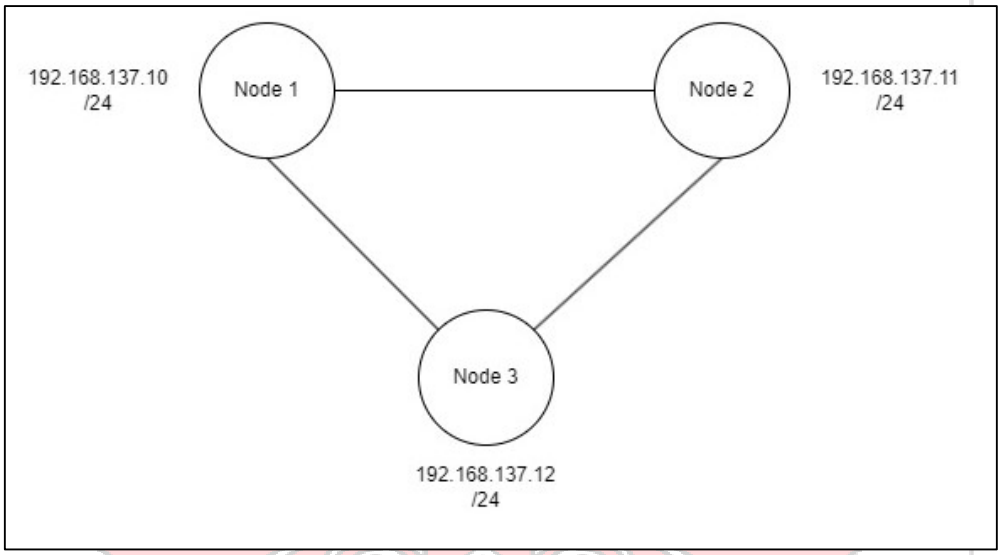
Berikut adalah gambaran mengenai alur kerja dari sistem penyimpanan terdistribusi berbasis IPFS pada penelitian ini berdasarkan gambar 3.2.

1. User menginput *file* melalui web dengan DApps, lalu *file* tersebut diubah menjadi *buffer*
2. Setelah proses konversi menjadi *buffer* telah selesai
3. Lalu, Ipfs-api akan mengunggah *file* menuju IPFS.
4. IPFS akan melakukan pengecekan isi *file*. Selanjutnya IPFS akan mengembalikan dalam bentuk IPFS *hash*.
5. IPFS akan mengirimkan beberapa blok ke sejumlah *node blockchain* yang terhubung.
6. IPFS akan mengirimkan data berupa *hash* dari *file* ke dalam sistem web DApp dan mengirimkan *raw data* berupa IPFS hash menuju Ethereum *network* yang ditulis kedalam *smart contract*. Pengiriman dilakukan dengan menggunakan ipfs-api
7. Pada saat proses pengiriman *raw data* IPFS *hash* menuju Ethereum network metamask akan meminta validasi dari transaksi kepada *user*.
8. Setelah *user* memberikan validasi maka pengiriman *raw data* berupa IPFS *hash* akan disimpan kedalam penyimpanan Blockchain. Sebagai bukti dari transaksi yang dilakukan ethereum akan mengirimkan data berupa *transaction hash*, *block*, serta *gas used*.

3.5 Rancangan Arsitektur

Berikut adalah rancangan arsitektur sistem penyimpanan terdistribusi berbasis IPFS yang dibuat pada penelitian ini. Rancangan arsitektur sistem meliputi, rancangan jaringan IPFS, dan rancangan sistem aplikasi yang digunakan oleh *client*.

3.5.1 Rancangan Arsitektur *Node* IPFS

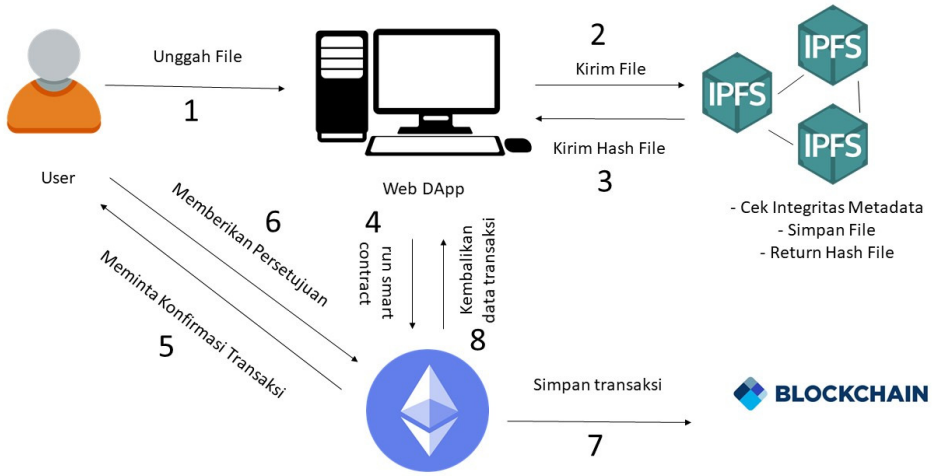


Gambar 3.3 Arsitektur *Node* IPFS

Pada sistem ini terdapat 3 komputer yang disebut dengan *node*. *Node* 1 berfungsi sebagai *node bootstrap*. *Node bootstrap* atau dapat disebut sebagai *node* utama. *Node* ini dijadikan sebagai pusat dari jaringan IPFS yang dibuat. Setiap *node* yang ingin bergabung dengan jaringan IPFS harus terhubung dengan *node* ini sebelum terhubung dengan *node* 2 dan 3.

Jaringan menggunakan *network address* 192.168.137.0 /24. Node 1 menggunakan alamat 192.168.137.10, node 2 menggunakan alamat 192.168.137.11, dan node 3 menggunakan alamat 192.168.137.12. Semua *node* saling terhubung satu sama lain.

3.5.2 Rancangan Arsitektur Aplikasi *Client*



Gambar 3.4 Arsitektur *Decentralized App* Berbasis IPFS

Pada penelitian ini dibangun suatu sistem atau aplikasi yaitu DApp (*Decentralize Application*) berbasis web dengan menggunakan NPM. Pada sistem web DApp dilakukan pengintegrasian IPFS dan *smart contract* Ethereum menggunakan ipfs-api. (Gambar 3.4)

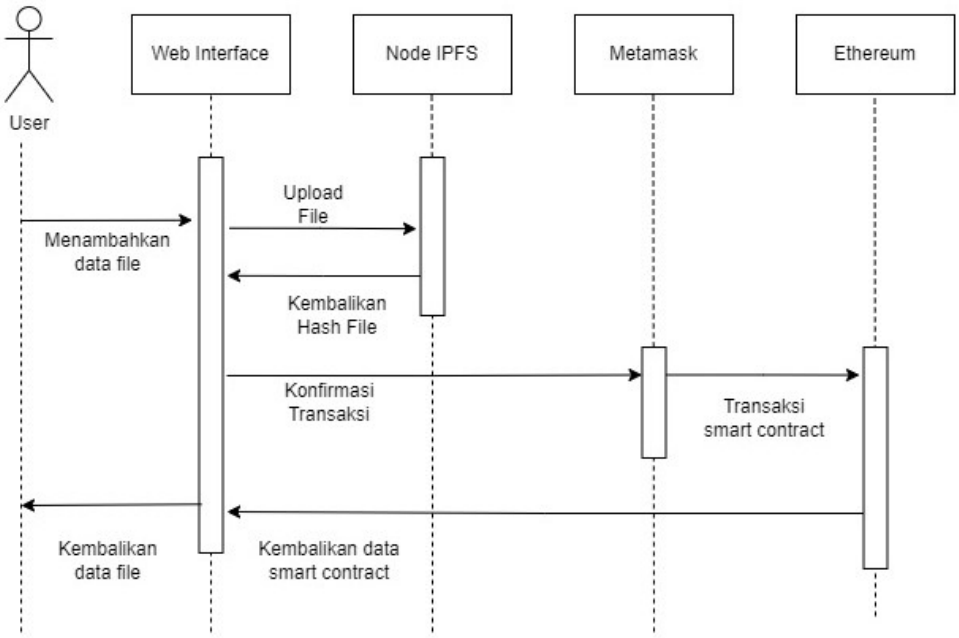
1. Masing – masing Laptop ter-*install* aplikasi, sistem operasi dan sistem web DApp yang sama
2. Pada saat membangun sistem web DApp dilakukan pembuatan kontrak menggunakan kode program yang ditulis kedalam Solidity melalui web remix.ethereum.org
3. Sistem web DApp berfungsi sebagai aplikasi yang menjalankan program Node.js dan NPM yang terdapat beberapa *dependency*.
4. User mengunggah *file* melalui sistem web DApp, selanjutnya *browser* akan mengubah *file* menjadi *buffer*.
5. Setelah proses pengubahan *file* menjadi *buffer* telah selesai, dilanjutkan dengan proses pengunggahan *file* menggunakan ipfs-api
6. Ipfs-api akan memproses pengunggahan *file* menuju IPFS. Pada saat proses pengiriman data melalui IPFS, konfigurasi IPFS menggunakan port 4001 pada saat proses pengiriman data.

7. IPFS akan melakukan pengecekan konten *file* berdasarkan *metadata* dari *file* tersebut. Selanjutnya IPFS akan mengembalikan dalam bentuk IPFS *hash*.
8. IPFS akan mengirimkan beberapa blok ke sejumlah *node* yang terhubung. Setelah proses pendistribusian *file* selesai dilakukan IPFS akan mengirimkan data berupa *hash* dari file ke dalam sistem web DApp dan mengirimkan *raw data* berupa IPFS *hash* menuju Ethereum network yang ditulis kedalam *smart contract*. Pengiriman *raw data* berupa IPFS *hash* dilakukan dengan menggunakan ipfs-api.
9. Pada saat proses pengiriman *raw data* IPFS hash menuju Ethereum *network metamask* akan meminta konfirmasi dari transaksi kepada *user*. Selanjutnya *user* akan memberikan persetujuan dari transaksi tersebut.
10. Setelah *user* memberikan persetujuan maka pengiriman *raw data* berupa IPFS *hash* akan disimpan kedalam penyimpanan *Blockchain*. Sebagai bukti dari transaksi yang dilakukan user akan meminta transaksi *receipt* dari Ethereum. Ethereum akan mengirimkan data berupa *transaction hash*, *block*, serta *gas used*.

3.5.3 Rancangan UML Sequence Diagram

Dalam menggambarkan perilaku dalam suatu skenario serta memberi gambaran bagaimana suatu entitas dan sistem saling berinteraksi, termasuk pesan yang digunakan dalam berinteraksi maka dapat dibuat dalam model *sequence diagram*. Visualisasi dari *sequence diagram* metode IPFS ditampilkan pada Gambar 3.5.

Pada Gambar 3.5 terdapat 1 pemeran dengan 4 objek yang meliputi Web DApp, API IPFS, MetaMask dan Ethereum. Proses yang dilakukan oleh user sesuai dengan yang ditampilkan pada sequence diagram metode IPFS diawali dengan user menambahkan data file dan diakhiri oleh sistem menampilkan kembali data file yang telah diidentifikasi oleh sistem.



Gambar 3.5 *Sequence Diagram* IPFS

3.6 Implementasi

Pada penelitian ini membuat sistem penyimpanan terdistribusi menggunakan blockchain berbasis IPFS. Tahap pertama dalam implementasi pada penelitian ini adalah melakukan proses instalasi node IPFS. Setelah itu membuat user interface (UI) sebagai sarana demonstrasi, dan yang terakhir. mengembangkan smart contract sebagai wadah penyimpanan data file.

3.6.1 Instalasi Node IPFS

Tahap ini adalah langkah awal untuk membuat sistem penyimpanan terdistribusi. Perangkat yang dibutuhkan adalah *personal computer* (PC) yang berbasis Ubuntu Linux. Dalam penelitian ini digunakan 3 buah perangkat PC yang disebut *node*. Berikut adalah langkah-langkah proses instalasi IPFS setiap node.

1. Instalasi IPFS

   Package yang digunakan adalah go-ipfs versi 0.4.18. Unduh package tersebut lalu unzip dan pindahkan ke dalam direktori /bin, dengan perintah

berikut

```
1. wget https://dist.ipfs.io/go-ipfs/v0.4.18/go-
   ipfs_v0.4.18_linux-amd64.tar.gz
2. tar xvfz go-ipfs_v0.4.18_linux-amd64.tar.gz
3. sudo mv go ipfs/ipfs /usr/local/bin/ipfs
```

Gambar 3.6 *Script* Untuk Instalasi IPFS

Pada perintah nomor 1, dilakukan proses unduh package go-ipfs dari halaman dist.ipfs.io dalam bentuk zip. Selanjutnya terjadi proses unzip package yang telah diunduh, dan pada akhirnya package yang telah di-*unzip* dipindahkan ke direktori /usr/local/bin/ipfs. Langkah ini dilakukan di semua *node.*

2. Inisialisasi IPFS pada setiap node

Setelah melakukan instalasi IPFS di semua *node*, langkah selanjutnya adalah menginisialisasi IPFS. Inisialisasi berfungsi sebagai penerapan konfigurasi awal dari IPFS agar dapat digunakan. Perintah untuk melakukan inisialisasi adalah sebagai berikut:

```
ipfs init
```

Gambar 3.7 *Script* Untuk Inisialisasi IPFS

Pada proses ini, sebuah node akan diberikan *peer identity*. *Peer identity* ini digunakan mengidentifikasi sebuah *node* pada jaringan IPFS. Selain itu proses ini juga akan memberikan konfigurasi awal untuk menjalankan IPFS. Mulai dari *bootstrap list* sampai dengan konfigurasi API IPFS.

3. Menghubungkan semua *node* di dalam jaringan IPFS

Setelah meng-*install* dan menginisialisasi semua node yang ada, hubungkan semua *node* tersebut satu sama lain. Yang dibutuhkan untuk menghubungkan semua *node* yang ada ialah *swarm* key. *Swarm key* berguna sebagai identifikator sebuah jaringan IPFS. Sebuah node yang memiliki *swarm key* tertentu, hanya dapat berkomunikasi dengan *node* yang memiliki *swarm key* yang sama. *Swarm key* menandakan bahwa *node-node* tersebut berada pada jaringan yang sama atau tidak. Key ini harus di-*generate* terlebih dahulu. Untuk

men-*generate swarm key* secara otomatis, perintah yang digunakan ialah sebagai berikut.

```
echo -e "/key/swarm/psk/1.0.0/\n/base16/\n`tr -dc 'a-f0-9' < /dev/urandom | head -c64`" > ~/.ipfs/swarm.key
```

Gambar 3.8 *Script* Untuk *Generate Swarm Key*

Perintah ini hanya dijalankan pada *node bootstrap* yakni *node* 1. Perintah ini akan men-*generate swarm key* yang tersimpan pada *file* swarm.key. Copy file swarm.key tersebut ke *node* 2 dan *node* 3 untuk menandakan *node-node* tersebut berada pada jaringan yang sama.

4. Menambahkan daftar *bootstrap*

Daftar *bootstrap* pada IPFS berisikan daftar *node bootstrap* yang dapat dihubungkan ketika menjalankan IPFS. Ketika melakukan inisialisasi, IPFS memberikan daftar *bootstrap default.* Untuk melihat daftar *bootstrap*, berikut adalah perintah yang digunakan:

```
ipfs bootstrap list
```

Gambar 3.9 Perintah Untuk Melihat Daftar *Bootstrap*

Berikut adalah hasil dari *script* pada gambar 4.8 :

```
/dnsaddr/bootstrap.libp2p.io/p2p/QmNnooDu7bfjPFoTZYxMNLWUQJyrVwtbZg5gBMjTezGAJN
/dnsaddr/bootstrap.libp2p.io/p2p/QmQCU2EcMqAqQPR2i9bChDtGNJchTbq5TbXJJ16u19uLTa
/dnsaddr/bootstrap.libp2p.io/p2p/QmbLHAnMoJPWSCR5Zhtx6BHJX9KiKNN6tpvbUcqanj75Nb
/dnsaddr/bootstrap.libp2p.io/p2p/QmcZf59bWwK5XFi76CZX8cbJ4BhTzzA3gU1ZjYZcYW3dwt
/ip4/104.131.131.82/tcp/4001/p2p/QmaCpDMGvV2BGHeYERUEnRQAwe3N8SzbUtfsmvsqQLuvuJ
```

Gambar 3.10 Daftar *Bootstrap Default*

Pada langkah ini, daftar default tersebut digantikan dengan *node bootstrap* yang tersedia pada jaringan IPFS yang telah dibuat. Untuk menghapus daftar bootstrap, gunakan perintah sebagai berikut :

```
ipfs bootstrap rm -all
```

Gambar 3.11 Perintah Untuk Menghapus Daftar *Bootstrap*

Setelah menghapus daftar *bootstrap default*, langkah selanjutnya ialah menambahkan alamat *node bootstrap* di jaringan IPFS. Hal tersebut dapat dilakukan dengan perintah berikut.

```
ipfs bootstrap add
/ip4/192.168.137.10/tcp/4001/ipfs/12D3KooWMRv2DgMgqZJBfjH1Nfoz9a
XfuJbYahy7k1VB6fzS3tjz
```

Gambar 3.12 Perintah Untuk Menambah Daftar Bootstrap

- 192.168.137.10, adalah alamat ipv4 *node* 1 sebagai *node bootstrap*
- 4001, adalah port dimana IPFS berjalan.
- 12D3KooWMRv2DgMgqZJBfjH1Nfoz9aXfuJbYahy7k1VB6fzS3tjz, adalah *peer identity* dari *node* 1 sebagai *node bootstrap*

Lakukan proses ini di semua node, baik *node* 1, *node* 2 dan *node* 3. Setelah semua node telah ditambahkan *bootstrap,* langkah selanjutnya ialah menjadikan jaringan IPFS yang telah dibuat menjadi jaringan *private*. Cara yang dilakukan yaitu dengan mengatur environment variable "LIBP2P_FORCE_PNET" menjadi 1. Berikut adalah perintah yang digunakan :

```
export LIBP2P_FORCE_PNET=1
```

Gambar 3.13 Perintah Untuk Mengubah Jaringan IPFS Menjadi *Private*

5. Menjalankan IPFS

   Setelah menambah daftar *bootstrap* pada setiap *node*, dan mengatur jaringan IPFS yang telah dibuat menjadi jaringan *private*. IPFS telah dapat dijalankan. Untuk menjalankan IPFS gunakan perintah berikut ini :

```
ipfs daemon
```

Gambar 3.14 Perintah Untuk Menjalankan IPFS

Untuk memastikan bahwa semua node telah terhubung satu sama lain ketika IPFS dijalankan, maka gunakan perintah sebagai berikut:

```
ipfs swarm peers
```

Gambar 3.15 Perintah Untuk Melihat *Node* IPFS Yang Terhubung

Berikut adalah hasil dari script pada gambar 4.14 di *node* 1:

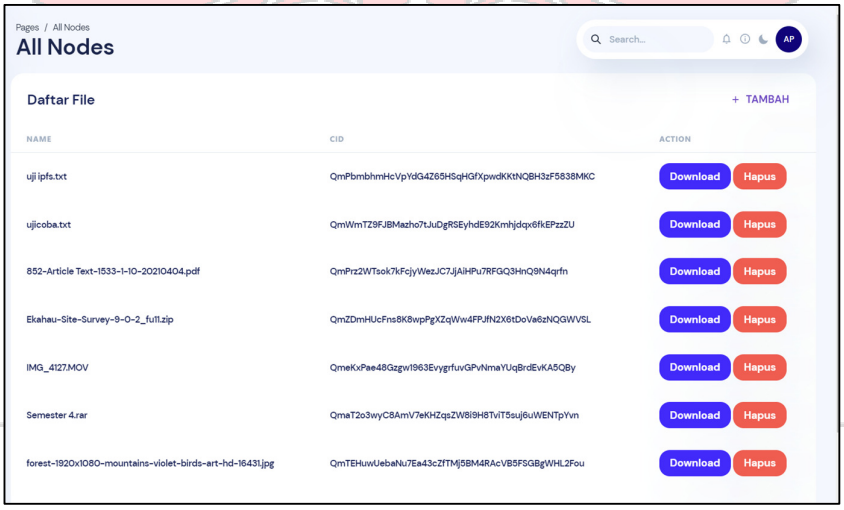
```
user@node1:~$ ipfs swarm peers
/ip4/192.168.137.11/tcp/4001/p2p/12D3KooWBZuz4ZMp8kq7zMBGQzzmPHAQBry4MhiVEsJ61yY2ToHc
/ip4/192.168.137.12/tcp/4001/p2p/12D3KooWPJ4cRLdf995Dc3H4PcaP1tMyeCsHu4i7yFMhWyh1cQDr
```

Gambar 3.16 *Node* IPFS Yang Terhubung Ke *Node* 1

Dapat dilihat pada gambar 3.16, bahwa *node* 1 telah terkoneksi dengan *node* 2 dan *node* 3.

3.6.2 Merancang *User Interface*

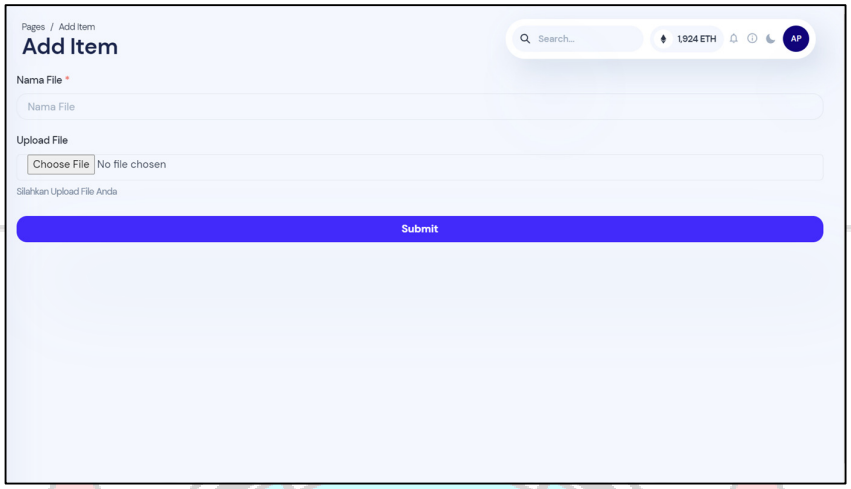
Merancang desain *user interface* bertujuan untuk membuat representasi bagaimana fungsi dari sistem penyimpanan IPFS pada penelitian ini. Desain user interface tersebut berisikan sebuah formulir. Formulir ini bertujuan untuk melakukan *submit file* pada *node* IPFS dan *smart contract* Ethereum. Data pada smart contract berisikan informasi dari file yang di-*upload* ke IPFS yaitu nama file, serta *hash file* yang didapatkan ketika mengupload file ke IPFS. Berikut desain *user interface* yang dibuat untuk web DApp berbasis IPFS seperti pada Gambar 3.17.



Gambar 3.17 *User Interface* DApp IPFS

Pada bagian tabel daftar ijazah terdapat tiga kolom, dimana kolom pertama berisikan nama file, yang kedua berisi hash file pada node IPFS, dan

yang terakhir berisikan tombol untuk mengunduh file dari IPFS. Data-data ini diambil dari blockchain melalui *smart contract*.

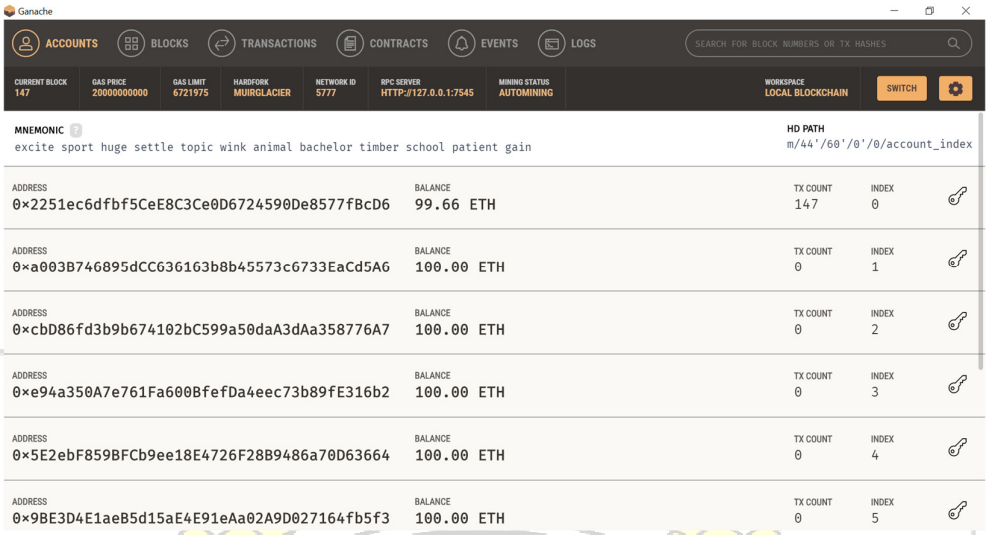


Gambar 3.18 *User Interface* Unggah File

Pada *form* data *file* terdapat dua data yang perlu dimasukkan yaitu input *file*, dan nama *file*, setelah itu terdapat tombol "submit" yang berfungsi sebagai perintah menyimpan *file* pada *node* IPFS dan *blockchain* Ethereum.

3.6.3 Pembuatan *Smart Contract* Dengan *Blockchain* Lokal

Pengembangan smart contract Ethereum dilakukan pada *blockchain* lokal, karena dengan menggunakan *blockhain* lokal, pengujian dan pengkodean smart contract dapat dilakukan secara gratis dan lebih efisien. Pada penelitian ini menggunakan *blockchain* lokal dari ganache seperti pada Gambar 3.19.



Gambar 3.19 *Blockchain Ethereum* Lokal Ganache

Ganache bertujuan untuk menjalankan *blockchain* versi *localhost* selayaknya *blockchain Ethereum* pribadi, sehingga dapat digunakan untuk menjalankan pengujian, menjalankan perintah, dan memeriksa keadaan sambal mengontrol bagaimana rantai beroperasi. Setelah itu *install framework* Truffle. *Framework* Truffle berfungsi untuk mengintegrasikan pengkodean website dengan pengkodean *smart contract*, selain itu Truffle dapat melakukan pengujian *smart contract* menggunakan Terminal. Setelah *framework* Truffle berhasil di-*install*, langkah selanjutnya membuat *smart contract* untuk mendaftarkan file yang telah disimpan pada node IPFS sebagai berikut:

```
pragma solidity >=0.4.22 <0.9.0;
contract Files {
    string fileHashes;
    function set(string memory _hashes) public {
        fileHashes = _hashes;
    }
    function get() public view returns (string memory) {
        return fileHashes;
    }
}
```

Gambar 3.20 Script Smart Contract

Pada *smart contract* gambar 3.20, terdapat dua *method*, yakni *get* dan *set*. *Method get* berfungsi untuk mengambil data dari blockchain. Sedangkan *method set* berguna untuk menuliskan data ke dalam *blockchain*. Data yang disimpan di dalam blockchain pada penelitian ini adalah sebuah *list* yang di-*encode* kedalam bentuk json. List tersebut berisi daftar nama dan *hash file* yang telah di-*upload* ke IPFS.

3.6.4 Integrasi IPFS dan Smart contract Ethereum

Setelah melakukan pengecekan *smart contract* pada *blockchain* lokal, langkah selanjutnya adalah melakukan integrasi pada IPFS. Pengintegrasian pada IPFS dilakukan dengan menggunakan API seperti pada *source code* berikut:

```
console.log("Submitting file to ipfs...")
var file = await ipfs.add(buffer)
var hash = await props.contracts.methods.get().call()
var lista = JSON.parse(hash)
var newList = lista.concat({ "name": name, "cid": file.path })
var jsonList = JSON.stringify(newList)
await props.contracts.methods.set(jsonList).send({ from:
props.accounts })
history.back()
```

Gambar 3.21 *Script Upload File* Ke IPFS & *Smart Contract Blockchain*

Pada *source code* gambar 3.21, file yang telah diubah kedalam bentuk buffer yang akan dikirim pada IPFS dan dikembalikan dalam bentuk hash. Setelah mendapatkan hash file, smart contract yang telah tersedia sebelumnya dalam bentuk json, di-*decode* kemudian ditambahkan dengan data file yang baru berupa nama dan hash file. Setelah itu, terjadi proses *encode* kedalam bentuk json dan kemudian dikirim ke *blockchain* lokal.

3.7 Langkah-langkah Pengujian Sistem

Tahap terakhir adalah melakukan pengujian sistem untuk menganalisis performa sistem sistem penyimpanan terdistribusi berbasis IPFS pada penelitian ini. Pengujian dibagi menjadi 2 yakni pengujian fungsionalitas dan pengujian

Quality of Service. Data yang digunakan dalam pengujian ini adalah 25 *file* dalam format yang berbeda-beda. Ukuran file yang digunakan berkisar antara 0,025 Kb – 1,6 Gb.

3.7.1 Pengujian Fungsionalitas Aplikasi

Tujuan dari pengujian ini adalah untuk mengetahui apakah perangkat lunak atau software yang telah kita bangun berfungsi seperti tujuan pada awalnya. Skenario pengujian fungsionalitas aplikasi dapat dilihat pada tabel 3.3

Tabel 3.3 Skenario Pengujian Fungsionalitas Aplikasi

Skenario Pengujian	Data Masukan	Keluaran yang Diharapkan
Upload file ke node ipfs	File (ekstensi txt, iso, rar, zip, pdf, jpg, docx, mp4)	File tersedia di semua node, dan mendapat hash cid sebagai kembalian
Pencatatan data ke smart contract	nama, dan hash ipfs, berbentuk json	Data nama dan hash ipfs tercatat di smart contract
Ambil data dari smart contract	-	Dapat mengambil data file dari smart contract dan menampilkannya di web interface

3.7.2 Pengujian *Quality Of Service*

Pengujian ini dilakukan untuk mengetahui kualitas dari sebuah jaringan dengan menggunakan beberapa parameter yaitu *throughput*, *packet loss*, dan *response time*. Pengujian dilakukan dengan cara mendata performa *throughput*, *packet loss*, dan *response time* pada saat melakukan submit file ke jaringan

IPFS. Selanjutnya dibuat grafik dari data tersebut untuk mengukur performa sistem. Setelah itu mencari rata-rata *throughput*, *packet loss*, dan *response time* dari kedua DApp untuk mengetahui DApp mana yang memiliki performa lebih baik dan seberapa banyak performa yang dtingkatkan

BAB IV HASIL & PEMBAHASAN

Hasil dari penelitian ini berupa aplikasi penyimpanan terdistribusi dengan menggunakan protokol IPFS (*Interplanetary File System*). Berdasarkan prosedur penelitian yang telah dipaparkan pada bab III, penelitian ini diawali dengan melakukan studi literatur sebagai dasar penelitian serta analisis kebutuhan yang bertujuan untuk melalukan persiapan sebaik-baiknya baik dari segi hardware maupun software yang dapat mendukung proses penelitian berjalan dengan lancar dan baik.

4.1 Sistem Penyimpanan Berbasis IPFS

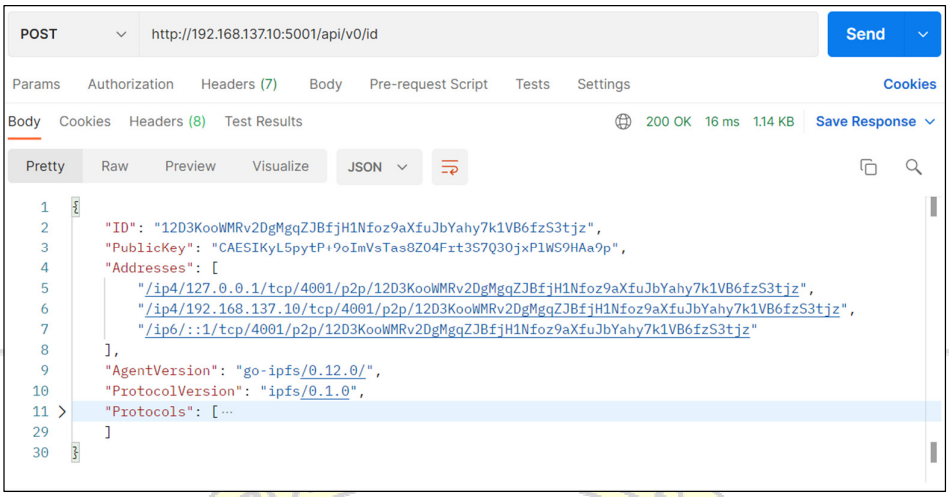
4.1.1 *Node* 1

Node 1 berfungsi sebagai *node* bootstrap. *Node* ini menjadi *node* utama dimana jika ada *node* yang ingin bergabung dengan jaringan IPFS, harus terhubung dengan *node* ini, maka *node* bootstrap akan menghubungkan *node* baru dengan jaringan yang ada.

```
user@node1:~$ ipfs swarm peers
/ip4/192.168.137.11/tcp/4001/p2p/12D3KooWBZuz4ZMp8kq7zMBGQzzmPHAQBry4MhiVEsJ61yY2ToHc
/ip4/192.168.137.12/tcp/4001/p2p/12D3KooWPJ4cRLdf995Dc3H4PcaP1tMyeCsHu4i7yFMhWyh1cQDr
```

Gambar 4.1 *Node* IPFS Yang Terhubung Ke *Node* 1

Pada gambar 4.1, dapat dilihat bahwa node 1 terhubung dengan node 2 dan node 3. Terdapat pula api IPFS yang digunakan untuk menghubungkan sistem penyimpanan ini dengan aplikasi client. Api IPFS pada node 1 dapat berjalan dengan baik seperti pada gambar 4.2.



Gambar 4.2 Api IPFS *Node* 1

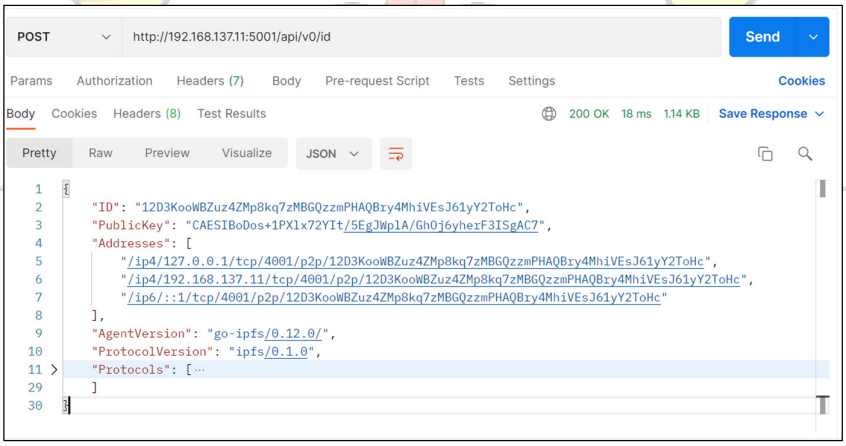
4.1.2 *Node* 2

Node 2 berfungsi sebagai salah satu *node peer*. *Node* ini dapat dikatakan sebagai *node* sekunder pada jaringan IPFS.

```
user@node2:~$ ipfs swarm peers
/ip4/192.168.137.10/tcp/4001/p2p/12D3KooWMRv2DgMgqZJBfjH1Nfoz9aXfuJbYahy7k1VB6fzS3tjz
/ip4/192.168.137.12/tcp/37514/p2p/12D3KooWPJ4cRLdf995Dc3H4PcaP1tMyeCsHu4i7yFMhWyh1cQDr
/ip4/192.168.137.12/tcp/4001/p2p/12D3KooWPJ4cRLdf995Dc3H4PcaP1tMyeCsHu4i7yFMhWyh1cQDr
```

Gambar 4.3 *Node* IPFS Yang Terhubung Ke *Node* 2

Pada gambar 4.3, dapat dilihat bahwa *node* 2 terhubung dengan *node* 1 dan *node* 3. Api IPFS pada *node* 2 dapat berjalan dengan baik seperti pada gambar 4.4.



Gambar 4.4 Api IPFS *Node* 2

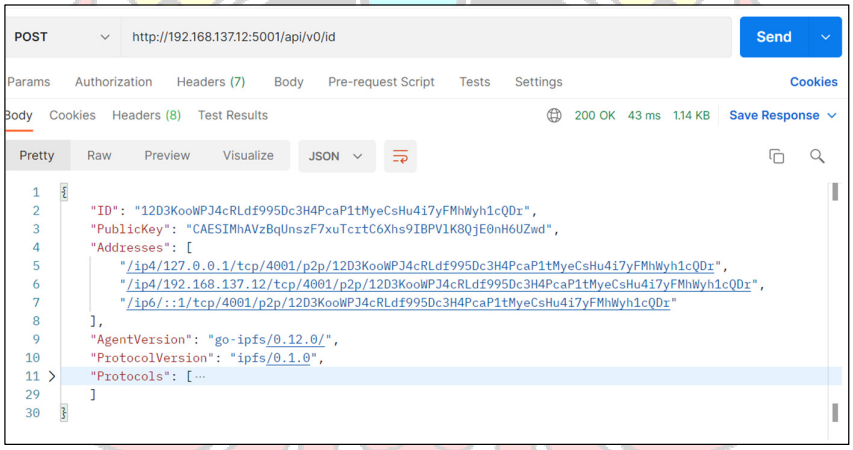
4.1.3 *Node* 3

Node 3 berfungsi sebagai salah satu *node peer*. *Node* ini dapat dikatakan sebagai *node* sekunder pada jaringan IPFS.

```
user@node3:~$ ipfs swarm peers
/ip4/192.168.137.10/tcp/4001/p2p/12D3KooWMRv2DgMgqZJBfjH1Nfoz9aXfuJbYahy7k1VB6fzS3tjz
/ip4/192.168.137.11/tcp/4001/p2p/12D3KooWBZuz4ZMp8kq7zMBGQzzmPHAQBry4MhiVEsJ61yY2ToHc
/ip4/192.168.137.11/tcp/4001/p2p/12D3KooWBZuz4ZMp8kq7zMBGQzzmPHAQBry4MhiVEsJ61yY2ToHc
```

Gambar 4.5 *Node* IPFS Yang Terhubung Ke *Node* 3

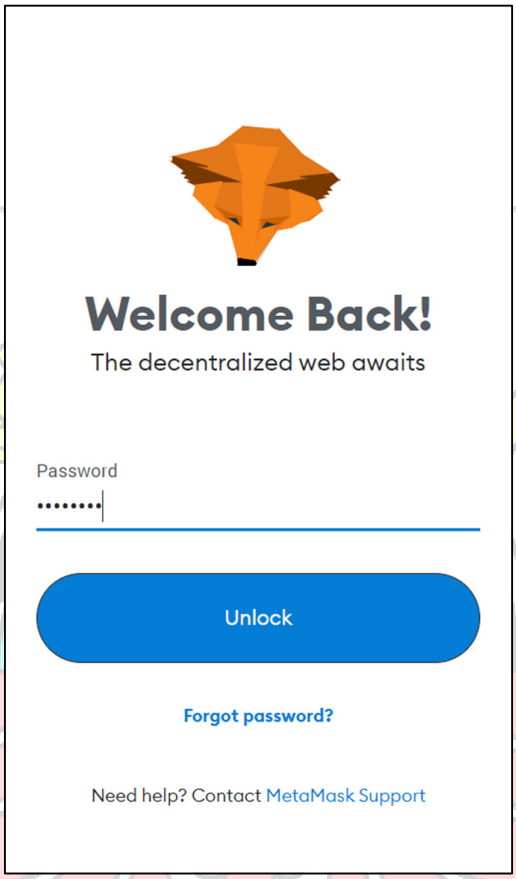
Pada gambar 4.5, dapat dilihat bahwa *node* 3 terhubung dengan *node* 1 dan *node* 2. Api IPFS pada *node* 3 dapat berjalan dengan baik seperti pada gambar 4.6.



Gambar 4.6 Api IPFS *Node* 3

4.2 Hasil Antarmuka Web DApp

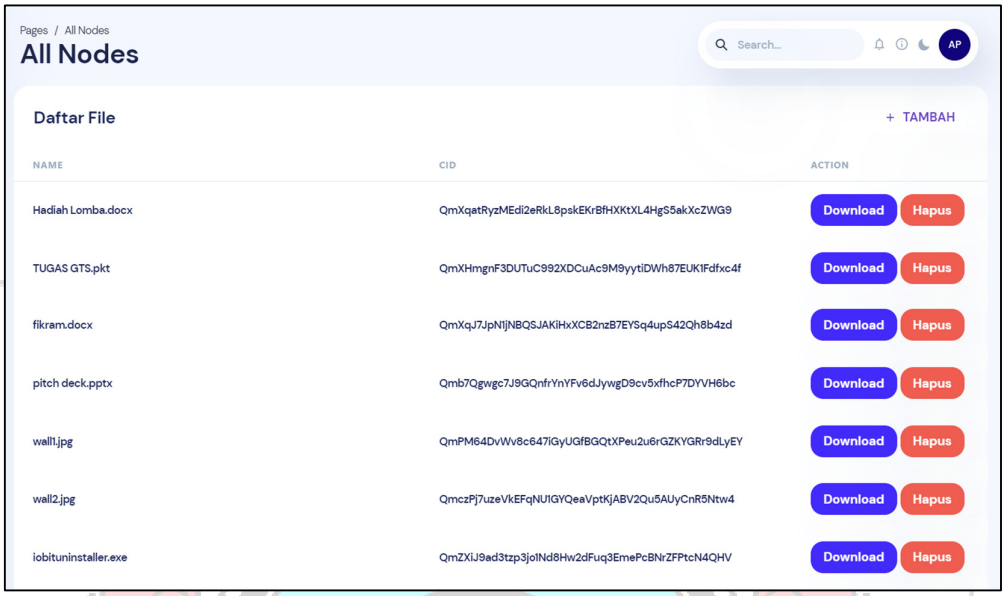
4.2.1 Login Melalui Metamask



Gambar 4.7 Login Metamask

Metamask dibutuhkan untuk menghubungkan antarmuka *web* dengan *smart contract* di jaringan *blockchain*. Untuk menggunakan metamask, pengguna diarahkan untuk *login* terlebih dahulu. Untuk *login*, pengguna diwajibkan mempunyai akun yang terhubung ke jaringan *blockchain* yang tersedia.

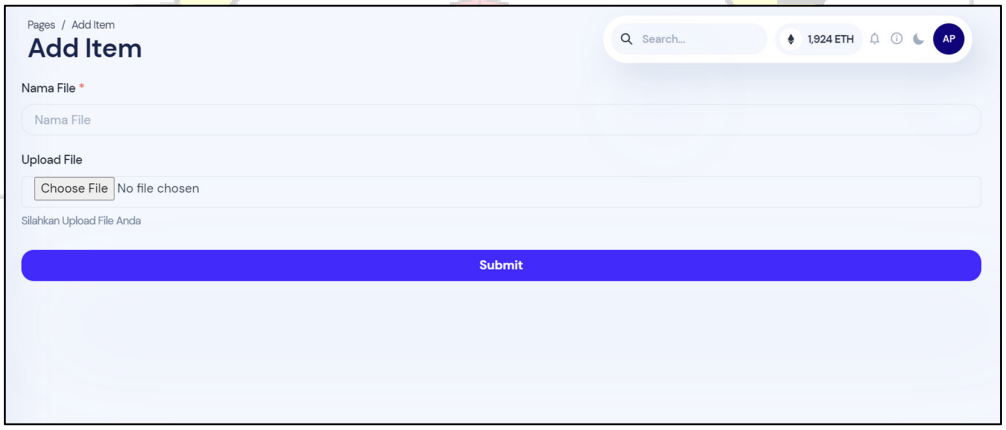
4.2.2 Halaman Index



Gambar 4.8 Halaman Index

Setelah login, pengguna diarahkan menuju halaman *index* terlebih dahulu. Dapat dilihat pada gambar 4.8, terdapat daftar semua *file* yang telah diunggah ke IPFS yang diambil melalui *blockchain*. Pada bagian tabel daftar *file* terdapat tiga kolom, dimana kolom pertama berisikan nama *file*, yang kedua berisi hash file pada *node* IPFS, dan yang terakhir berisikan tombol untuk mengunduh file dari IPFS.

4.2.3 Halaman Tambah *File*

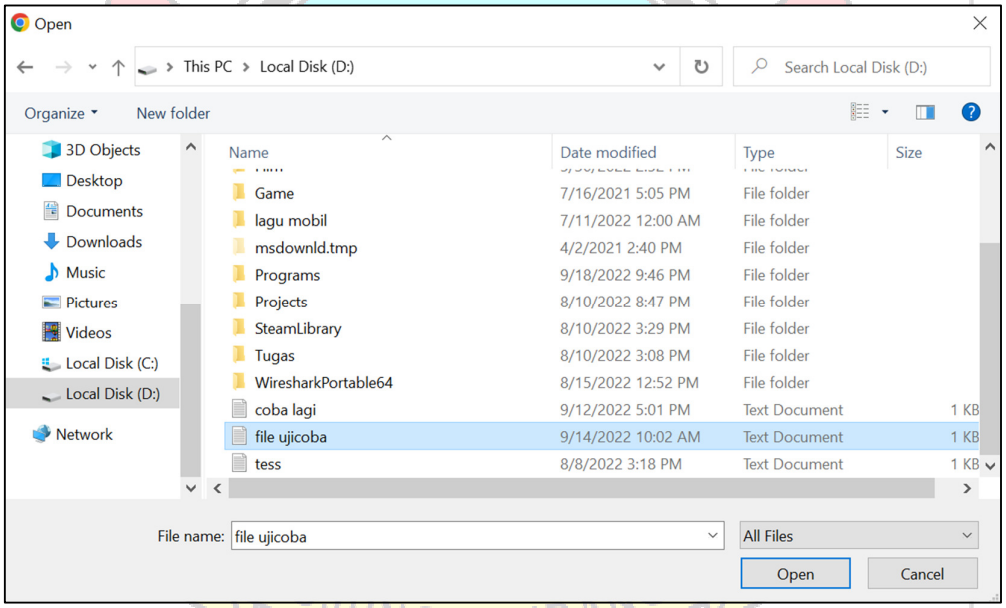


Gambar 4.9 Halaman Tambah *File*

Dapat dilihat pada gambar 4.9, di halaman ini terdapat *form* untuk mengunggah *file* ke IPFS. Pada *form* data *file* terdapat dua data yang perlu dimasukkan yaitu input *file*, dan nama *file*, setelah itu terdapat tombol "*submit*" yang berfungsi sebagai perintah menyimpan *file* pada *node* IPFS dan *blockchain* Ethereum.

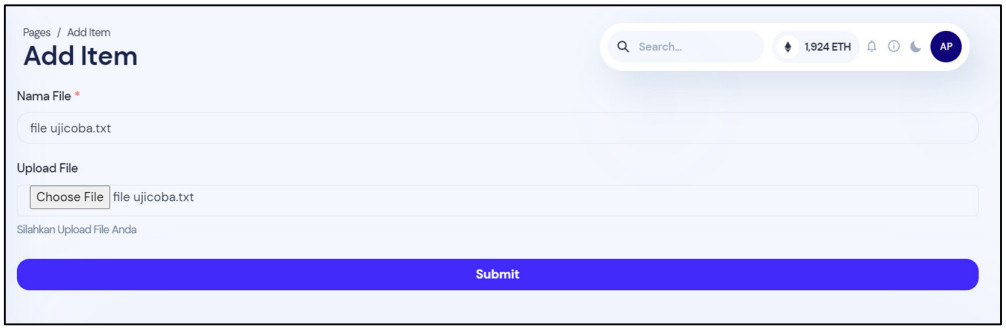
4.2.4 Proses Unggah *File*

Untuk menggunggah *file* ke IPFS, langkah pertama yang dilakukan ialah memilih *file* yang diunggah. Untuk memilih file, pengguna menekan tombol "*choose file*" pada halaman tambah file yang dapat dilihat pada gambar 4.9. Setelah itu akan muncul jendela *windows explorer* untuk memilih *file* seperti pada gambar 4.10.



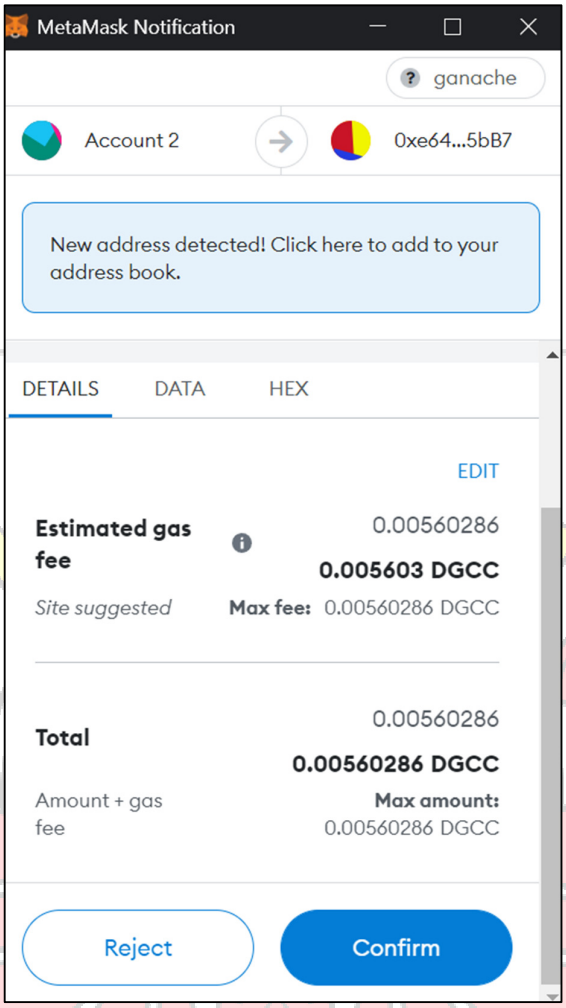
Gambar 4.10 Jendela *Windows Explorer* Untuk Memilih *File*

Setelah memilih *file*, pengguna akan diarahkan kembali ke halaman tambah *file*, dan pada kolom nama *file* dan upload *file* telah terisi dengan *file* yang dipilih seperti pada gambar 4.11.



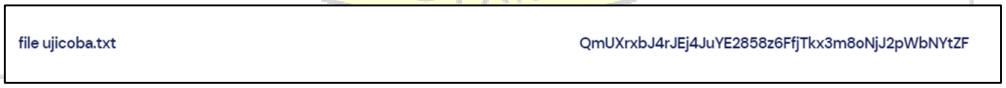
Gambar 4.11 Halaman Tambah *File* Setelah Memilih *File*

Selanjutnya, pengguna menekan tombol "*submit*" untuk mengirimkan data *file* ke IPFS. Aplikasi akan mengirimkan data *file* berupa yang telah dikonversi menjadi *buffer* ke salah satu *node* IPFS melalui api IPFS. Kemudian, api IPFS akan mengembalikan *hash file* yang telah diunggah. Setelah itu aplikasi akan mengirimkan nama dan *hash file* yang didapatkan dari IPFS ke jaringan *blockchain*. Proses ini dilakukan melalui metamask, metamask menghubungkan aplikasi ke *smart contract* yang ada di jaringan *blockchain.* sebelum memasukkan data ke jaringan blockchain, metamask akan meminta konfirmasi pengguna mengenai *cost* transaksi. Setiap transaksi di jaringan blockchain mempunyai *cost* ketika pengguna melakukannya. *Cost* ini bervariasi tergantung dari banyaknya data yang dikirimkan. Proses konfirmasi transaksi dapat dilihat pada gambar 4.12



Gambar 4.12 Proses Konfirmasi Transaksi Metamask

Setelah pengguna mengkonfirmasi transaksi unggah *file* ke jaringan *blockchain*, pengguna akan diarahkan ke halaman *index*. Di halaman *index*, file yang telah diunggah tampil pada tabel daftar *file* seperti pada gambar 4.13



Gambar 4.13 File Pada Halaman Index

4.3 Pengujian Fungsionalitas Aplikasi

Setelah semua sistem sudah terintegrasi langkah selanjutnya adalah pengujian fungsionalitas aplikasi. Tujuan dari pengujian ini adalah untuk

mengetahui apakah perangkat lunak atau software yang telah kita bangun berfungsi seperti tujuan pada awalnya.

Tabel 4.1 Hasil Pengujian Fungsionalitas Aplikasi

Skenario Pengujian	Data Masukan	Keluaran yang Diharapkan	Pengamatan	Kesimpulan	Keterangan
Upload file ke node ipfs	*File* (ekstensi txt, iso, rar, zip, pdf, jpg, docx, mp4)	*File* tersedia di semua *node*, dan mendapat hash cid sebagai kembalian	*File* tersedia di semua *node*, terkecuali *file* iso	Gagal di *file* iso	Gagal dikarenakan *file* iso tidak dapat dikonversi kebentuk buffer untuk kemudian dikirimkan melalui jaringan ke IPFS
Pencatatan data ke smart contract	nama, dan hash ipfs, berbentuk json	Data nama dan hash ipfs tercatat di smart contract	Data nama dan hash ipfs tercatat di smart contract	Berhasil	-
Ambil data dari smart contract	-	Dapat mengambil data file dari smart contract dan menampilkannya di web interface	dapat mengambil data file dari smart contract dan tampil di web interface	Berhasil	-

4.4 Pengujian *Quality of Service*

Pengujian *quality of service* dilakukan pada sistem penyimpanan terdistribusi berbasis IPFS menggunakan data file dengan ukuran berbeda-beda dengan rentang antara 0,025 kb – 1,67 Gb. Rekapitulasi hasil pengujian sistem penyimpanan terdistribusi berbasis IPFS seperti pada Tabel 4.2.

Tabel 4.2 Pengujian *Quality Of Service*

No.	Ukuran File	*Response Time* (s)	*Throughput*	*Packet Loss* (%)
1.	0,025 Kb	0,137	0,182 Kbps	0
2.	16,15 Kb	0,178	90,7 Kbps	0
3.	21,5 Kb	0,299	71,9 Kbps	0,001
4.	45,6 Kb	0,046	991,3 Kbps	0
5.	237 Kb	0,182	1,3 Mbps	0,001
6.	304,3 Kb	33,4	9,1 Kbps	0,003
7.	374 Kb	0,1	3,7 Mbps	0,002
8.	1,18 Mb	0,482	2,4 Mbps	0,005
9.	1,3 Mb	0,387	3,4 Mbps	0,005
10.	5,8 Mb	0,613	9,5 Mbps	0,01
11.	7,8 Mb	0,812	9,6 Mbps	0,009
12.	19,6 Mb	1,91	10,3 Mbps	0,013
13.	27,5 Mb	2,4	11,5 Mbps	0,016
14.	41,4 Mb	3,6	11,5 Mbps	0,02
15.	45,6 Mb	4,8	9,5 Mbps	0,028
16.	48,8 Mb	4,4	11,1 Mbps	0,021
17.	74,8 Mb	7,4	10,1 Mbps	0,036
18.	77,2 Mb	7,5	10,3 Mbps	0,039
19.	111,3 Mb	13,5771	6,15 Mbps	0,04
20.	196 Mb	24,395	8 Mbps	0,047
21.	209 Mb	15,5	13,3 Mbps	0,059
22.	419 Mb	42,5	9,8 Mbps	0,06
23.	629 Mb	58,5	10,75 Mbps	0,07

Tabel 4.2 Pengujian *Quality Of Service*

24.	839 Mb	86,8	9,66 Mbps	0,079
25.	1,6 Gb	178,5	9,4 Mbps	0,1
Rata-rata	174,211 Mb	19,5 s	6,8 Mbps	0,026 %

Dari pengujian performa pada sistem penyimpanan terdistribusi berbasis IPFS, maka rata – rata *throughput* adalah 6,8 Mbps, rata – rata *packet loss* adalah 0,026%, dan rata – rata *response time* adalah 19,5 s. Pada pengujian ini menggunakan data file dengan ukuran rata – rata 174,211 MB.

4.5 Pengujian Ukuran File

Pengujian ukuran file dilakukan dengan membandingkan ukuran file yang di-upload dengan ukuran file yang tersimpan di sistem IPFS. Perbandingan ukuran file asli dengan ukuran file yang tersimpan pada sistem penyimpanan terdistribusi berbasis IPFS seperti pada Tabel 4.3

Tabel 4.3 Perbandingan Ukuran File Asli & File yang Tersimpan di IPFS

No.	Ukuran File	Ukuran File Tersimpan	Format File
1.	0,025 Kb	0,025 Kb	Txt
2.	16,15 Kb	16,15 Kb	Bat
3.	21,5 Kb	21,5 Kb	Txt
4.	45,6 Kb	45,6 Kb	Pdf
5.	237 Kb	237 Kb	Jpg
6.	304,3 Kb	304,3 Kb	Docx
7.	374 Kb	374 Kb	Pptx
8.	1,18 Mb	1,18 Mb	Exe
9.	1,3 Mb	1,3 Mb	Zip
10.	5,8 Mb	5,8 Mb	Rar
11.	7,8 Mb	7,8 Mb	Mp3
12.	19,6 Mb	19,6 Mb	Exe

Tabel 4.3 Perbandingan Ukuran File Asli & File yang Tersimpan di IPFS

13.	27,5 Mb	27,5 Mb	Zip
14.	41,4 Mb	41,4 Mb	Exe
15.	45,6 Mb	45,6 Mb	Rar
16.	48,8 Mb	48,8 Mb	Mov
17.	74,8 Mb	74,8 Mb	Avi
18.	77,2 Mb	77,2 Mb	Rar
19.	111,3 Mb	111,3 Mb	Zip
20.	196 Mb	196 Mb	Mp4
21.	209 Mb	209 Mb	Exe
22.	419 Mb	419 Mb	Exe
23.	629 Mb	629 Mb	Exe
24.	839 Mb	839 Mb	Rar
25.	1,6 Gb	1,6 Gb	Mkv

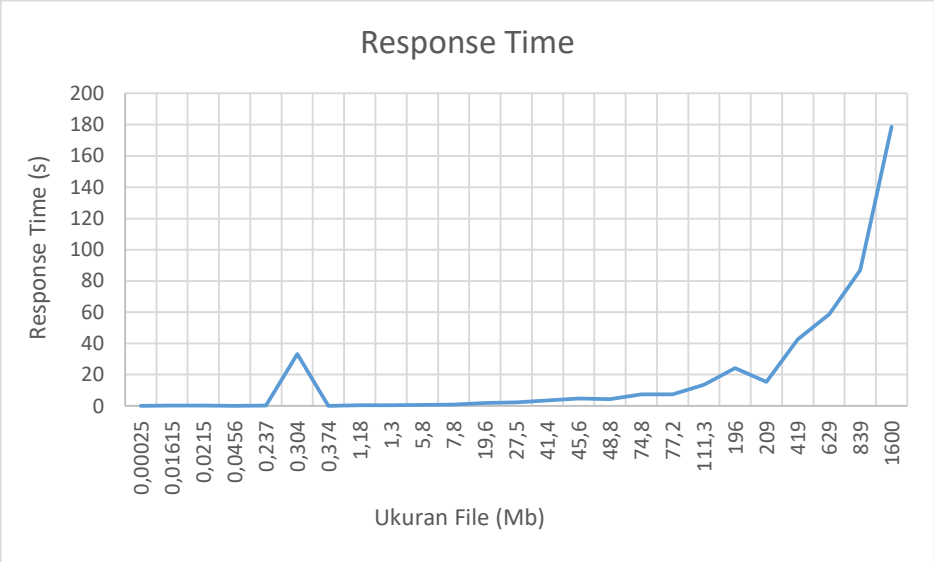
Dapat dilihat pada tabel 4.3, bahwa semua file yang diupload ke sistem penyimpanan berbasis IPFS terjaga keutuhannya.

4.6 Analisis Performa Aplikasi

Setelah melakukan pengujian performa pada sistem penyimpanan terdistribusi berbasis IPFS, langkah selanjutnya adalah melihat perbandingan performa response time, throughput, dan packet loss untuk mengambil kesimpulan. Berikut perbandingannya:

1. *Response Time*

   Hasil pengujian *response time* pada saat input data file pada *node* IPFS direkap dan ditampilkan dalam bentuk grafik untuk menganailisis performa sistem.

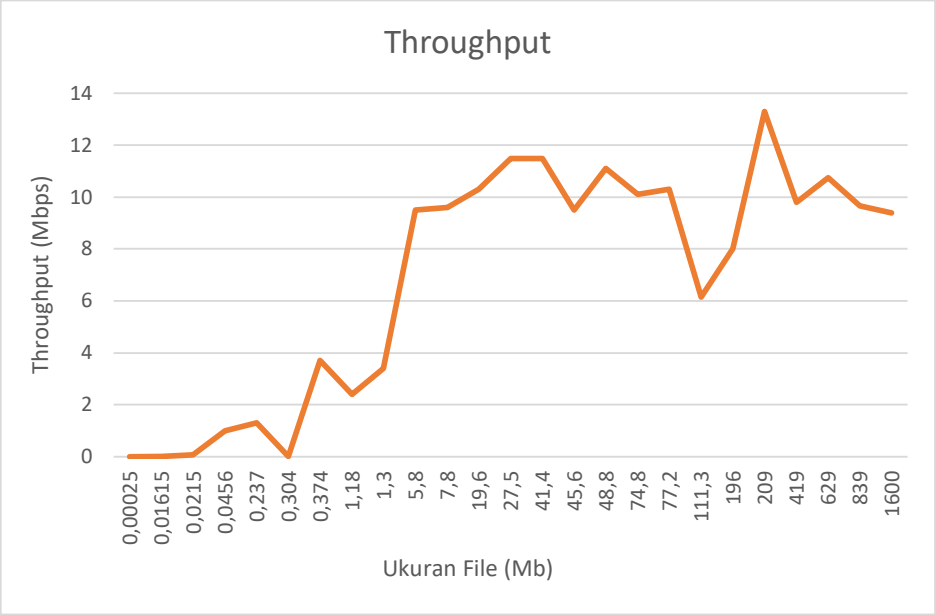


Gambar 4.14 Response Time

Gambar 4.14 adalah grafik performa *response time* berdasarkan 20 data yang diuji. Dapat disimpulkan bahwa *response time* ketika meng-*upload* data ke node IPFS akan semakin besar sebanding dengan ukuran file yang di-upload. Rata-rata *response time* yang didapatkan adalah 19,5 s. Dapat dilihat Pada gambar 4.14, *response time* yang didapatkan mengalami peningkatan seiring dengan ukuran *file* yang dikirim. Ada beberapa faktor selain ukuran *file* yang mempengaruhi *response time*, diantaranya konektivitas jaringan dan *throughput* yang didapatkan.

2. *Throughput*

Hasil pengujian *throughput* pada saat input data file pada *node* IPFS direkap dan ditampilkan dalam bentuk grafik untuk menganailisis performa sistem.

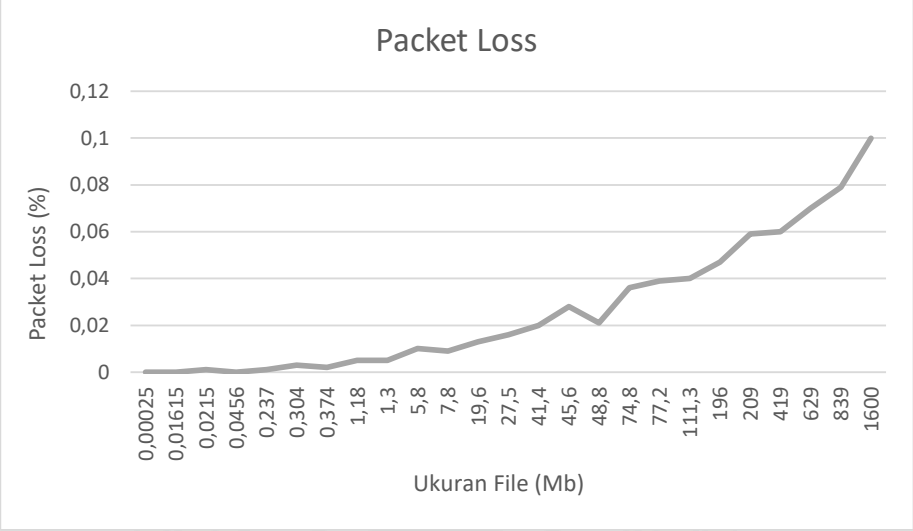


Gambar 4.15 *Throughput*

Gambar 4.15 menunjukkan grafik performa *throughput* berdasarkan 20 data yang diuji. Rata-rata *throughput* yang didapatkan adalah 6,8 Mbps. Nilai *throughput* tertinggi yaitu 13,3 Mbps yang diperoleh ketika mengunggah file sebesar 209 Mb. *Throughput* terendah yaitu 0,182 Kbps yang diperoleh pada *upload file* berukuran 0,025 Kb. Gambar 4.15 juga menunjukkan nilai *throughput* meningkat seiring dengan bertambahnya ukuran *file* yang di-*upload*, namun nilai throughput mengalami penurunan ketika *file* yang dikirimkan mencapai 45 Mb – 55 Mb. Hal ini menunjukkan bahwa, sistem penyimpanan berbasis IPFS ini dapat berfungsi secara optimal pada file berukuran 0 sampai 45 Mb.

3. *Packet Loss*

Hasil pengujian *packet loss* pada saat input data file pada *node* IPFS direkap dan ditampilkan dalam bentuk grafik untuk menganailisis performa sistem.



Gambar 4.16 *Packet Loss*

Pada gambar 4.16, dapat dilihat bahwa *packet loss* yang terjadi ketika proses kirim *file* bervariasi. Packet loss dipengaruhi dari konektivitas jaringan yang tersedia ketika proses upload file.

Berdasarkan hasil seluruh pengujian performa aplikasi, berikut hasil rekapitulasi rata – rata perhitungan nilai throughput, packet loss dan indeks serta keterangan berdasarkan standar atau rekomendasi dari *Telecommunications and Internet Protocol Harmonization Over Networks* (TIPHON).

Tabel 4.4 Qos TIPHON

Parameter	**Rata-rata nilai**	**Indeks**	**Kategori**
Throughput	6,8 Mbps	4	Sangat Baik
Packet Loss	0,026%	4	Sangat Baik
Rata-rata Indeks		4	Sangat Baik

Berdasarkan tabel 4.4, rata-rata indeks yang didapatkan adalah 4 dengan kategori sangat baik. Nilai rata-rata throughput 6,8 Mbps termasuk dalam kategori sangat baik dengan indeks 4, begitupun *packet loss* dengan nilai rata-rata 0,068% termasuk dalam kategori sangat baik.

4.6.1 Perbandingan Performa Aplikasi dengan Sistem Tersentralisasi

Selanjutnya hasil pengujian performa pada sistem penyimpanan terdistribusi berbasis IPFS akan dibandingkan performa dari sistem penyimpanan tradisional yang tersentralisasi. Sistem tersentralisasi yang digunakan adalah sistem penyimpanan berbasis blob. Sistem ini tetap menggunakan blockchain sebagai penyimpanan alamat dari file tersebut, tetapi alih-alih menggunakan IPFS sebagai media penyimpanan file sistem ini menggunakan database sql dan file yang disimpan berbentuk blob. Performa sistem ini diukur menggunakan data yang sama seperti pengujian performa sistem penyimpanan berbasis IPFS. Berikut hasil pengujian *quality of service* dari sistem penyimpanan berbasis blob yang dijabarkan pada tabel 4.5.

Tabel 4.5 Pengujian *Quality Of Service* Sistem Penyimpanan Berbasis Blob

No.	Ukuran File	*Response Time* (s)	*Throughput*	*Packet Loss* (%)
1.	0,025 Kb	0,147	0,17 Kbps	0
2.	16,15 Kb	0,180	89,7 Kbps	0
3.	21,5 Kb	0,32	67,18 Kbps	0
4.	45,6 Kb	0,088	518,1 Kbps	0,002
5.	237 Kb	0,214	1,1 Mbps	0,001
6.	304,3 Kb	8,8	950,9 Kbps	0,003
7.	374 Kb	0,145	2,5 Mbps	0,002
8.	1,18 Mb	0,482	2,44 Mbps	0,005
9.	1,3 Mb	0,387	3,36 Mbps	0,005
10.	5,8 Mb	0,655	8,85 Mbps	0,01
11.	7,8 Mb	0,9	8,66 Mbps	0,009
12.	19,6 Mb	2,0	9,8 Mbps	0,013
13.	27,5 Mb	2,3	11,95 Mbps	0,016
14.	41,4 Mb	3,8	10,89 Mbps	0
15.	45,6 Mb	5,0	9,12 Mbps	0,028
16.	48,8 Mb	5,7	10,6 Mbps	0,021
17.	74,8 Mb	7,8	9,58 Mbps	0,036

Tabel 4.5 Pengujian *Quality Of Service* Sistem Penyimpanan Berbasis Blob

18.	77,2 Mb	7,8	9,89 Mbps	0,039
19.	111,3 Mb	13,77	8,082 Mbps	0,04
20.	196 Mb	25,1	7,81 Mbps	0,047
21.	209 Mb	15,7	13,3	0,059
22.	419 Mb	42,5	9,85	0,06
23.	629 Mb	58,5	10,75	0,07
24.	839 Mb	86,9	9,65	0,08
25.	1,6 Gb	180	8,89	0
Rata-rata	174,211 Mb	19,5 s	6,75 Mbps	0,021 %

Berdasarkan hasil seluruh pengujian performa aplikasi penyimpanan berbasis blob, berikut perbandingan rekapitulasi rata – rata perhitungan nilai throughput, packet loss dan indeks serta keterangan berdasarkan standar atau rekomendasi dari *Telecommunications and Internet Protocol Harmonization Over Networks* (TIPHON) serta *response time* yang didapatkan dengan sistem penyimpanan berbasis IPFS.

Tabel 4.6 Perbandingan QoS sistem penyimpanan IPFS dan sistem penyimpanan Blob

Sistem Penyimpanan	**Rata-rata *Throughput***	**Indeks *Throughput***	**Kategori *Throughput***	**Rata-rata *Packet Loss***	**Indeks *Packet Loss***	**Kategori *Packet Loss***	**Rata-rata *Response Time***
IPFS	6,8 Mbps	4	Sangat Baik	0,026%	4	Sangat Baik	19,5 s
BLOB	6,75 Mbps	4	Sangat Baik	0,021%	4	Sangat Baik	19,5 s

Berdasarkan tabel 4.6 dapat dilihat bahwa perbandingan antara sistem penyimpanan berbasis IPFS yang terdesentralisasi dengan sistem penyimpanan berbasis blob yang tersentralisasi cukup variatif, dimana dari segi *throughput*, sistem IPFS unggul dengan rata-rata 6,8 Mbps dibanding sistem blob dengan rata-rata 6,75 Mbps. Namun *packet loss* dari sistem blob sedikit lebih baik daripada sistem IPFS, dengan masing-masing 0,021% dan 0,026%. *Packet loss*

sendiri bergantung pada kondisi jaringan yang ada pada saat pengujian. Selain itu rata-rata response time dari kedua sistem ini relatif sama yakni di angka 19,5 s. dapat disimpulkan bahwa sistem penyimpanan berbasis IPFS memiliki throughput lebih besar daripada sistem penyimpanan tradisional yang berbasis blob. Adapun packet loss dan response time yang didapatkan tergantung pada kondisi jaringan.

BAB V PENUTUP

5.1 Kesimpulan

Pada penelitian tentang rancang bangun penyimpanan terdistribusi berbasis IPFS ini, dapat ditarik kesimpulan sebagai berikut. Penelitian ini berhasil membangun sistem penyimpanan berbasis IPFS dengan tujuan untuk menyediakan penyimpanan file yang lebih terdistribusi dan aman. Terdapat beberapa tahapan dalam membangun sistem ini. Salah satunya tahap analisis, dalam tahap analisis dimana dalam kebutuhan pengembangannya membutuhkan perangkat *node* IPFS dan blockchain lokal yaitu ganache untuk pengujian *smart contract* sebelum di *deploy*. Lalu tahap selanjutnya adalah tahap integrasi, pada tahap integrasi sistem menggunakan API dari IPFS untuk proses penyimpanan file pada jaringan *blockchain*. Dari beberapa tahapan tersebut dilakukan pengujian fungsionalitas aplikasi dimana sistem penyimpanan berbasis IPFS bisa digunakan sesuai skenario pengujian.

5.2 Saran

Adapun saran untuk pengembangan penyimpanan terdistribusi berbasis IPFS dimasa yang akan datang, diantaranya adalah:

1. Penelitian selanjutnya diharapkan menggunakan jumlah node IPFS lebih dari tiga.
2. Penelitian selanjutnya dapat menerapkan sistem penyimpanan berbasis IPFS pada satu bidang tertentu, salah satu contohnya pada bidang pendidikan untuk menyimpan materi ajar.
3. Penelitian selanjutnya dapat menggunakan *blockchain* yang berbeda, yakni EOS atau *Ripple*.

DAFTAR PUSTAKA

Arief, L., & Sundara, T. A. (2017). Studi atas Pemanfaatan Blockchain bagi Internet of Things (IoT). *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, *1*(1), 70-75.

Aziz, A. M., Budiyono, A., & Widjajarto, A. (2019). Analisis Dan Implementasi Komunikasi Antar Node Ipfs (Interplanetary File System) Pada Smart Contract Ethereum. eProceedings of Engineering, 6(2).

Benet, J. (2014). IPFS-content addressed, versioned, P2P file system (DRAFT 3). arXiv preprint arXiv:1407.3561.

Bhosale, K., Akbarabbas, K., Deepak, J., & Sankhe, A. (2019). Blockchain based Secure Data Storage. International Research Journal of Engineering and Technology (IRJET), 06(03), 4.

Budi, S. (2006). Konsep dan Aplikasi Client Server dan Sistem Terdistribusi. Andi Offset, Yogyakarta.

Cachin, C., & Vukolić, M. (2017). Blockchain Consensus Protocols in the Wild. ArXiv:1707.01873 [Cs]. http://arxiv.org/abs/1707.01873

Fajar, M. H. (2020). Quality of service ethereum blockchain berbasis IPFS untuk validasi ijazah sekolah (Doctoral dissertation, UIN Sunan Ampel Surabaya).

Lone, A. H., & Mir, R. N. (2019). Consensus protocols as a model of trust in blockchains. Int. J. Blockchains and Cryptocurrencies, 1, 15.

Nizamuddin, N., Hasan, H. R., & Salah, K. (2018). IPFS-Blockchain-Based Authenticity of Online Publications. In S. Chen, H. Wang, & L.-J. Zhang (Eds.), Blockchain – ICBC 2018 (Vol. 10974, pp. 199–212). Springer International Publishing. https://doi.org/10.1007/978-3-319-94478-4_14

Politou, E., Alepis, E., Patsakis, C., Casino, F., & Alazab, M. (2020). Delegated content erasure in IPFS. Future Generation Computer Systems, 112, 956-964.

Rajalakshmi, A., Lakshmy, K. V., Sindhu, M., & Amritha, P. P. (2018). A blockchain and IPFS based framework for secure Research record keeping. International Journal of Pure and Applied Mathematics, 119, 1437–1442

Sanka, A. I., Irfan, M., Huang, I., & Cheung, R. C. (2021). A survey of breakthrough in blockchain technology: Adoptions, applications, challenges and future research. Computer Communications.

Sasmito, G. W. (2017). Penerapan Metode Waterfall Pada Desain Sistem Informasi Geografis Industri Kabupaten Tegal. Jurnal Informatika: Jurnal Pengembangan IT, 2(1), 6-12.

Setyoadi, W., Wintolo, H., & Indrianingsih, Y. (2012). Otomatisasi Penerimaan Dan Pengiriman Pesan Dengan Sistem Terdistribusi Untuk Mendukung Penyebaran Informasi Akademik. Compiler, 1(1).

Steichen, M., Fiz, B., Norvill, R., Shbair, W., & State, R. (2018, July). Blockchain-based, decentralized access control for IPFS. In *2018 Ieee international conference on internet of things (iThings) and ieee green computing and communications (GreenCom) and ieee cyber, physical and social computing (CPSCom) and ieee smart data (SmartData)* (pp. 1499-1506). IEEE.

Sultan, K., Ruhi, U., & Lakhani, R. (2018). Conceptualizing blockchains: Characteristics & applications. *arXiv preprint arXiv:1806.03693*.

Sun, J., Yao, X., Wang, S., & Wu, Y. (2020). Blockchain-based secure storage and access scheme for electronic medical records in IPFS. *IEEE Access*, *8*, 59389-59401.

Tenorio-Fornés, A., Jacynycz, V., Llop-Vila, D., Sánchez-Ruiz, A., & Hassan, S. (2019, January). Towards a decentralized process for scientific publication and peer review using blockchain and IPFS. In Proceedings of the 52nd Hawaii International Conference on System Sciences.

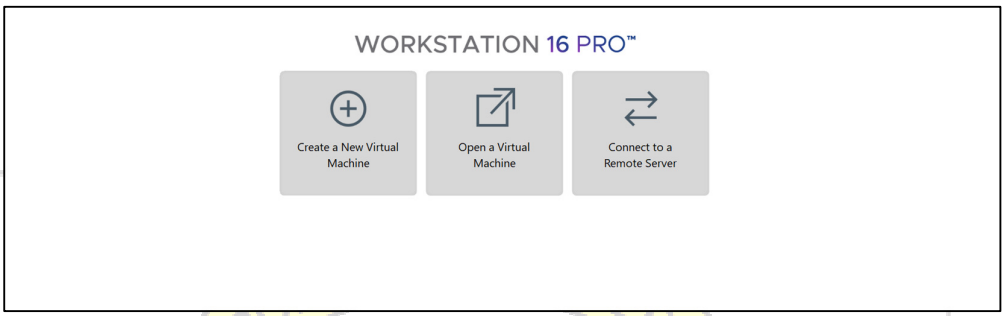
Wulandari, R. (2016). Analisis QoS (Quality of Service) pada jaringan internet (studi kasus: upt loka uji teknik penambangan jampang kulon–lipi). *Jurnal teknik informatika dan sistem informasi*, *2*(2).

Yli-Huumo, J., Ko, D., Choi, S., Park, S., & Smolander, K. (2016). Where Is Current Research on Blockchain Technology? - A Systematic Review. PLOS ONE, 11(10), e0163477. https://doi.org/10.1371/journal.pone.0163477

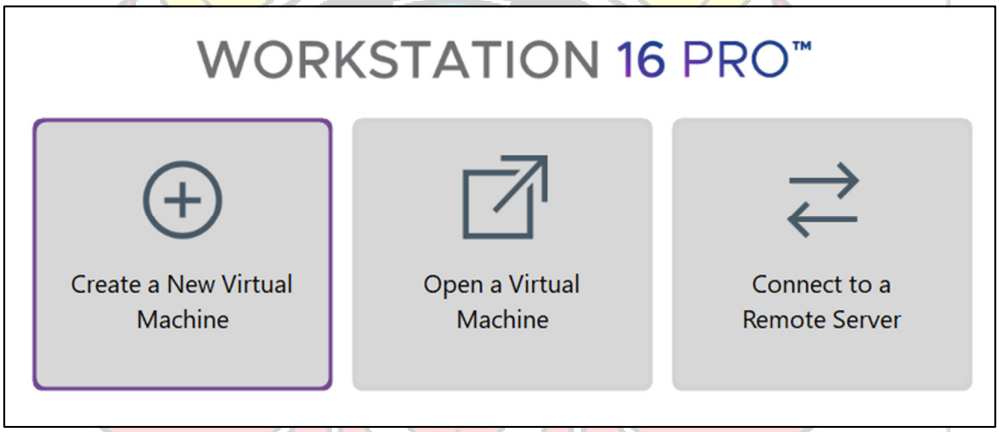
LAMPIRAN

Lampiran 1 Proses Pembuatan *Virtual Machine*

1. Jalankan aplikasi Vmware Workstation

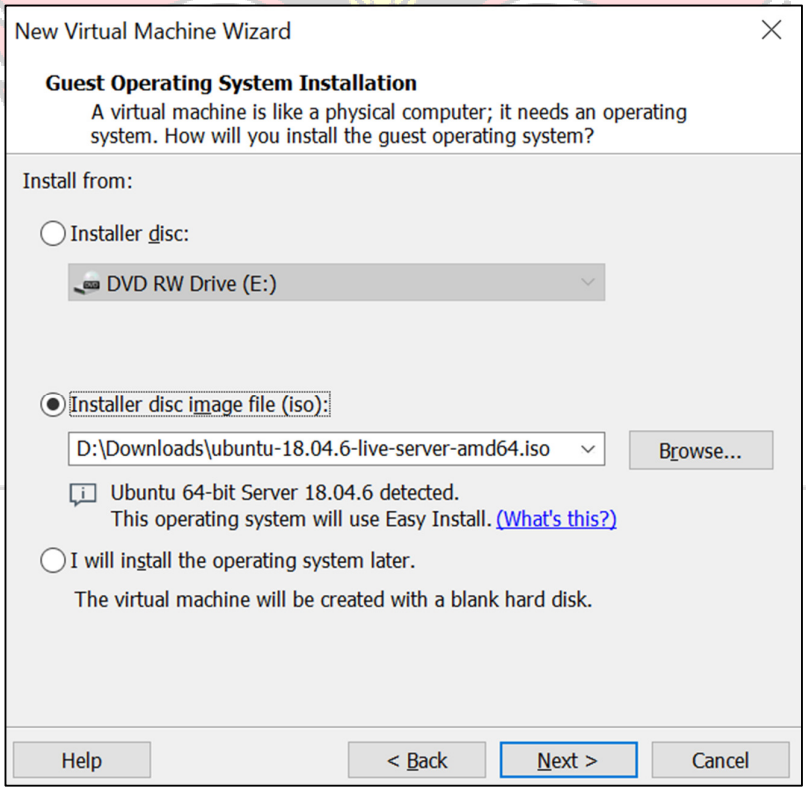


2. Pilih "*Create a New Virtual Machine*"

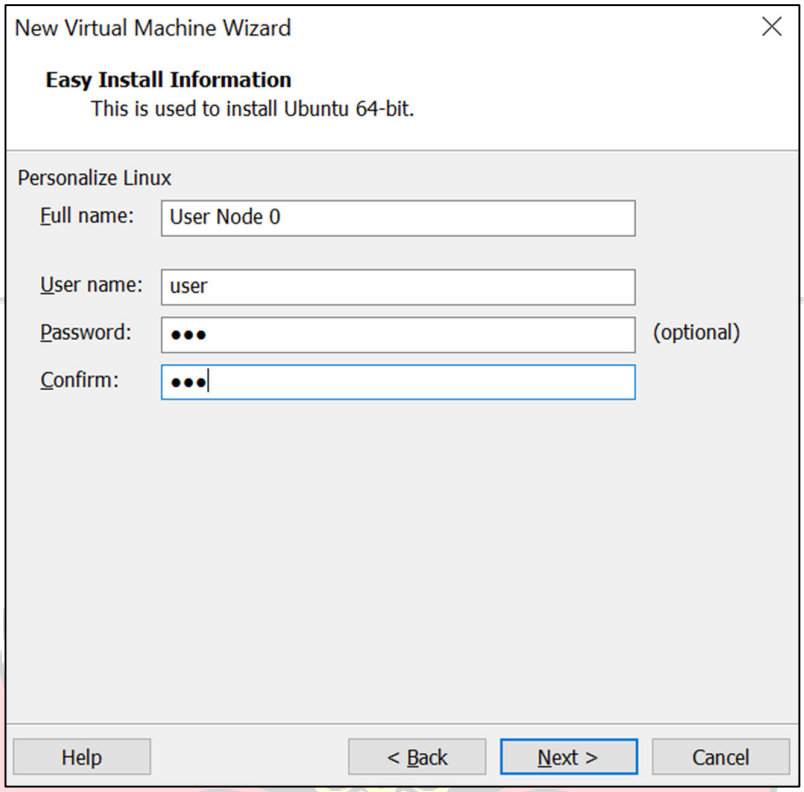


3. Memillih tipe konfigurasi *virtual machine* "*typical*"

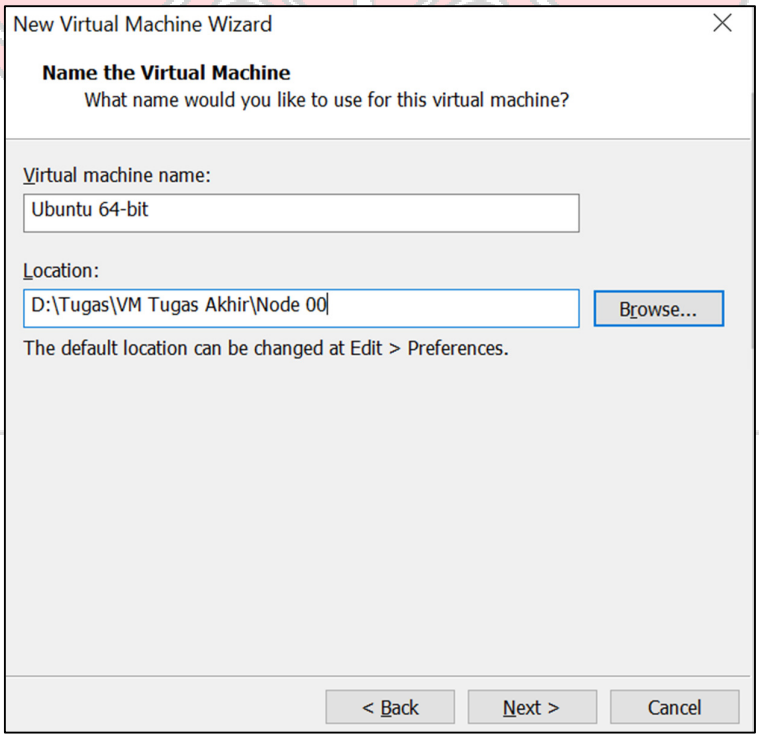
4. Memilih installer sistem operasi yang digunakan pada *virtual machine*



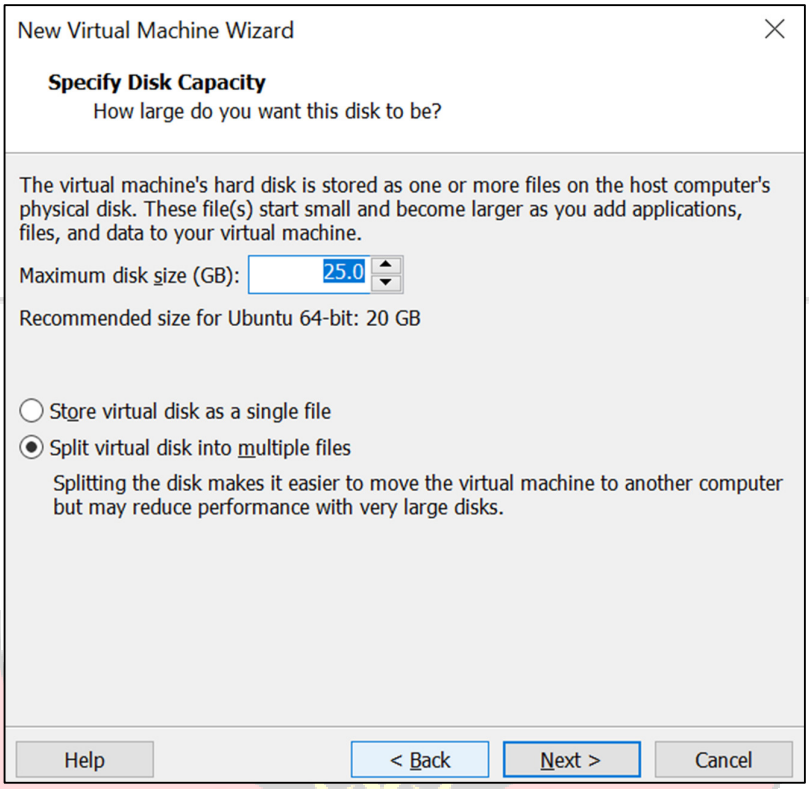
5. Konfigurasi *username* dan *password* untuk sistem operasi



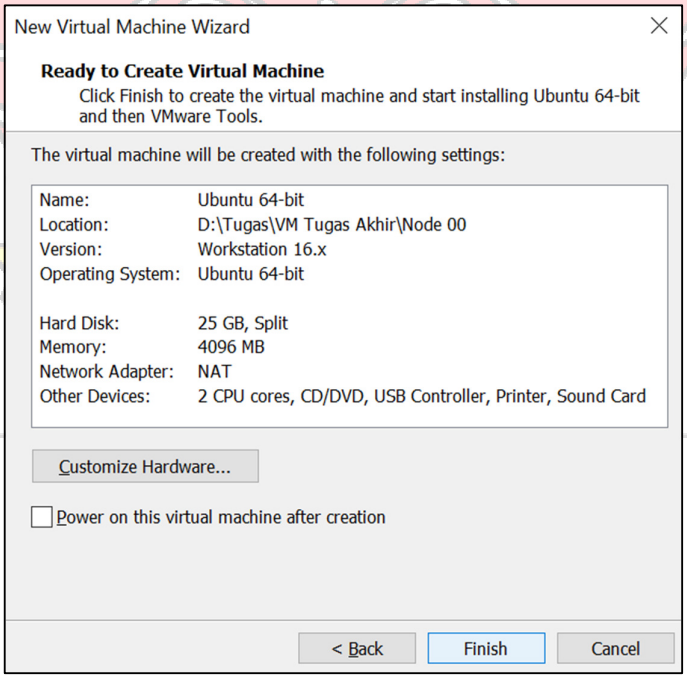
6. Memilih lokasi penyimpanan *file virtual machine*.



7. Menentukan ukuran *virtual machine*



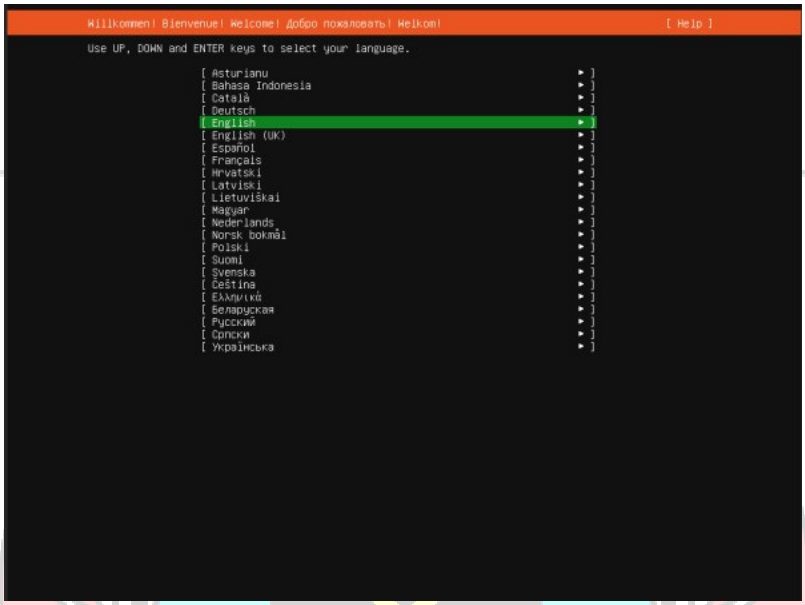
8. Mengkonfirmasi virtual machine yang dibuat



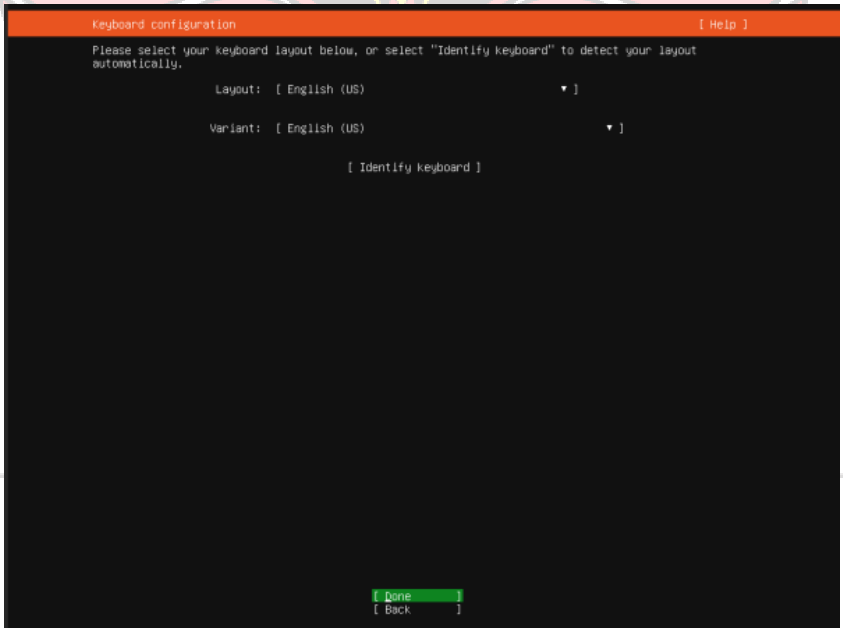
9. Selesai

Lampiran 2 Proses Instalasi Ubuntu Server

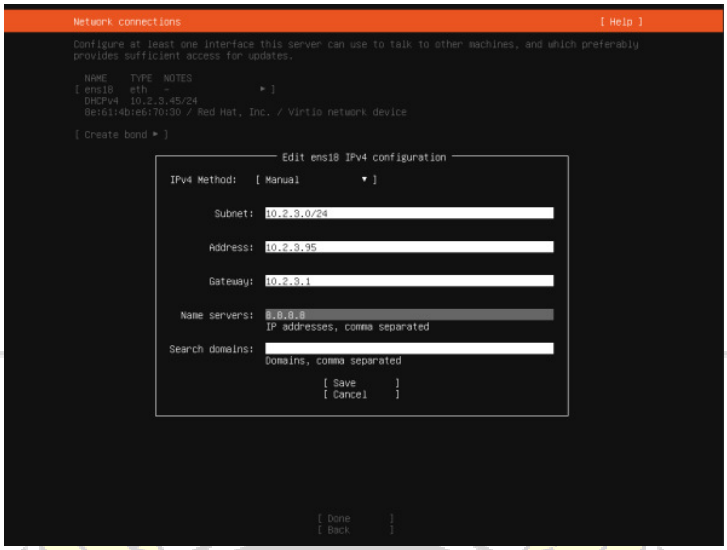
1. Menjalankan *virtual machine*.
2. Memilih *English* sebagai *language*.



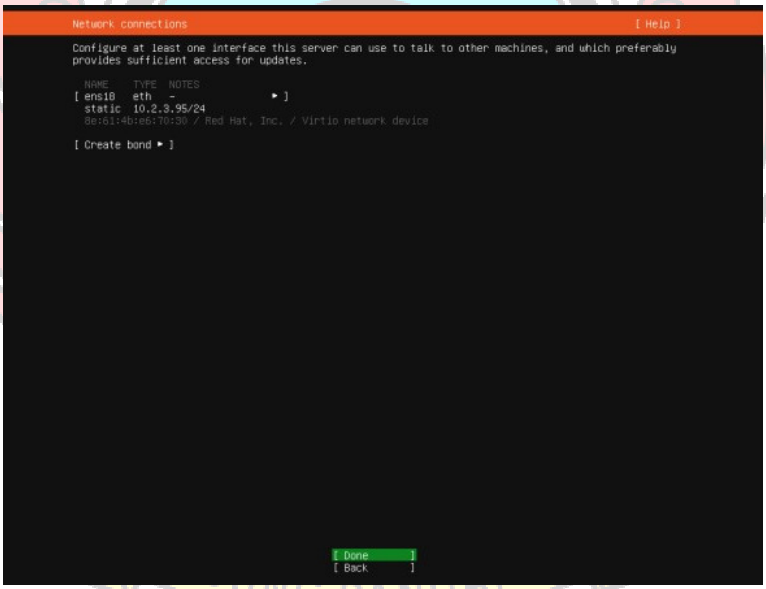
3. Memilih *layout* dan *variant English (US)* sebagai *keyboard layout*.



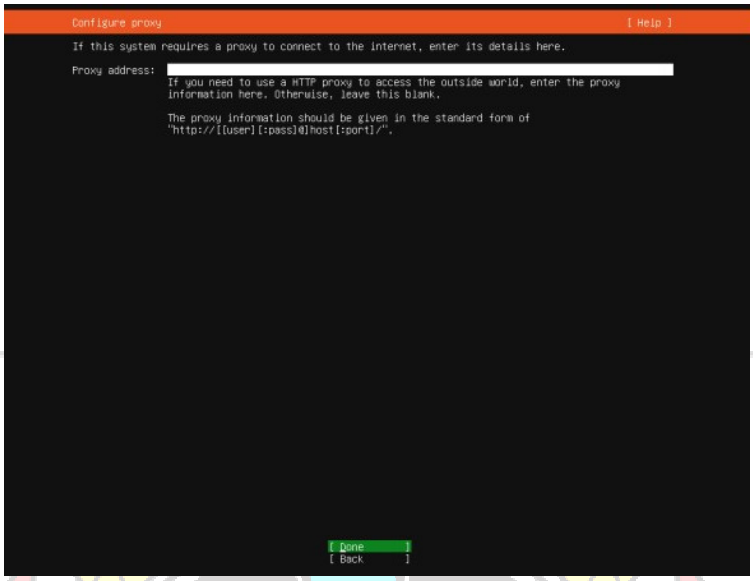
4. Mengatur manual *Subnet, Address, Gateway, dan Name servers* yang akandigunakan, kemudian pilih *Save*.



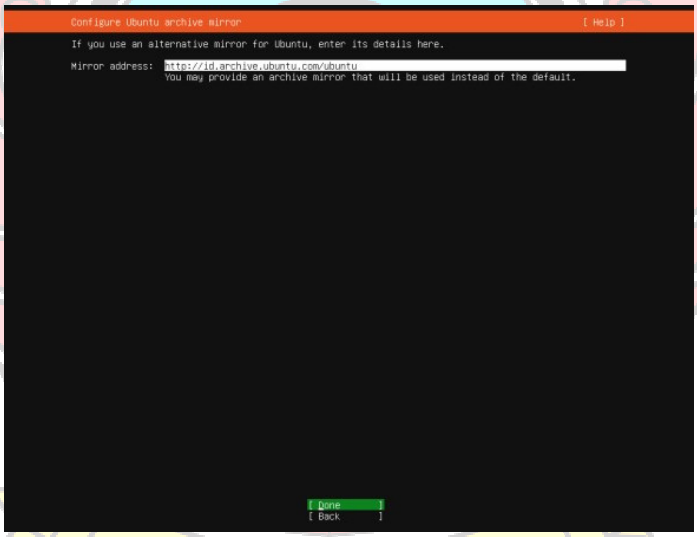
5. Memilih *Done*.



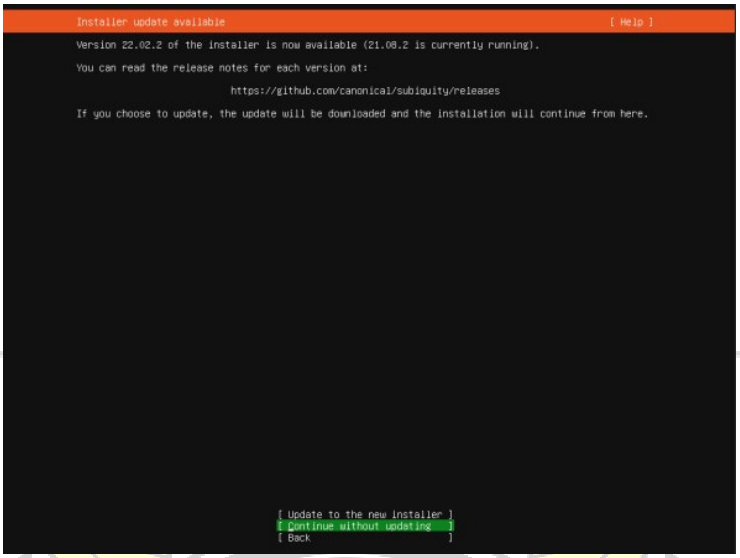
6. Memilih *Done* pada menu *Configure proxy*.



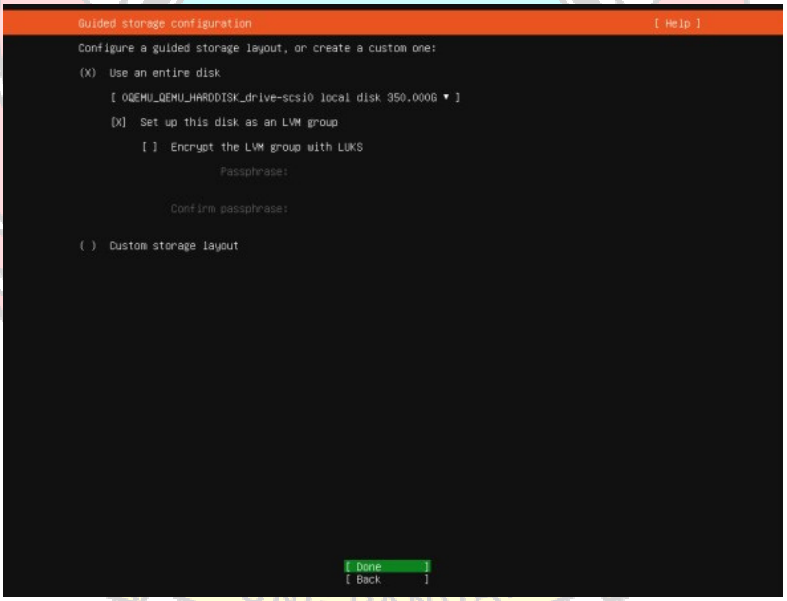
7. Memilih *Done* pada menu *Configure Ubuntu archive mirror*.



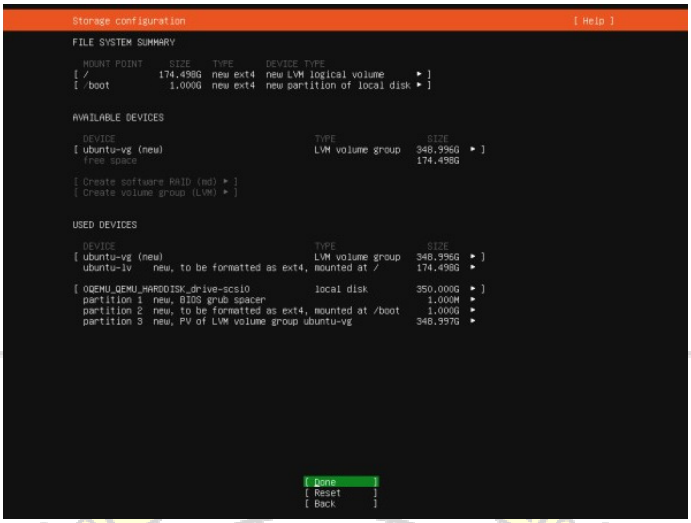
8. Pilih *Continue without updating* pada menu *Installer update available*.



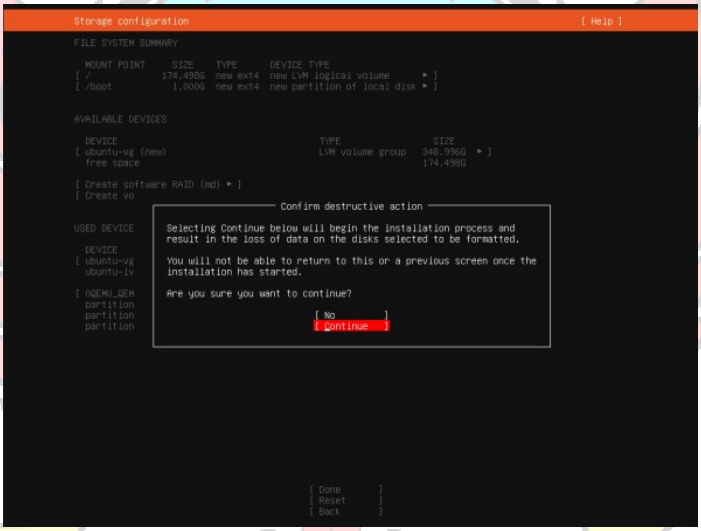
9. Memilih *Done* pada menu *Guided storage configuration*.



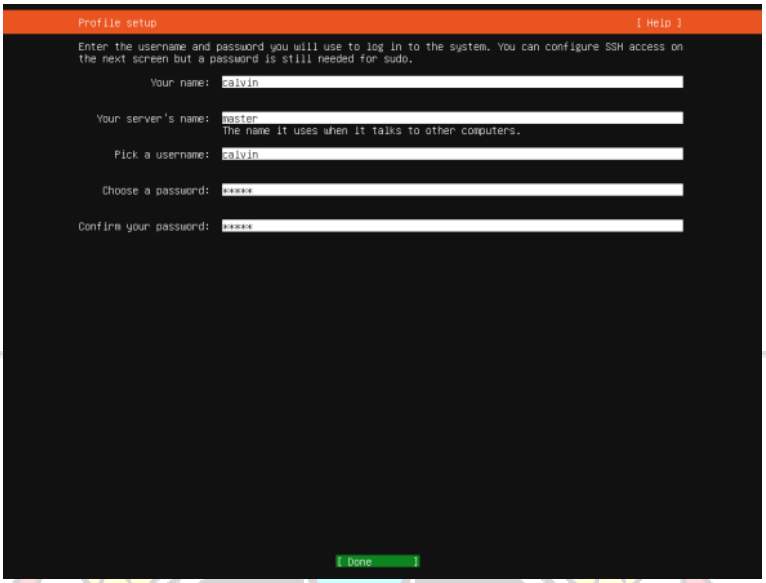
10. Memilih *Done* pada menu *Storage configuration*.



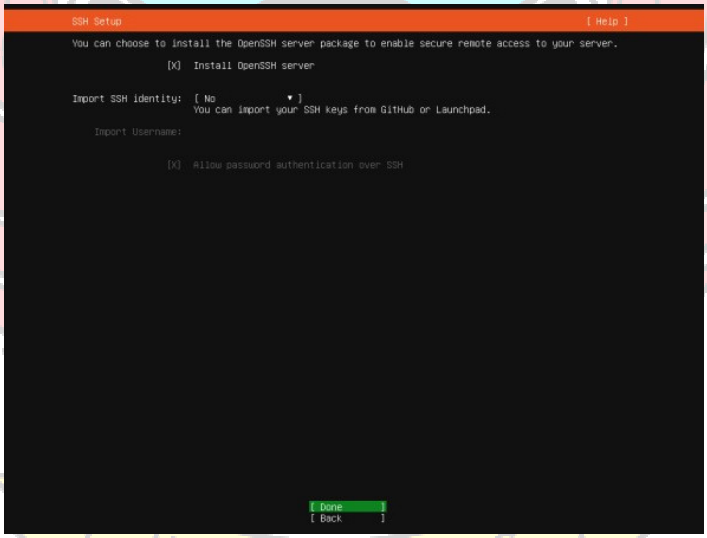
11. Memilih *Continue* pada *Confirm destructive action*.



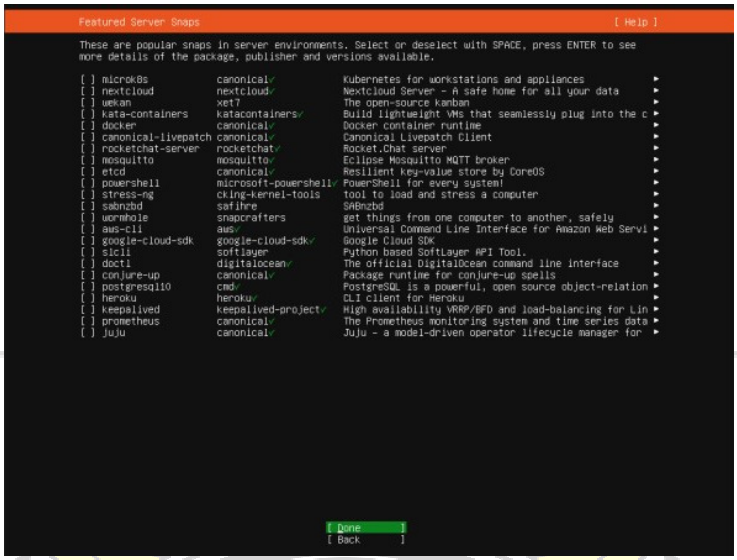
12. Mengisi *Your name, Your server's name, Pick a username, Choose a password, Confirm your password* yang akan digunakan. Kemudian pilih *Done*.



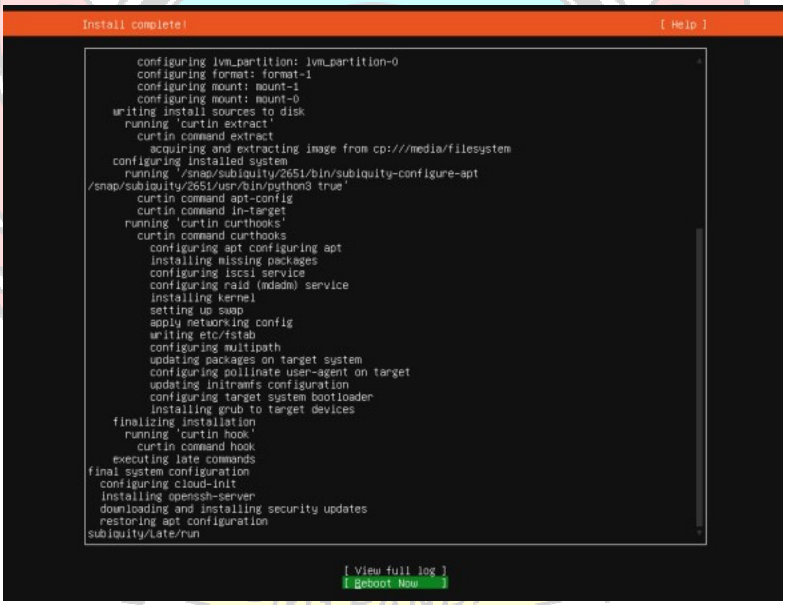
13. Memilih *Install OpenSSH server*, kemudian pilih *Done*.



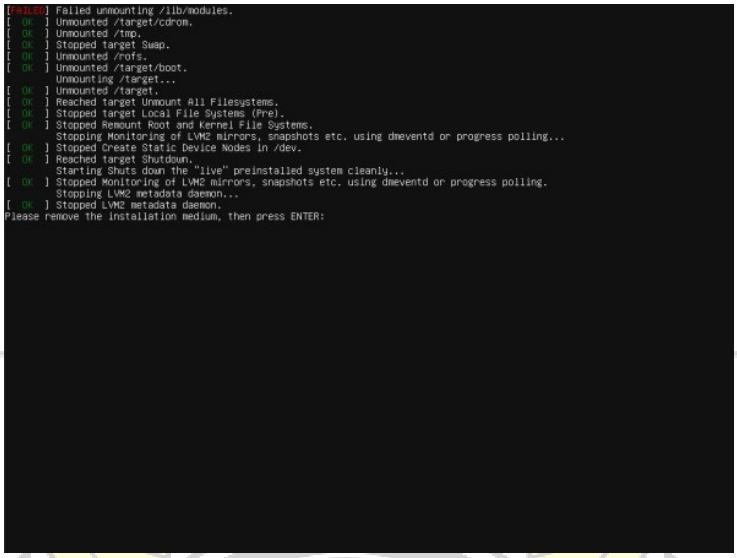
14. Memilih *Done* pada menu *Featured Server Snaps*.



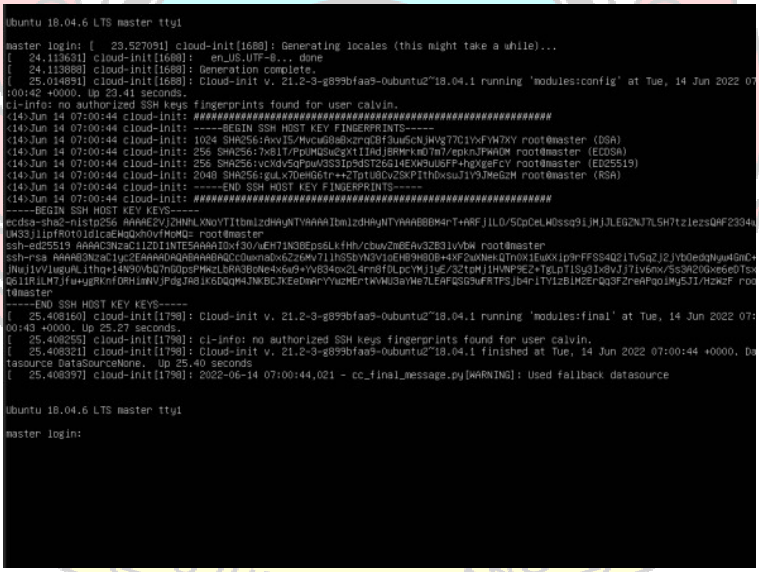
15. Menunggu proses *Installing system* berhasil. Kemudian pilih *Reboot Now*.



16. Menekan Enter.

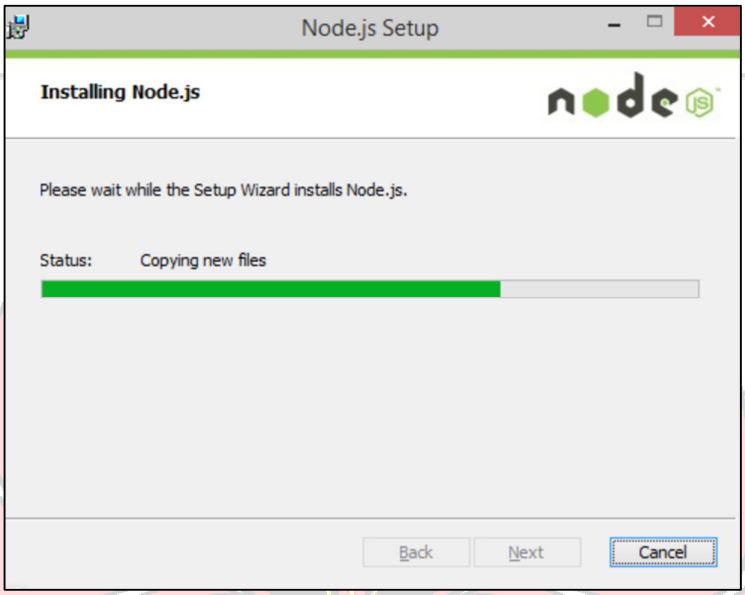


17. Instalasi Ubuntu 18.04 telah berhasil.

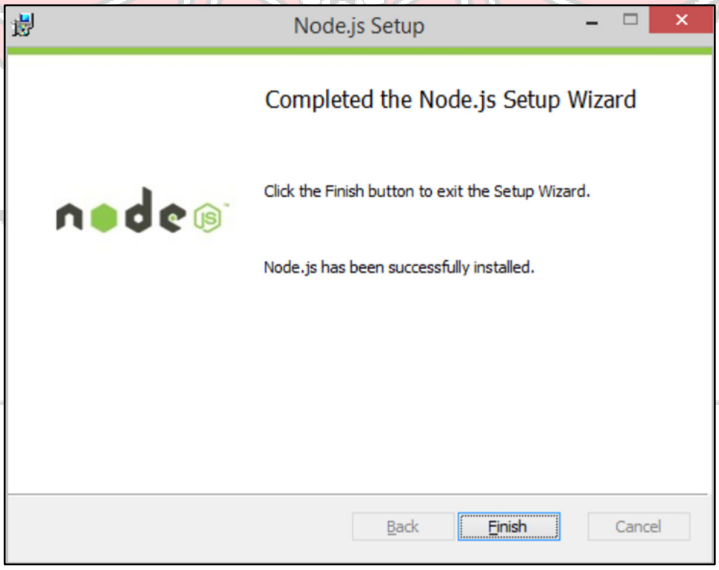


Lampiran 3 Proses Instalasi Node Js

1. Unduh Node.JS *Installer* melalui website resminya https://nodejs.org/download/
2. Jalankan *file* msi yang sudah diunduh.
3. Ikuti petunjuk dan *User Agreement*. Dan tunggu sampai instalasi selesai.



4. Klik “Finish” untuk menyelesaikan proses instalasi.

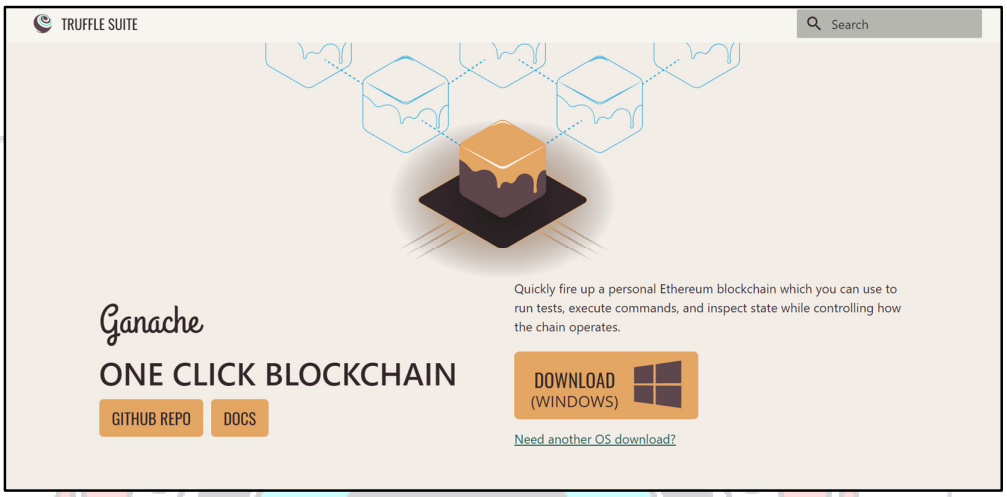


5. Proses instalasi node js telah berhasil

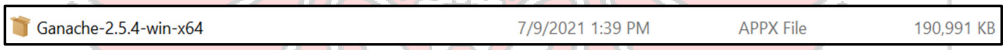
```
C:\Users\Fadhil>node -v
v18.6.0
```

Lampiran 4 Proses Instalasi Ganache

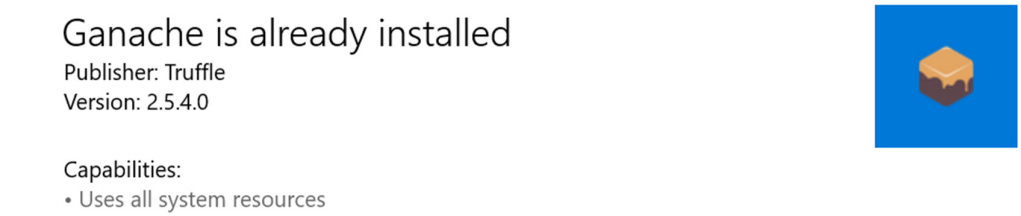
1. Unduh *installer* ganache melalui situs resminya https://trufflesuite.com/ganache/

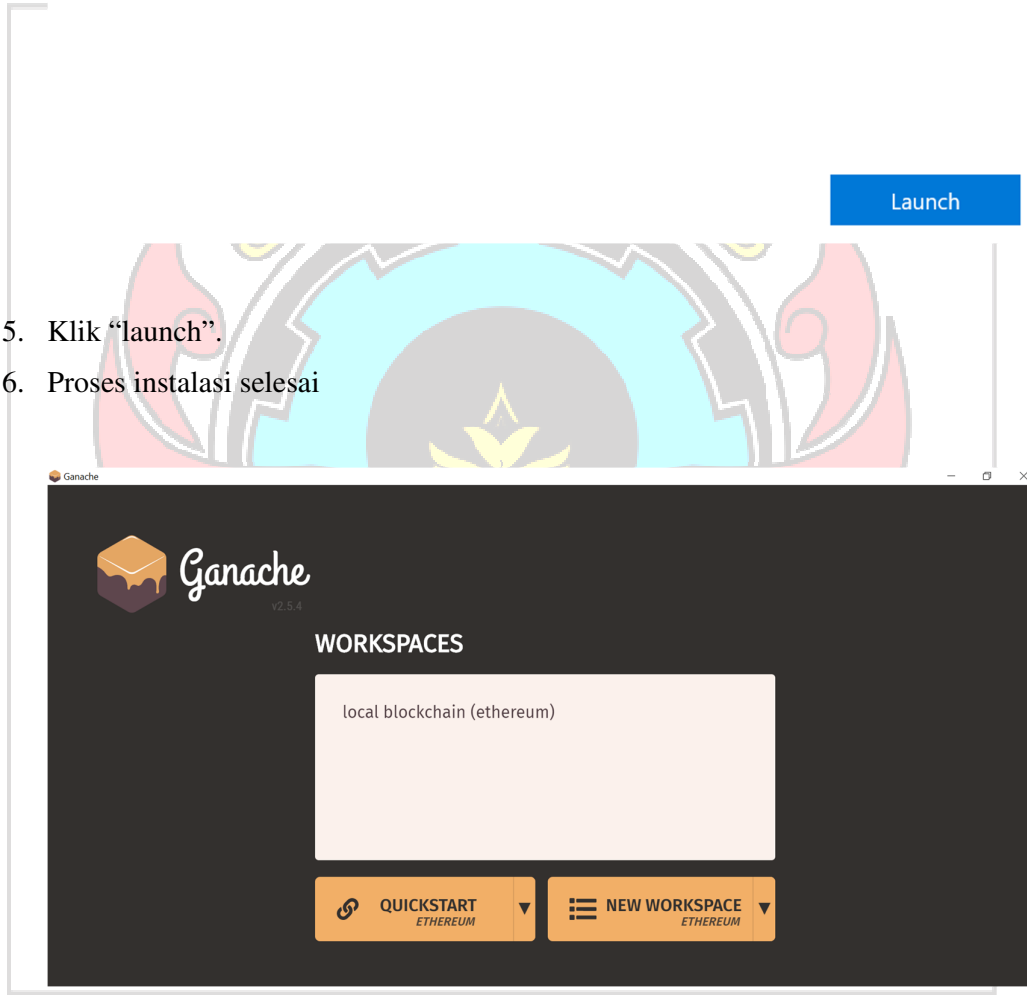


2. Setelah itu buka *installer* yang telah diunduh, installer tersebut tersedia dalam format appx.



3. Ketika dibuka file tersebut akan mengarahkan pengguna ke *windows store* (jika menggunakan sistem operasi windows), dan proses instalasi akan berjalan di sana.
4. Ketika proses instalasi selesai, maka akan muncul tampilan berikut





5. Klik “launch”.
6. Proses instalasi selesai