Blockchain Based Ticket Management

Capstone Intermediate Milestone Submission

Blockchain Group 1 (blockchaing1bbtm)

Ankit Shah
Kalaivanan S
Mohammed Shuaib Mohiuddin
Preeti Aggarwal

Apr 24, 2022

Overview

Develop a unique blockchain based ticket management system to entice users with transparency, and automated refunds and delay penalties for one Eagle Airlines.

Set-up a private Ethereum blockchain and develop a base ticketing contract, that would allow immediate refund in case of cancellations and predefined penalty payment in case of delays.

Scope

01 Infrastructure - EC2 instances (2) to run Geth-CliquePoA nodes of the Private Blockchain network that hosts the Airline Ticket Management system and related Airline & Customer Accounts

02 Core Application - Solidity contracts to implement the Ticket Management System's interaction with Airline & Customers.

03 Advanced Features
Infra - Hyperledger Besu nodes with Tessera to mark transactions as private.
Application - Implement Oracle to fetch Flight details. Multiple cancellation & penalty scenarios. If possible, dApp creation using Truffle - one for Airline and one for Customers.

Private Blockchain Network Setup

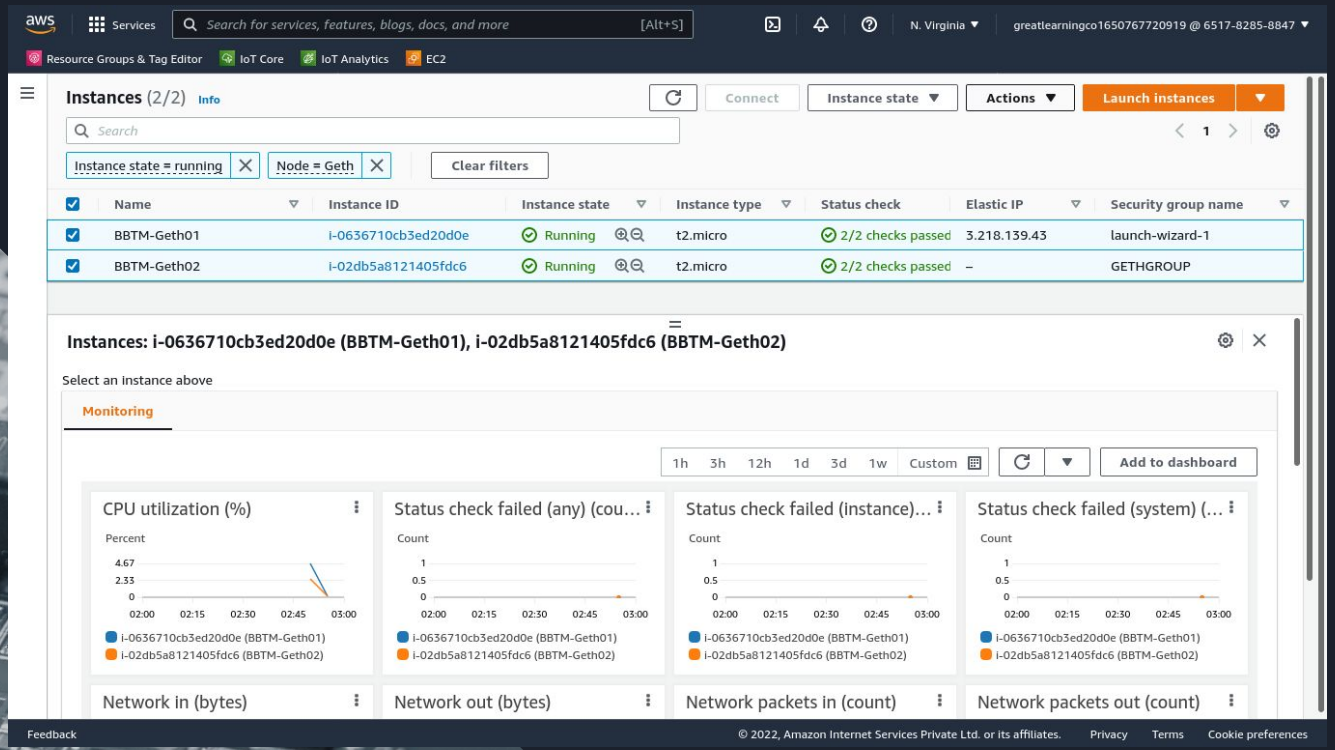
01 Infrastructure

EC2 instances
2 x t2.micro

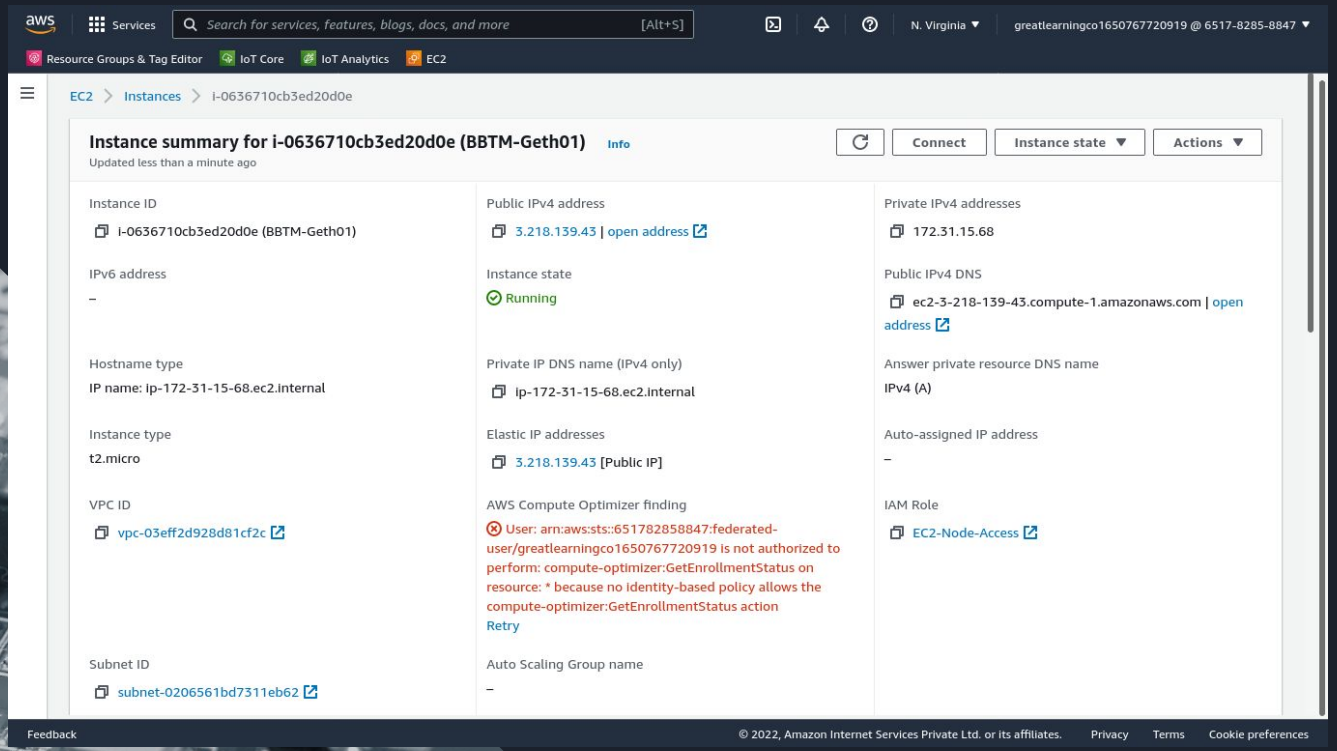
Blockchain Ethereum Clique
- 2 geth instances (one in each EC2) with 3 nodes
Instance-1 - 1 bootnode + 1 peer node
Instance-2 - 1 miner node

BC Accounts
2 Airline Accounts + 4 Passenger Accounts

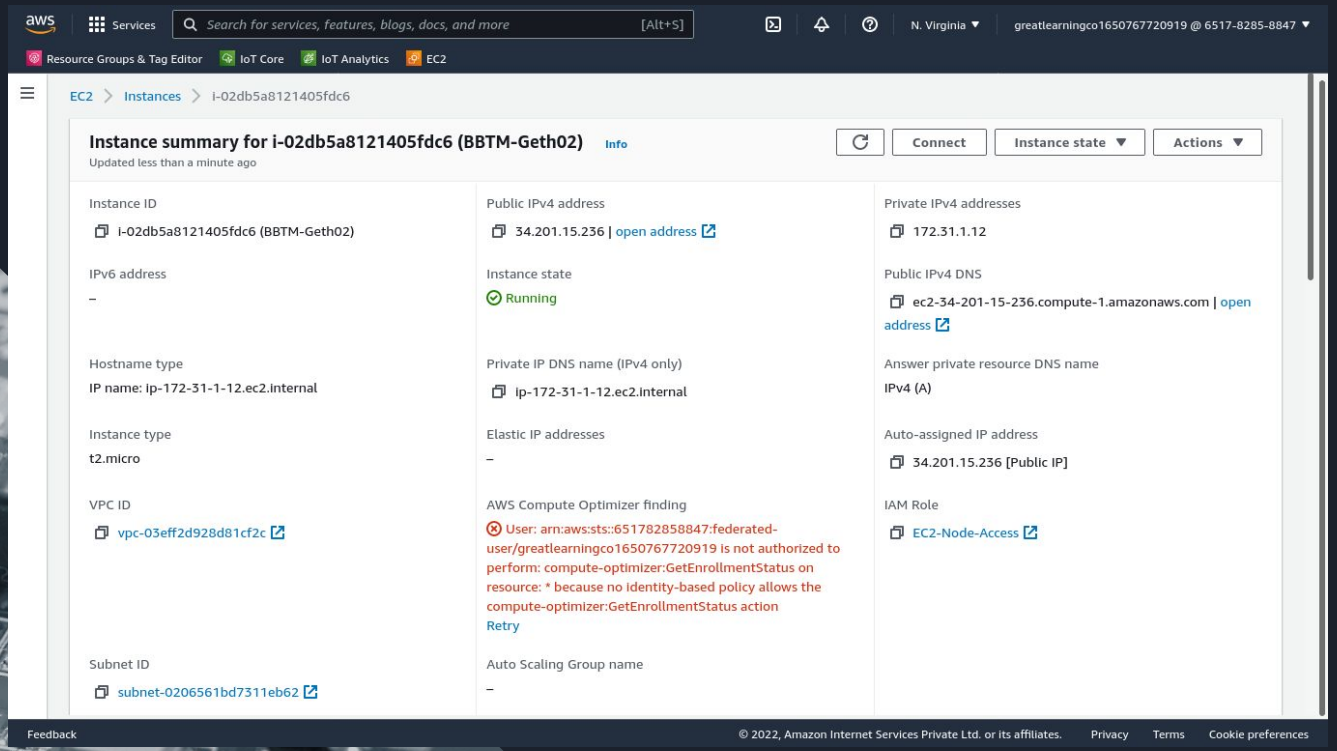
EC2 instances



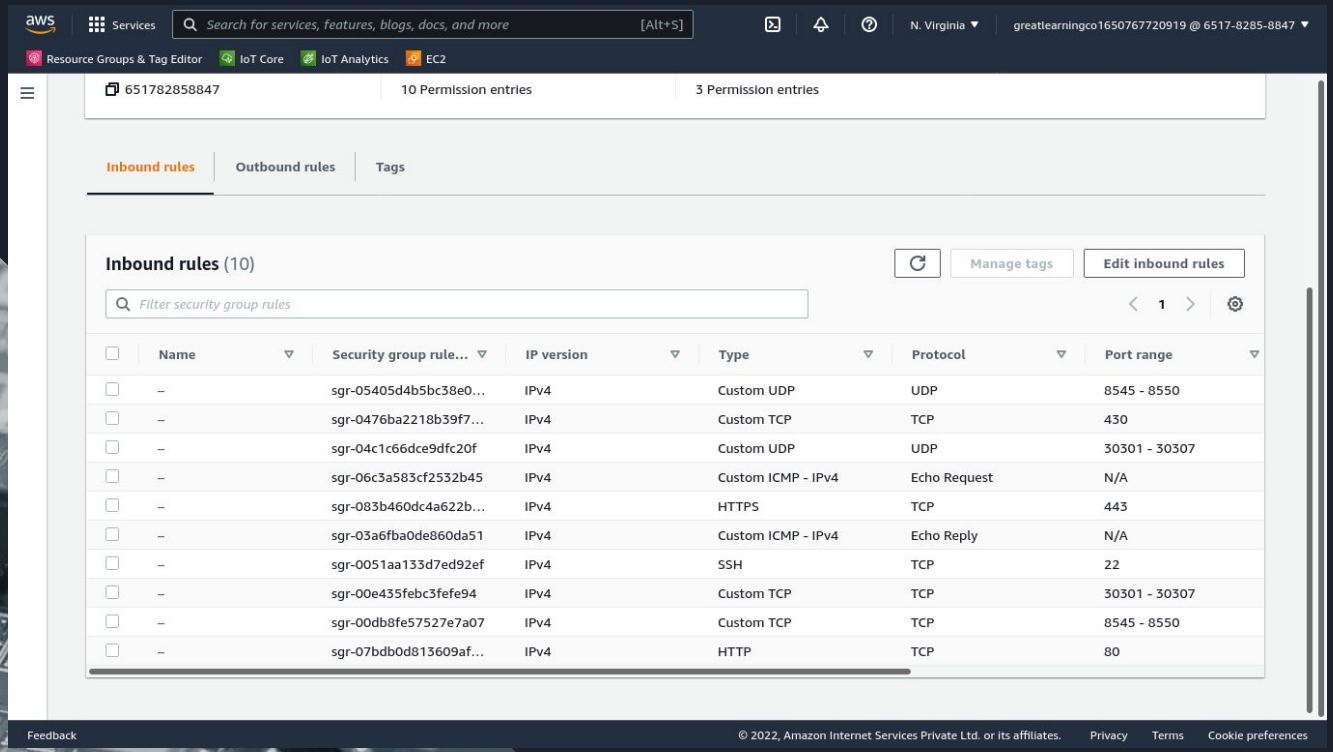
EC2 instances - BBTM-Geth01



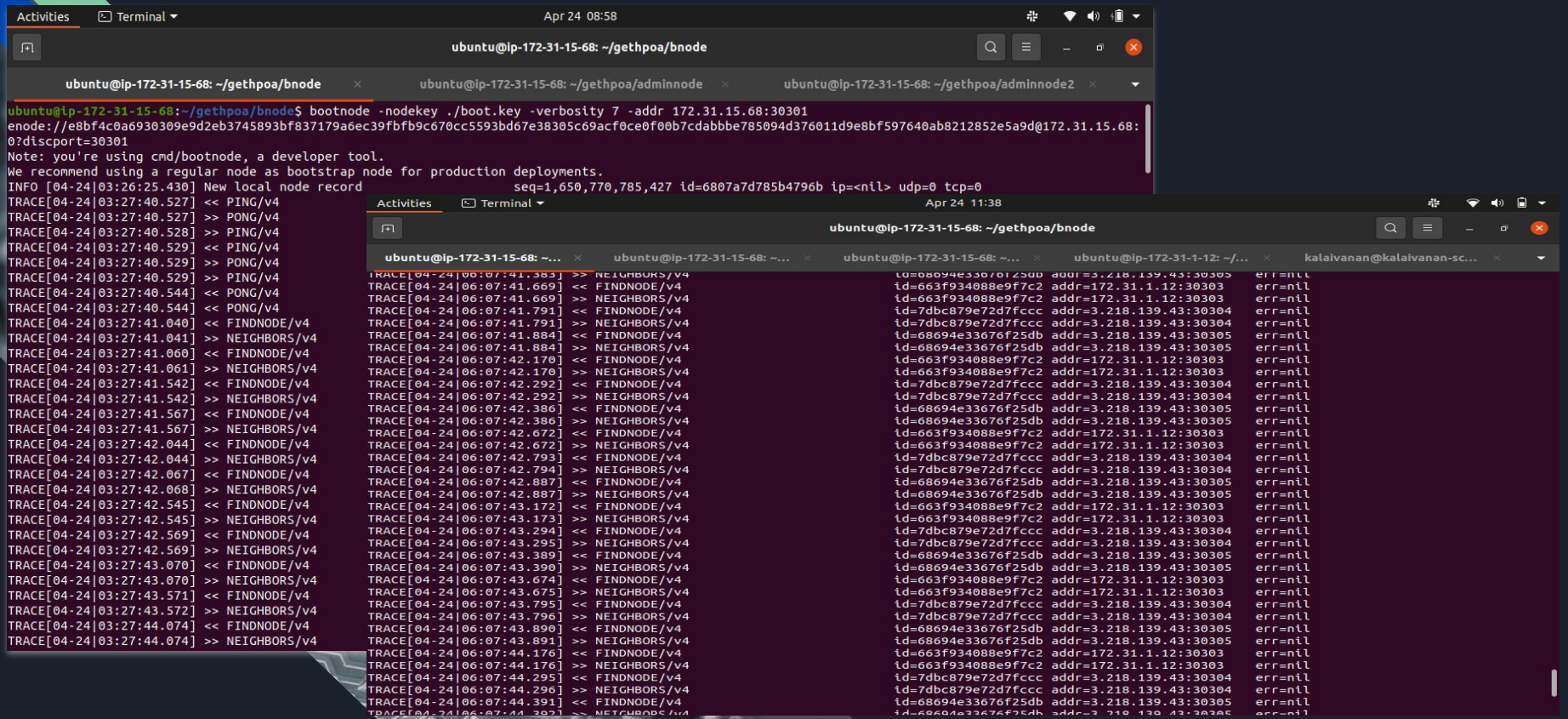
EC2 instances - BBTM-Geth02



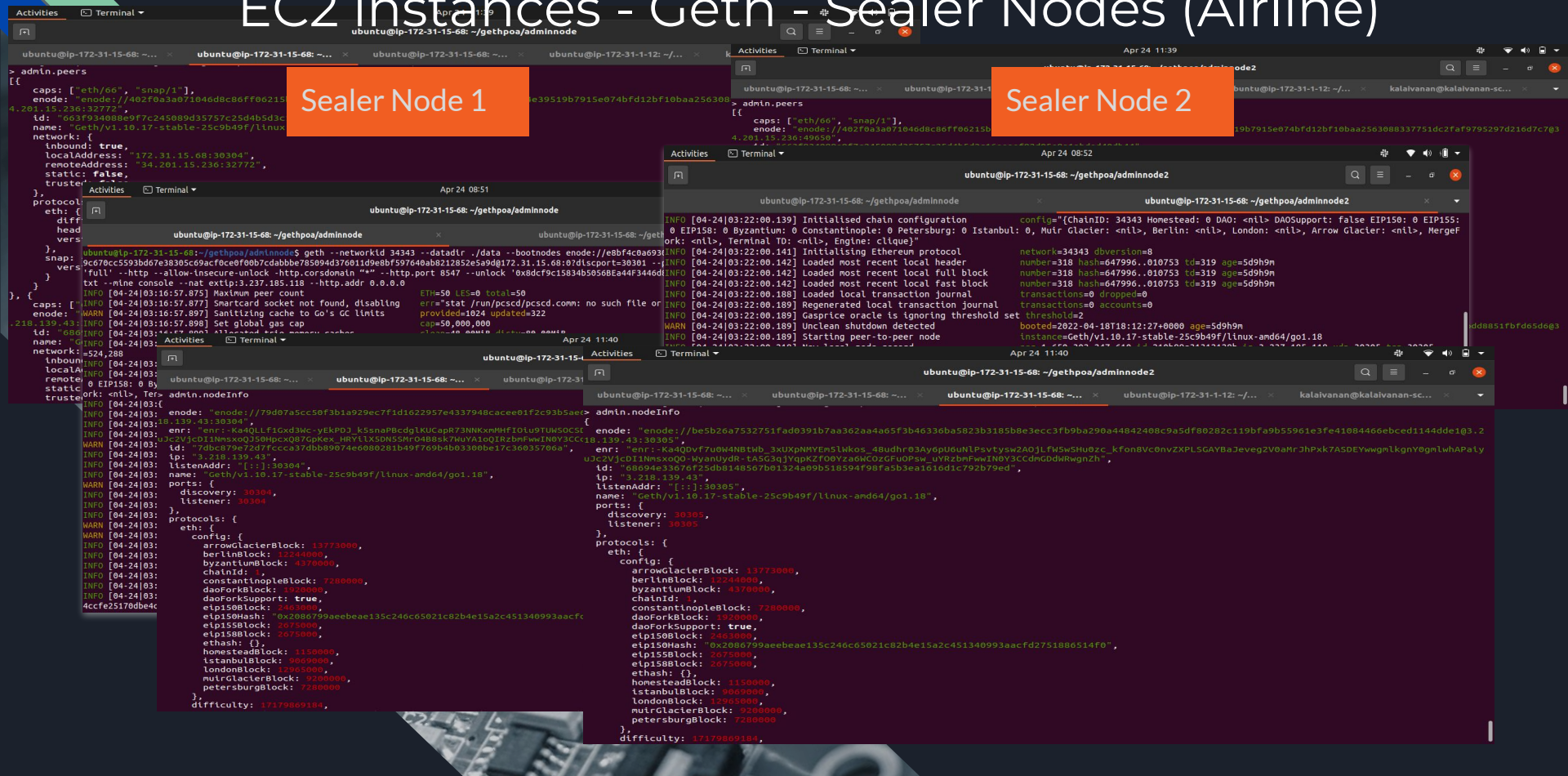
EC2 instances - Inbound Rules



EC2 instances - Geth - Boot Node

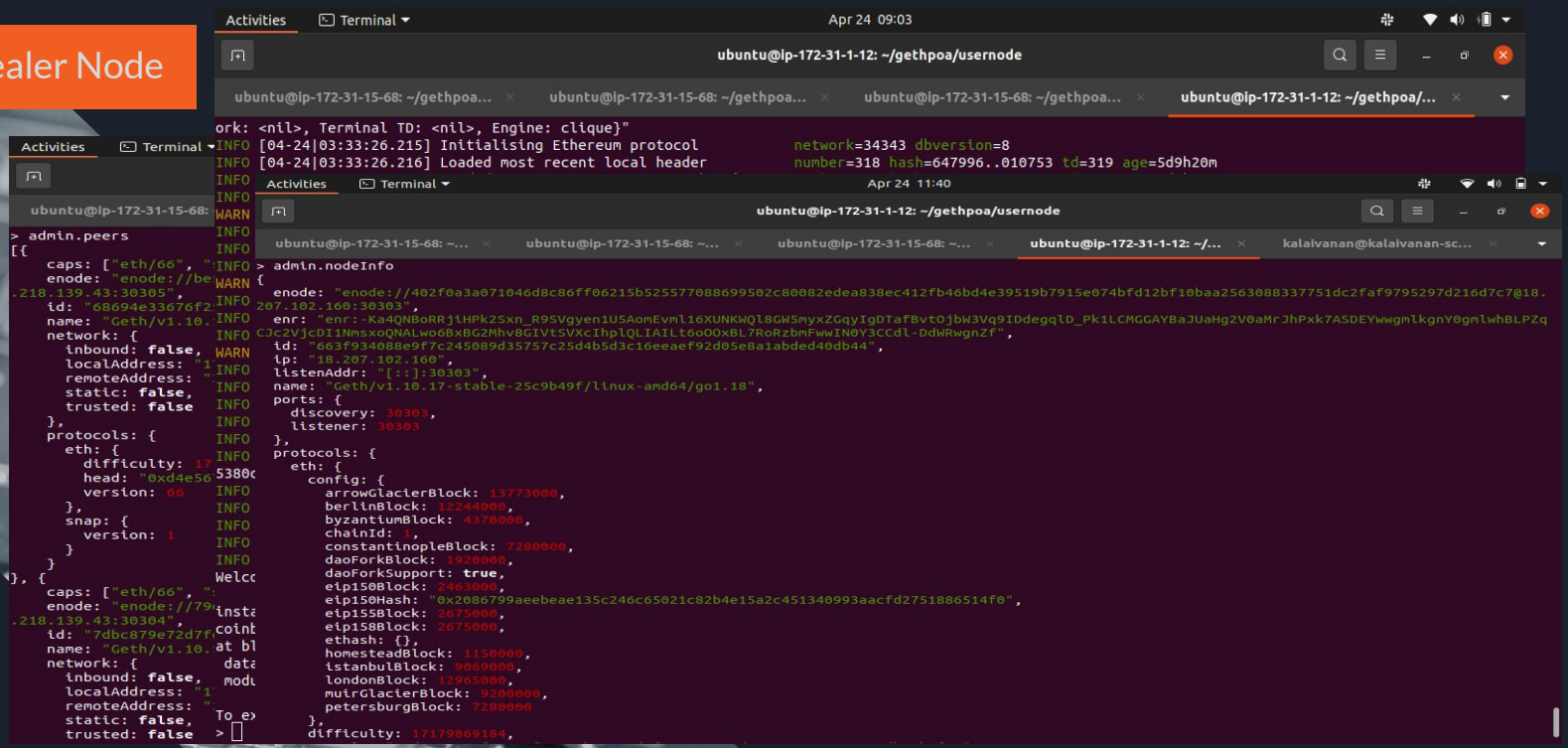


EC2 instances - Geth - Sealer Nodes (Airline)

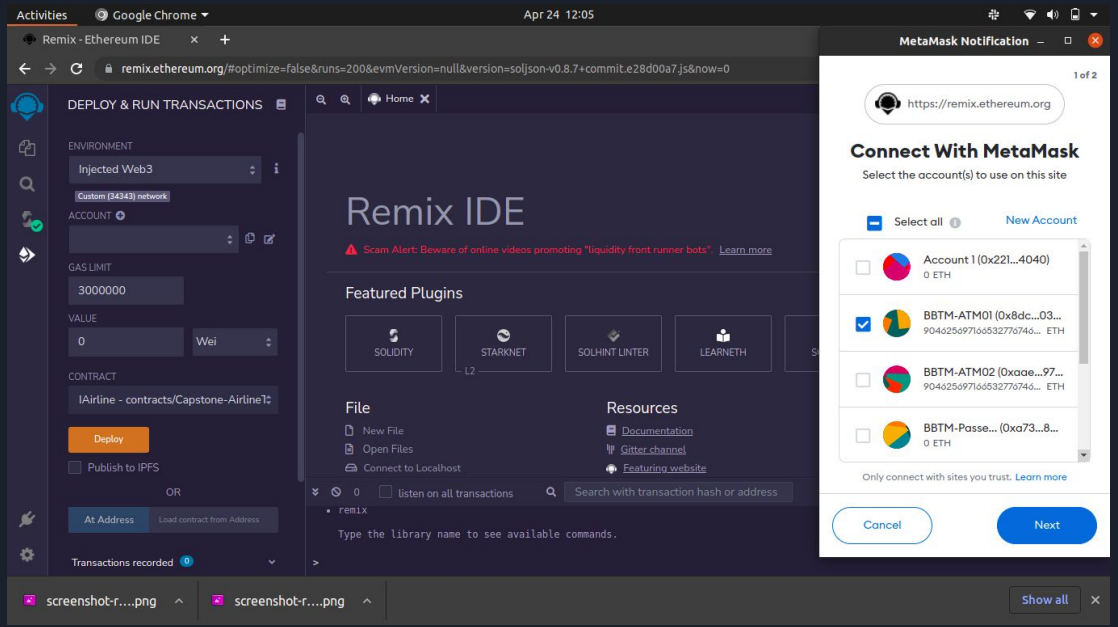


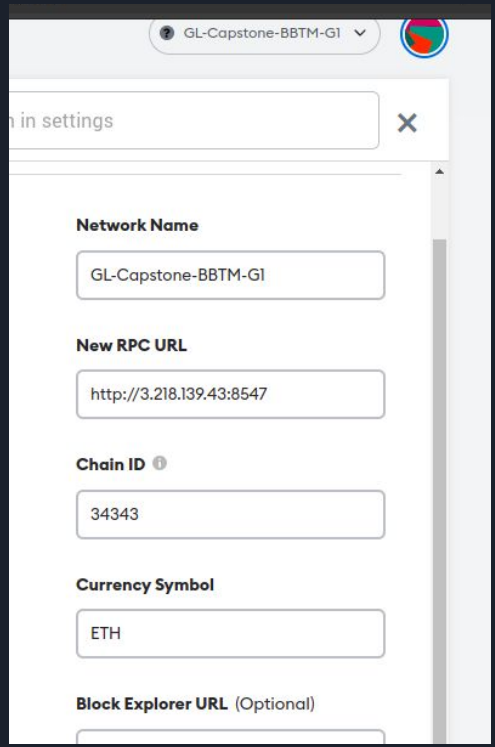
EC2 instances - Geth Passenger Node

Non Sealer Node



Connecting BBTM - Private Blockchain Network via Metamask



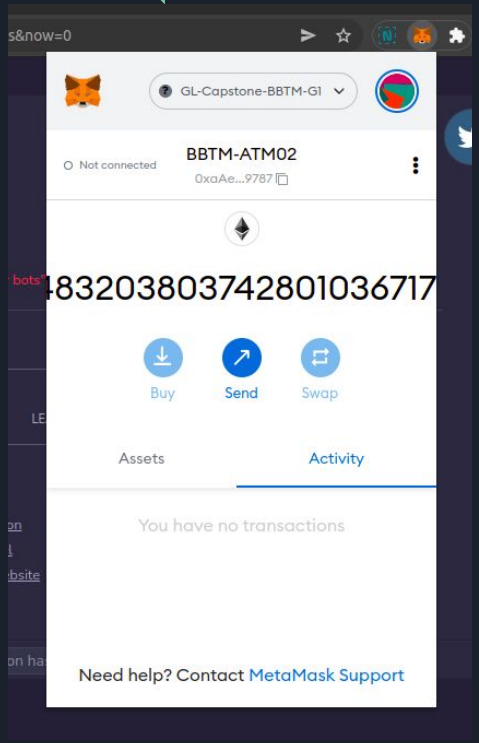


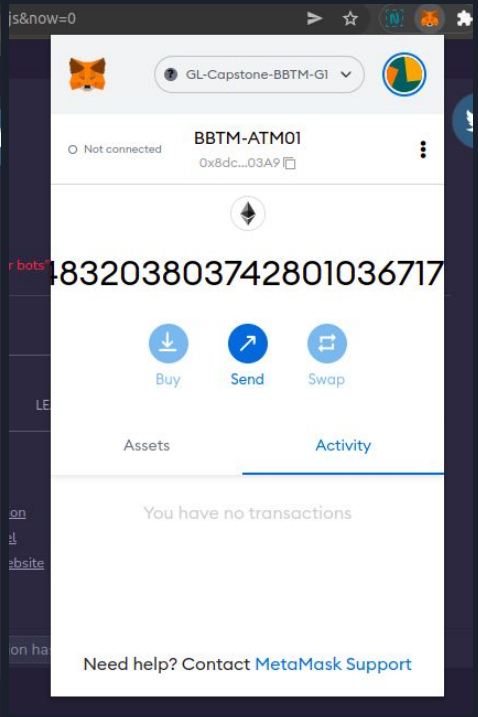
Airline Addresses

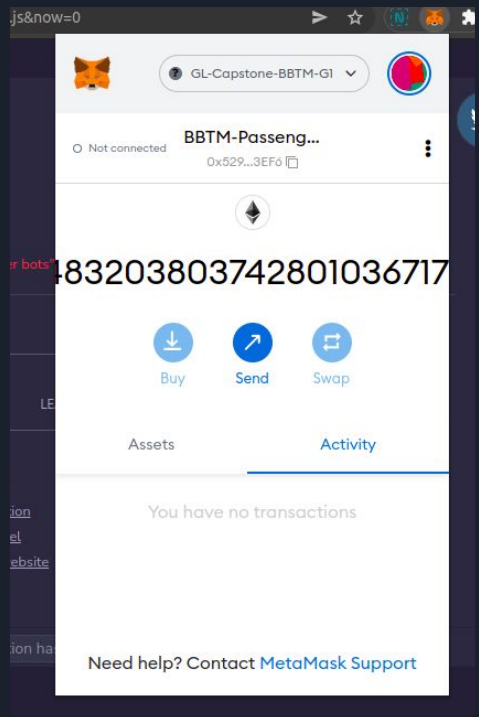
BBTM-ATM01
0x8dcf9c15834b5056BEa44F3446d860bE532903A9
BBTM-ATM02
0xaAeF6375968F1DA352F7Df62FD597b8B91739787

Passenger Addresses

BBTM-Passenger001 - 0xa7386bBA929DA248F83577723AeE5dEE7B128AD7
BBTM-Passenger002 - 0x6759bf46596B16e00450D9e965F33C0108fca65c
BBTM-Passenger003 - 0x529A5feEcB002B47B611bB32cEcCc944f1873EF6
BBTM-Passenger004 - 0x92FcF68DbAe5c4C08D30870Ea156772aC2B46b7B



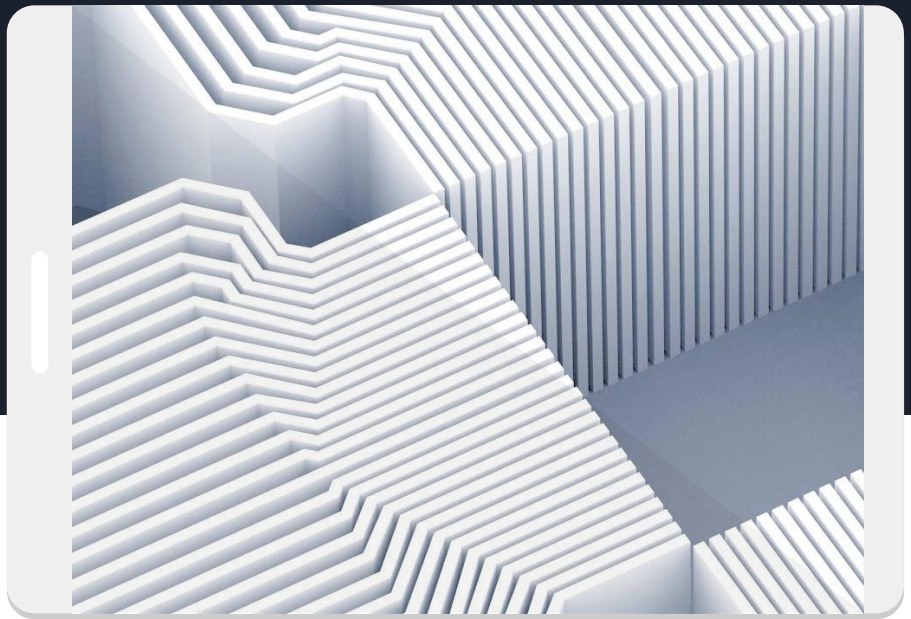




Private Blockchain creation - Challenges & Learnings

1. Initially nodes running were not able to find any peers, as they other node was running on another ec2 instance. We used local host port 30301 to run boot node, And node on another EC2 kept searching for boot node on local 30301 port. USE of private ip on boot node along with 30301 port made possible for another node on ec2 to attach to that particular boot node and find peers connected to that bootnode.
2. Initially both sealer admin accounts were assigned to one node and due to that they were not able to mine block. **Running both miner accounts** separately **on different sealer node** made possible for them to mine blocks one at a time.
3. Initially metamask couldn't connected to chain, as there was no public IP assigned to node. By default 127.0.0.1 was assigned to each node. **Using "--nat extip", we overcome this hurdle**.
4. Even after assigning Public Ip seemed not working, and Use of METAMASK & injected web3 was not possible Initially as only rpc calls to local host only allowed. HOWEVER with use of flag "--http.addr 0.0.0.0", all rpc calls from anywhere were allowed and we could connect to chain using a valid rpc URL

02 Core Application



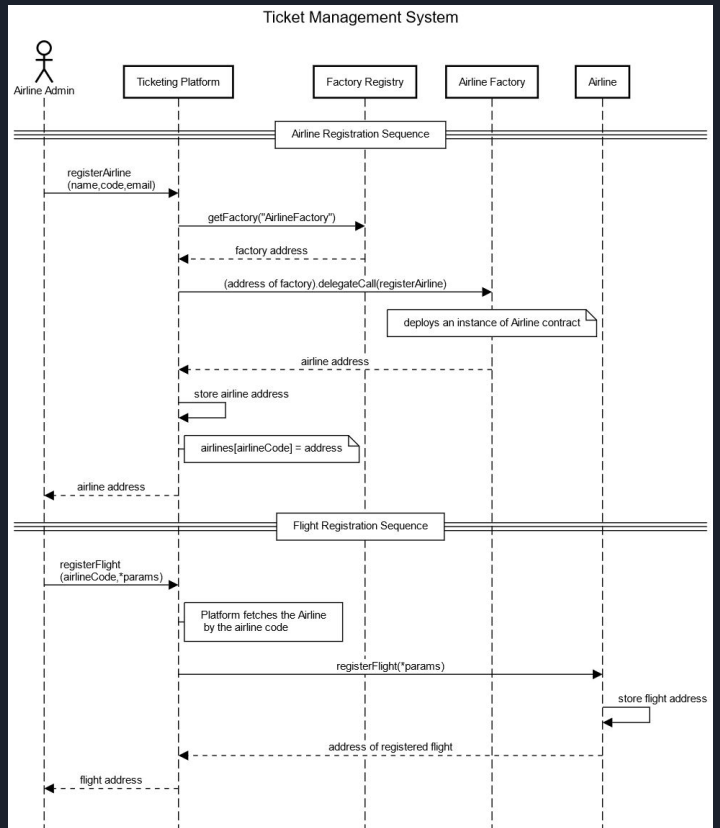
Feature Summary

Register Airlines, Flights, Customers

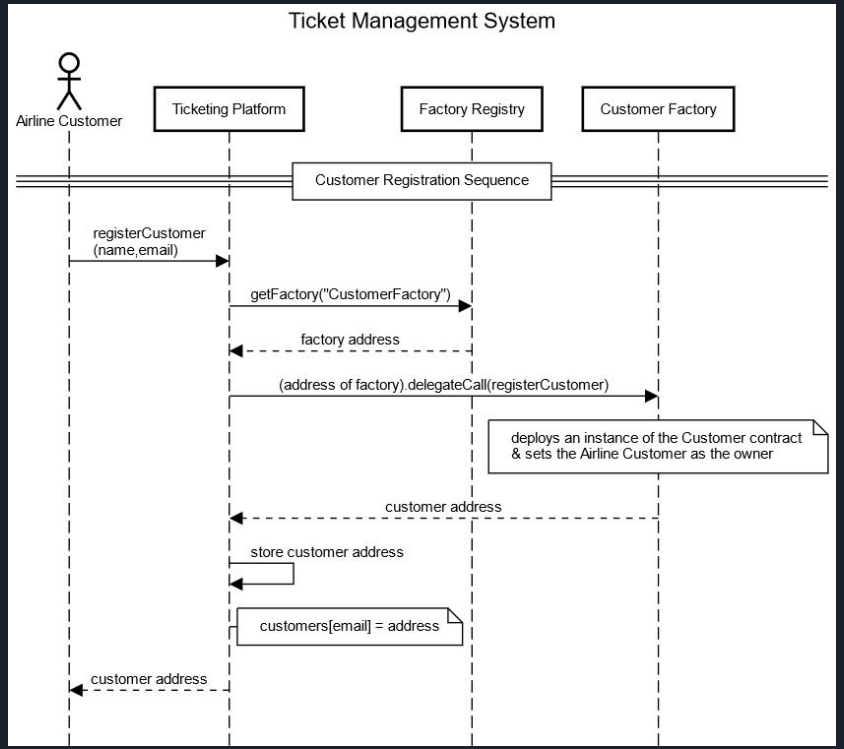
List Available Seats, Book & Cancel Tickets, Penalties & Refunds

Contract Design - Registrations

Registering Airline & Flight

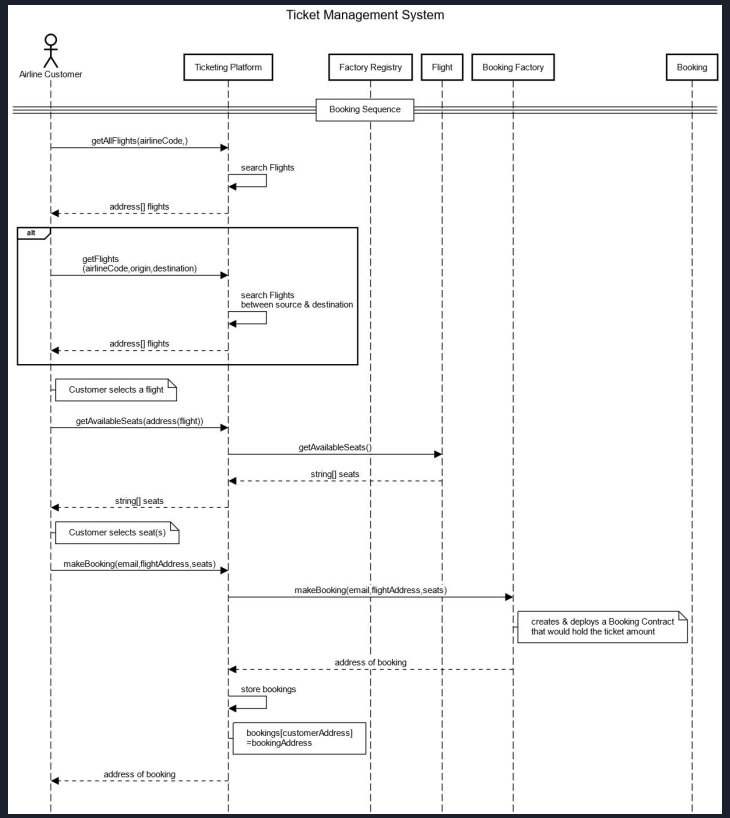


Registering a Customer

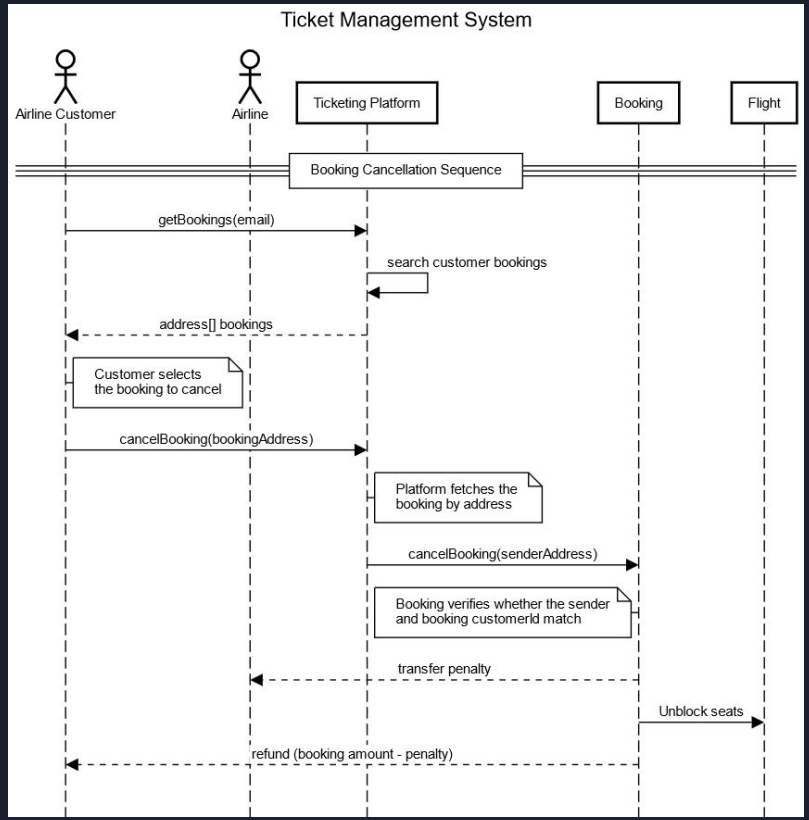


Contract Design - Customer Booking & Cancellation

Ticket Booking

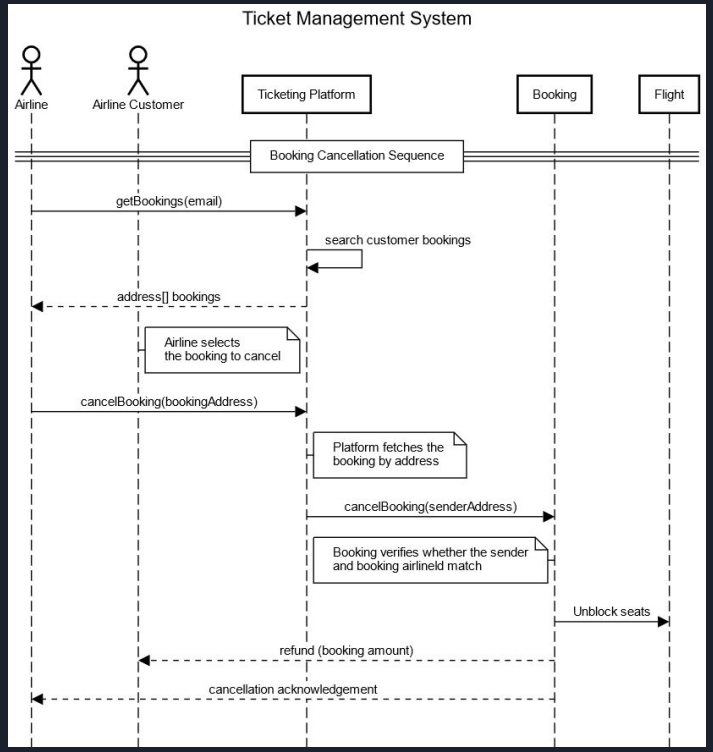


Ticket Cancellation by Customer

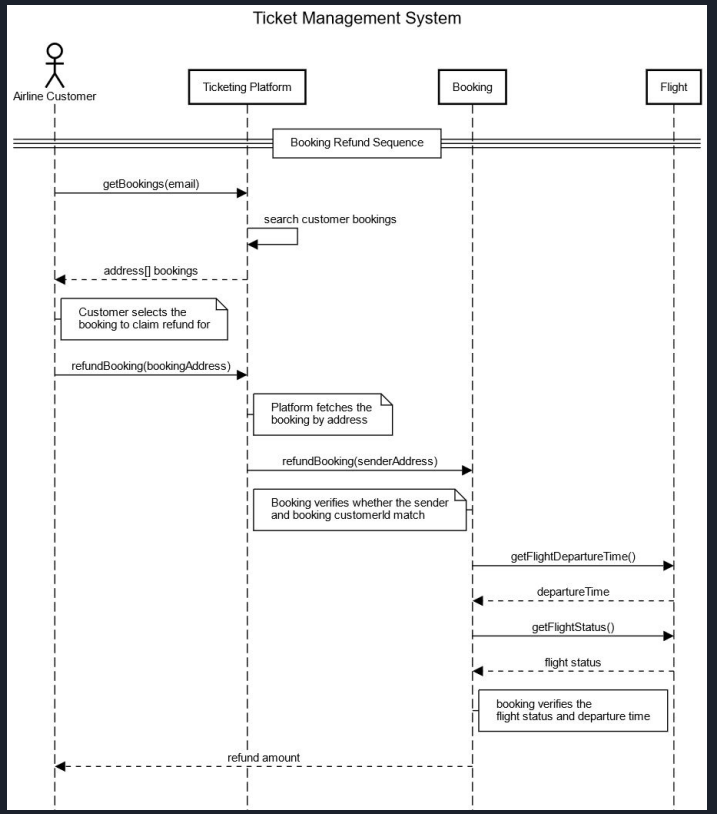


Contract Design - Airline Cancel & Claim Refund

Ticket Cancellation by Airline



Refund Claim by Customer



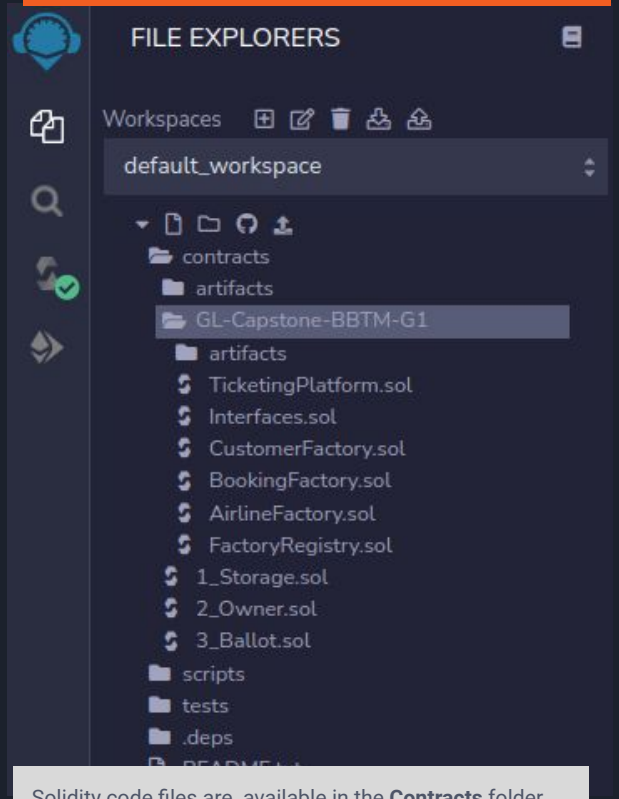
Initializing the BBTM on the Private Blockchain

1. Connect to one of the Airline Accounts (say, BBTM-ATM01) in Metamask and connect via InjectedWeb3 in Remix IDE
2. Deploy the below contracts* (no particular order)
    - AirlineFactory
    - BookingFactory
    - CustomerFactory
    - TicketingPlatform
3. Call the **getRegistryAddress** function of the deployed **TicketingPlatform** contract and copy the Address returned
4. Call the **updateRegistry** function of the Factory contracts - AirlineFactory, BookingFactory and CustomerFactory, filling in the copied Registry Address
5. System is ready for Ticket Management. Start by registering Airlines, Customers and Flights.

* Solidity code files are available in the **Contracts** folder

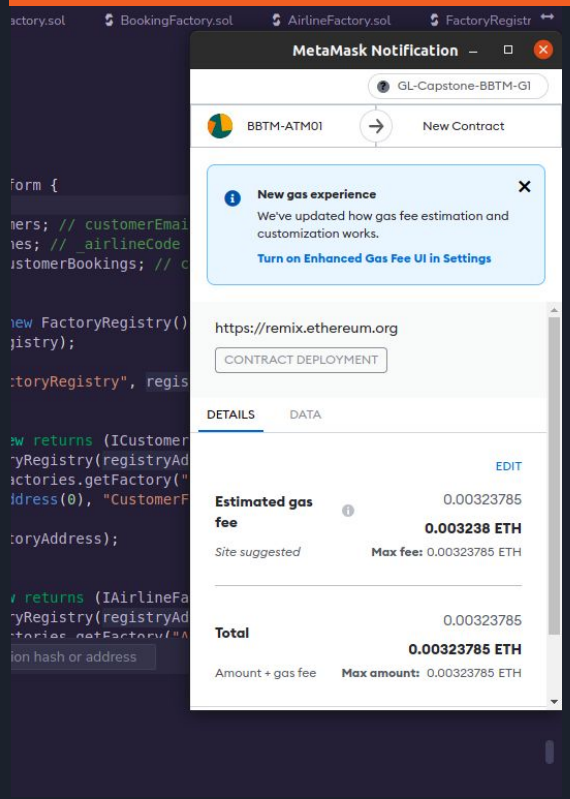
Contract Deployment

1. Import Contract Files

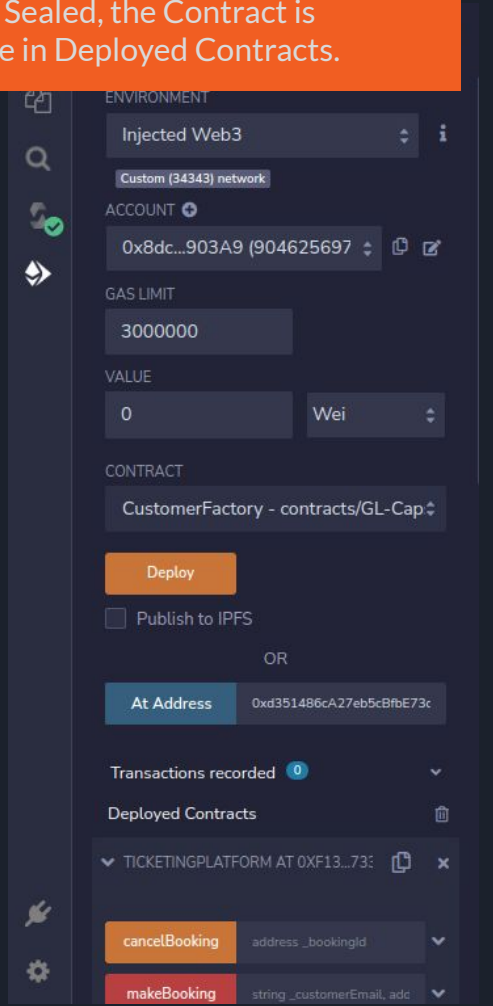


Solidity code files are available in the **Contracts** folder

2. Approve Contract Deployment

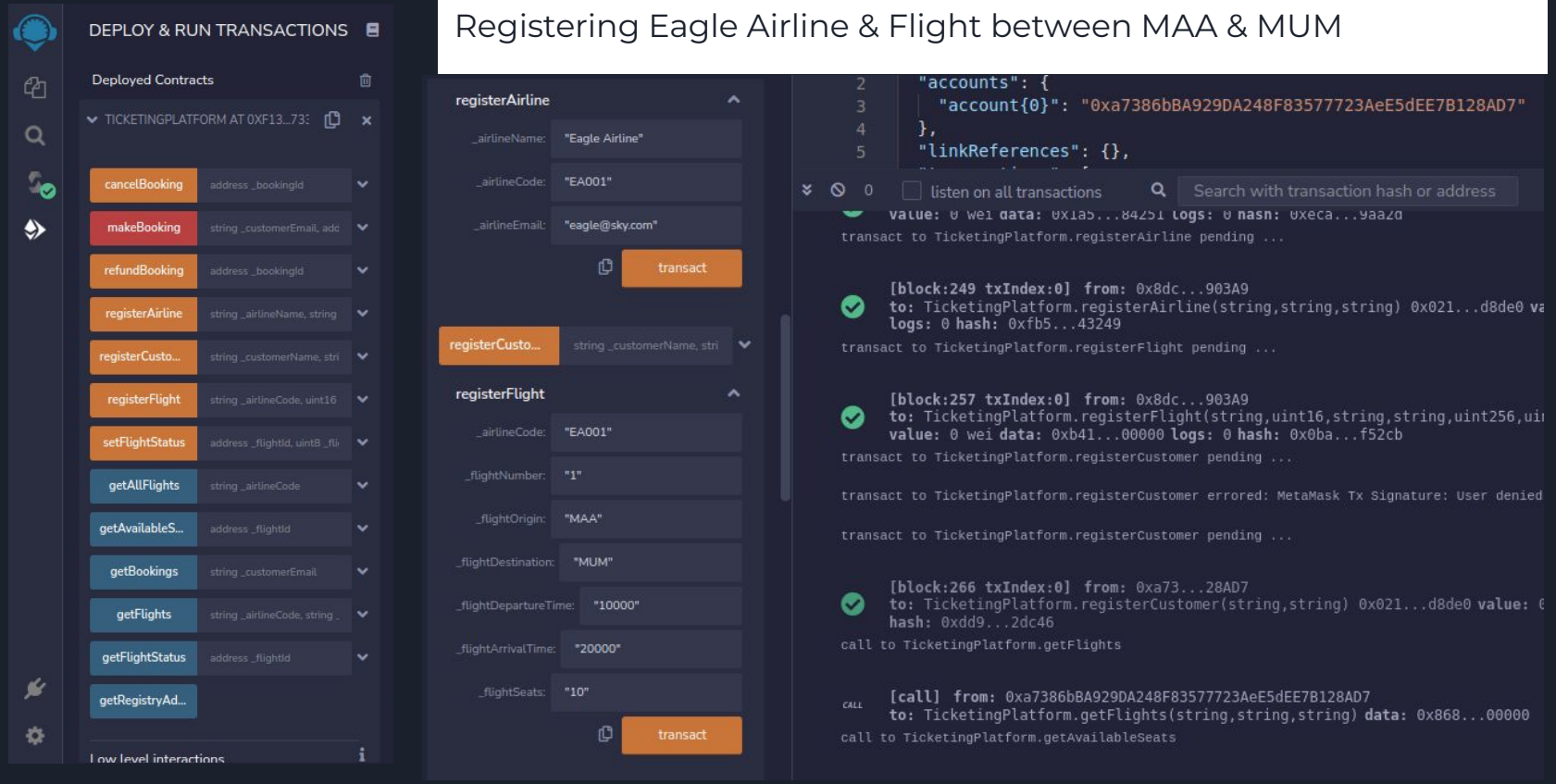


3. Once Sealed, the Contract is available in Deployed Contracts.



#BBTM: In Action

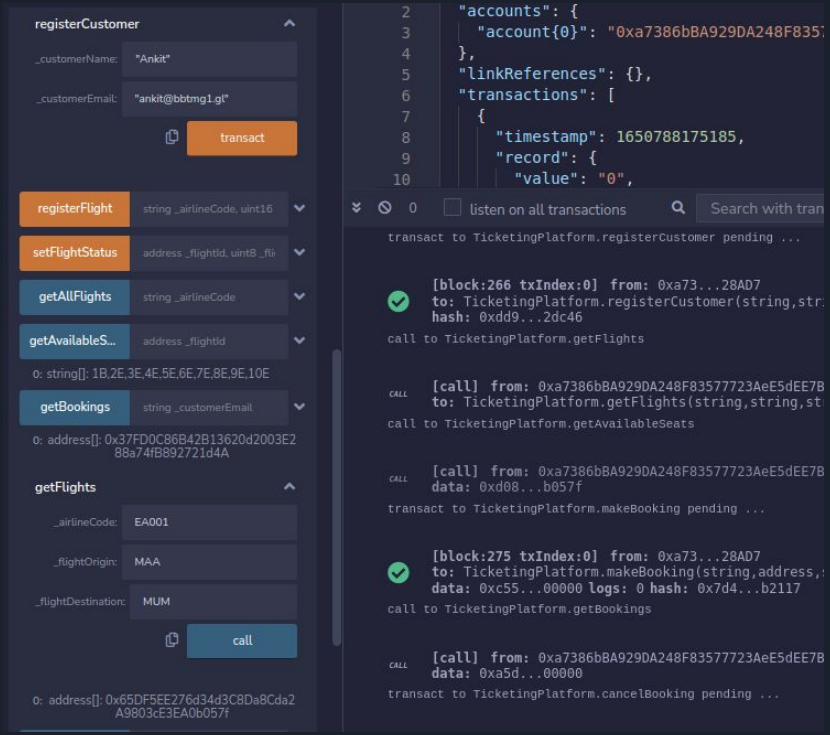
Registering Eagle Airline & Flight between MAA & MUM



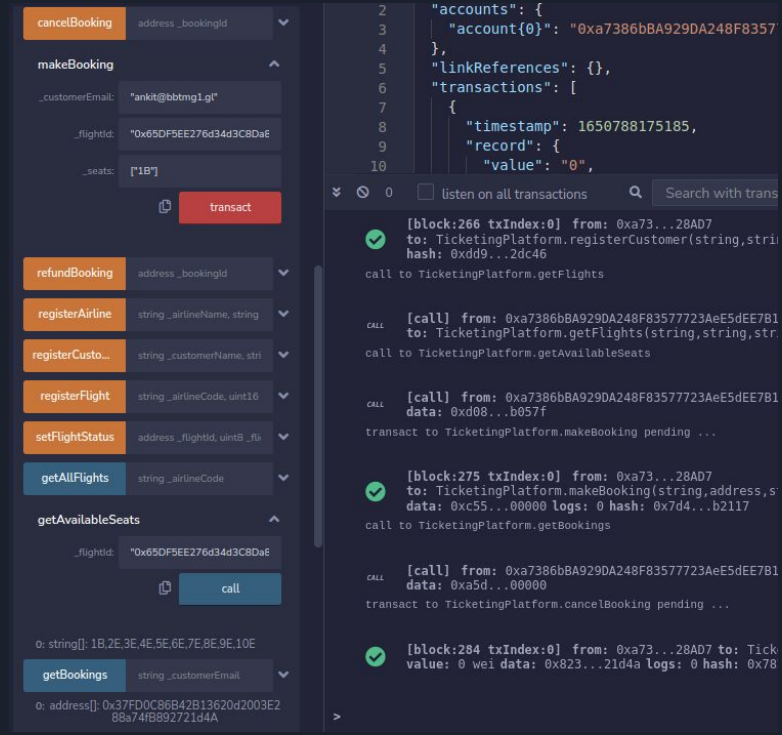
Watch videos in the **Artifacts** → **Demo Videos** folder for better understanding

BBTM: In Action

Registering a Customer & Searching a Flight



Seat Search, Booking & Cancellation



Watch videos in the **Artifacts** → **Demo Videos** folder for better understanding

Smart Contract Development - Challenges & Learnings

1. Deciding on what functions to keep and what to remove when code size hits the limit of 24Kb. Solved by distributing the features across multiple smaller contracts by using **Interface Proxies** and **delegatecall** functions.

Contract code available in the **Contracts** folder

Responsibilities

01 Infrastructure - Configuring EC2 instances & Geth nodes - Ankit Shah & Kalaivanan S

02 Core Application - Base Contract Development - Preeti Aggarwal & Shuaib Mohammed

Artifacts

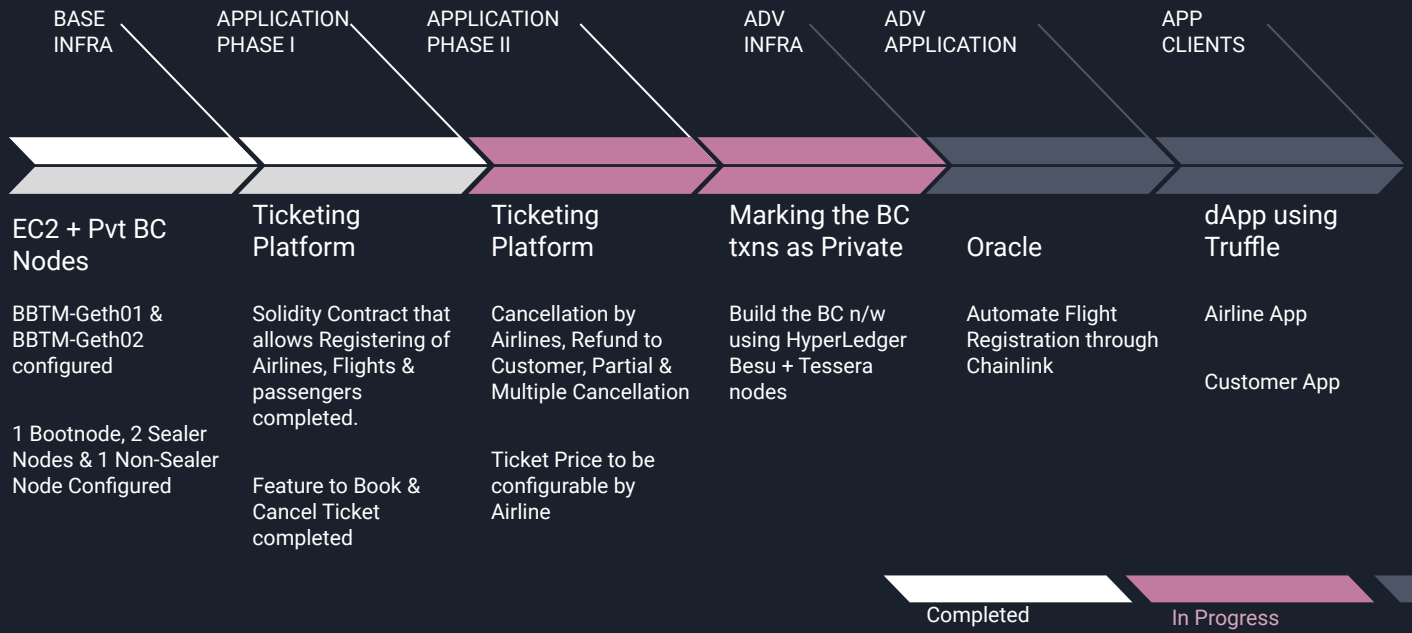
EC2 & BC Node Commands - Ankit Shah

Sequence Diagrams - Shuaib Mohammed

README - Preeti Aggarwal

Scope Document & Demo Screen Captures / Videos - Kalaivanan S

Project Status



Thank you!