**Eagle Airlines Ticket Management System – Final Submission**

Blockchain - Group 2 – Dec 2021

* Team
  + Ashhar Mohammad
  + Gigo Mathew Idicula
  + Kumaran Murugesan (Leader)
  + Santosh Venkatesh Yadav
  + Satyendra Kumar

Contents

[Assumptions 3](#_Toc88861165)

[Architecture 3](#_Toc88861166)

[Functions 4](#_Toc88861167)

[Infrastructure Setup (Geth EVM on AWS) 4](#_Toc88861168)

[Configure security group in AWS 12](#_Toc88861169)

[Configure nodes and accounts 13](#_Toc88861170)

[Configure Genesis File 18](#_Toc88861171)

[Initialize Nodes 20](#_Toc88861172)

[Create and Start Boot Node 22](#_Toc88861173)

[Start Our nodes 22](#_Toc88861174)

[Contracts 27](#_Toc88861175)

[Testing 27](#_Toc88861176)

[TMS Airline 27](#_Toc88861177)

[TMS Portal 44](#_Toc88861178)

[TMS Ticket 47](#_Toc88861179)

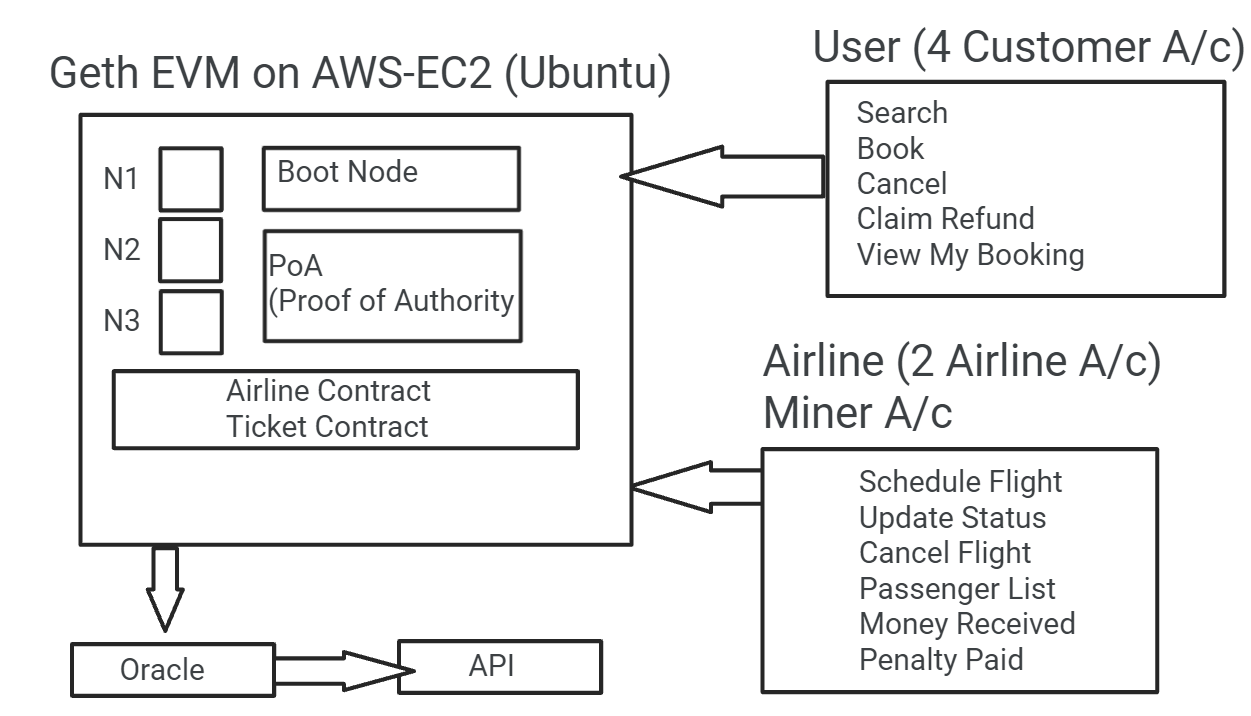
[Testing Advanced Feature 54](#_Toc88861180)

[Challenges Faced 61](#_Toc88861181)

# Assumptions

1. Eagle Airline operates only in single class – Economy
2. For demo purposes, each customer can book any number of seats in a flight.
3. For demo purposes, we will limit the number of seats in a flight
4. We are booking only domestic flights, example route DEL → PNQ → COK → HYD → DEL
5. The price range is currently for domestic flights only. International flights will have higher price range difference.
6. Flight doesnt take any cargo at this point (out of scope)
7. Ticket fixed base price is input as Wei
8. Seat Number will be allocated by Airline once customer pays the fare

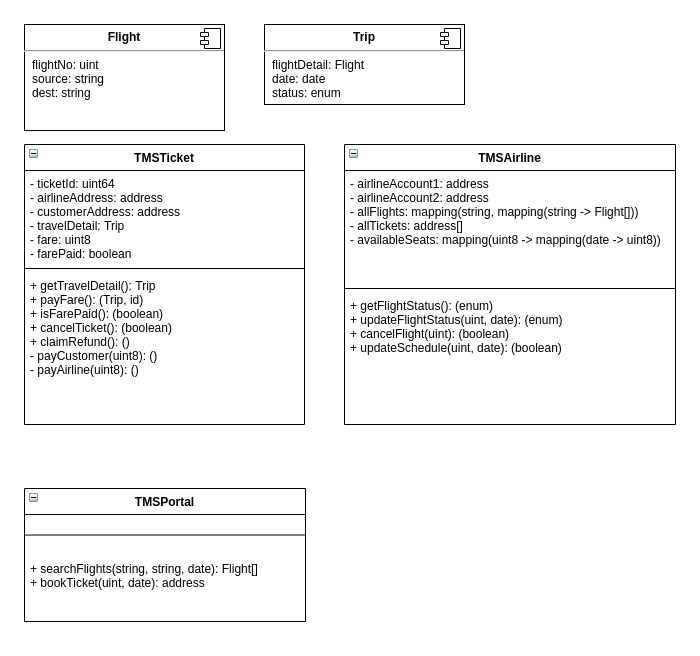
# Architecture



The architecture consists of 3 Geth EVM on AWS cloud. All EVM’s are of type “t2.micro” (1 vCPUs, 2.5 GHz, 1 GiB memory). The account distribution is as below.

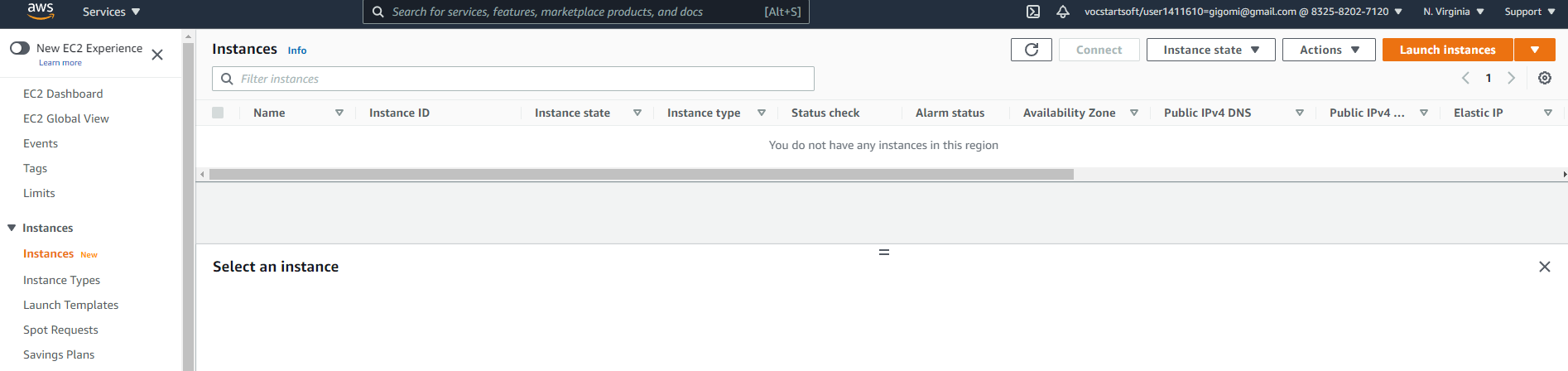
1. TMSGethEVM01
   1. Boot Node - port 30301
   2. Node 1
      1. Airline Account 1 - **Sealer** - port 30303
2. TMSGethEVM02
   1. Node 2
      1. Airline Account 2 - **Sealer** - port 30305
3. TMSGethEVM03
   1. Node 3
      1. Customer Account 1 - port 30401
4. Metamask
   1. Customer Accounts – 4/5/6

# Functions

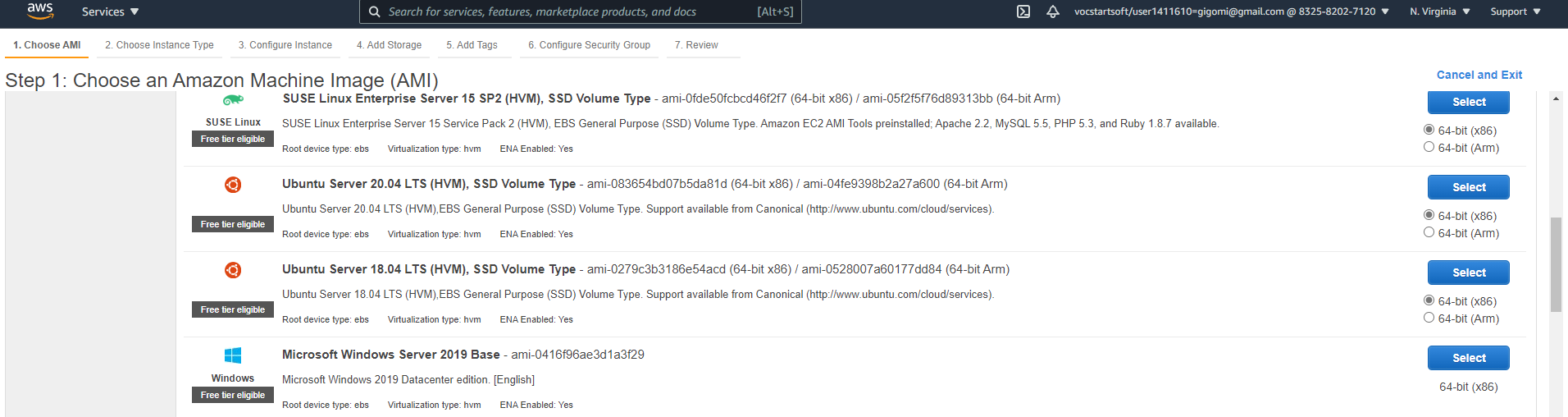


# Infrastructure Setup (Geth EVM on AWS)

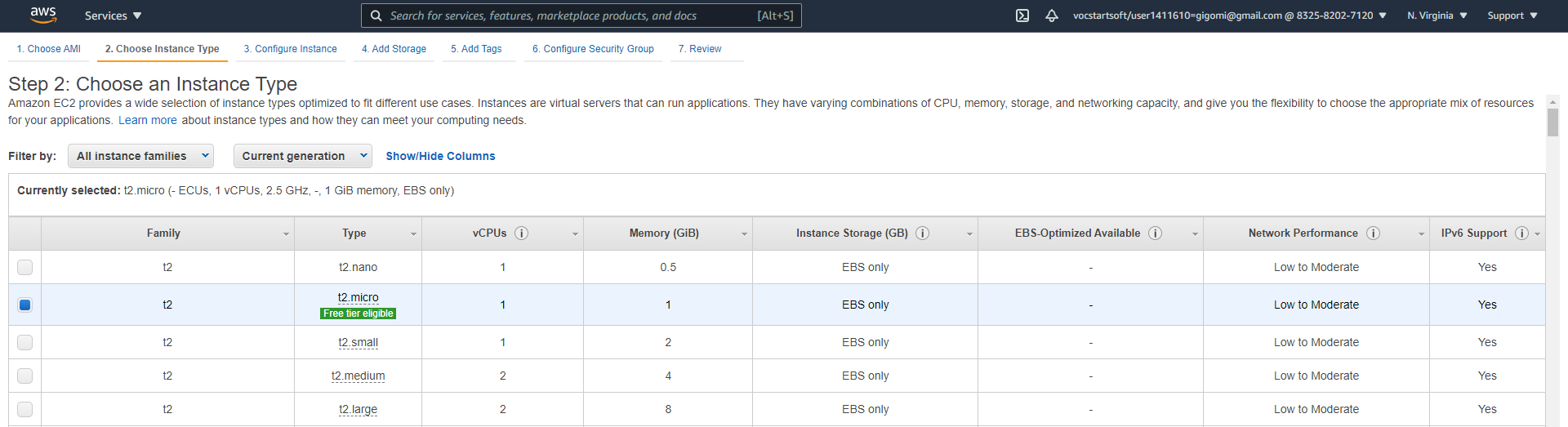
1. In AWS console, select EC2 instances & click on Launch instances



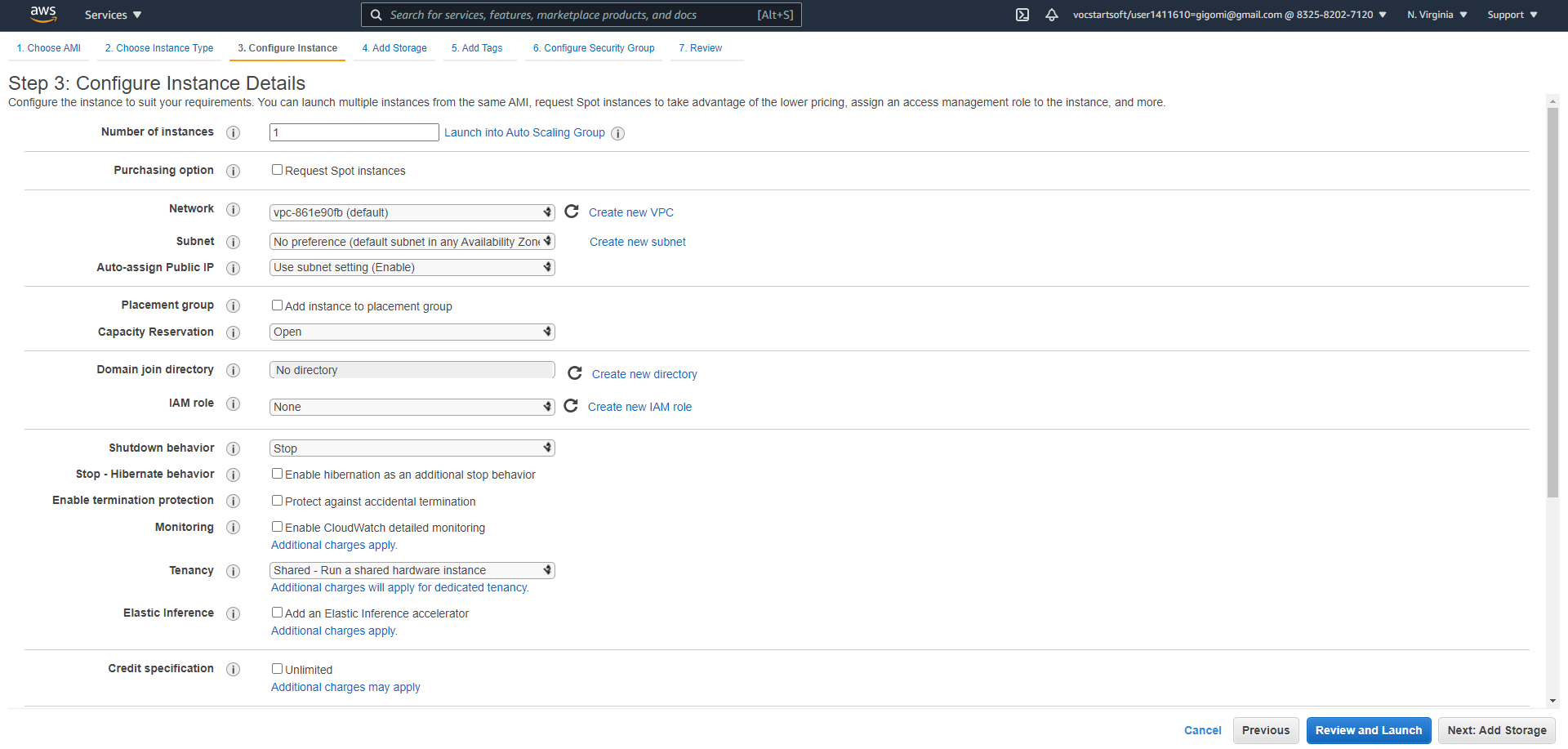
1. Select latest version of Ubuntu (20.04)



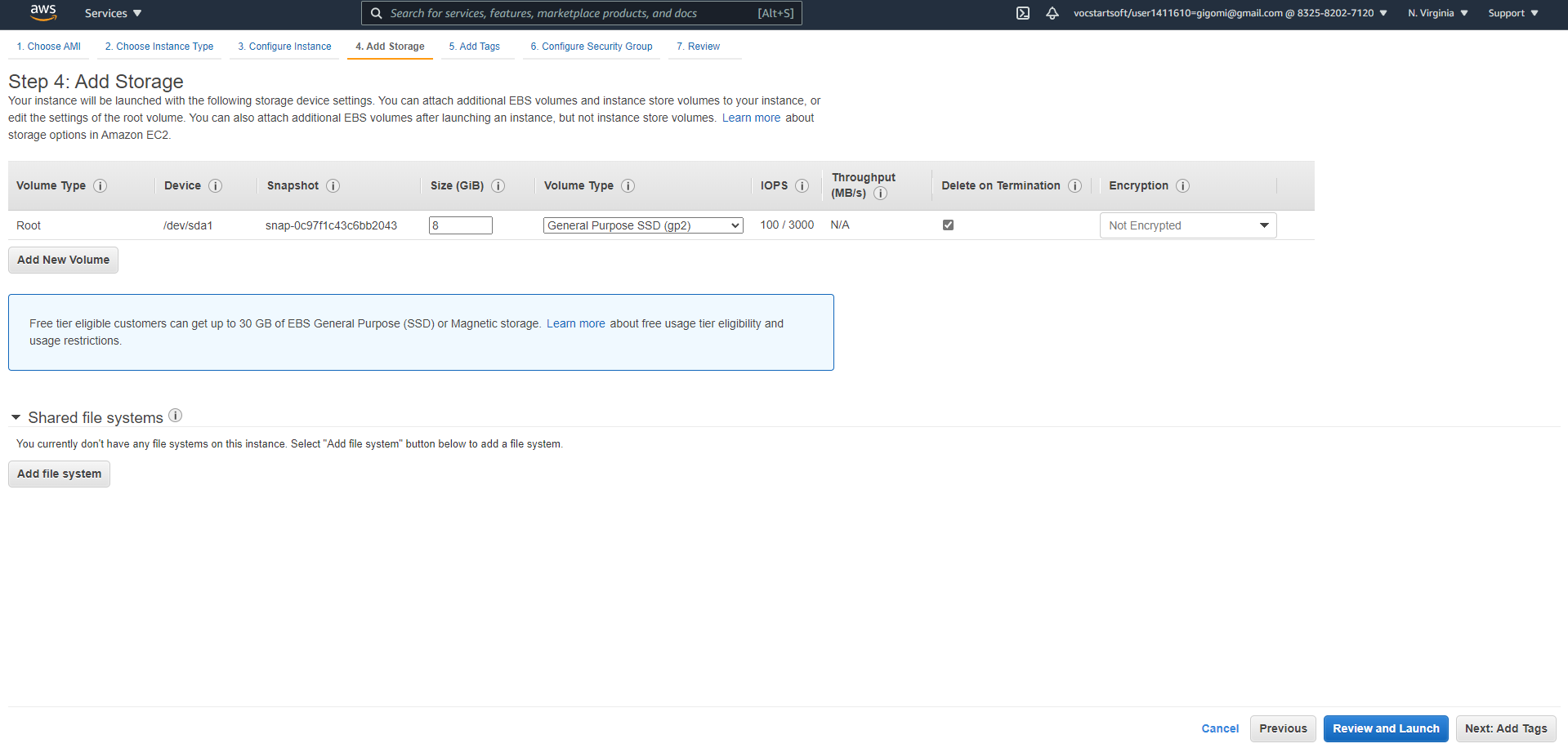
1. Keep default instance & proceed to configure Instance details



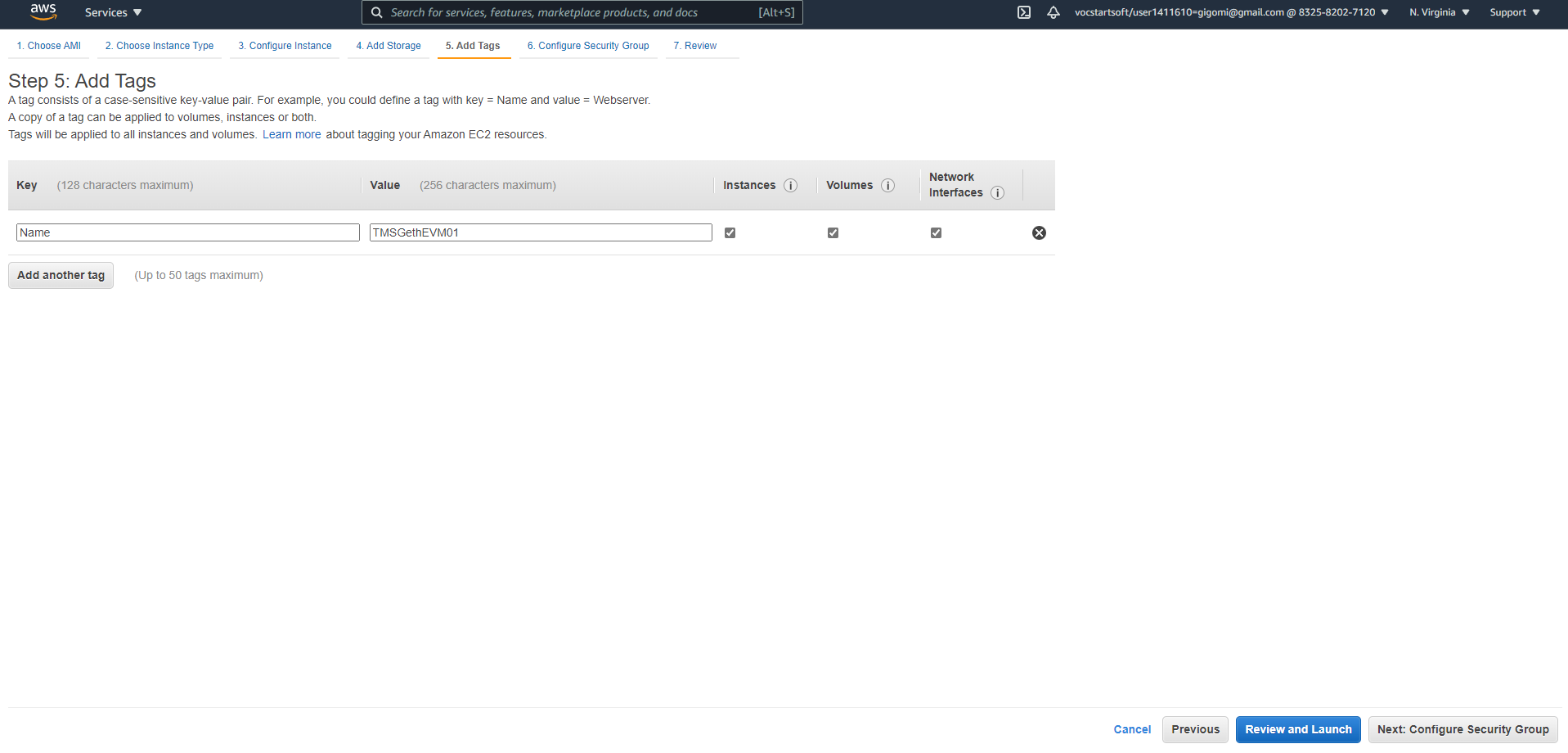
1. Keep default values and click on add storage



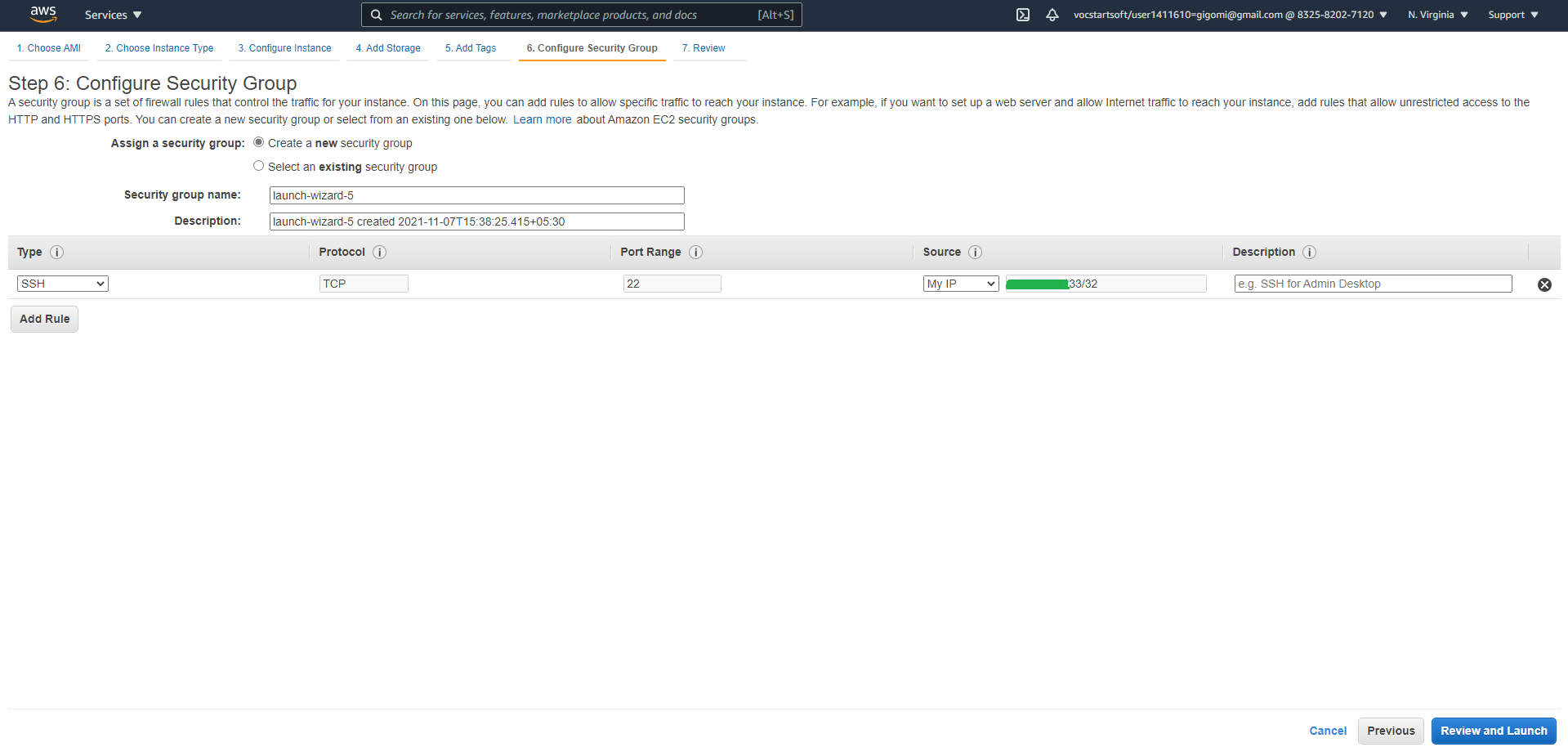
1. Keep default values in storage and proceed to add tags



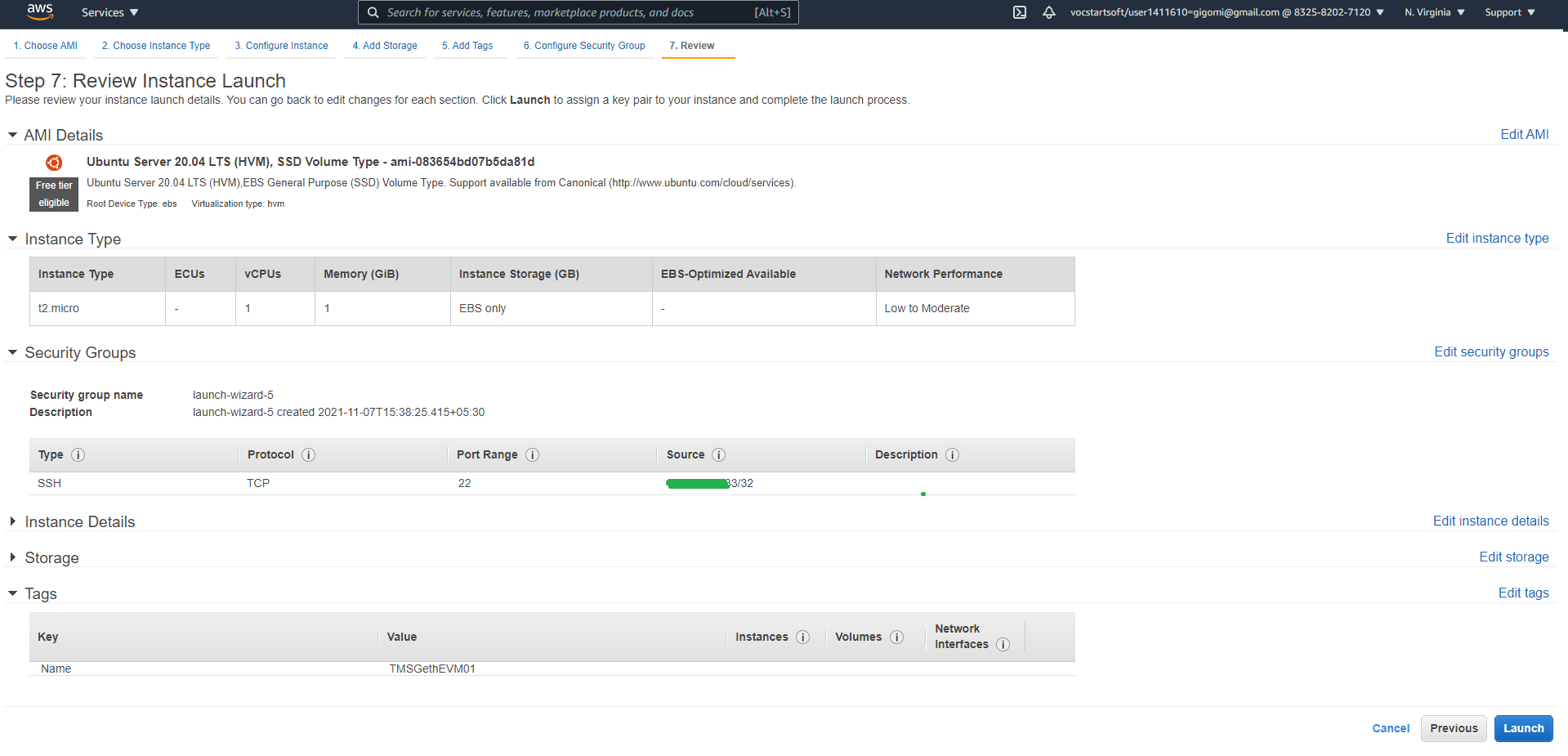
1. Add tag & go to security group



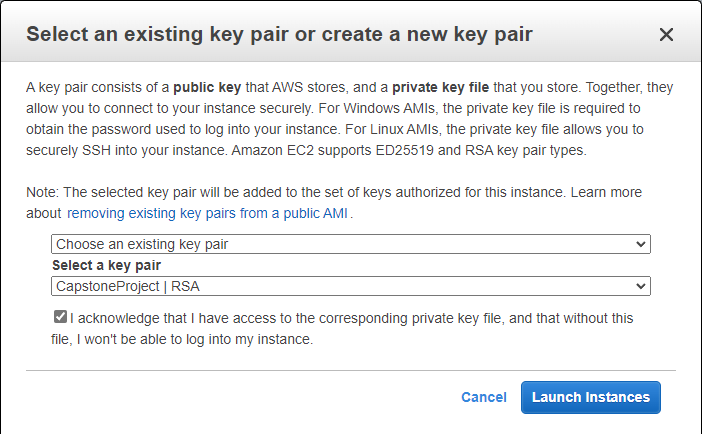
1. Change from custom IP to My IP. Proceed to review and launch



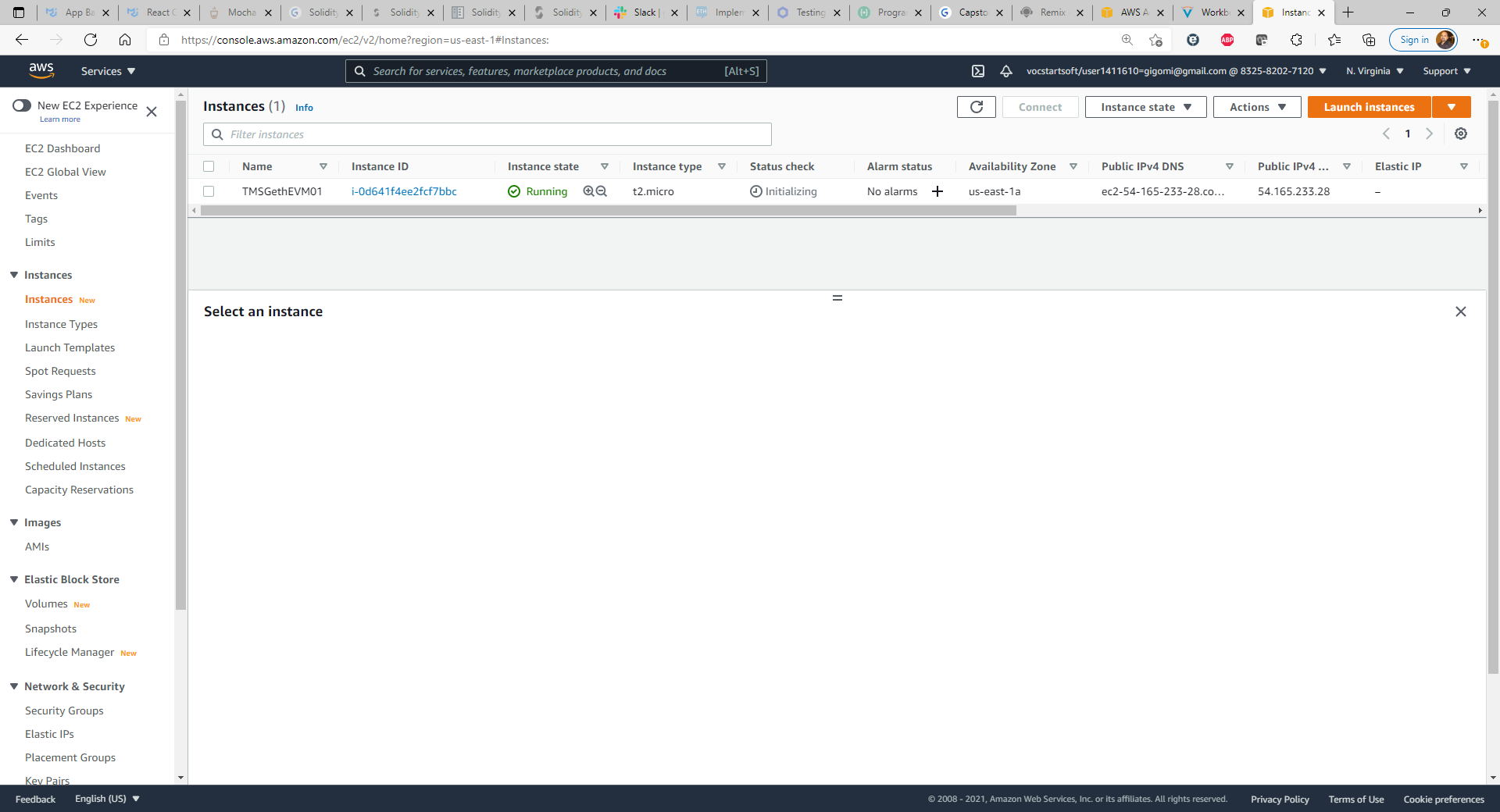
1. Proceed to Launch



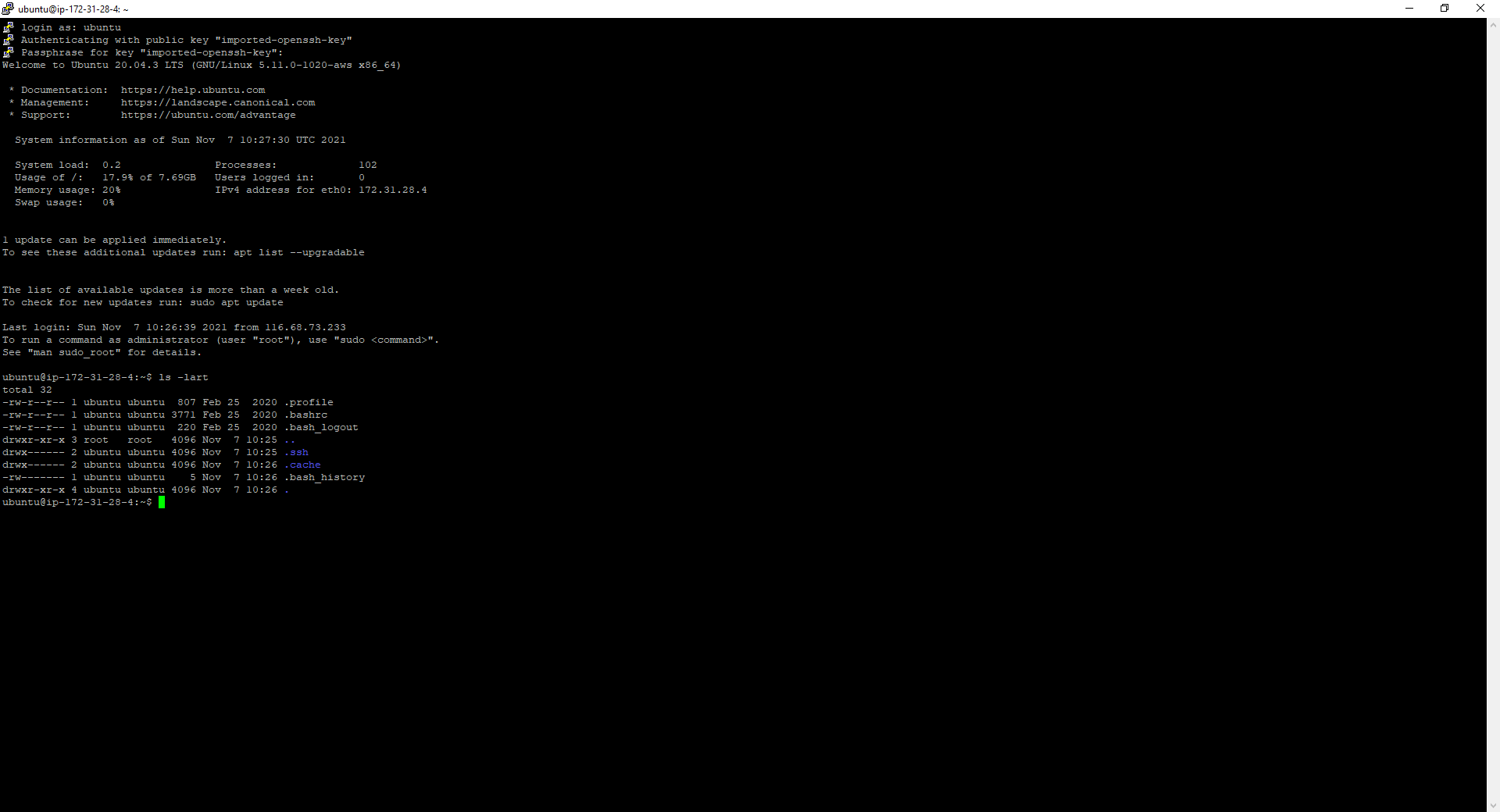
1. Either select existing key pair or go for new one. Proceed to launch instances.



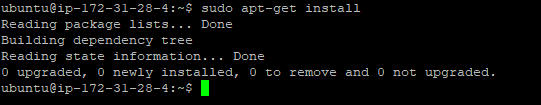
1. Confirm instances are running



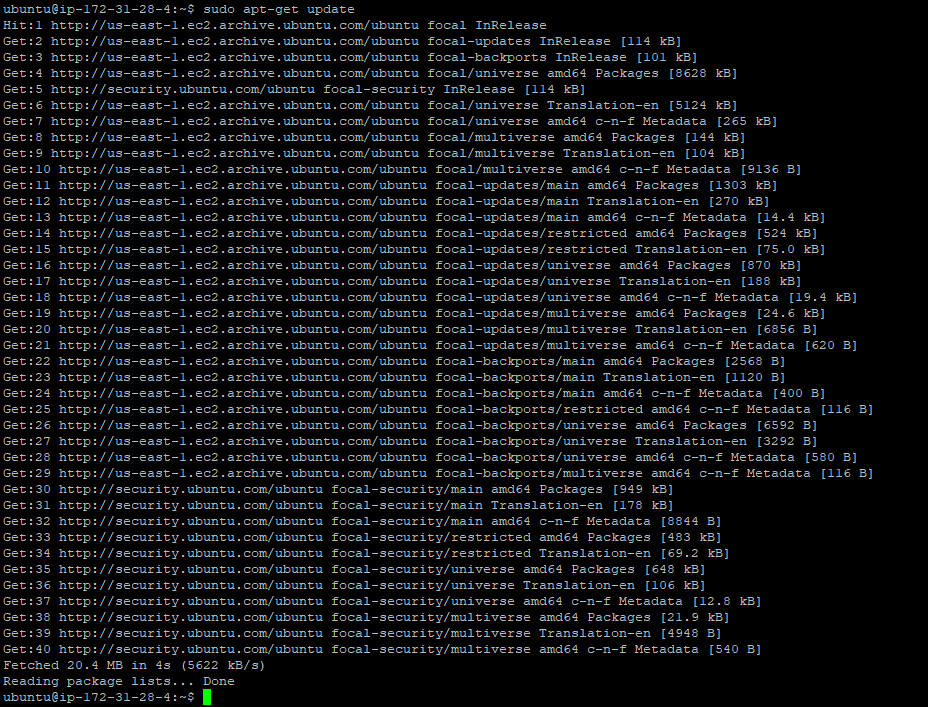
1. Login to instance using putty. Username is “ubuntu” and give your key password.



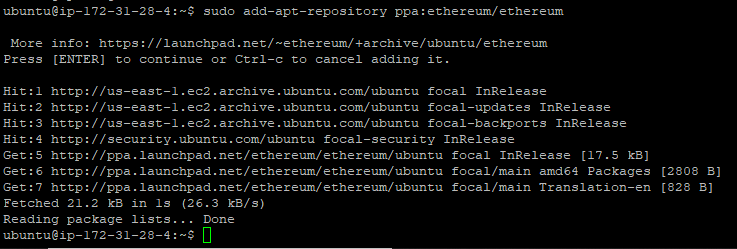
1. Run “**sudo apt-get install**” to install APT (Advanced Package Tools)



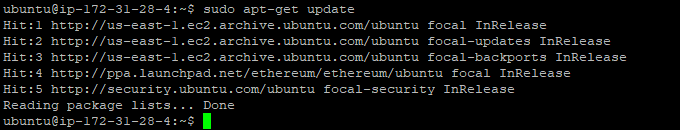
1. Run “**sudo apt-get update**” for updating the packages



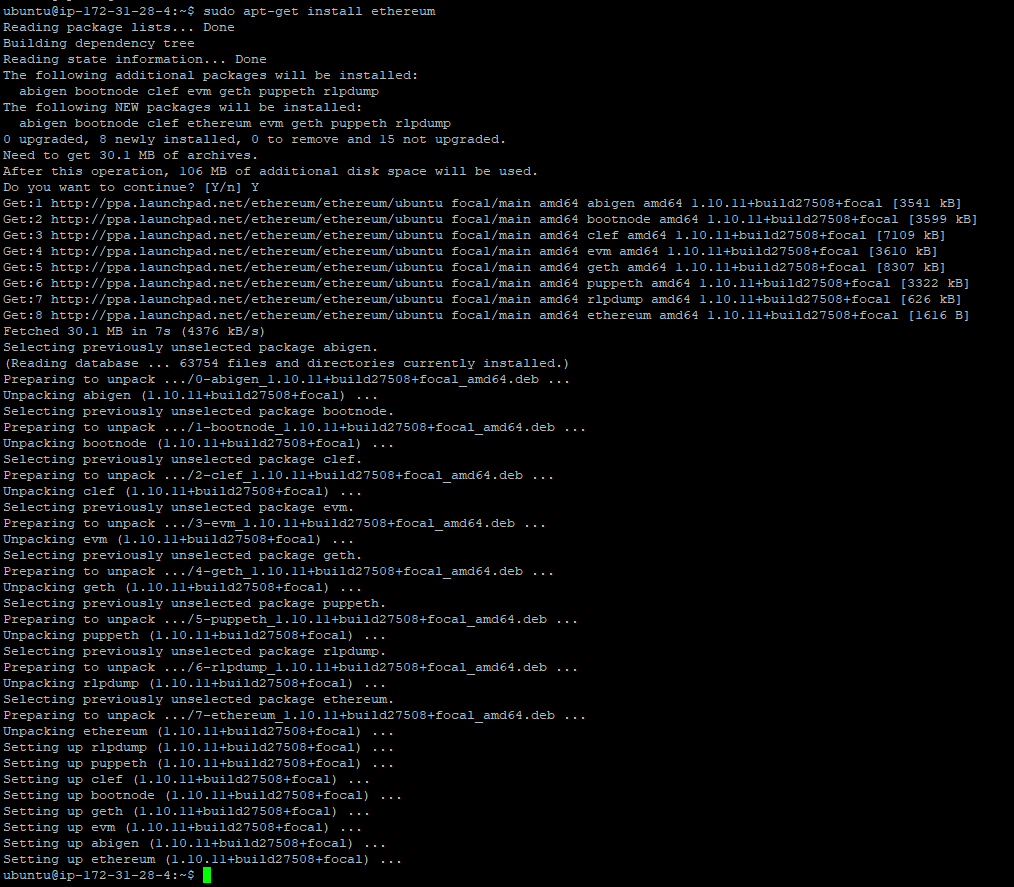
1. Run “**sudo add-apt-repository ppa:ethereum/ethereum**” to install Ethereum repository



1. Run “**sudo apt-get update**” to update packages, if any



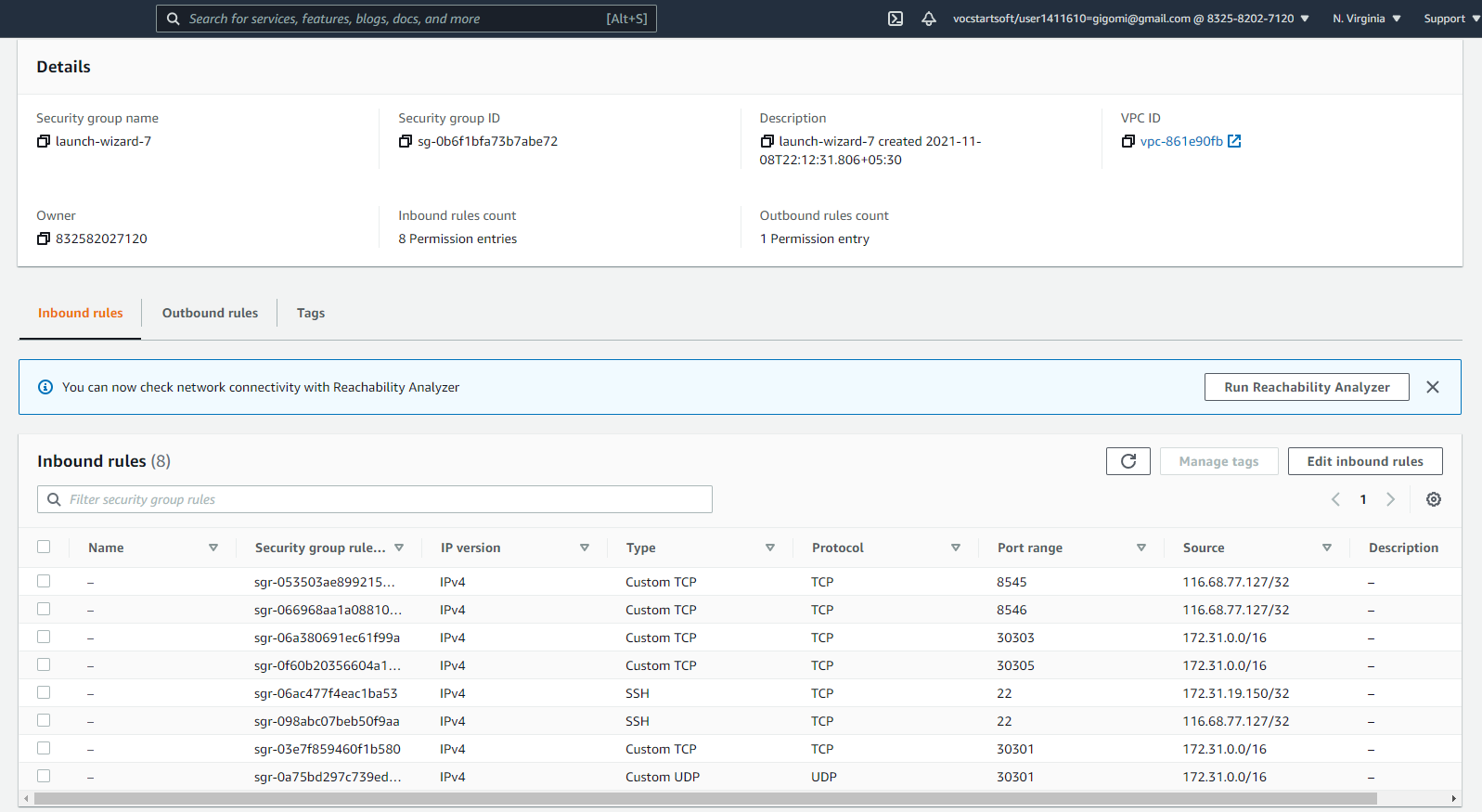
1. Run “**sudo apt-get install ethereum**” to install latest version of “Go Ethereum” (Geth)



1. This completes setup of first Geth EVM node (TMSGethEVM01). Repeat same for remaining 2 Geth EVM nodes (TMSGethEVM02, TMSGethEVM03)

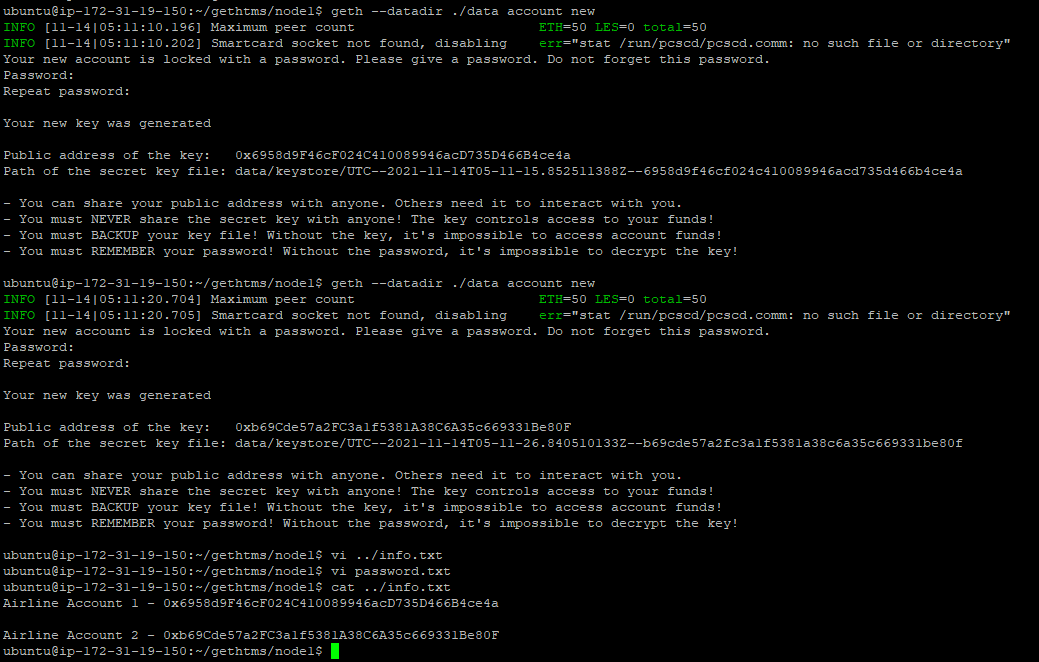
# Configure security group in AWS

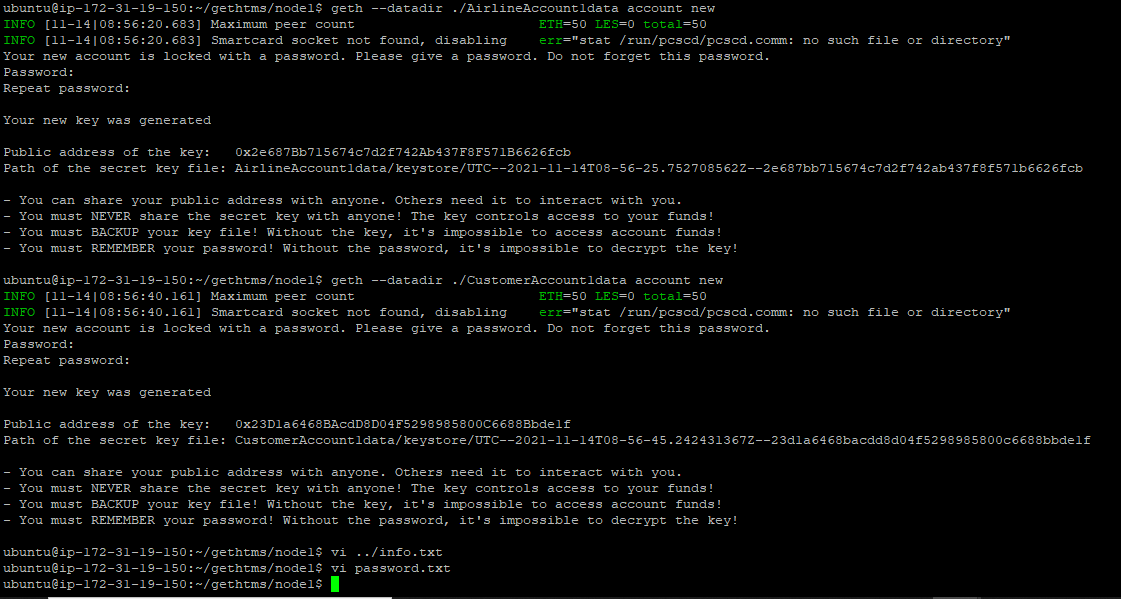
Following are the security groups defined in AWS. They are required to open connectivity for various purposes. Port 22 is opened from local IP for access to environment. UDP to boot node port is mandatory for nodes running on different EVM’s to connect to it. Rest is opened for communication to various ports for testing, like 8545

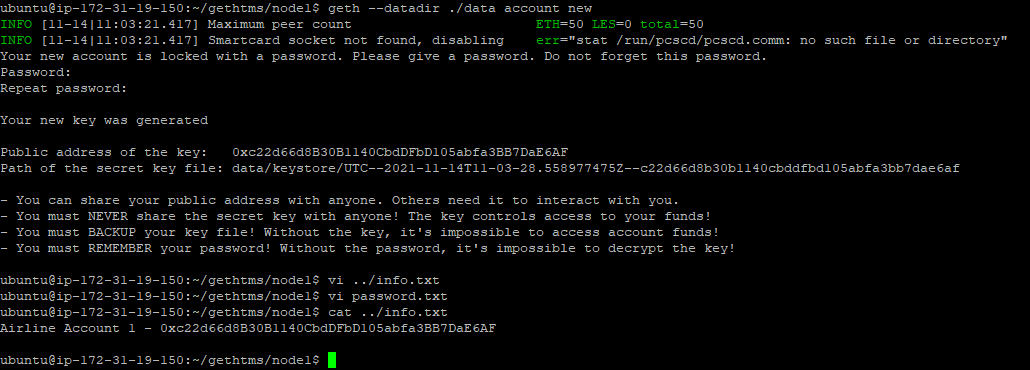


# Configure nodes and accounts

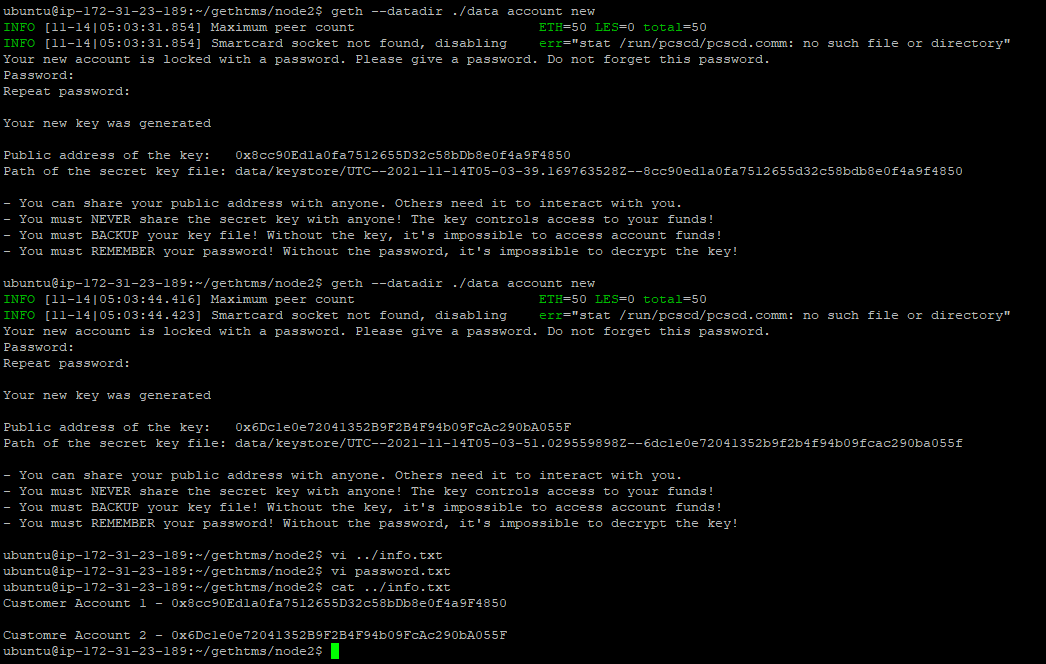
1. The design is as in the Architecture section.
2. Create directory gethtms and subdirectory bnode, node1 on TMSGethEVM01
3. Run command to create new account in node 1 (*geth --datadir ./data account new*). We will designate it as Airline Account 1. Save its address in info.txt file under ~gethtms for easy reference. Save password in password.txt file.

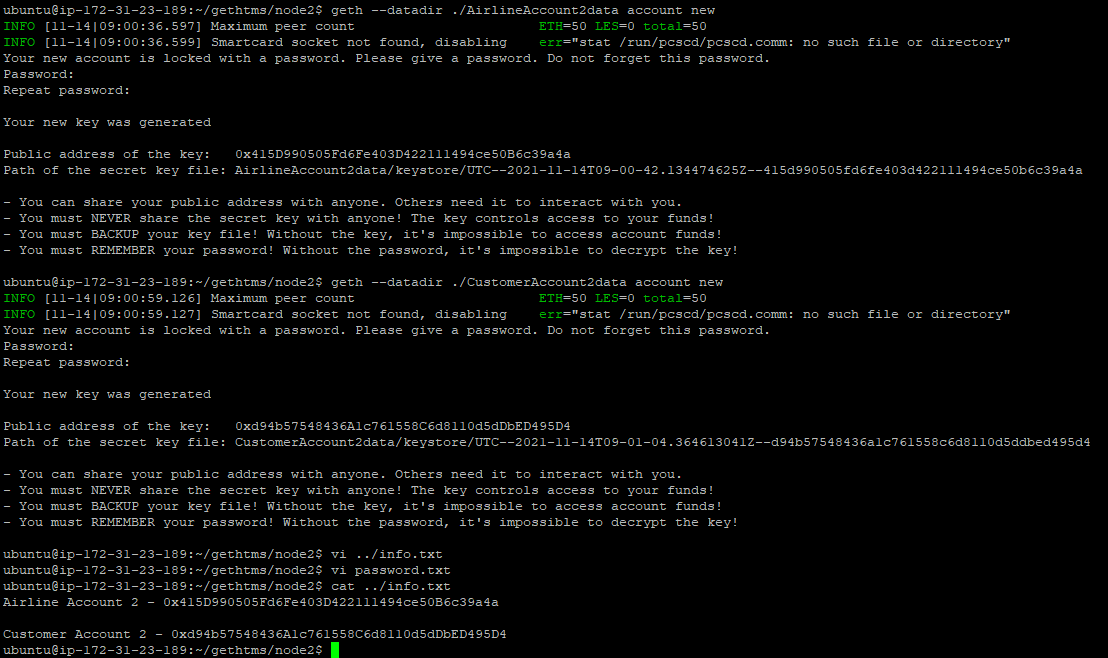


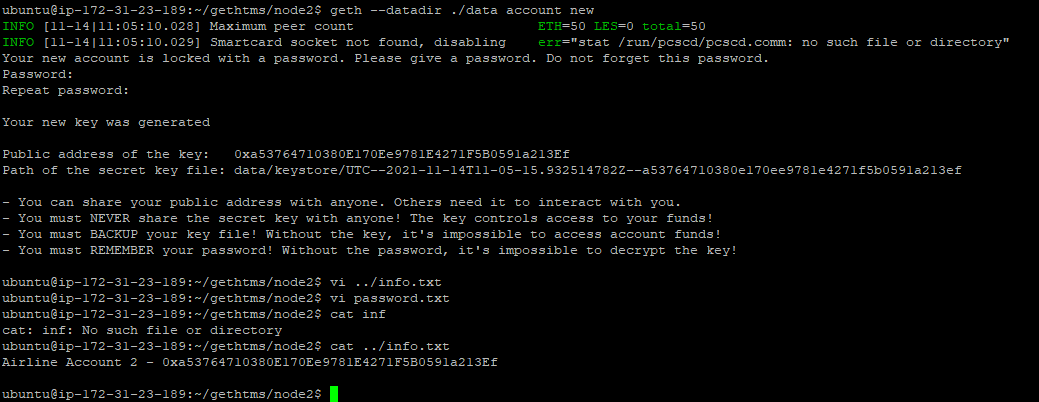




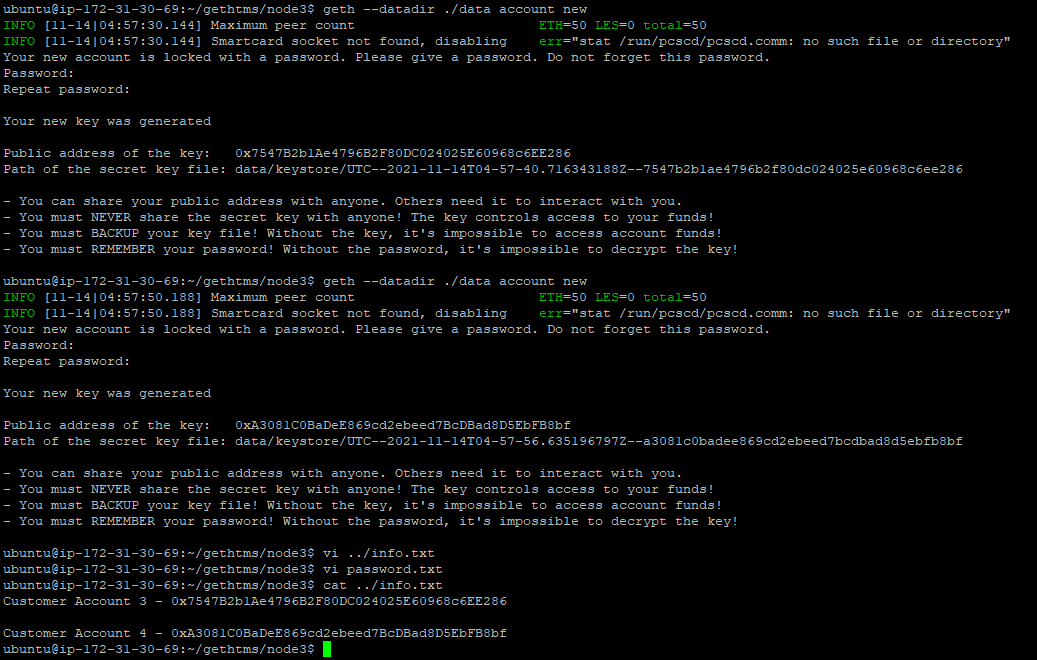
1. Create directory gethtms and subdirectory node2 on TMSGethEVM02
2. Run command to create new account in node 2 (*geth --datadir ./data account new*). We will designate it as Customer Account 2. Save its address in info.txt file under ~gethtms for easy reference. Save password in password.txt file.

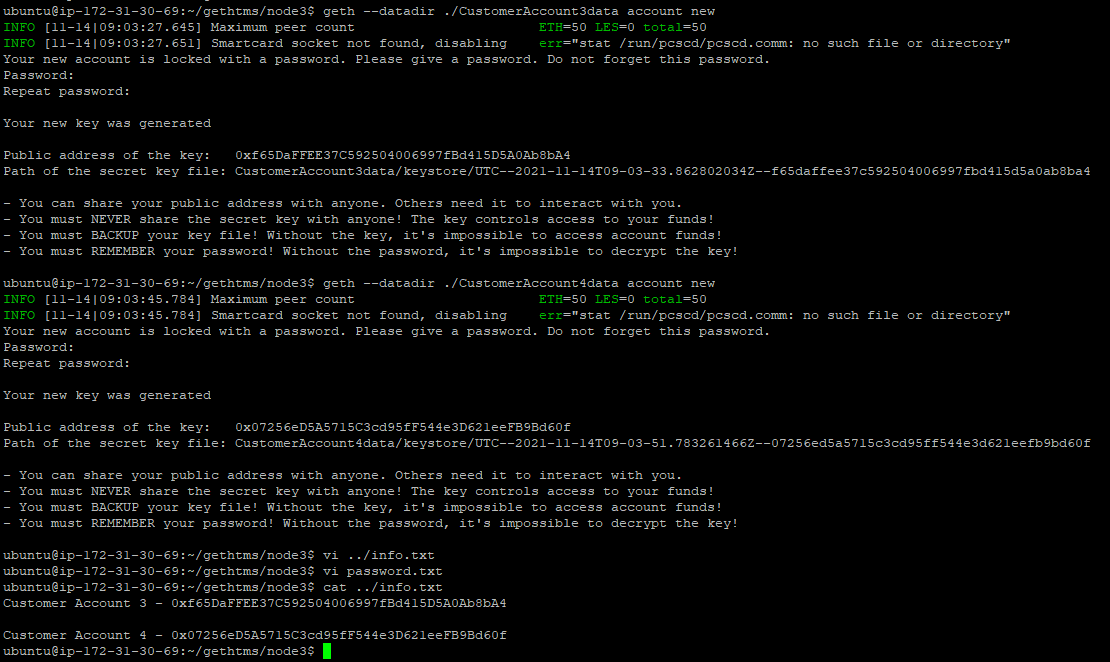


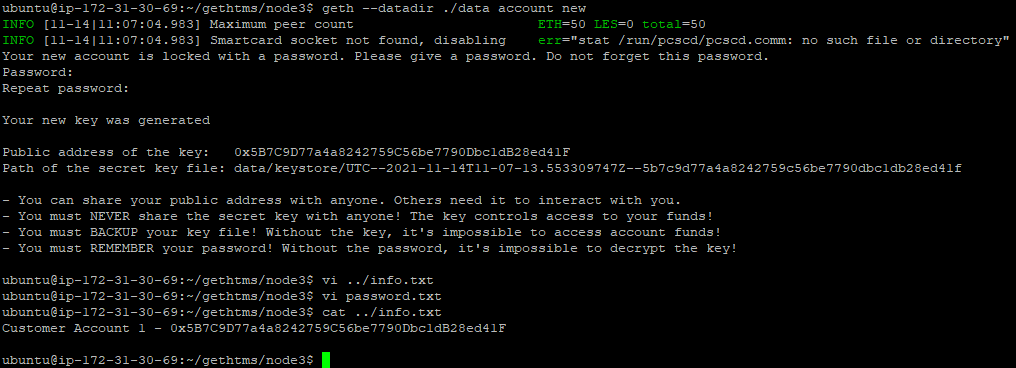




1. Create directory gethtms and subdirectory node3 on TMSGethEVM03
2. Run command to create new account in node 3 (*geth --datadir ./data account new*). We will designate them as Customer Account 1. Save their address in info.txt file under ~gethtms for easy reference. Save password in password.txt file.



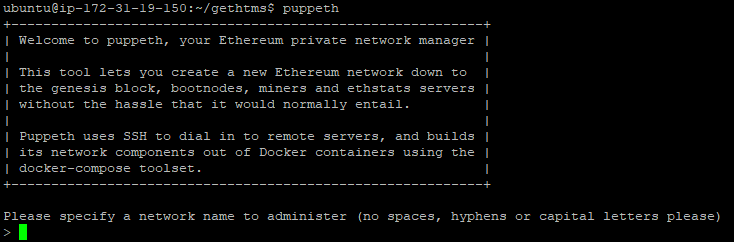




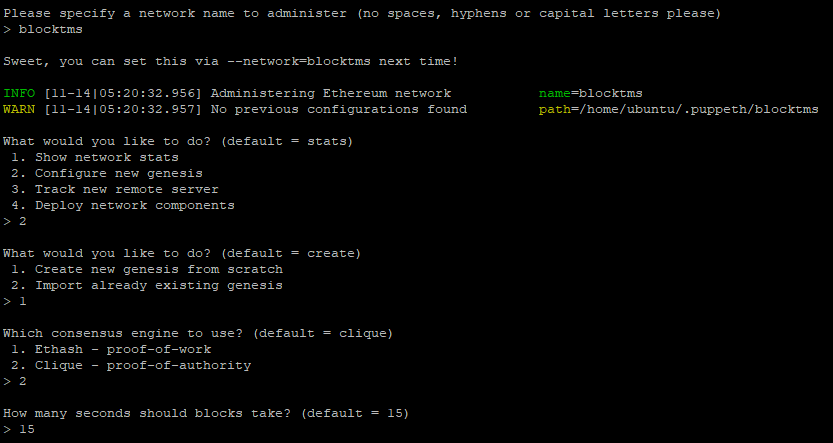
# Configure Genesis File

We will create genesis file to define the consensus protocol as “Proof of Authority”, define 2 Airline Accounts as sealers/miners and all Customer Accounts as normal ones.

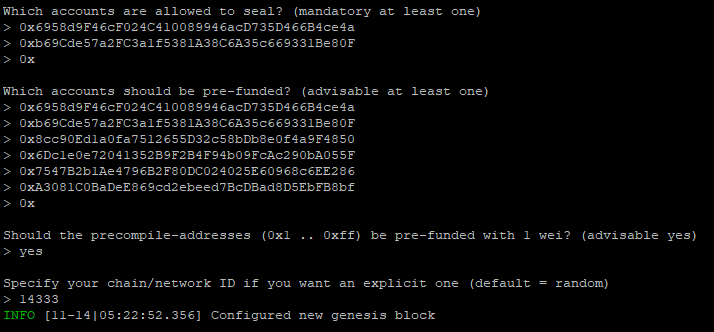
1. From gethtms folder run “puppeth” tool



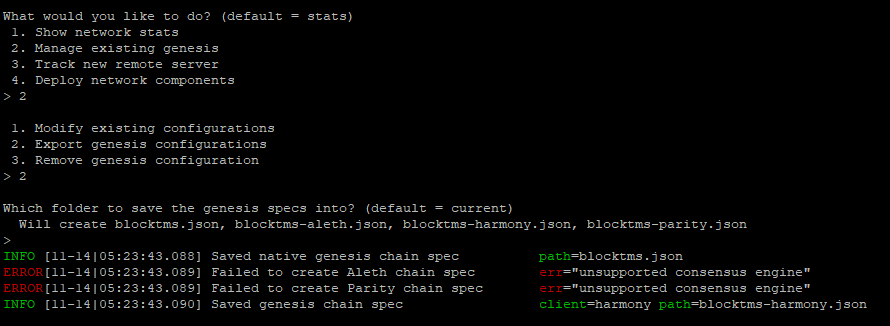
1. Provide network name, then options to configure new genesis from scratch. We will also select the protocol POA here and go for default time for block creation interval



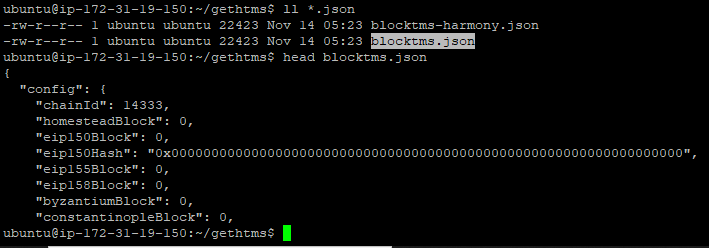
1. We will give both airline accounts as sealers/miners. We will prefund all the accounts with 1 wei each. Also provide network ID.



1. Proceed to export the genesis configuration to json files

` 

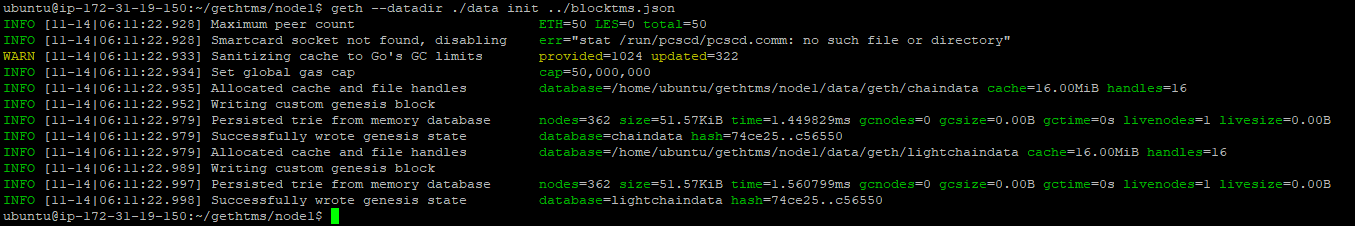
1. We are done with the genesis configuration and can see the json files created

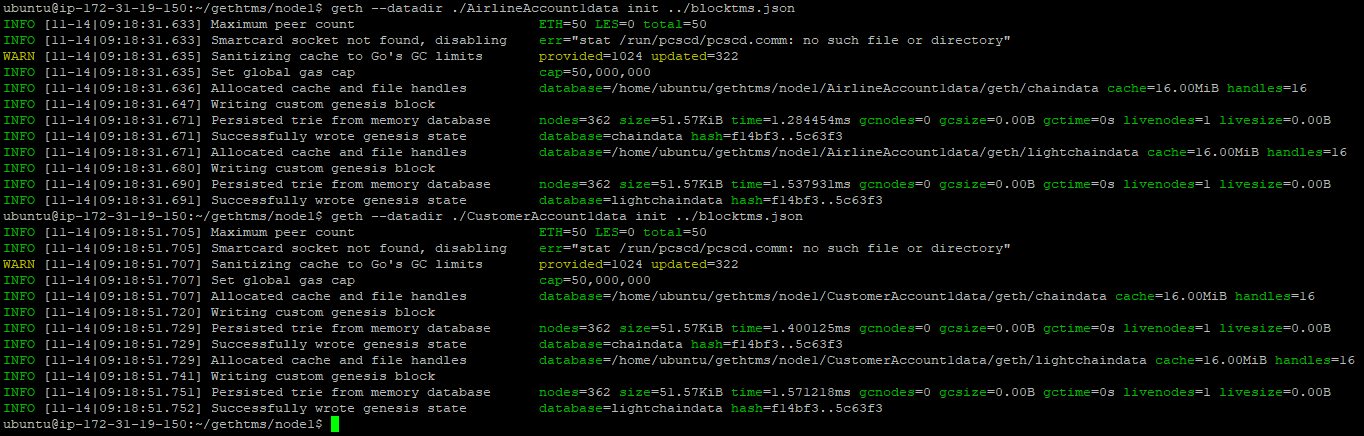


1. Copy blocktms.json to gethems directory on other 2 EVM’s

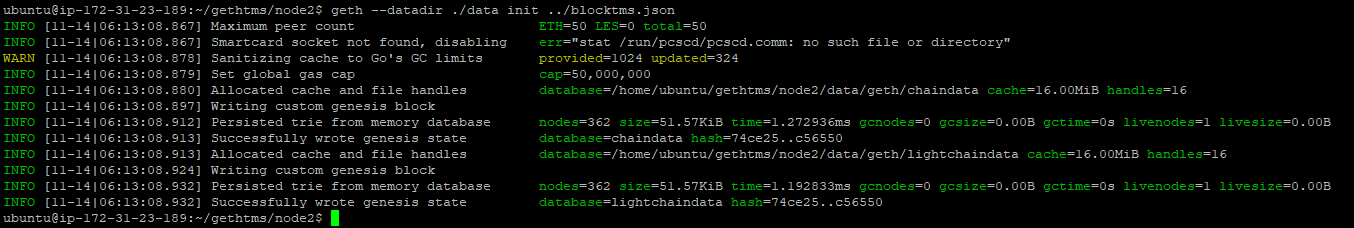
# Initialize Nodes

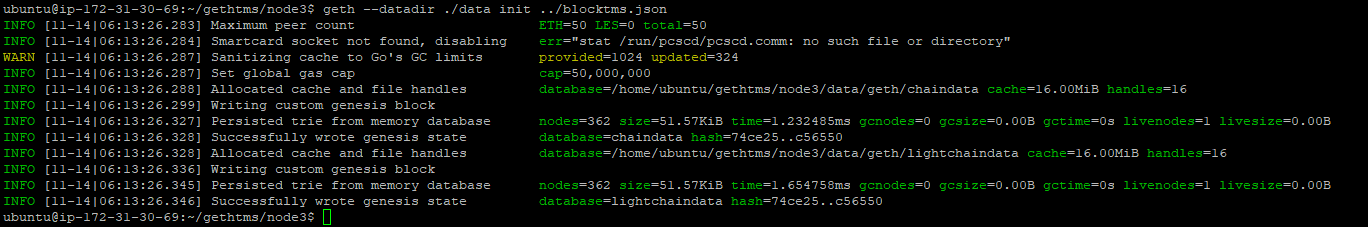
1. Initialize Node 1 (*geth --datadir ./data init ../blocktms.json*) on TMSGethEVM01

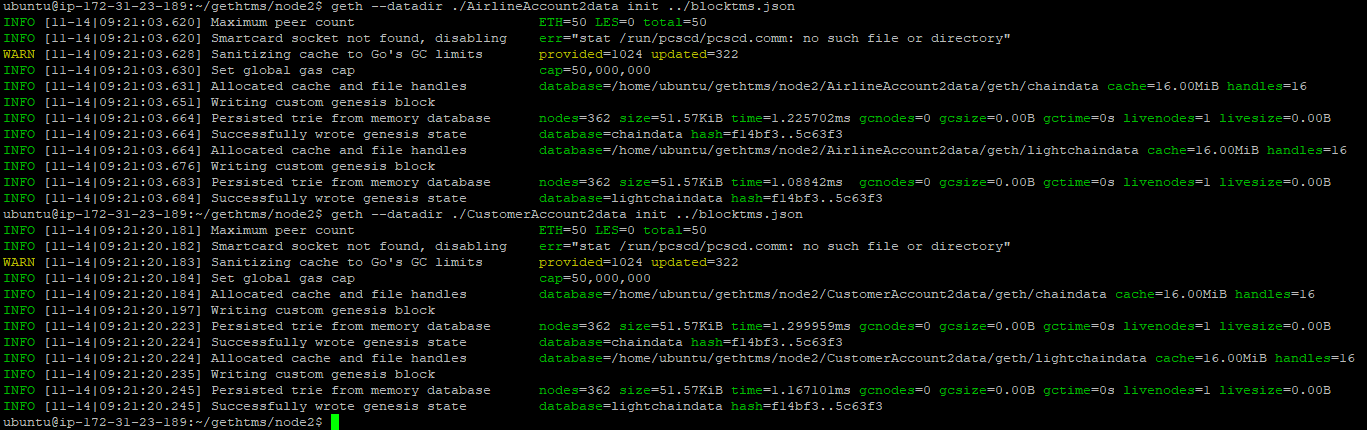


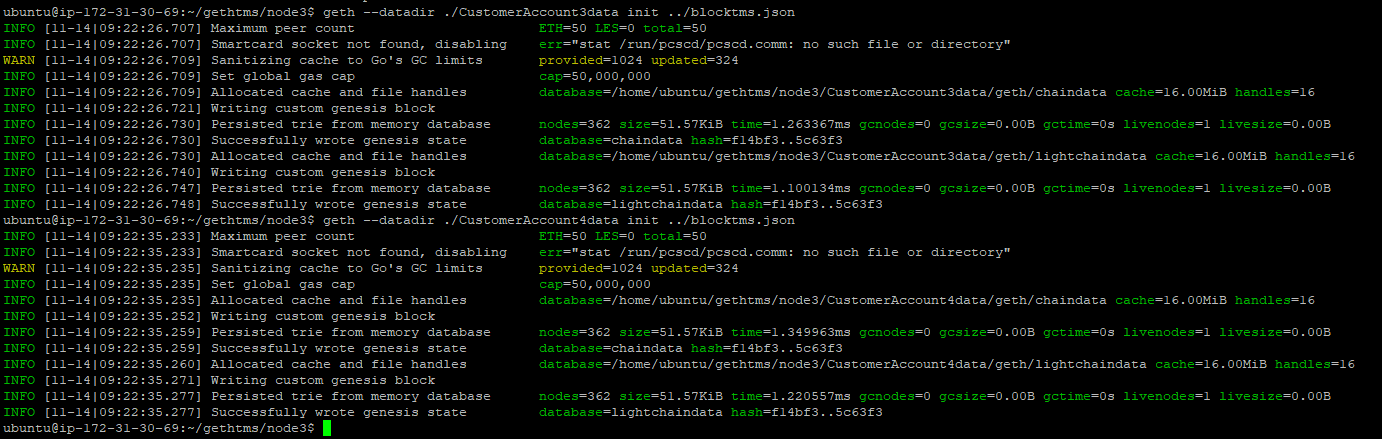


1. Similarly, for node 2 & node 3 (*geth --datadir ./data init ../blocktms.json*) on TMSGethEVM02 & TMSGethEVM03





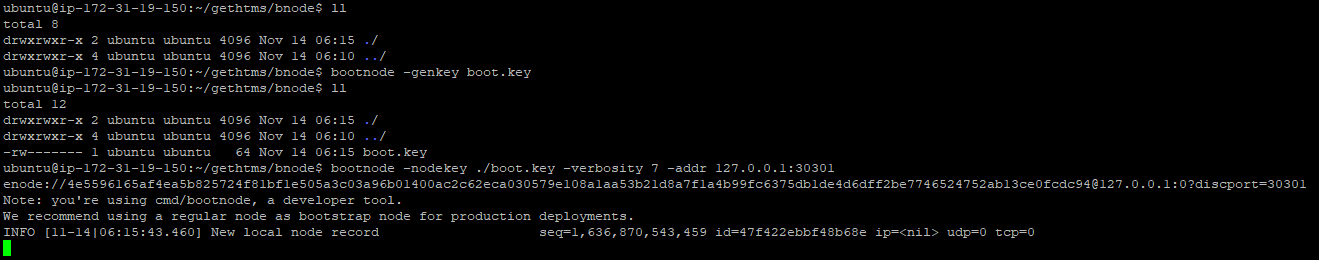




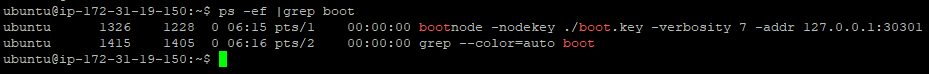
1. Initialization is complete

# Create and Start Boot Node

1. Create and start boot node on port 30301

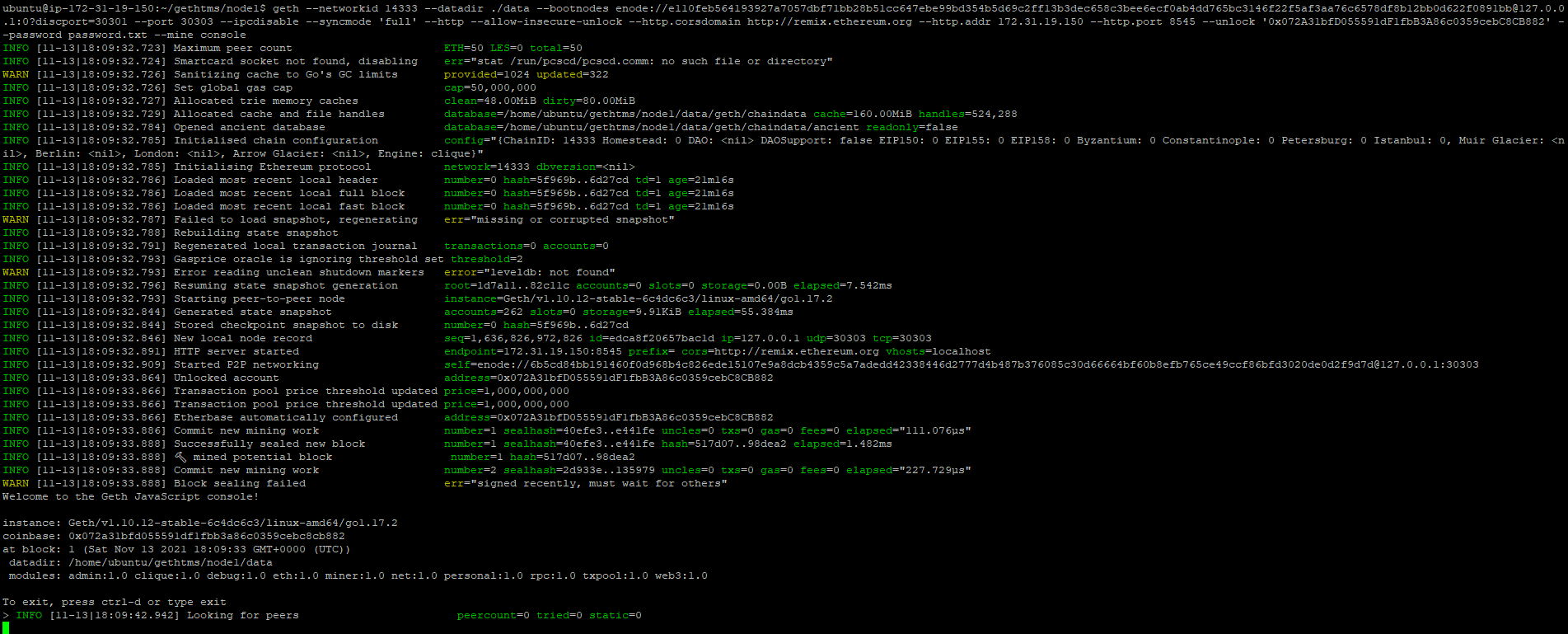


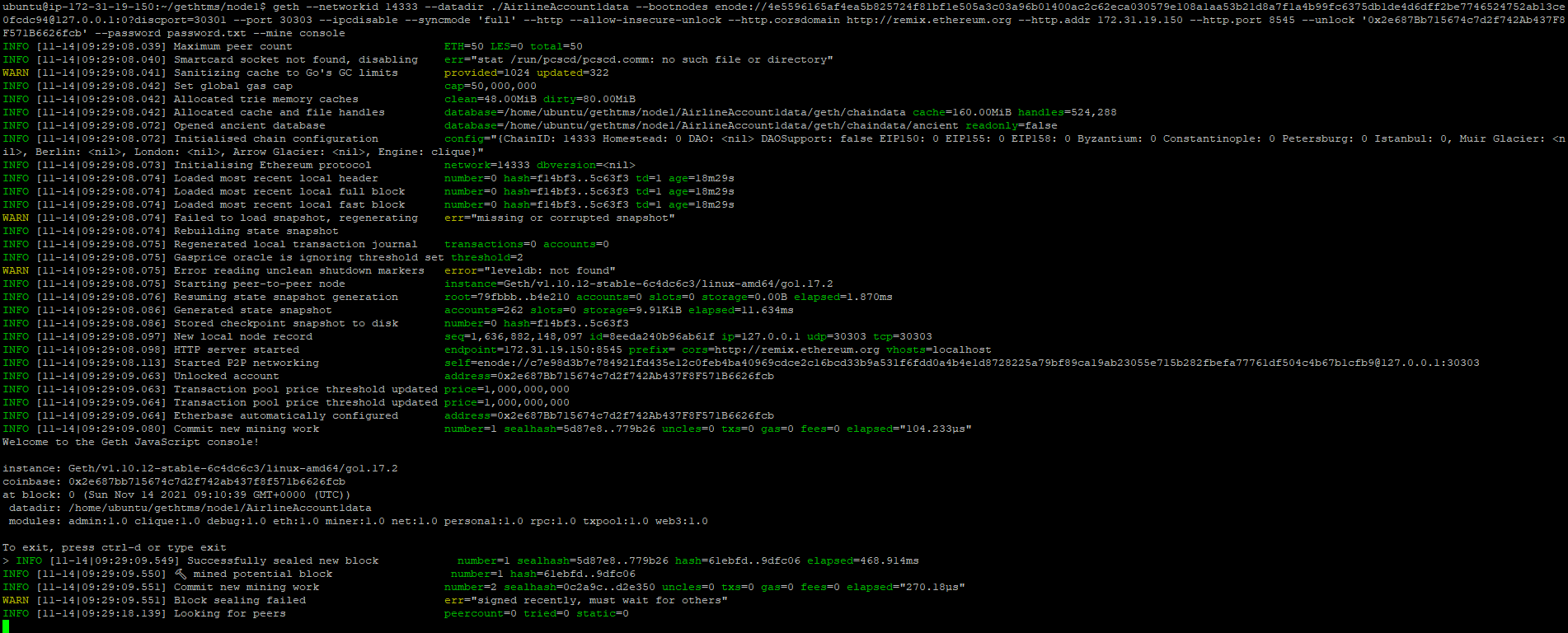
1. Command to generate key 🡪 “*bootnode -genkey boot.key*”
2. Startup Command 🡪 “*bootnode -nodekey ./boot.key -verbosity 7 -addr 172.31.19.150:30301*”
3. Save the enode info to info.txt file
4. Verify boot node is running



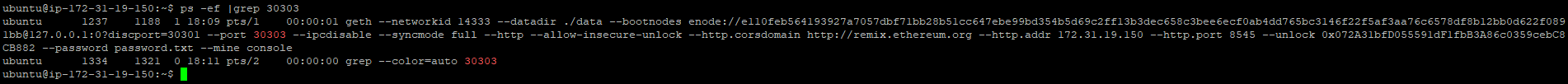
# Start Our nodes

1. Start Node1 AirlineAccount 1 as miner on port 30303

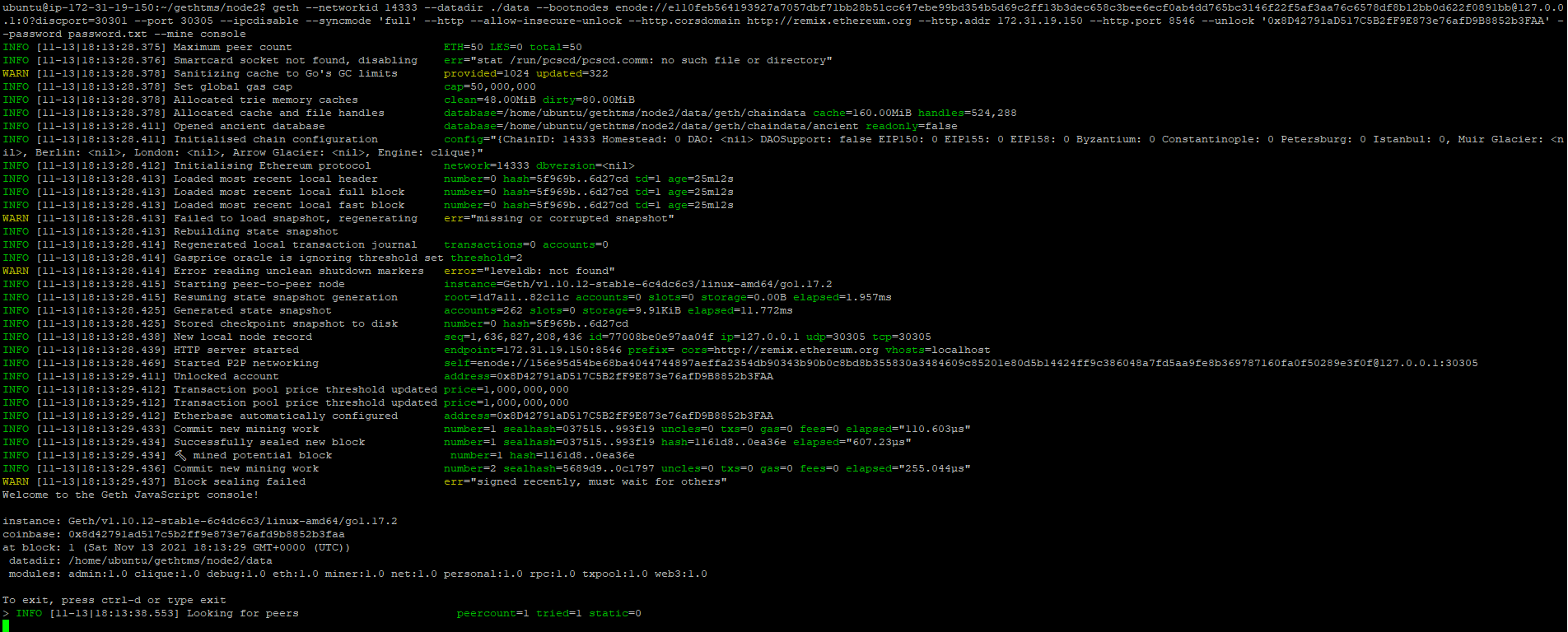




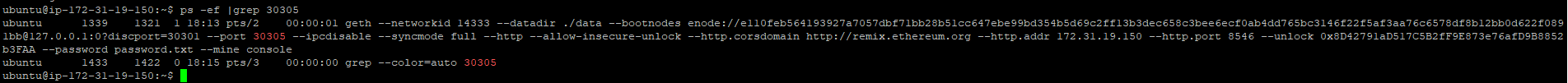
1. Startup Command 🡪 *geth --networkid 14333 --datadir ./data --bootnodes enode://0df5d75715588100bd88a8c2f3afa8554edaaede9f247394dd25e77bba965436a425431bb601defbd2955b1c1fb6e3dffc2c4cfaf47f9d9b5077c052b5b88405@172.31.19.150:30301 --port 30303 --ipcdisable --syncmode 'full' --http --allow-insecure-unlock --http.corsdomain "http://remix.ethereum.org, chrome-extension://nkbihfbeogaeaoehlefnkodbefgpgknn" --http.addr 172.31.19.150 --http.port 8545 --unlock '0xc22d66d8b30b1140cbddfbd105abfa3bb7dae6af' --password password.txt --mine console --rpc.txfeecap 0*
2. Verify its up & running



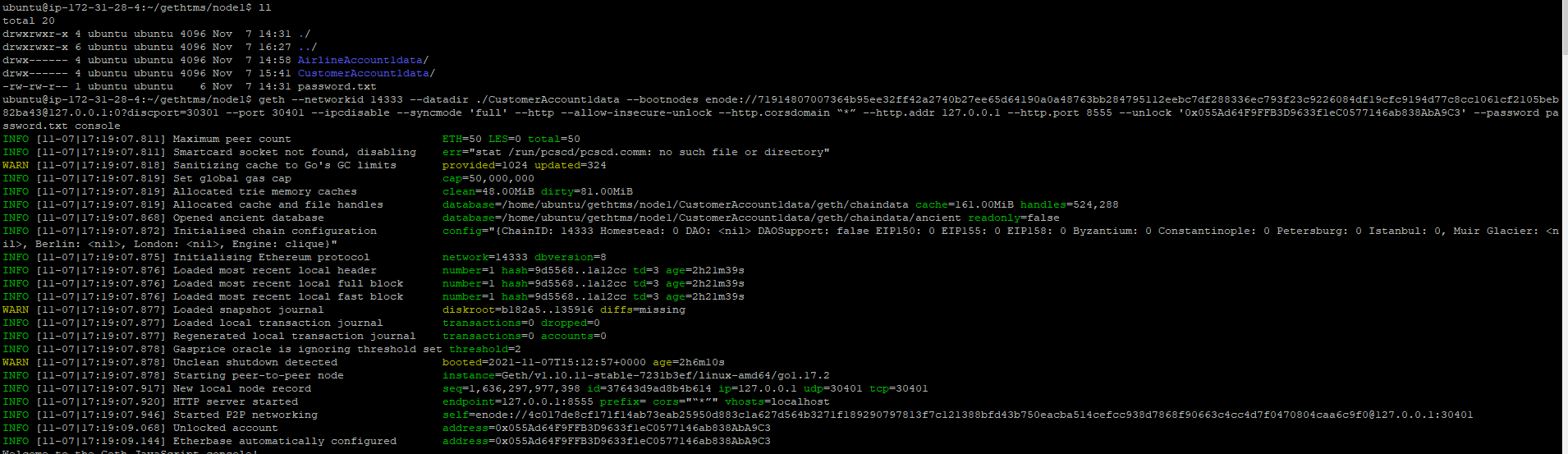
1. Start Node2 Airline Account 2 as miner on port 30305



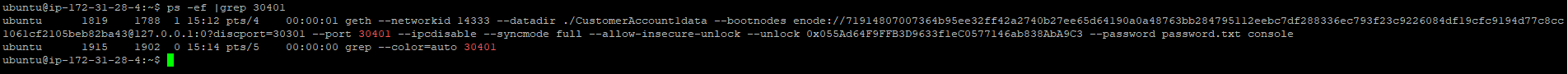
1. Startup Command 🡪 *geth --networkid 14333 --datadir ./data --bootnodes enode://0df5d75715588100bd88a8c2f3afa8554edaaede9f247394dd25e77bba965436a425431bb601defbd2955b1c1fb6e3dffc2c4cfaf47f9d9b5077c052b5b88405@172.31.19.150:30301 --port 30305 --ipcdisable --syncmode 'full' --http --allow-insecure-unlock --http.corsdomain "http://remix.ethereum.org, chrome-extension://nkbihfbeogaeaoehlefnkodbefgpgknn" --http.addr 172.31.23.189 --http.port 8546 --unlock '0xa53764710380E170Ee9781E4271F5B0591a213Ef' --password password.txt --mine console --rpc.txfeecap 0*
2. Verify its up & running



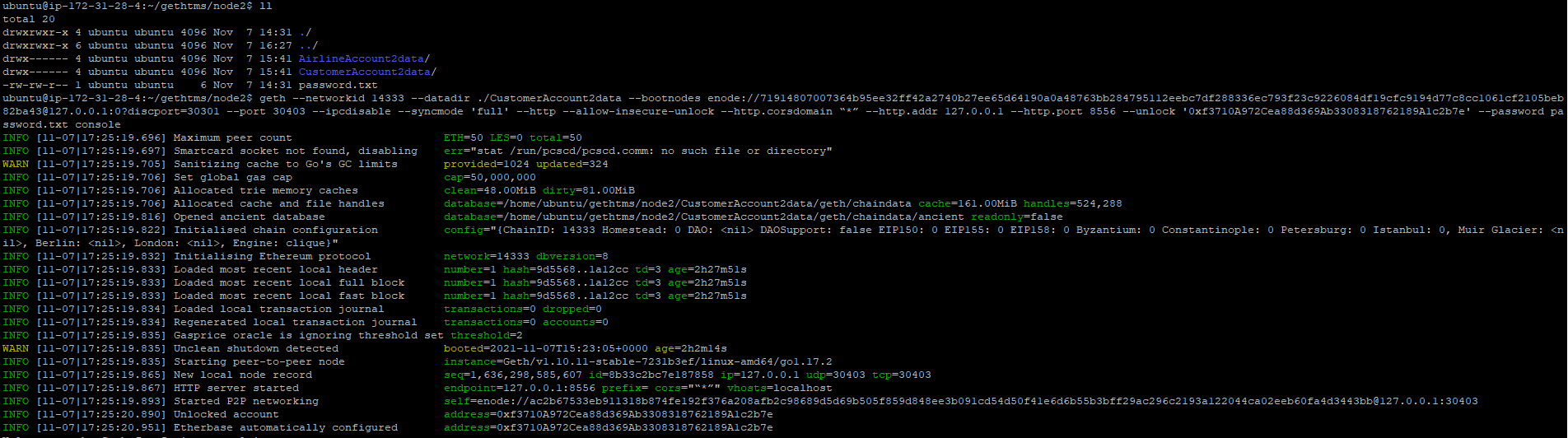
1. Start Node3 Customer Account 1 on port 30401 as non-miner



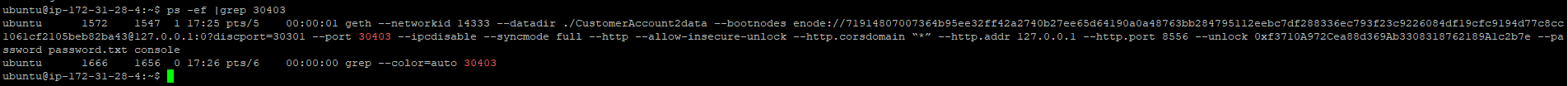
1. Startup Command 🡪 “*geth --networkid 14333 --datadir ./data --bootnodes enode://0df5d75715588100bd88a8c2f3afa8554edaaede9f247394dd25e77bba965436a425431bb601defbd2955b1c1fb6e3dffc2c4cfaf47f9d9b5077c052b5b88405@172.31.19.150:30301 --port 30401 --ipcdisable --syncmode 'full' --http --allow-insecure-unlock --http.corsdomain "http://remix.ethereum.org, chrome-extension://nkbihfbeogaeaoehlefnkodbefgpgknn" --http.addr 172.31.30.69 --http.port 8547 --unlock '0x5B7C9D77a4a8242759C56be7790Dbc1dB28ed41F' --password password.txt console --rpc.txfeecap 0*”
2. Verify its up & running



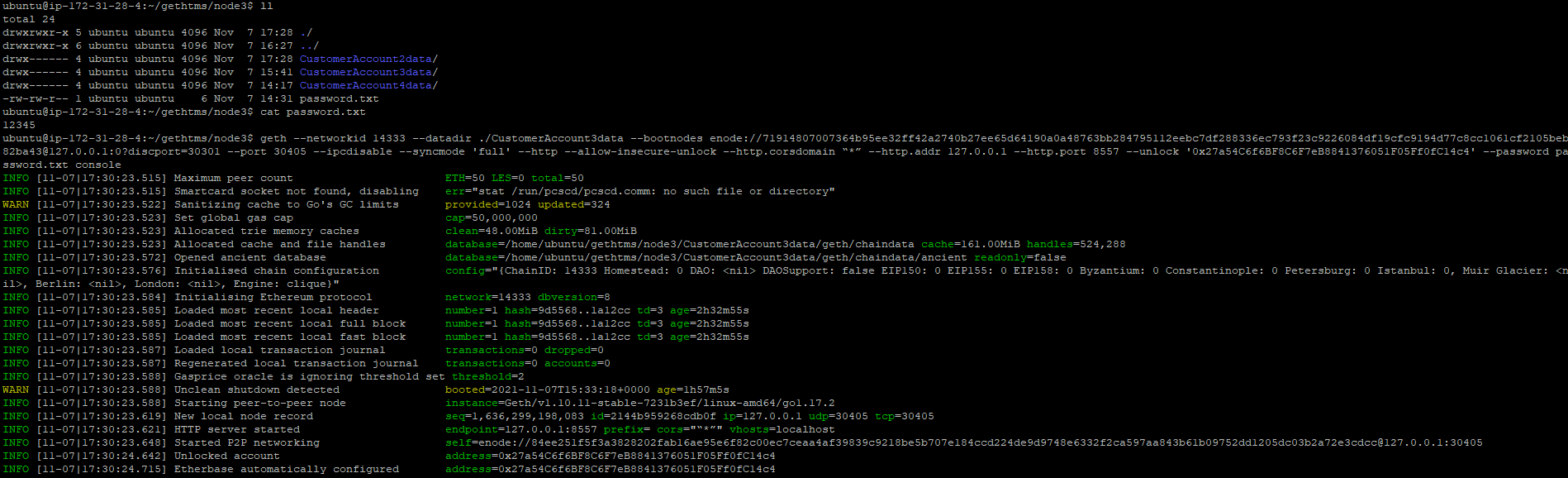
1. Start Customer account 2 on port 30403



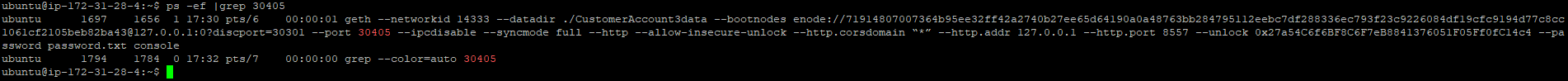
1. Startup Command 🡪 “*geth --networkid 14333 --datadir ./data --bootnodes enode://0df5d75715588100bd88a8c2f3afa8554edaaede9f247394dd25e77bba965436a425431bb601defbd2955b1c1fb6e3dffc2c4cfaf47f9d9b5077c052b5b88405@172.31.19.150:30301 --port 30403 --ipcdisable --syncmode 'full' --http --allow-insecure-unlock --http.corsdomain "http://remix.ethereum.org, chrome-extension://nkbihfbeogaeaoehlefnkodbefgpgknn" --http.addr 172.31.23.189 --http.port 8548 --unlock '0xd94b57548436A1c761558C6d8110d5dDbED495D4' --password password.txt console --rpc.txfeecap 0*”
2. Verify its up & running



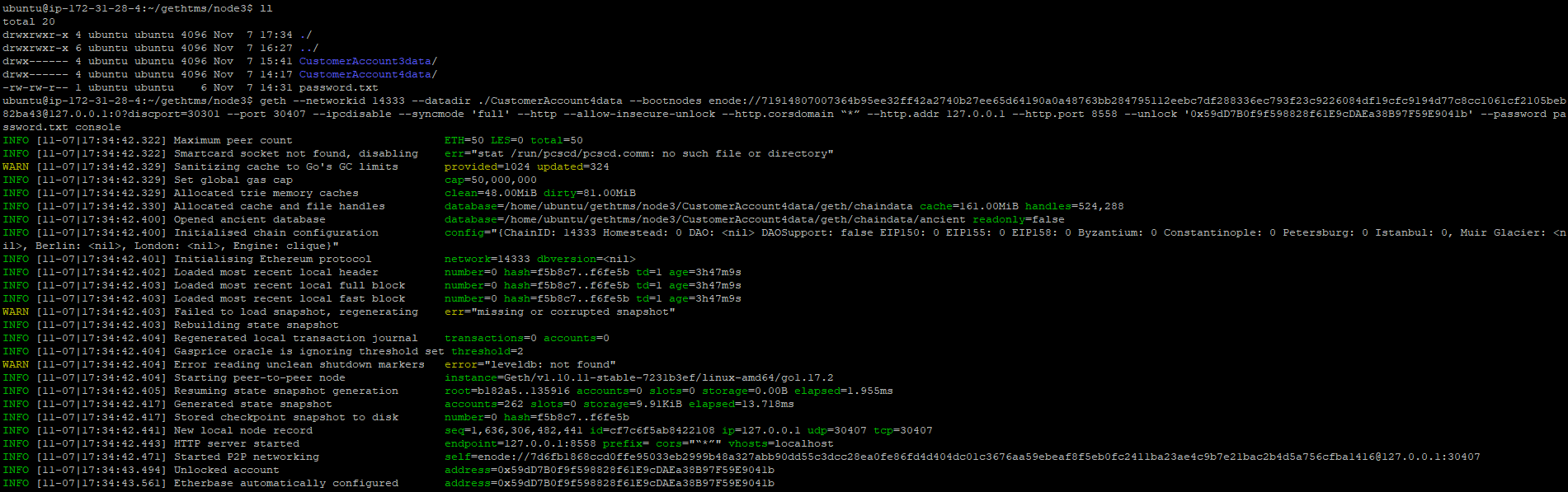
1. Start Customer account 3 on port 30405



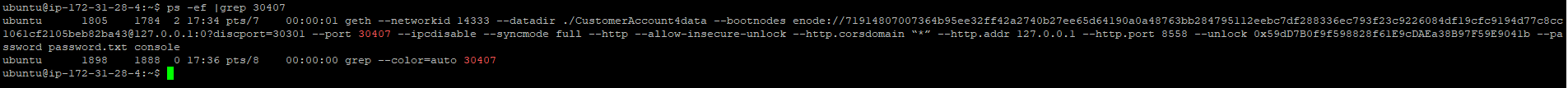
1. Startup Command 🡪 “*geth --networkid 14333 --datadir ./data --bootnodes node://0df5d75715588100bd88a8c2f3afa8554edaaede9f247394dd25e77bba965436a425431bb601defbd2955b1c1fb6e3dffc2c4cfaf47f9d9b5077c052b5b88405@172.31.19.150:30301 --port 30405 --ipcdisable --syncmode 'full' --http --allow-insecure-unlock --http.corsdomain "http://remix.ethereum.org, chrome-extension://nkbihfbeogaeaoehlefnkodbefgpgknn" --http.addr 172.31.30.69 --http.port 8549 --unlock '0xf65DaFFEE37C592504006997fBd415D5A0Ab8bA4' --password password.txt console --rpc.txfeecap 0*”
2. Verify its up & running



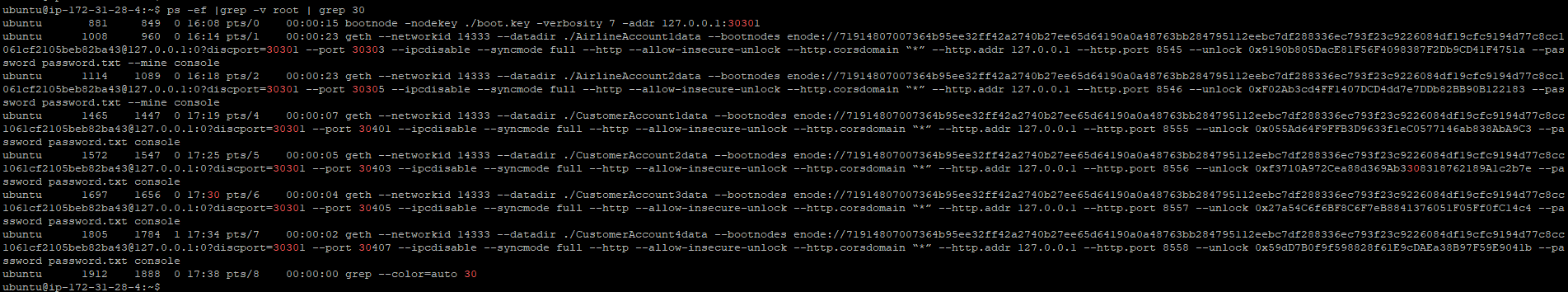
1. Start Customer account 4 on port 30407



1. Startup Command 🡪 “*geth --networkid 14333 --datadir ./data --bootnodes node://0df5d75715588100bd88a8c2f3afa8554edaaede9f247394dd25e77bba965436a425431bb601defbd2955b1c1fb6e3dffc2c4cfaf47f9d9b5077c052b5b88405@172.31.19.150:30301 --port 30407 --ipcdisable --syncmode 'full' --http --allow-insecure-unlock --http.corsdomain "http://remix.ethereum.org, chrome-extension://nkbihfbeogaeaoehlefnkodbefgpgknn" --http.addr 172.31.30.69 --http.port 8550 --unlock '0x07256eD5A5715C3cd95fF544e3D621eeFB9Bd60f' --password password.txt console --rpc.txfeecap 0*”
2. Verify its up & running



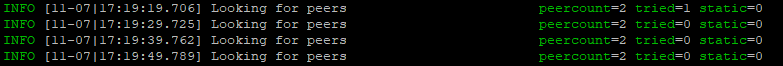
1. Here is the complete list of processes started in this section:

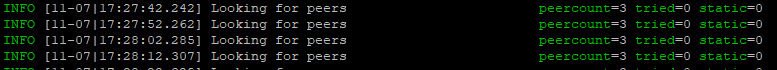


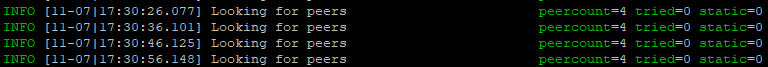
1. Verified that peer count was increasing each time













# Contracts

Following are the contract files.



TMSPortal – allows customer to search for flight, book tickets

TMSAirline – for airline to setup flight, update status etc

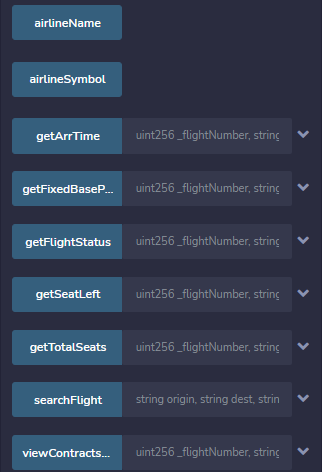
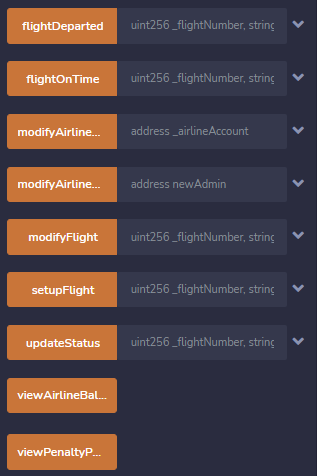
TMSTicket – Customer can payfare or claim refund, while airline can settle ticket

AirlineLibraries -

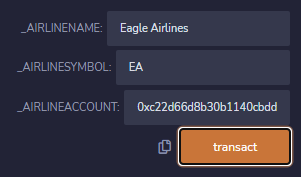
# Testing

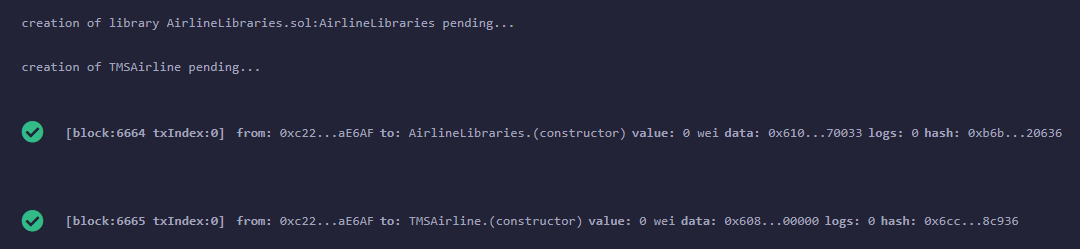
## TMS Airline

* 1. TMS Airline has the following functions implemented

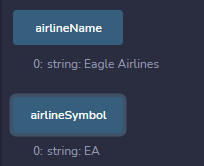


* 1. While deploying contract, give Airline Name, Symbol and account address.





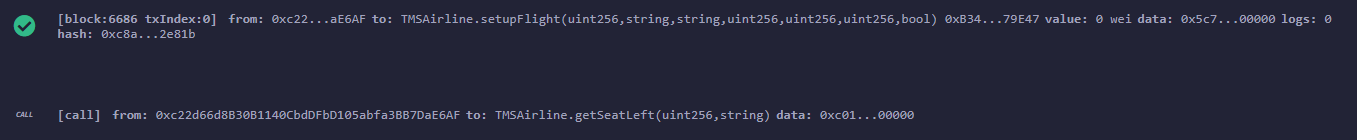
* 1. Verify Airline Name & Symbol with call functions – airlineName / airlineSymbol



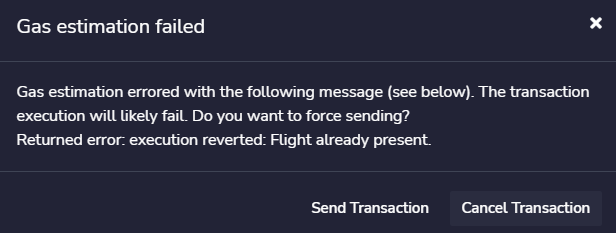


* 1. Now we can setup flight
     + Need to give epoch time (<https://www.epochconverter.com/>) in arrival and departure

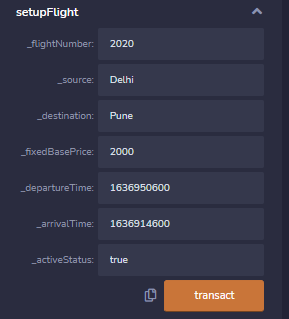


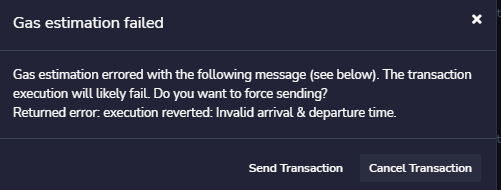


* + - If we retry same, it returns following valid error message:

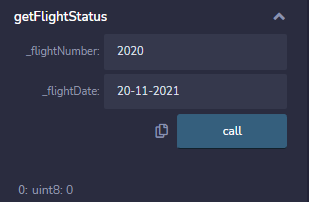


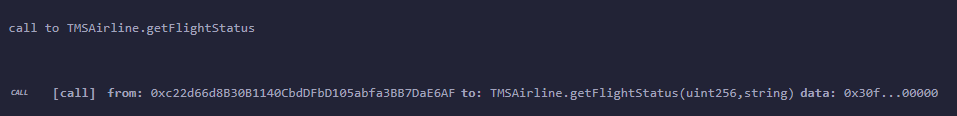
* + - If arrival time is given before departure time, then it gives valid error message.

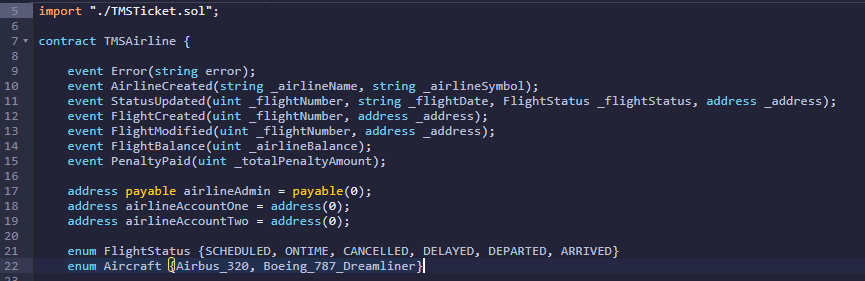




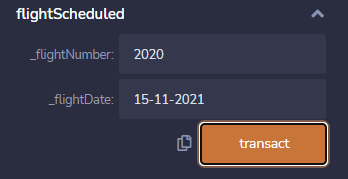
* 1. Check the status by calling flightStatus. 0 is for SCHEDULED as in the code snippet below.

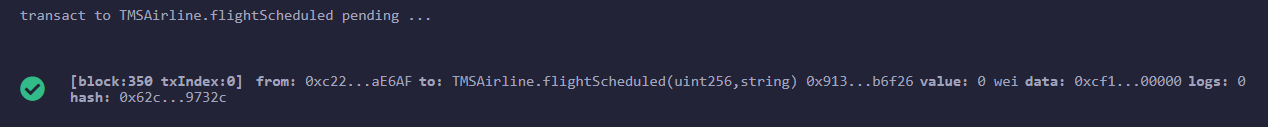




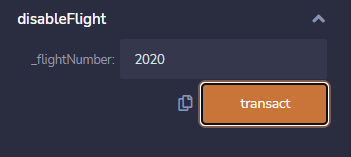


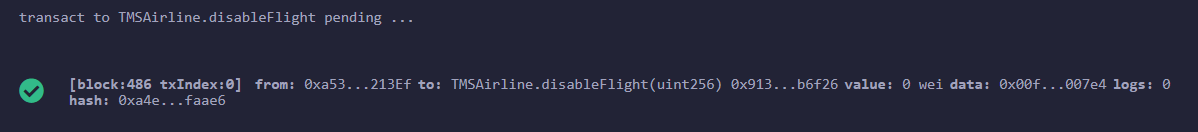
* 1. In condition where we have given “active status” as “false” in “setupFlight”, probably for some much advanced date or the one that was awaiting regulatory permission, we can schedule the flight.



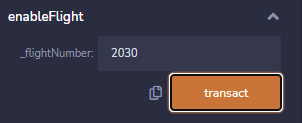


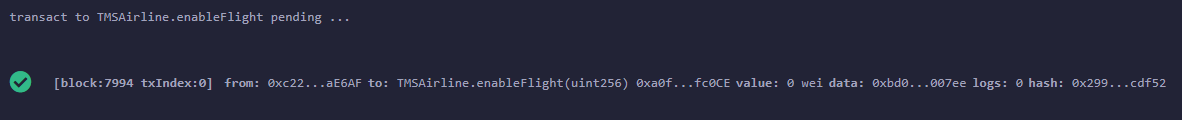
* 1. There is option to disable flight that we have scheduled earlier



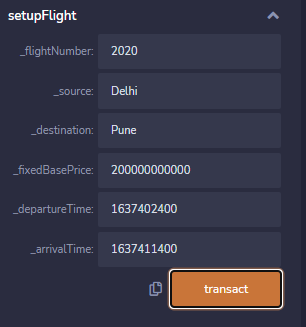


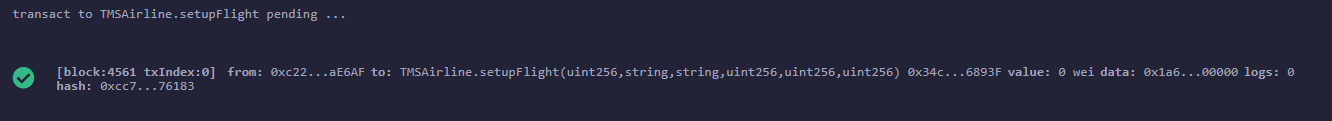
1. And an option to enable flight, if needed



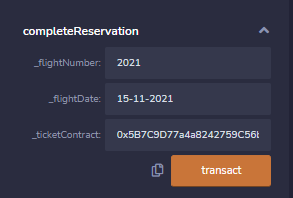


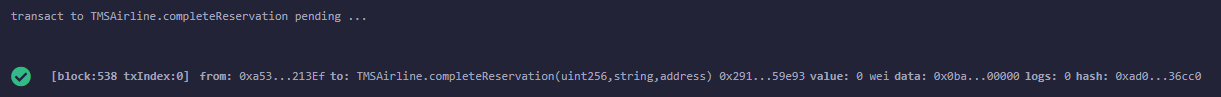
1. Now setup a new flight & schedule it (Fixed Base Price is in wei)



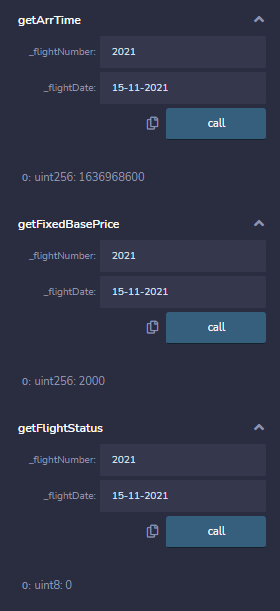


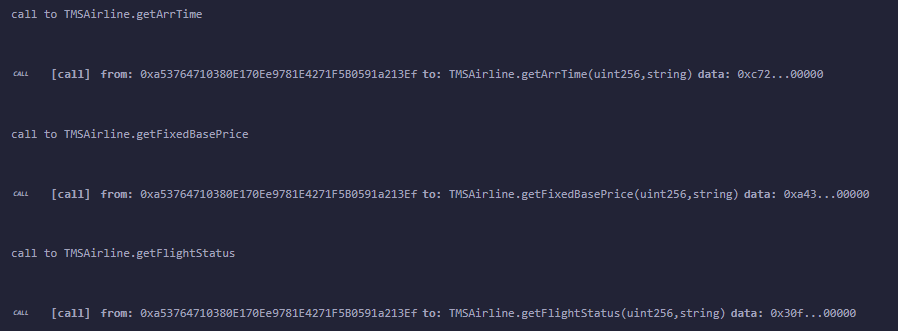
1. Complete Reservation part



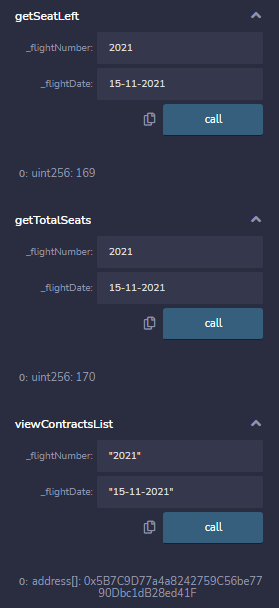


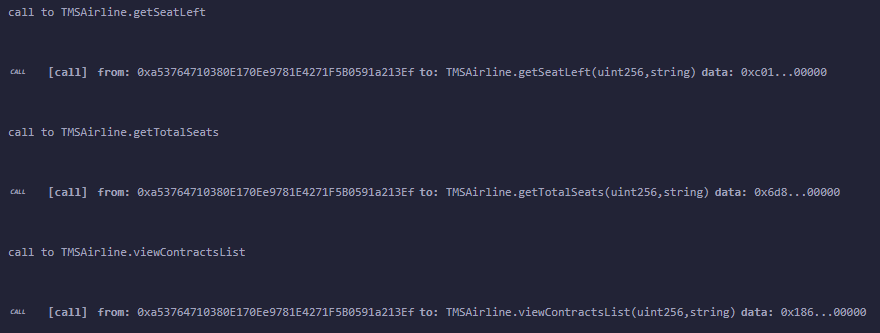
1. Once flight is scheduled, we can get Arrival time, fixed base price and flight status.



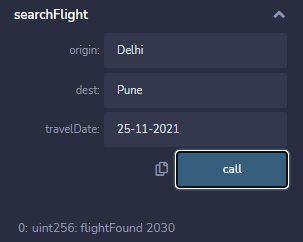


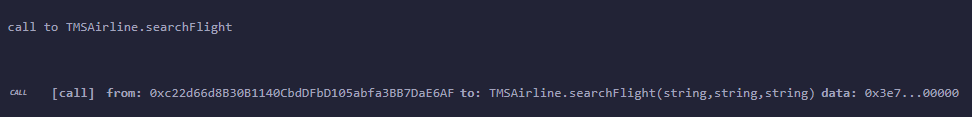
1. Can also get the remaining seats, total seats on flight and the contract list for a flight.



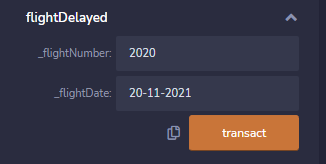


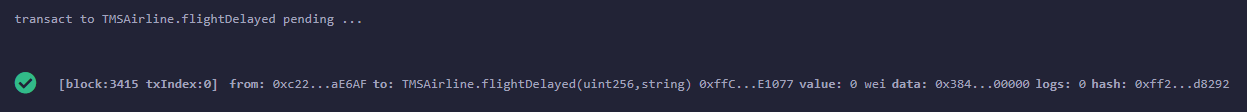
1. Can do a search flight operation



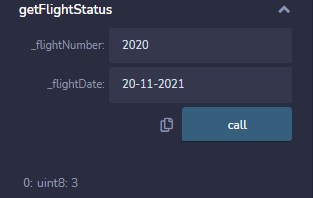


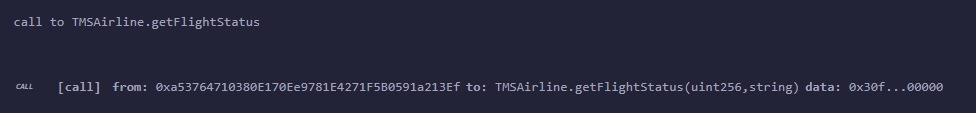
1. Update status as delayed by airline

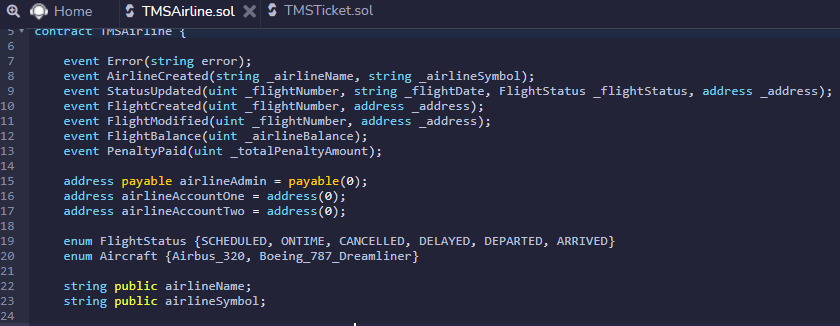




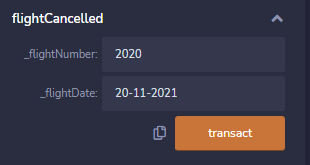
1. Check the flight status, after above (Status 3 is for DELAYED, as in code snippet below)

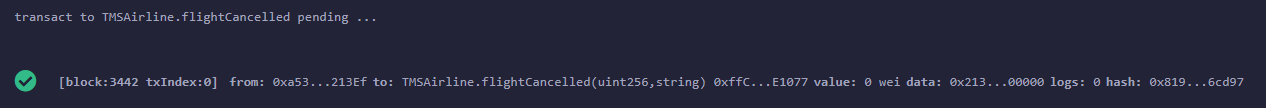




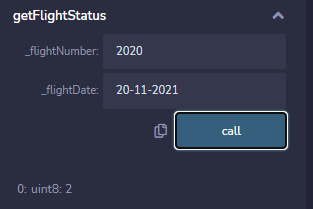


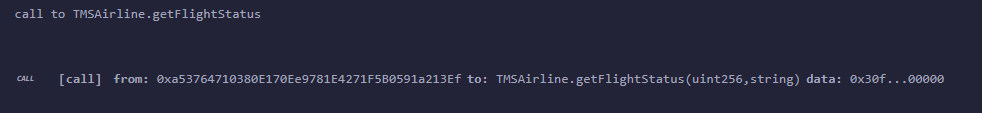
1. Cancel the flight



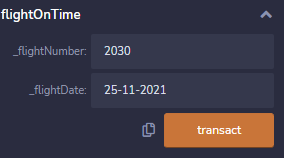


1. Now check the status, after cancelling (2- CANELLED)

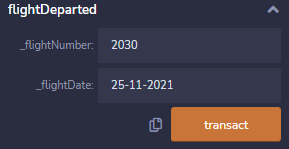




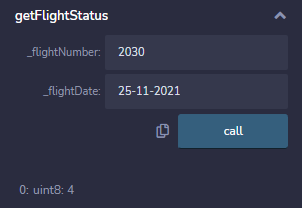
1. Setup a new flight and update that flight is on time



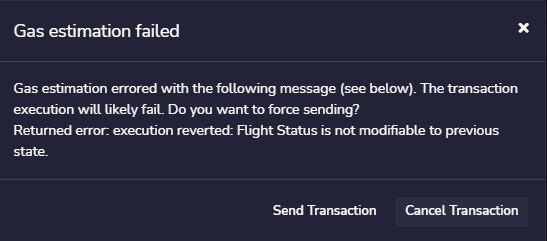
1. We can update the status as flight Departed



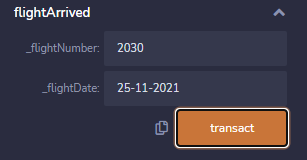
& now if we do getFlightStatus, we will that status is changed to 4 (DEPARTED)



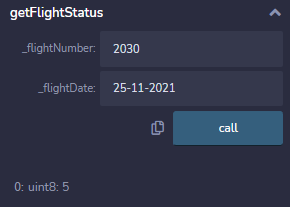
1. If we try to do an earlier operation, like “flightOnTime” now, it gives valid error message



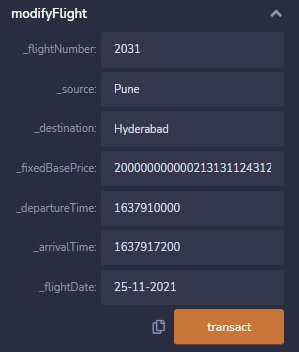
1. We can set the status to flight Arrived

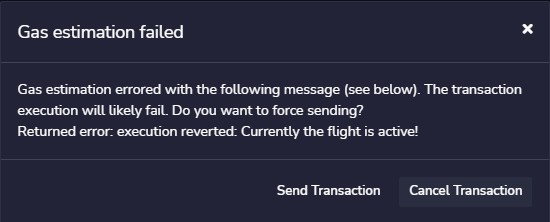


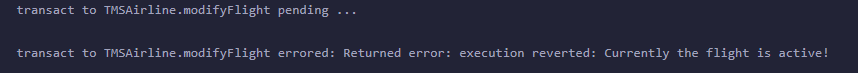
& now if we check status of flight, it will be status 5 (ARRIVED)



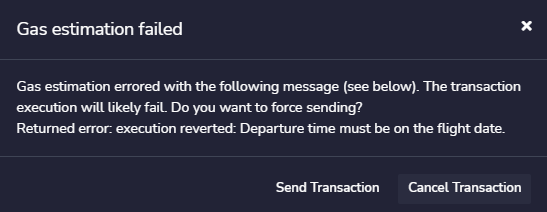
1. We can modify flight schedule, provided its in disabled state. We get valid error message if we try to modify flight schedule of an active flight

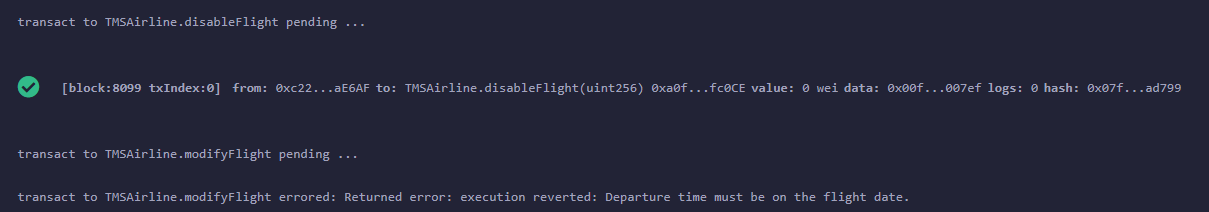




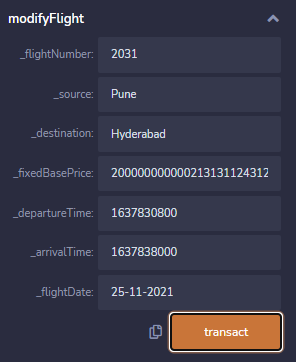


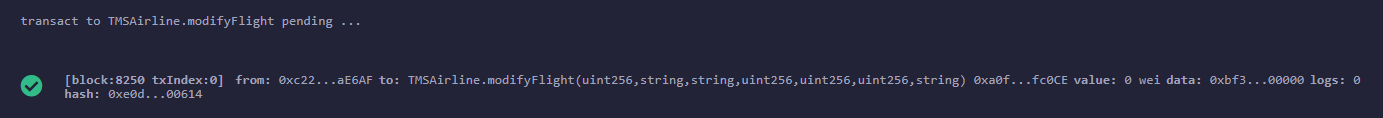
1. So, disable the flight and try to reschedule the flight to another day. It gives valid error message.



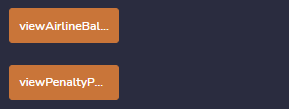


1. Modify flight to another time slot on same day and is successful

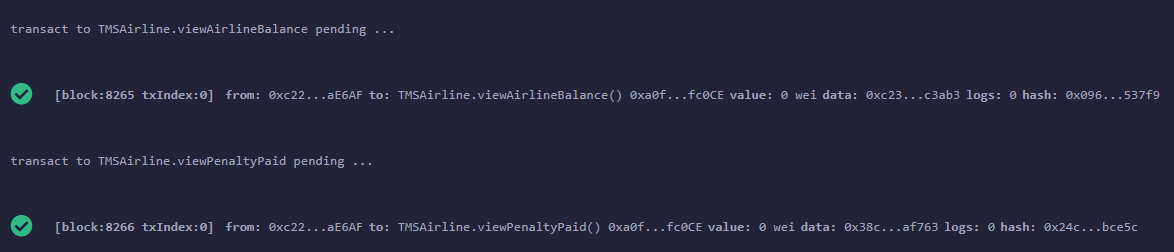




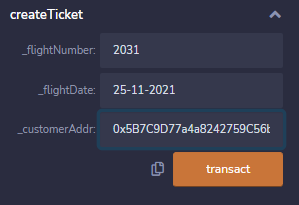
1. Functions are provided to view Airline Balance & also view Penalty Paid by Airline

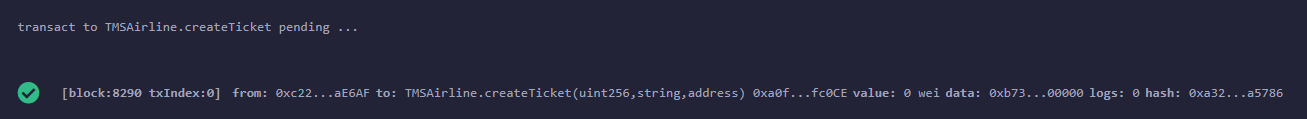


The information for it is present in log files

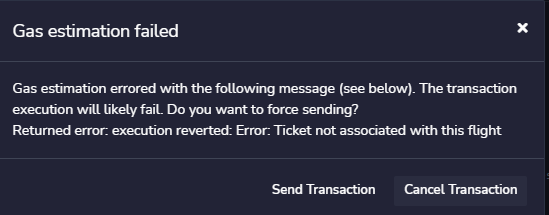


1. Airline can create a ticket for customer (as part of services provided at the airport). However, airline will not be able to pay the fare and customer will have to do it. Only with payFare done, seat will be allocated by Airline to the customer.



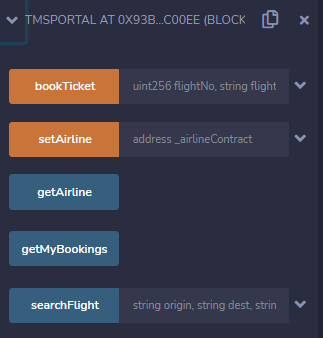


1. Airline is given provision to cancel the ticket for a particular customer, once payfare is done. But if we try to cancel reservation, prior to it, it will give valid error

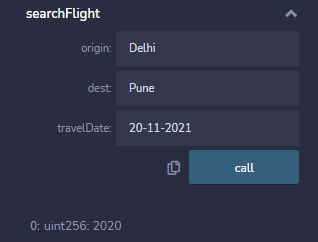


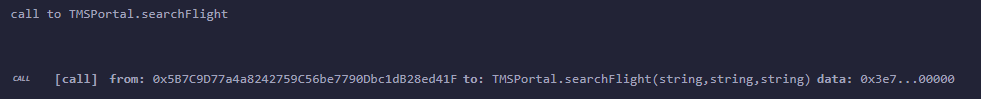
## TMS Portal

1. We have deployed following functions in the Portal Contract

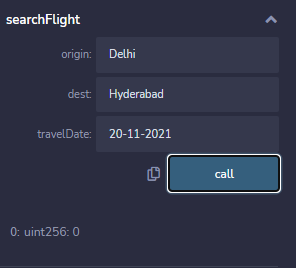


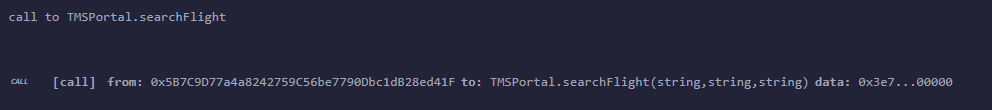
1. Search Flight – for flight that is scheduled



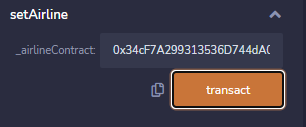


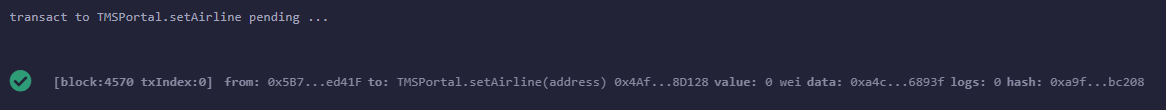
1. Search Flight – for flight that is not Setup/Scheduled. It returns empty flight number.



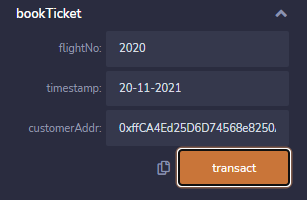


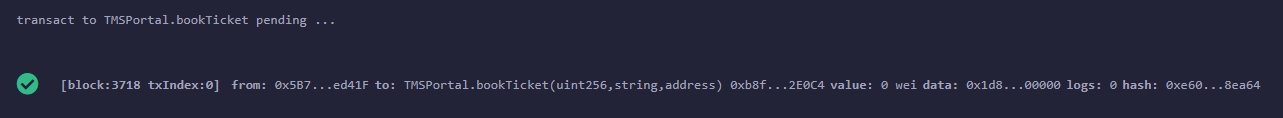
1. setAirline (Useful if one wants to switch to a subsidiary airline or to a newly deployed airline contract or connect back to old contract to view tickets)



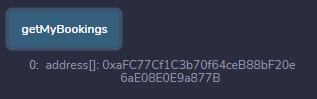


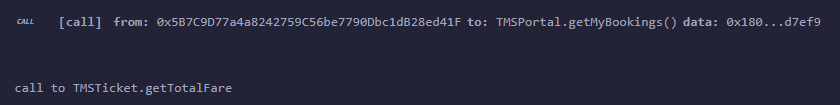
1. Book Ticket





1. Get My Bookings





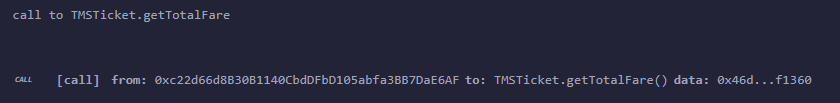
## TMS Ticket

1. Deploy TMSTicket contract with the ticket contract address retrieved while doing create Ticket. It will be either seen in logs or can do getMyBookings function as described earlier
2. We have deployed following functions



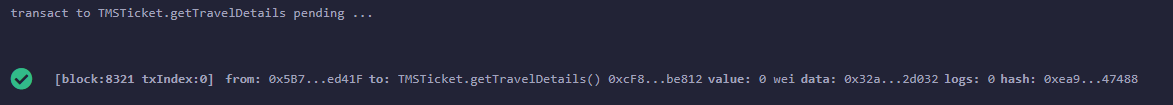
1. Get Total Fare





1. Get Trave Details



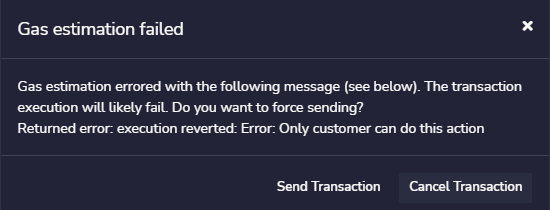


From logs, we can see the details of reservation, however, seat number is not allocated until payFare is done by customer.



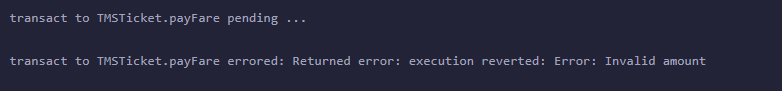
1. Try to do payfare from airline. It gives proper permission error message





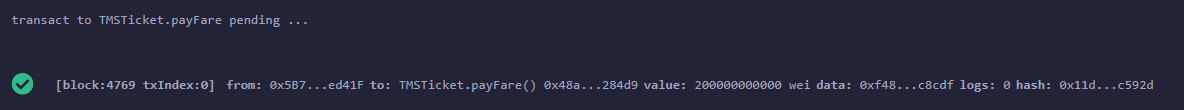
1. Click on payfare without updating the ticket value. It gives valid error message



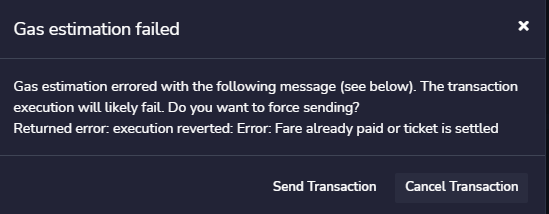


1. Update value to the amount of ticket (retrieved from getTotalFare function) & click on payFare





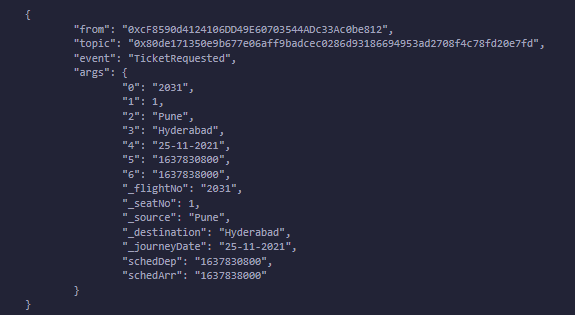
1. Repeat the same & it gives valid error



1. Once we payFare, we can do “getTravelDetails” once again.

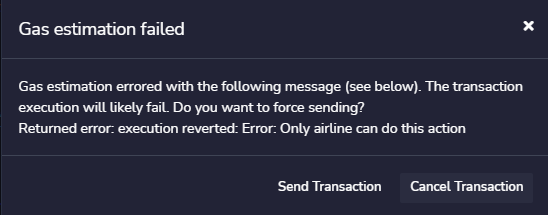


From logs, we can see that seat Number is now allocated



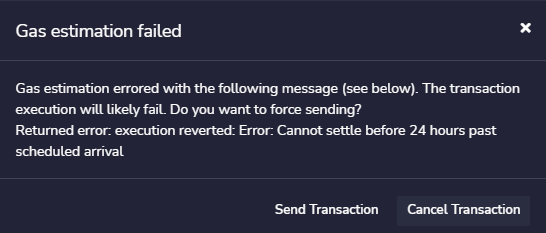
1. Do settle ticket from customer account





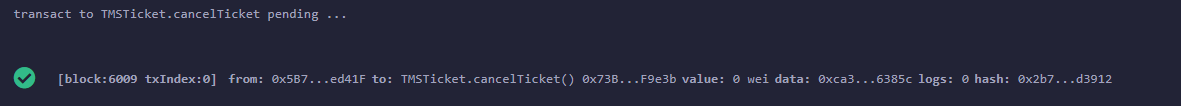
1. Claim Refund before 24 hours past the arrival time. Gets valid message

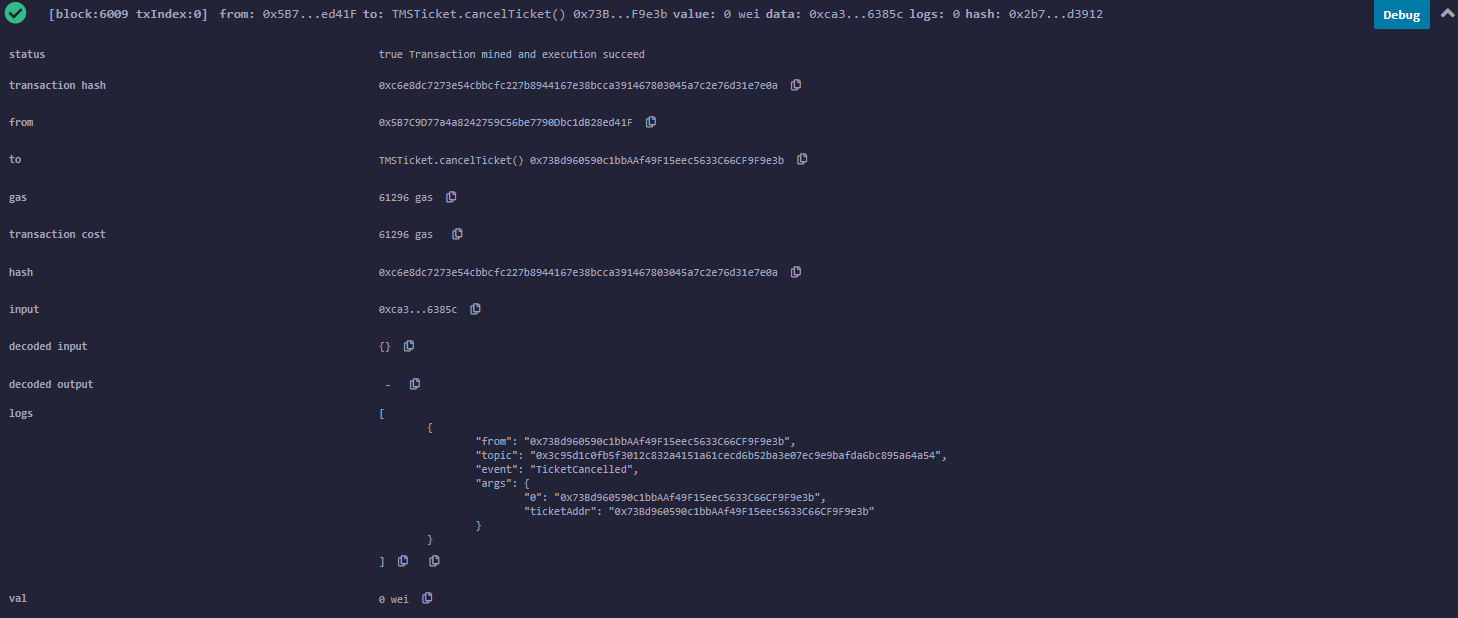




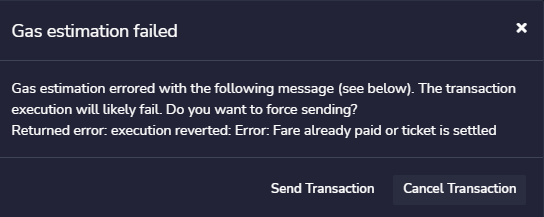
1. Cancel Ticket



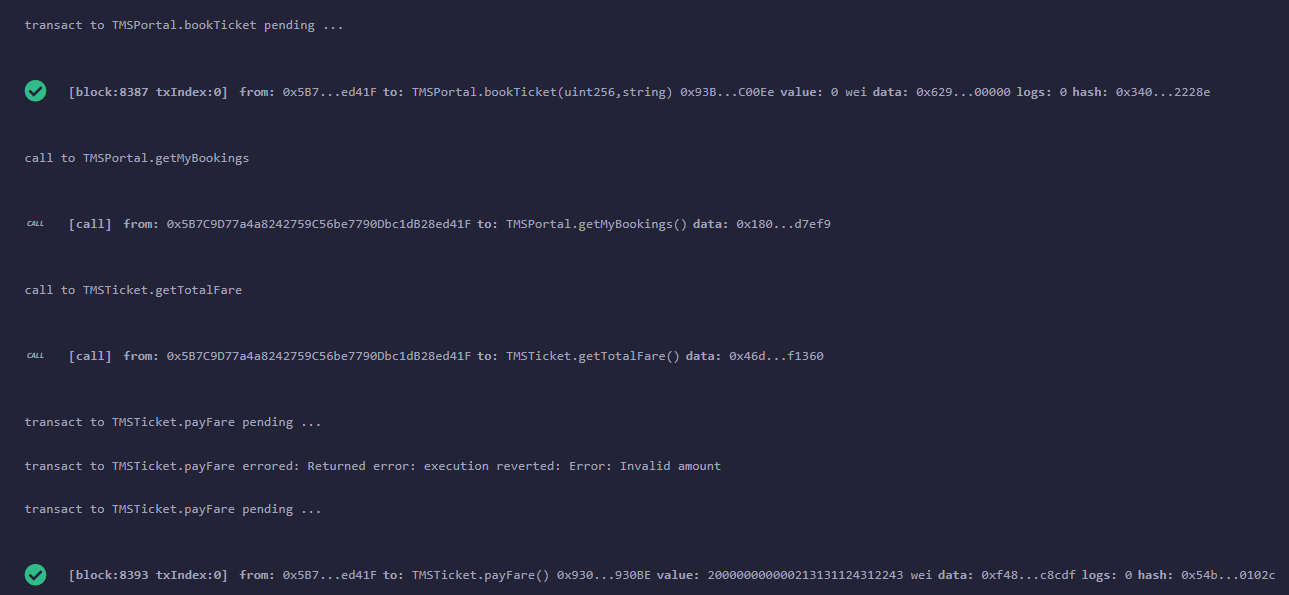




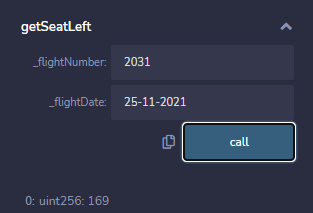
1. Try to do payFare on cancelled ticket and gives valid error message



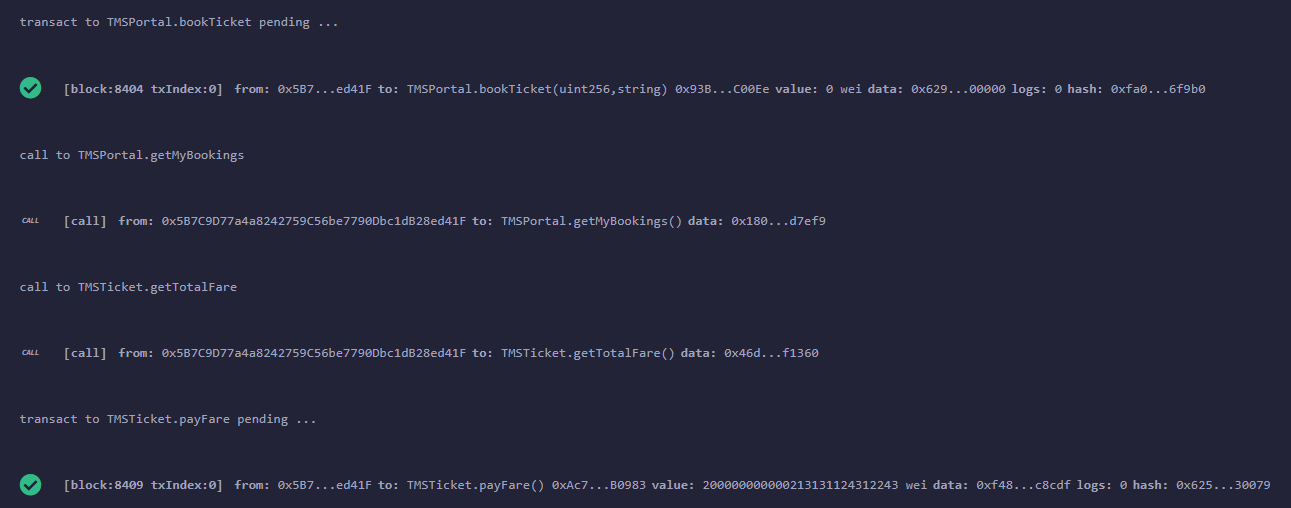
1. Book a new ticket and pay Fare.



Go to Airline contract and check for Seats Left. The seat has reduced by 1 from 170 to 169



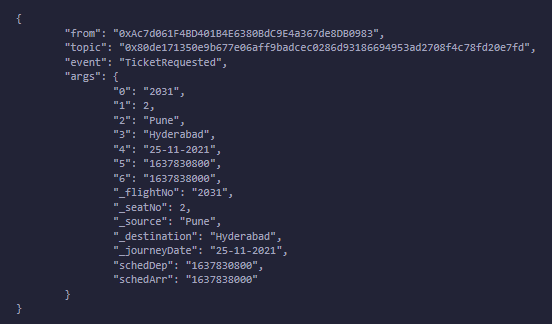
1. Do the booking for another ticket.



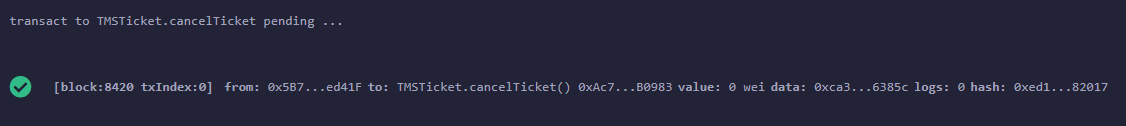
1. Now the seats left has reduced by 1 more



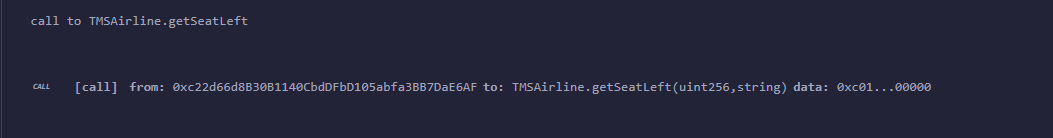
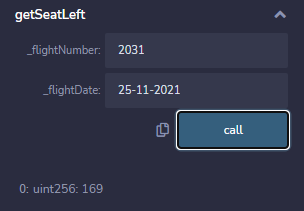
1. Perform getTravelDetails. We can see that Airline has automatically allocated seat Number to it



1. Now lets perform cancelTicket

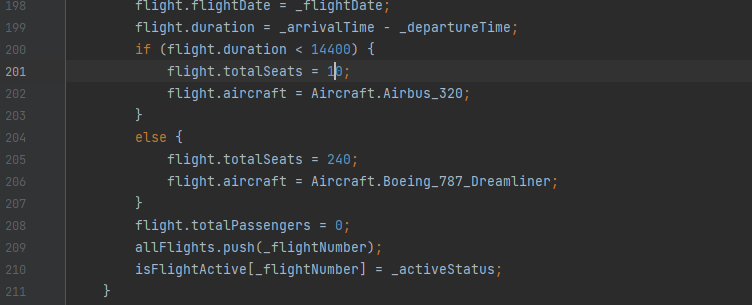


1. Check on getSeatLeft from airline and can see that count has increased by 1, due to cancellation. This confirms end to end flow is working fine.



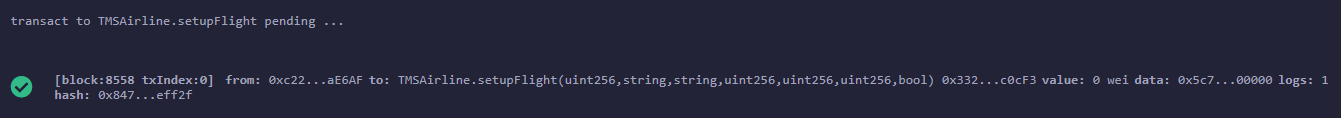
## Testing Advanced Feature

1. To test various ticket prices, changed the number of seats in airline to 10 in Airline.sol file line 201. Also single customer can book any number of seats (as in section: “Assumptions”, points 2 & 3)

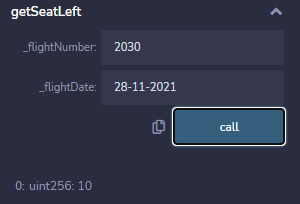


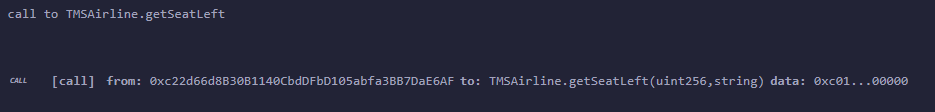
1. Setup the flight from airline account



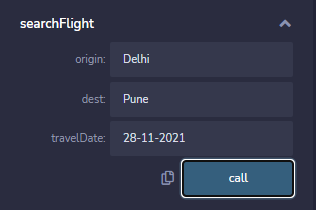


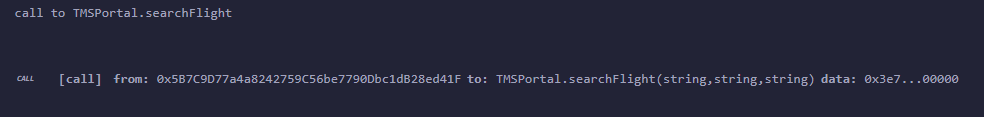
1. Verify the seats available is 10



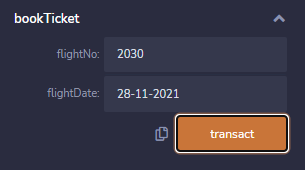


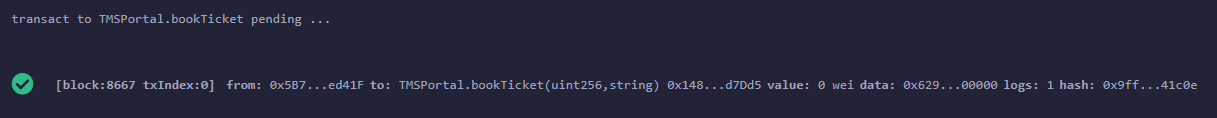
1. Deploy Portal and do Search Flight



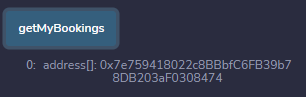


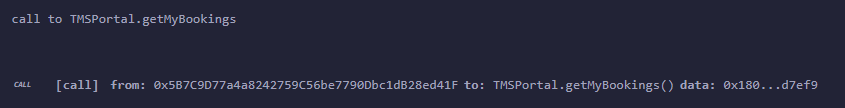
1. Book 1 ticket



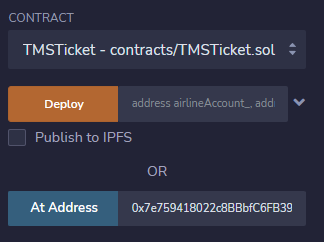


1. Retrieve the ticket id

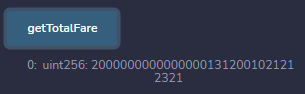


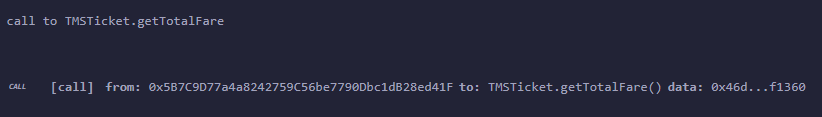


1. Deploy TMSTicket contract with the ticket contract address retrieved above



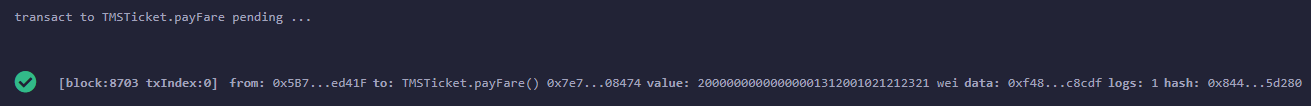
1. Get the fare price





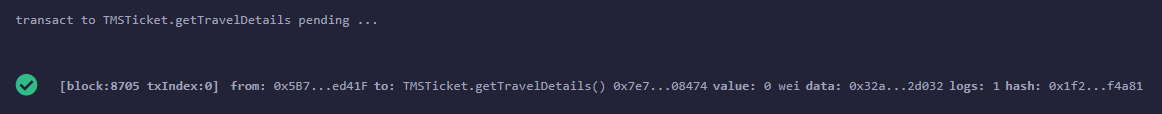
1. Give this price in value and do payFare





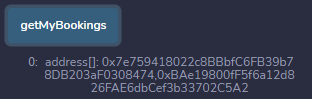
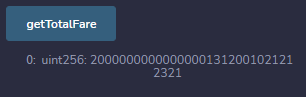
1. Get travel details & verify from log that seat number 1 is assigned

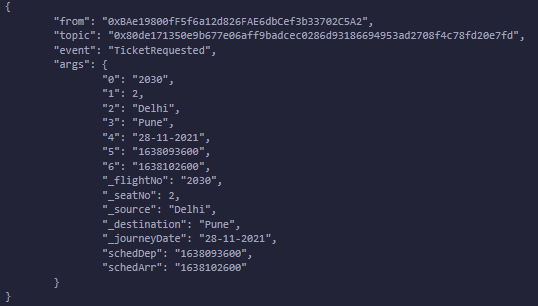


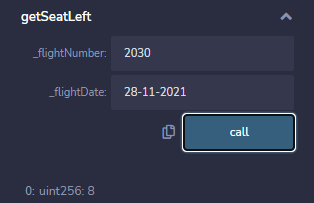




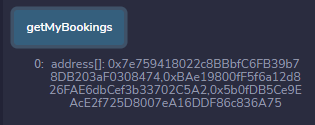
1. Repeat steps 5 to 10 above (Book ticket, retrieve ticket id, deploy ticket contract, get fare price, do payFare & retrieve the travel details. Verify that seat number 2 is assigned. From airline contract, verify that seats left is only 8 now.

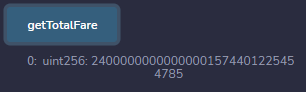
 



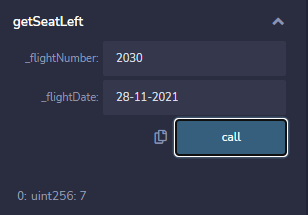


1. Repeat steps 5 to 10 above (Book ticket, retrieve ticket id, deploy ticket contract, get fare price, do payFare & retrieve the travel details. Verify that seat number 3 is assigned and the total seat left is only 7 now. We can see that the fare price has changed now from the basic one by 20%.

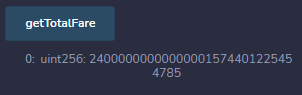


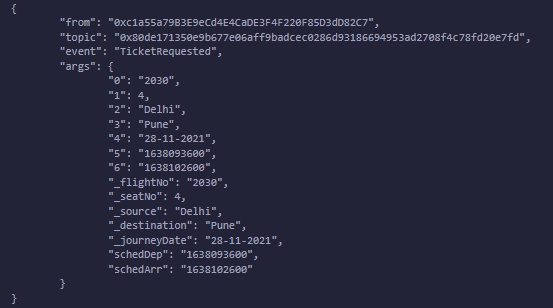


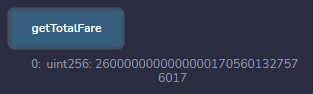


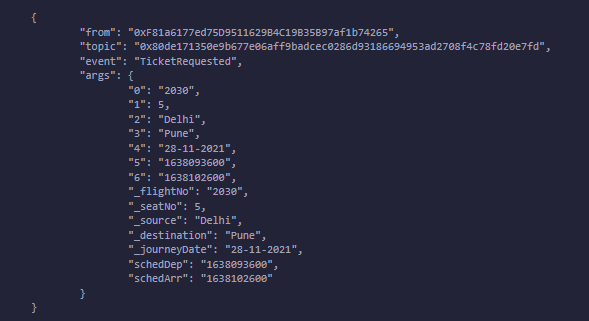


1. Now lets do it for 2 more times. For seat number 4 the price will be same, but it will increase by 40% for seat number 5.









1. This confirms that the fare price advanced feature is working fine. Similarly, we have implemented the cancellation penalty and delay penalty in Ticket sol file.

# Challenges Faced

We faced following challenges while working in the project and successfully overcame them.

1. RPC is deprecated in v0.8.7 and works only with http.
2. No proper version control mechanism and managed with Google Drive
3. Handling Epoch Time
4. Gas Estimation Failed error. Used “--rpc.txfeecap 0” to set it to unlimited.
5. Remix integration
6. Metamask integration
7. AWS Educate free tier allows only 3 EC2 instances to be created
8. Spurious Dragon – Overcame by reducing code size & also “Enabling Optimization” option in remix