# Assignment 1

Name – Sati, Ankit Date – 5/18/2022

Section – 001

SID – as14128

Total in points (Maximum 100 points)–

Professors Comments –

**Part 1 – Local Deployment**

**Step 1 - Compilation of Casino.sol file using truffle**

Following the tutorial, we create and compile a smart contract using truffle. Truffle is a development environment, testing framework and asset pipeline for Ethereum with features for smart contract compilation, linking, deployment and binary management. This screenshot shows the successful compilation of Casino.sol file using truffle.

Graphical user interface, text, application, email

Description automatically generated

**Step 2 – deployment on Ropsten testnet**

Next, we use Remix IDE to deploy our smart contract to the Testnet Ropsten Blockchain. We obtain free ether for the testnet using an ether faucet, and we get ether in our metamask account. This ether, which has no real value, is used as payment to deploy our smart contract. In this screenshot, we can see that our smart contract has been deployed successfully on the Ropsten testnetText

Description automatically generated

**Step 3 - Connect front-end application built in ReactJS to contract.**

The next step is to connect our front-end application built in ReactJS to our smart contract. We copy the ABI generated by Remix into our code along with the address where our smart contract is deployed. Then, we test the application by betting some ether and we see Metamask ask us to confirm the transaction.

**Step 4 – Getting Ethereum**

The following screenshot shows a transaction that has completed successfully using our front-end application and Metamask  
(Note: we have to use the "@metamask/legacy-web3" library in our frontend because web3 API support has been deprecated in newer versions of Metamask. The recommended solution is to use ether.js API but this solution works with minimal changes required to the code)

Graphical user interface, text, application

Description automatically generated

**Part 2 – Deploying the application above on the cloud platforms**

1. AWS

Document followed - <https://docs.aws.amazon.com/managed-blockchain/latest/hyperledger-fabric-dev/managed-blockchain-get-started-tutorial.html>

Git - <https://github.com/awsdocs/amazon-managed-blockchain-ethereum-developer-guide/tree/main/doc_source>

Step 1 – Creating the network

Graphical user interface, application

Description automatically generated

Role

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Adding Member 2

Graphical user interface, text, application, email

Description automatically generated

Steo 2 – VPC endpoint

Graphical user interface, application, email

Description automatically generated

Step 3 – AMI

Graphical user interface, application

Description automatically generated

Table

Description automatically generated

Permissions

Graphical user interface, text, application, email

Description automatically generated

Docker image

Graphical user interface, text, application, email

Description automatically generated

Test Channel

A black and white document

Description automatically generated with low confidence

Deploying Chain code

Graphical user interface, text, application

Description automatically generated

Linking with front end

Repo shared above

Text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

1. IBM

Graphical user interface, application, Teams

Description automatically generated

Graphical user interface, application, Teams

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, text, application, Teams

Description automatically generated