2) Configuring Network Settings

To connect to Ropsten Network, You need to specify the network, Infura Project ID and mnemonic in **truffle-config.js** file.

1) Install HDWalletProvider Dependency

```
$ npm install truffle-hdwallet-provider --save
```

2) Open **truffle-config.js.** You can see a Bunch of code commented. We only need to uncommented the required Configuration.

3) Uncomment and assign the infura project id and mnemonic key as shown below.

```
const HDWalletProvider = require('truffle-hdwallet-provider');
const infuraKey = "yourInfuraProjectID";

const mnemonic = "this is a mnemonic this is a mnemonic this is a mnemonic this is a mnemonic";
```

4) Uncomment the ropsten configuration as shown below.

```
ropsten: {
  provider: () => new HDWalletProvider(mnemonic, `https://ropsten.infura.io/${infuraKey}`),
  network_id: 3,       // Ropsten's id
  gas: 5500000,        // Ropsten has a lower block limit than mainnet
  confirmations: 2,    // # of confs to wait between deployments. (default: 0)
  timeoutBlocks: 200,  // # of blocks before a deployment times out  (minimum/default: 50)
  skipDryRun: true     // Skip dry run before migrations? (default: false for public nets )
},
```

Server-Side Signed Txn (Part 2 of 4)

3) Write, Compile and Migrate Contract

Here, we going to write, compile and migrate contract to Ropsten network

1) Copy the Contract to **contracts/Auctions.sol**

```
pragma solidity ^0.5.0;

contract Auction{
    //Declare All State Variables here
    address internal auction_owner;
    uint256 public auction_end;
    uint256 public highestBid;
    address payable public highestBidder;

    //Define a constructor for your contract
    constructor(uint _biddingTime, string  memory _brand, string memory
_Rnumber) public {
        auction_owner=msg.sender;
        auction_end =now +_biddingTime * 1 minutes;
        Mycar.Brand =_brand;
        Mycar.Rnumber =_Rnumber;
        Mycar.owner = auction_owner;
    }

    //Function for get auction details
    function getAuctionDetails() public view returns
(uint256,uint256,address,address) {
       return (auction_end,highestBid,highestBidder,auction_owner);
    }

    //Define a structure for Vehicle Details
    struct car{
        string Brand;
        string Rnumber;
        address owner;
    }
    car public Mycar;

    //Mapping that accepts the bidder's address as the key, and with the
value type being the corresponding bid
    mapping(address => uint) public bids;
    event BidEvent (address indexed highestBidder,uint256 highestBid);
    event WithdrawalEvent(address withdrawer,uint256 amount);
```

Server-Side Signed Txn (Part 2 of 4)

```
//Checks whether the bid is can be done
    modifier bid_conditions(){
        require(now<= auction_end,"auction timeout");
        require(bids[msg.sender]+msg.value > highestBid, "cant't bid,
make a higher Bid");
        require(msg.sender != auction_owner, "Auction owner cant bid");
        require(msg.sender != highestBidder, "Current HighestBidder cant
bid");
        _;
    }

    //makes the contract ownable
    modifier only_owner(){
        require(msg.sender == auction_owner);
        _;
    }

    //Define Bidding function
    function bid() public payable bid_conditions returns (bool){
        highestBidder=msg.sender;
        bids[msg.sender]=bids[msg.sender]+msg.value;
        highestBid=bids[msg.sender];
        emit BidEvent(highestBidder,highestBid);
        return true;
    }

    // check auction status
    function auction_status() public view returns(bool state){
        state = now < auction_end;
    }

    //Withdraw function for loosers
    function getAmount() public returns (bool){
        require(now> auction_end, "can't withdraw, Auction is still
open");
        require(msg.sender != auction_owner, "owner cant withdraw");
        require(msg.sender != highestBidder, "HighestBidder cant
withdraw");
        uint amount = bids[msg.sender];
        bids[msg.sender]=0;
        msg.sender.transfer(amount);
        emit WithdrawalEvent(msg.sender,amount);
        return true;
    }
```

```
//Withdraw Bid amount to owner address
    function withdraw() public only_owner returns (bool){
        require(now> auction_end, "can't withdraw, Auction is still
open");
        msg.sender.transfer(highestBid);
        Mycar.owner = highestBidder;
        emit WithdrawalEvent(msg.sender,highestBid);
        return true;
    }
}
```

2) Write the following code to **migrations/2_deploy_auction.js**

```
const Auction = artifacts.require("Auction");

const Duration = 30; //Value in Minutes
const brand = "BMW";
const rNumber = "RN00091"

module.exports = function (deployer) {
deployer.deploy(Auction, Duration, brand, rNumber);
};
```

3) Run the following command to compile the Contract.

```
$ truffle compile
```

___Truffle Output____

```
Compiling your contracts...
===========================
> Compiling ./contracts/Auction.sol
> Compiling ./contracts/Migrations.sol
> Artifacts written to /home/sabir/Workplace/KBA/CED Lab/Auction Server-Side-Txn/build/contracts
> Compiled successfully using:
   - solc: 0.5.0+commit.1d4f565a.Emscripten.clang
```

4) Run the below command to deploy the contract to ropsten

```
$ truffle migrate --network ropsten
```

Note:

If you are trying to redeploy the contract, you have to specify **--reset** attribute to the above command

```
$ truffle migrate --network ropsten --reset
```

___Truffle Output____

```
Compiling your contracts...
===========================
> Everything is up to date, there is nothing to compile.


Starting migrations...
======================
> Network name:    'ropsten'
> Network id:      3
> Block gas limit: 8000000


1_initial_migration.js
======================

   Deploying 'Migrations'
   ----------------------
   > transaction hash:    0xc00134964a44c574be32ba92a04e0a0b9f0fb549aee5cf03679eef9311bd9084
   > Blocks: 1            Seconds: 21
   > contract address:    0x749c62D7524Bdc5C3A7102Fe115d446b579dBe4F
   > account:             0xD7dED5948EdE7264E3cEa5b8F958887caffB9C3a
   > balance:             4.884878239963401
   > gas used:            284908
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.00569816 ETH

   Pausing for 2 confirmations...
   ------------------------------
   > confirmation number: 1 (block: 5227522)
   > confirmation number: 2 (block: 5227523)

   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:          0.00569816 ETH


1552641525_auction.js
=====================
0xD7dED5948EdE7264E3cEa5b8F958887caffB9C3a

   Deploying 'Auction'
   -------------------
   > transaction hash:    0xdf29eba1a70bc90be3071ca910b0649025f00e3cc58c8120281b22e5e38541cb
   > Blocks: 1            Seconds: 33
   > contract address:    0x9C0e09588c76329F4fe073a8E554d0A8100bd6eC
   > account:             0xD7dED5948EdE7264E3cEa5b8F958887caffB9C3a
   > balance:             4.863641259963401
   > gas used:            1019815
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.0203963 ETH

   Pausing for 2 confirmations...
   ------------------------------
   > confirmation number: 1 (block: 5227528)
   > confirmation number: 2 (block: 5227529)

   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:           0.0203963 ETH


Summary
=======
> Total deployments:   2
> Final cost:          0.02609446 ETH
```