2) Demonstrate the steps involved in the process of deploying your smart contract to a live network.

Ans. To deploy a smart contract to a live network using Hardhat, follow the steps involving configuration, deploying the contact and verifying the deployment. For this demonstration, we will deploy the same SimpleStorage contract which we used in Testing *1) Question*.

Steps for Deploying a Smart Contact to a live network:
1. Install necessary dependencies.
2. Create an Alchemy or Infura project (for RPC URL)
3. Set up environment variables (Private key and RPC URL)
4. Configure Hardhat for the live network
5. Write the deployment script
6. Deploy the contract

Step 1. Install necessary dependencies. You need some additional packages for deploying the contract. These include dotenv (to manage environment variables)
- Install dotenv for environment variable management:
npm install dotenv –save

Step 2. Create an Alchemy or Infura project

You will need an RPC URL to connect to the Ethereum network.

1. **Create an Alchemy account**:
    - Go to Alchemy and sign up.
    - Create a new project (choose the Ethereum mainnet or testnet like Goerli).
    - Copy the **RPC URL** of the project.
2. **Create an Infura account** (optional, as an alternative to Alchemy):
    - Go to Infura and sign up.
    - Create a new project.
    - Copy the **RPC URL** of the project.

Step 3. Set up Environment variable

You need to secure your private key and RPC URL in environment variables to avoid hardcoding them.

- Create a .env file in the root of your project:
 touch .env

- Add your private key and RPC URL to the .env file.

- Update your hardhat.config.js to read from .env

Step 4. Configure Hardhat for Live network

You need to configure the network you'd like to deploy to in hardhat.config.js. This file should now look something like this for deployment on Goerli testnet:

```
require("@nomiclabs/hardhat-waffle");

require("dotenv").config();

const { ALCHEMY_API_URL, PRIVATE_KEY } = process.env;

module.exports = {

  solidity: "0.8.0",

  networks: {

    goerli: {

      url: ALCHEMY_API_URL,

      accounts: [`0x${PRIVATE_KEY}`]

    },

   }

  };
```

Step 5. Write the deployment script
Next, you need to write a deployment script for your contract. Create a new file scripts/deploy.js

```
async function main() {
  // Get the contract to deploy
  const SimpleStorage = await ethers.getContractFactory("SimpleStorage");
  console.log("Deploying SimpleStorage contract...");
  const simpleStorage = await SimpleStorage.deploy();
  await simpleStorage.deployed();
  console.log("SimpleStorage deployed to:", simpleStorage.address);
}
main()
  .then(() => process.exit(0))
  .catch((error) => {
    console.error(error);
    process.exit(1);
  });
```

Step 6. Deploy the contract

To deploy the contact to the Goerli testnet or any other live network, run the deployment script with the following command.

Npx hardhat run scripts/deploy.js –network goerli

Summary of the Deployment Process:

1. Set up the project and configure Hardhat for the live network.
2. Create a deployment script to deploy the contract.

3. Run the deployment script using Hardhat on the desired network (e.g., Goerli testnet or Ethereum mainnet).