**2. Smart Contract**

This step ensures that your smart contract (likely written in Solidity) is correctly compiled, deployed, and integrated into the rest of your application (backend/frontend).

**• Verify BvsBContract.sol compiles and is deployed.**

* **BvsBContract.sol** is the Solidity file containing your smart contract logic.
* You need to **compile** this contract using a tool like:
  + **Hardhat**
  + **Truffle**
  + **Remix IDE**
* Once compilation is successful, you should **deploy** it to your desired network (e.g., local development network like Ganache, or a testnet/mainnet like Goerli, Sepolia, or Ethereum Mainnet).
* **Verification means**:
  + There are no compiler errors.
  + The contract is deployed and has a valid address on the blockchain.
  + Optionally, you might want to verify the source code on a block explorer (like Etherscan).

**• Confirm ABI is exported and used in backend/clients.**

* The **ABI (Application Binary Interface)** is a JSON file generated during compilation that describes how to interact with your smart contract.
* It is **required by frontend (like React.js apps using Web3.js or Ethers.js)** and backend (like Node.js services) to:
  + Call functions in the contract
  + Read data
  + Send transactions
* You must ensure:
  + The ABI file is **exported** (usually located in something like artifacts/contracts/BvsBContract.json)
  + The correct ABI is **imported** into your backend or frontend code

js

CopyEdit

import BvsBABI from './abis/BvsBContract.json';

const contract = new web3.eth.Contract(BvsBABI.abi, contractAddress);

**• Confirm contract address is stored in .env.**

* The **contract address** is the location of your deployed contract on the blockchain.
* Instead of hardcoding it in your app, it's best practice to store it in a **.env file** for configuration purposes.
  + Example:

env

CopyEdit

REACT\_APP\_CONTRACT\_ADDRESS=0x1234567890abcdef...

* This makes it:
  + Easier to manage across environments (local/testnet/mainnet)
  + More secure and cleaner for CI/CD workflows
* Your app or backend can then load it using environment variables:

js

CopyEdit

const contractAddress = process.env.REACT\_APP\_CONTRACT\_ADDRESS;

**✅ Summary**

To complete step 2 properly:

| **Task** | **What You Do** |
| --- | --- |
| **Compile contract** | Run Hardhat/Truffle compile |
| **Deploy contract** | Deploy using a script or CLI |
| **Export ABI** | Use the compiled output (JSON) |
| **Integrate ABI** | Import in frontend/backend |
| **Store address** | Save contract address in .env |

Let me know if you need code examples for any of these steps.