Creating a blockchain-based identity involves several technical and conceptual steps, which usually encompass generating cryptographic keys, creating a digital identity record, and recording that identity on a blockchain.

Below are the broad steps to create a basic blockchain identity. Please note, however, that depending on the specific requirements and the blockchain you choose, the steps might differ:

||| 1. Generate Cryptographic Keys
First, you need a pair of cryptographic keys – a public key and a private key.

```bash
# Using OpenSSL to generate a 2048-bit RSA private key
openssl genpkey -algorithm RSA -out private_key.pem -pkeyopt rsa_keygen_bits:2048

# Extract the public key from the private key
openssl rsa -pubout -in private_key.pem -out public_key.pem
```

||| 2. Create a Digital Identity Record
Create a JSON object to represent the identity.

```json
{
  "publicKey": "YourPublicKeyHere",
  "identityInformation": {
    "username": "JohnDoe",
    "attributes": {
      "country": "Nomadland",
      "email": "john.doe@example.com"
    }
  }
}
```

||| 3. Hash the Identity Record
Generate a SHA-256 hash of your identity record to ensure integrity.

```bash
# Assuming your identity record is saved in a file identity.json
openssl dgst -sha256 identity.json
```

||| 4. Record the Identity on a Blockchain
Choose a blockchain where you want to store the identity record. For example, you can use Ethereum and smart contracts to store the hash.

Here is a simple Ethereum smart contract written in Solidity to store identity hashes:

```solidity
```

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract IdentityRegistry {
    mapping(address => bytes32) public identities;

    function registerIdentity(bytes32 identityHash) public {
        identities[msg.sender] = identityHash;
    }
}
```

Deploy this contract to the Ethereum blockchain using a development environment like Truffle or Remix.

||| 5. Interact with the Smart Contract
Use a Web3 library like web3.js or ethers.js to interact with the smart contract and record your identity hash.

```javascript
const contract = new web3.eth.Contract(abi, contractAddress);
const identityHash = '0x...'; // The SHA-256 hash of your identity JSON
const account = '0xYourEthereumAccount';

contract.methods.registerIdentity(identityHash).send({ from: account });
```

||| 6. Verification of Identity
To verify an identity, a user would retrieve the hash from the blockchain and compare it with a hash generated from the identity data provided by the identity holder.

These are simplified steps for educational purposes and don't include aspects such as gas fees for Ethereum, identity verification mechanisms, or smart contract deployment details. Moreover, there's an assumption of some level of familiarity with command-line tools, Ethereum, Solidity, and Web3.js.

For a production environment, the process would need to be significantly more robust, with additional considerations for security, privacy, and compliance with any relevant regulations such as KYC (Know Your Customer) and AML (Anti-Money Laundering). Always consult with blockchain legal experts when handling personal data.

Remember to always keep your private key secure, and never share it or expose it in your applications.