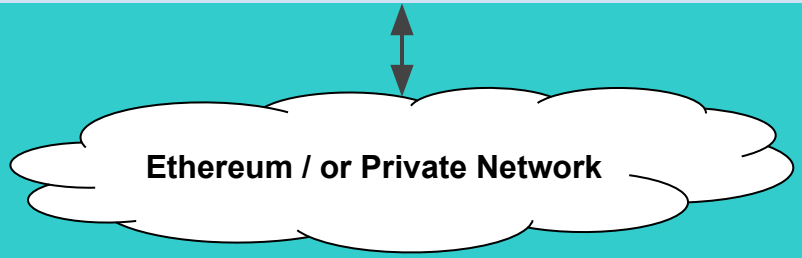
Building Ethereum apps (dapps) with

Blockchain Monterrey

Deploying Smart Contracts

DAPP (using Web3.js)

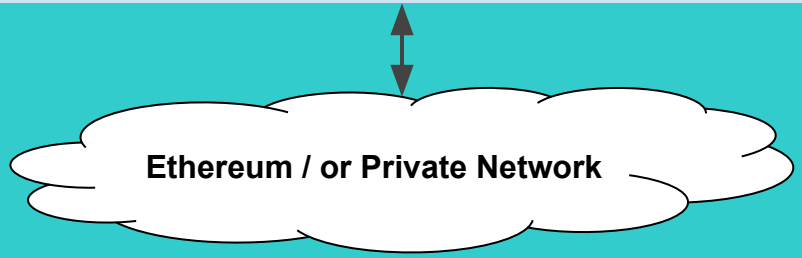
Ethereum Node

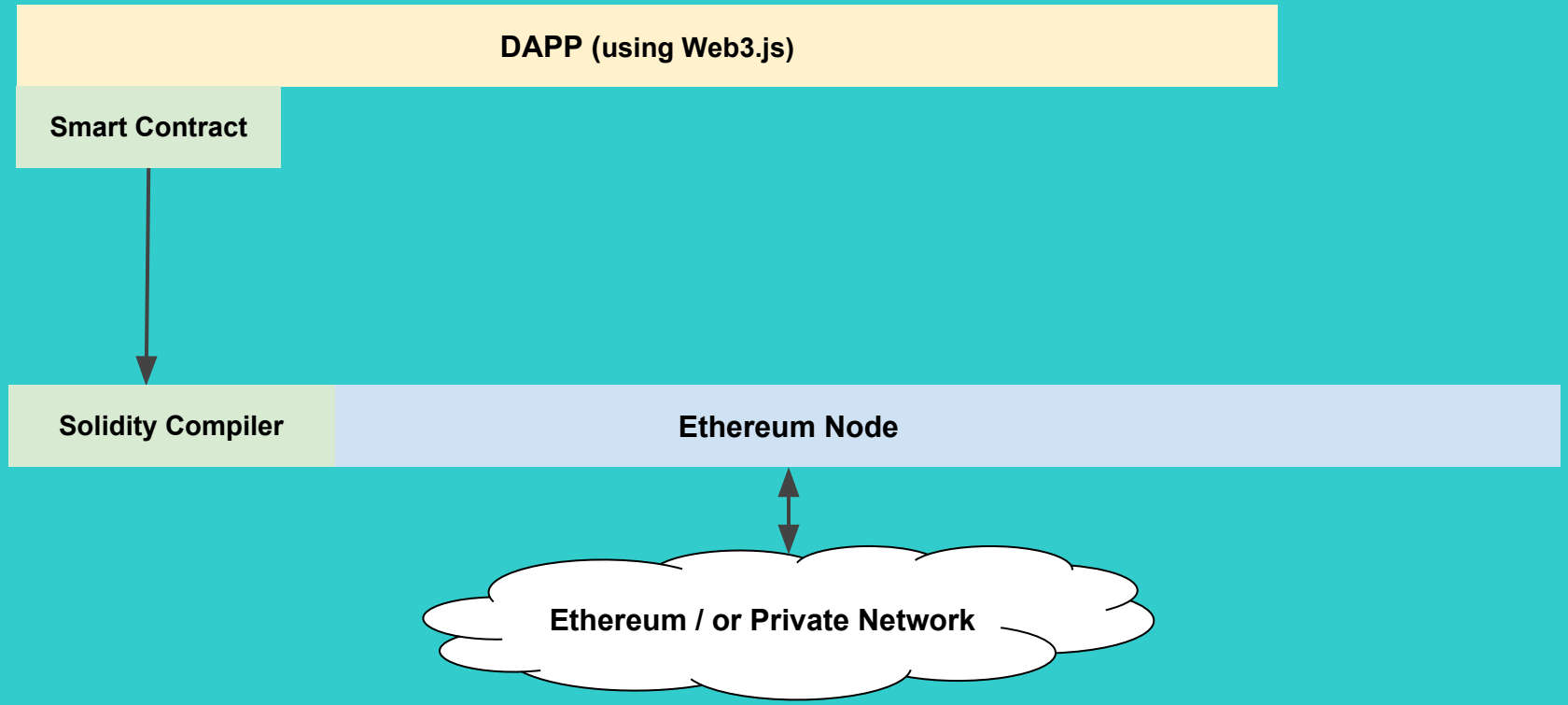


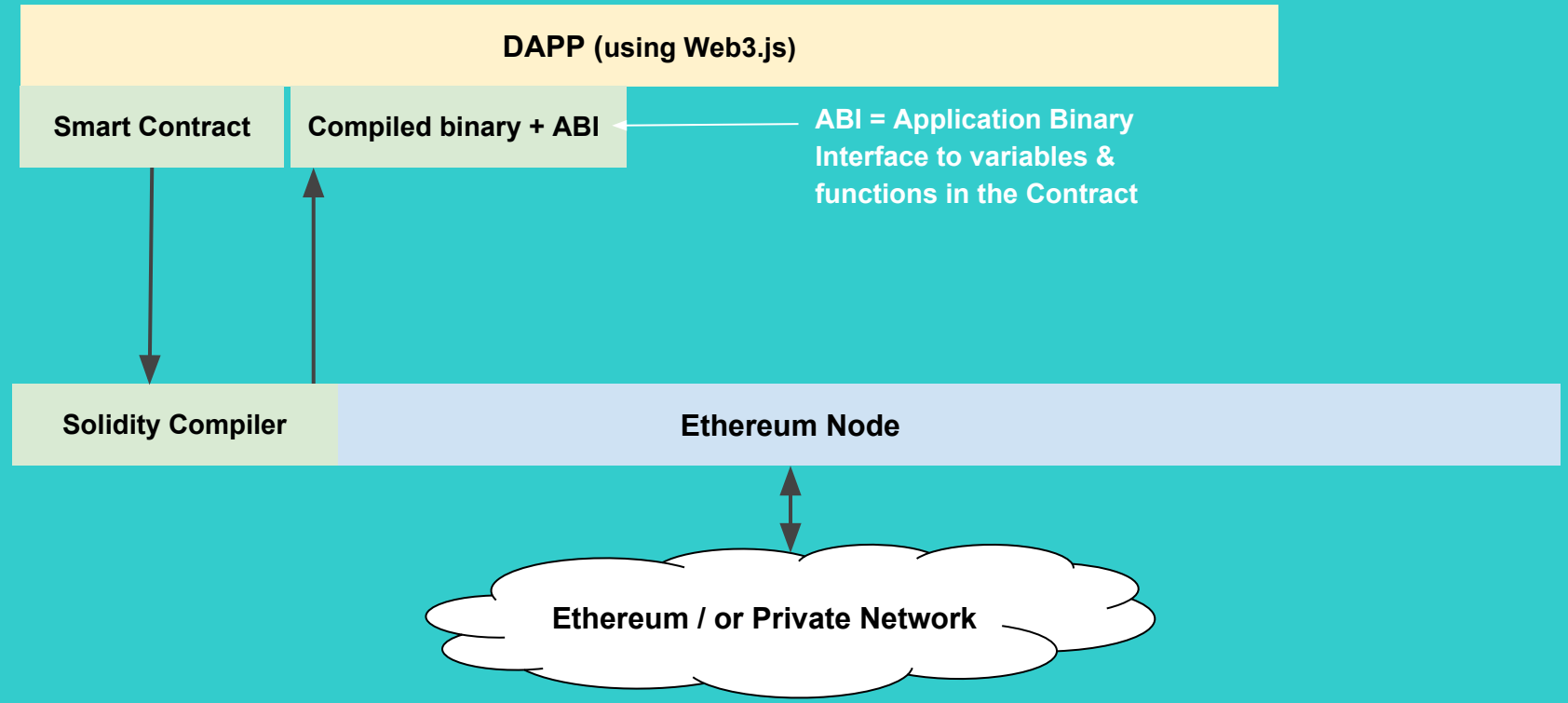
DAPP (using Web3.js)

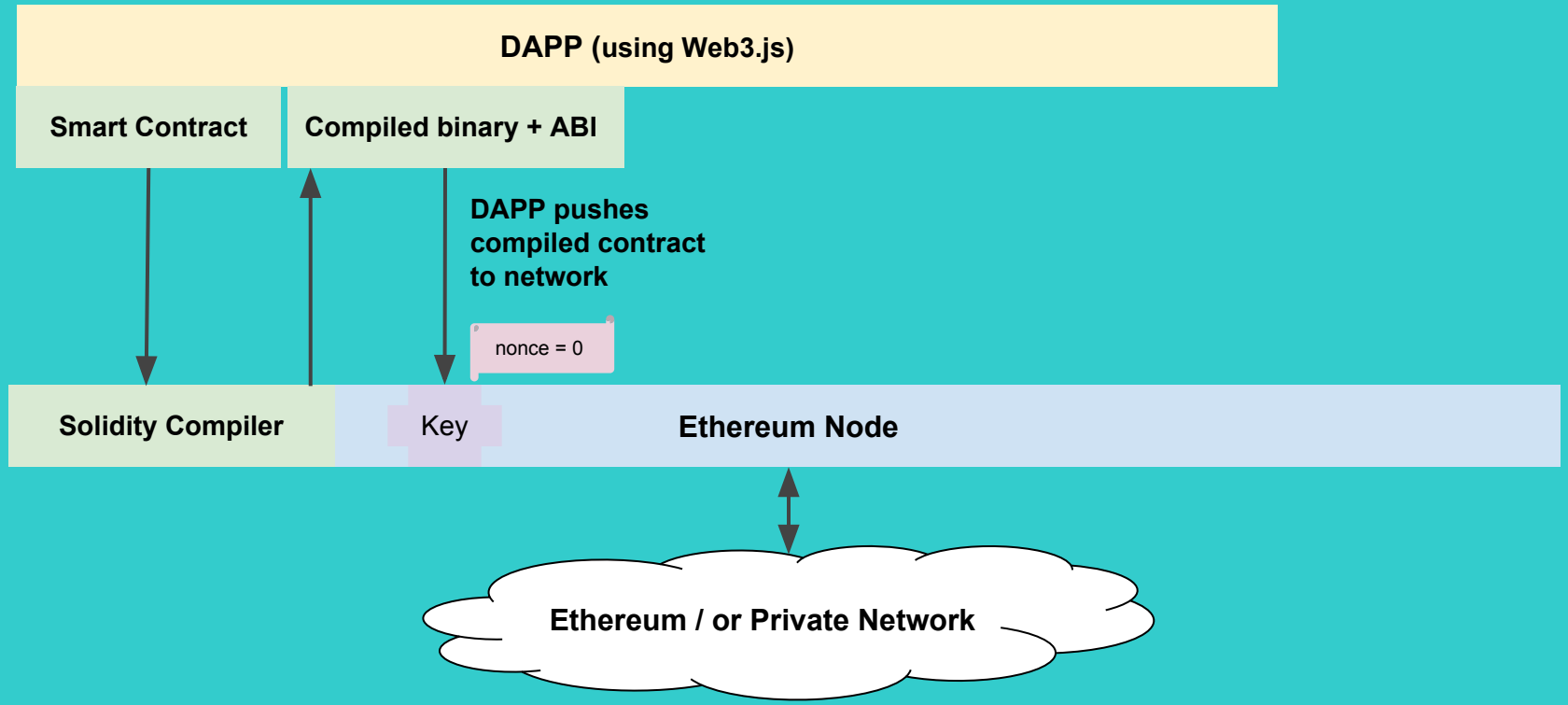
Smart Contract

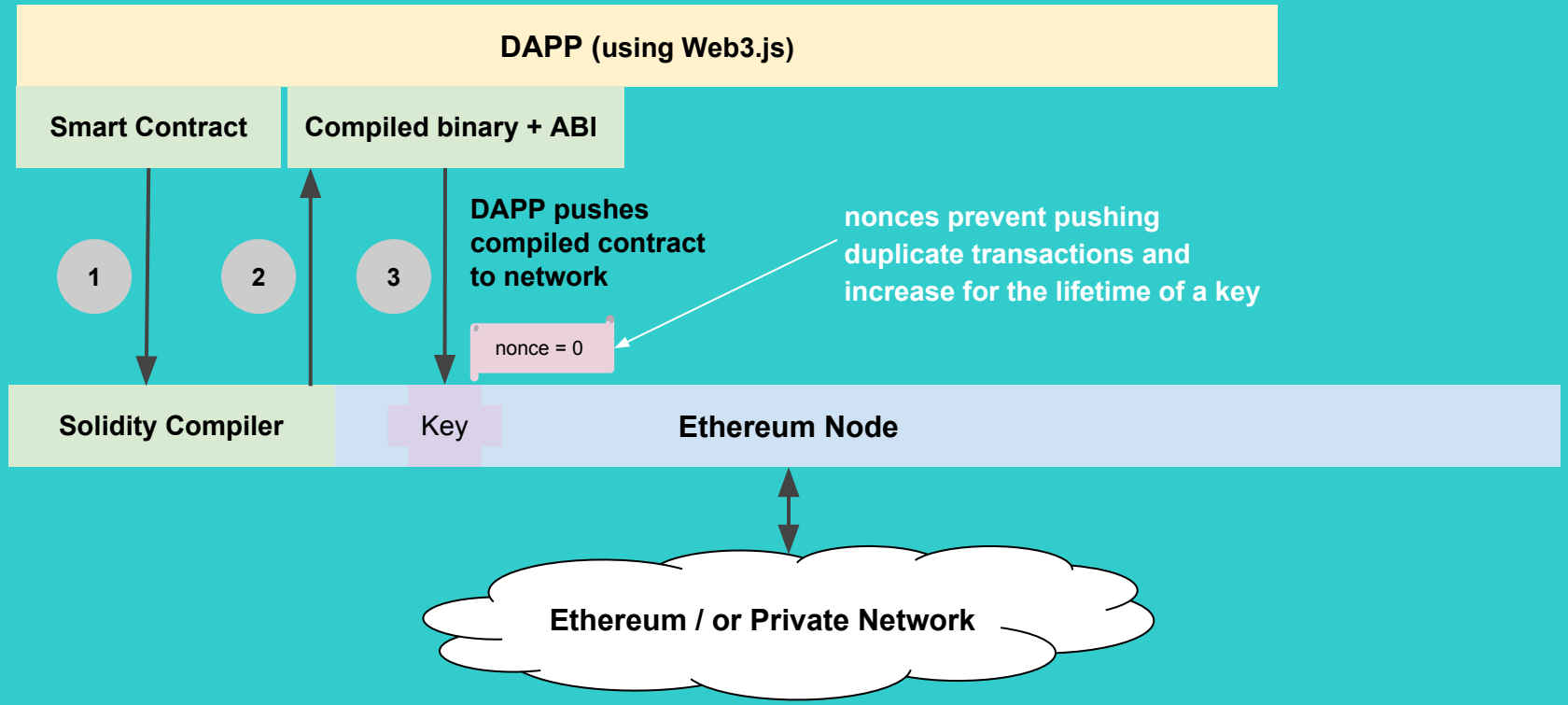
Ethereum Node

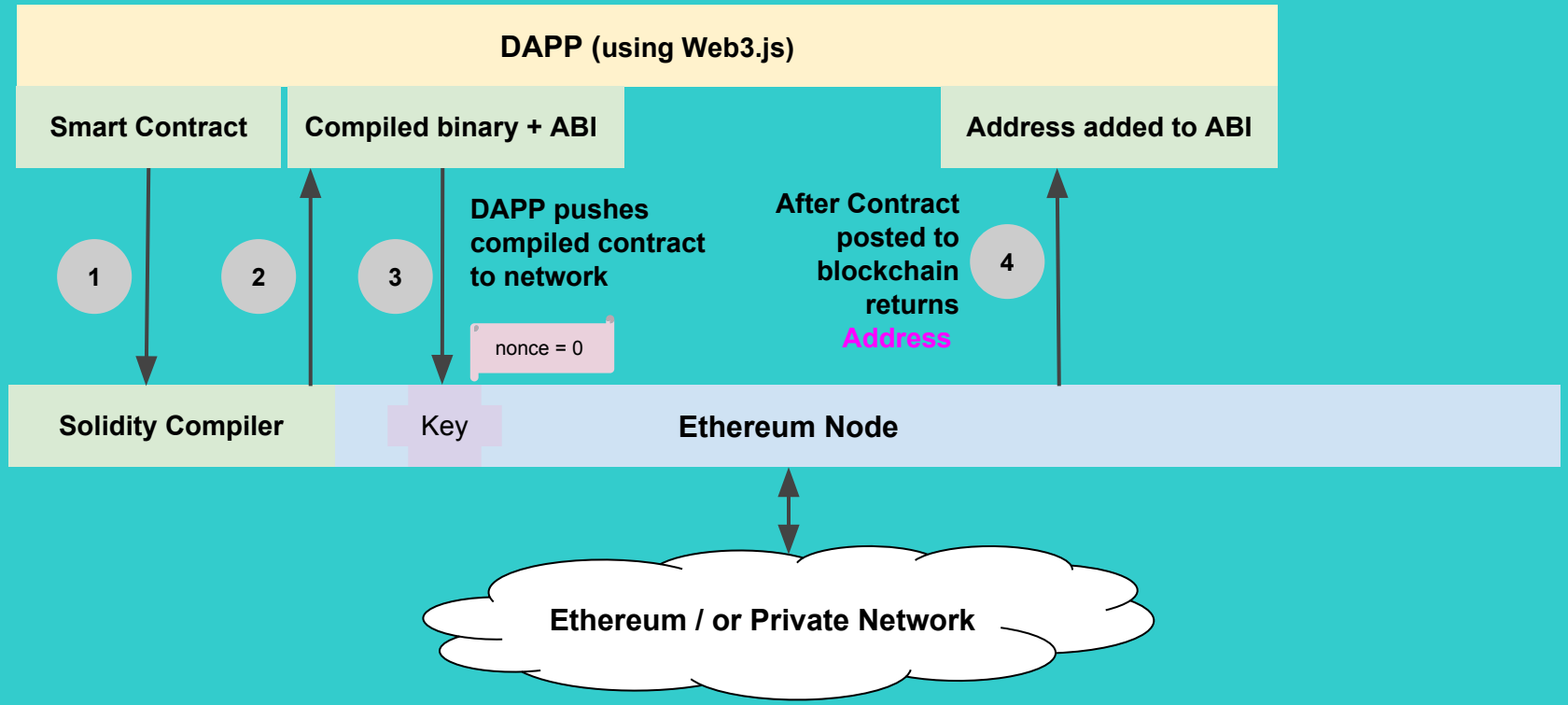



DAPP (using Web3.js)
Smart Contract
Solidity Compiler
Ethereum Node
Ethereum / or Private Network

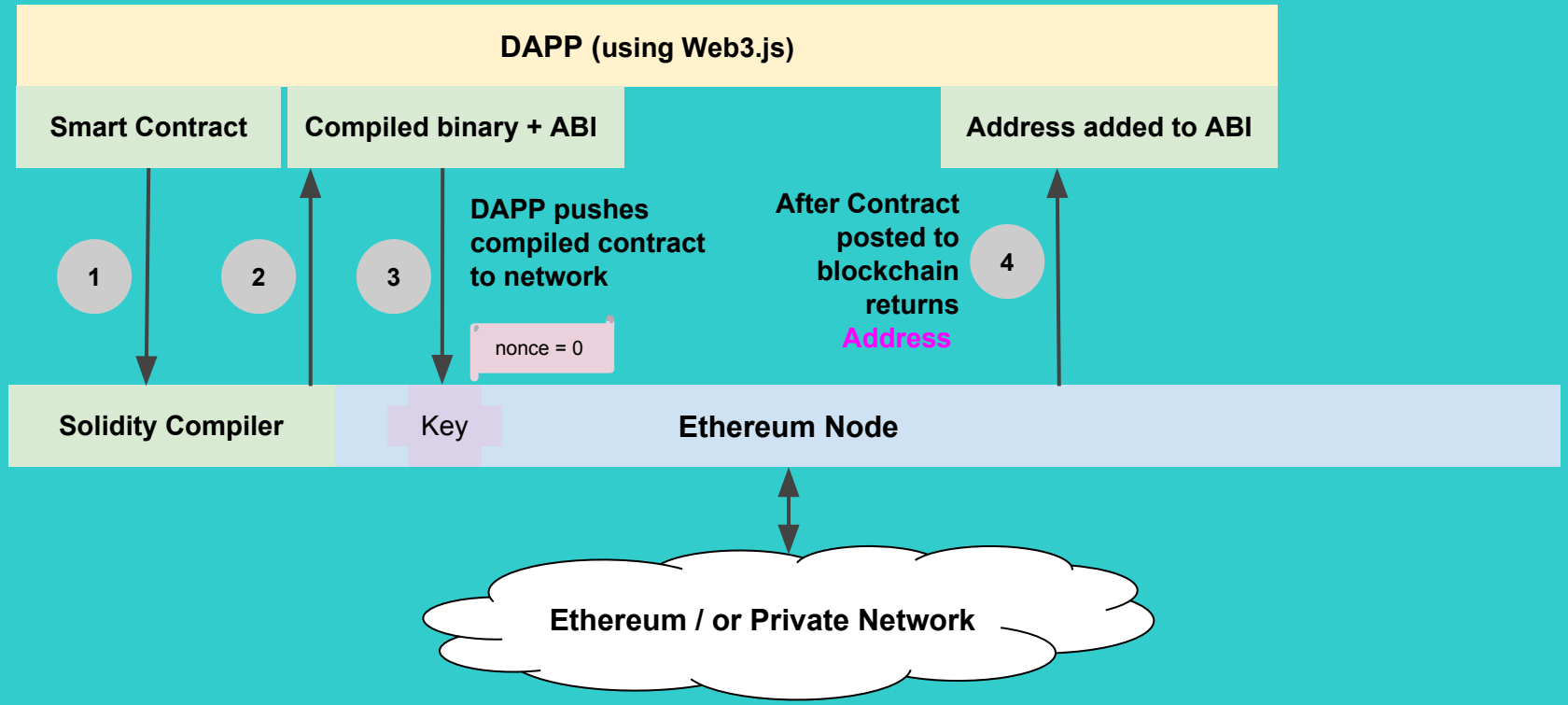

DAPP (using Web3.js)
Smart Contract
Compiled binary + ABI
ABI = Application Binary Interface to variables & functions in the Contract
Solidity Compiler
Ethereum Node
Ethereum / or Private Network

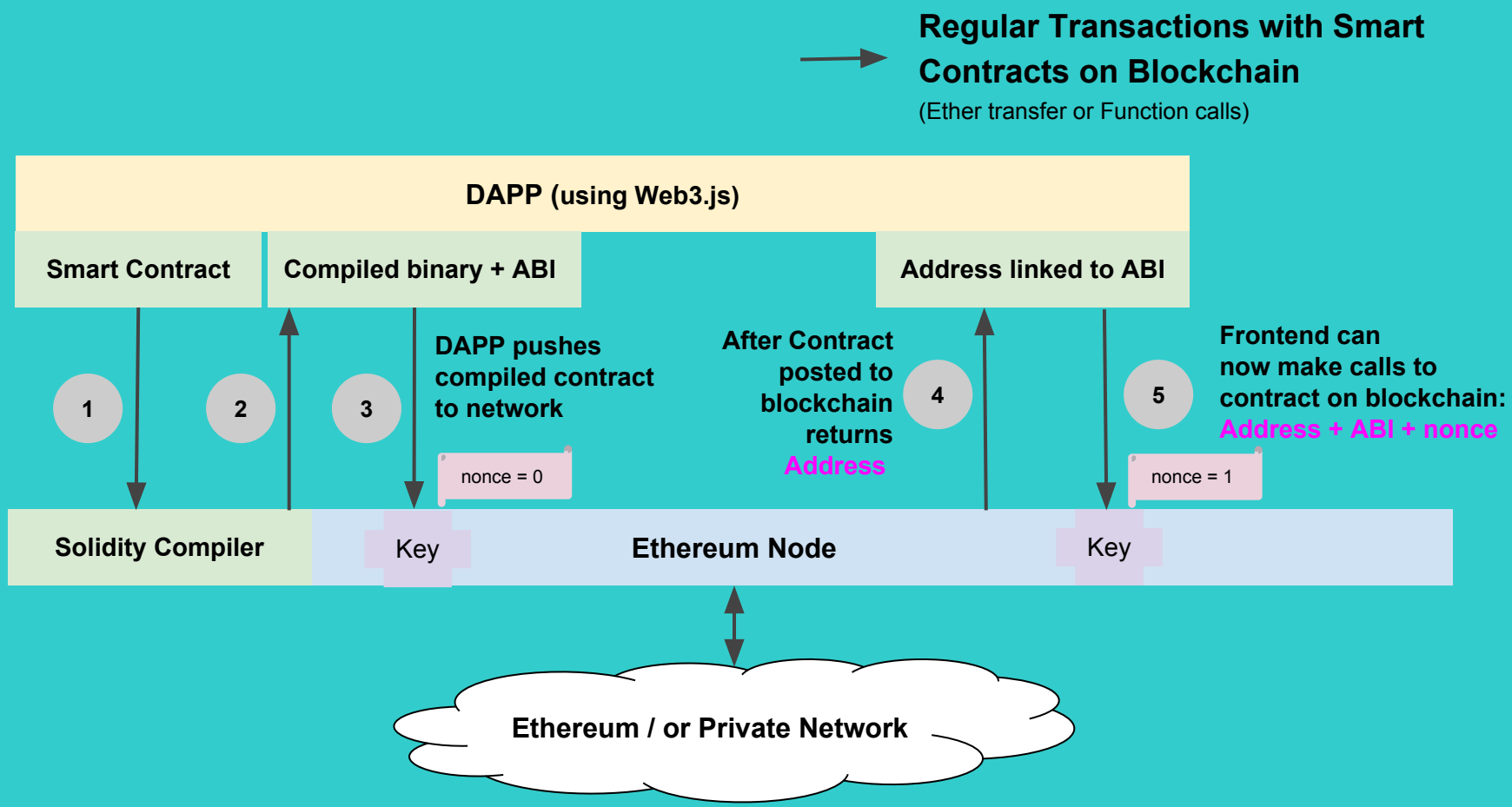

DAPP (using Web3.js)
Smart Contract
Compiled binary + ABI
DAPP pushes
compiled contract
to network
nonce = 0
Solidity Compiler
Key
Ethereum Node
Ethereum / or Private Network


DAPP (using Web3.js)
Smart Contract
Compiled binary + ABI
1
2
3
DAPP pushes compiled contract to network
nonces prevent pushing duplicate transactions and increase for the lifetime of a key
nonce = 0
Solidity Compiler
Key
Ethereum Node
Ethereum / or Private Network


DAPP (using Web3.js)
Smart Contract
Compiled binary + ABI
Address added to ABI
1
2
3
DAPP pushes compiled contract to network
nonce = 0
After Contract posted to blockchain returns Address
4
Solidity Compiler
Key
Ethereum Node
Ethereum / or Private Network

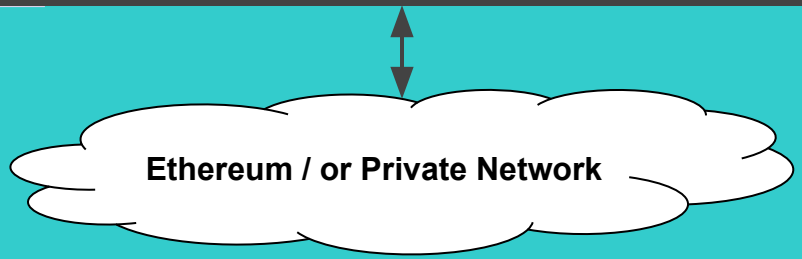
Contract Creation Transaction




Regular Transactions with Smart Contracts on Blockchain
(Ether transfer or Function calls)
DAPP (using Web3.js)
Smart Contract
Compiled binary + ABI
Address linked to ABI
1
2
3
DAPP pushes compiled contract to network
nonce = 0
After Contract posted to blockchain returns Address
4
5
Frontend can now make calls to contract on blockchain: Address + ABI + nonce
nonce = 1
Solidity Compiler
Key
Ethereum Node
Key
Ethereum / or Private Network

DAPP

... is a command-line tool to help you
compile, deploy, link and test
smart contracts
... and build a front-end

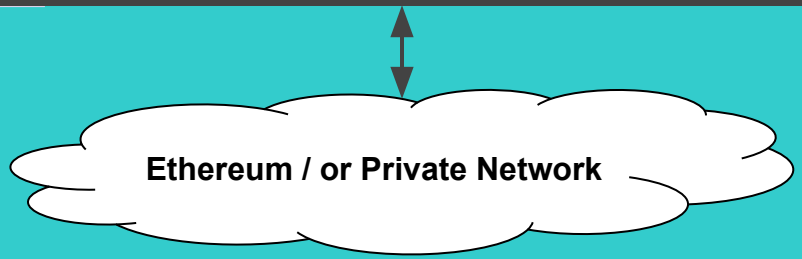


DAPP

... is a command-line tool to help you
compile, deploy, link and test smart contracts
... and build a front-end

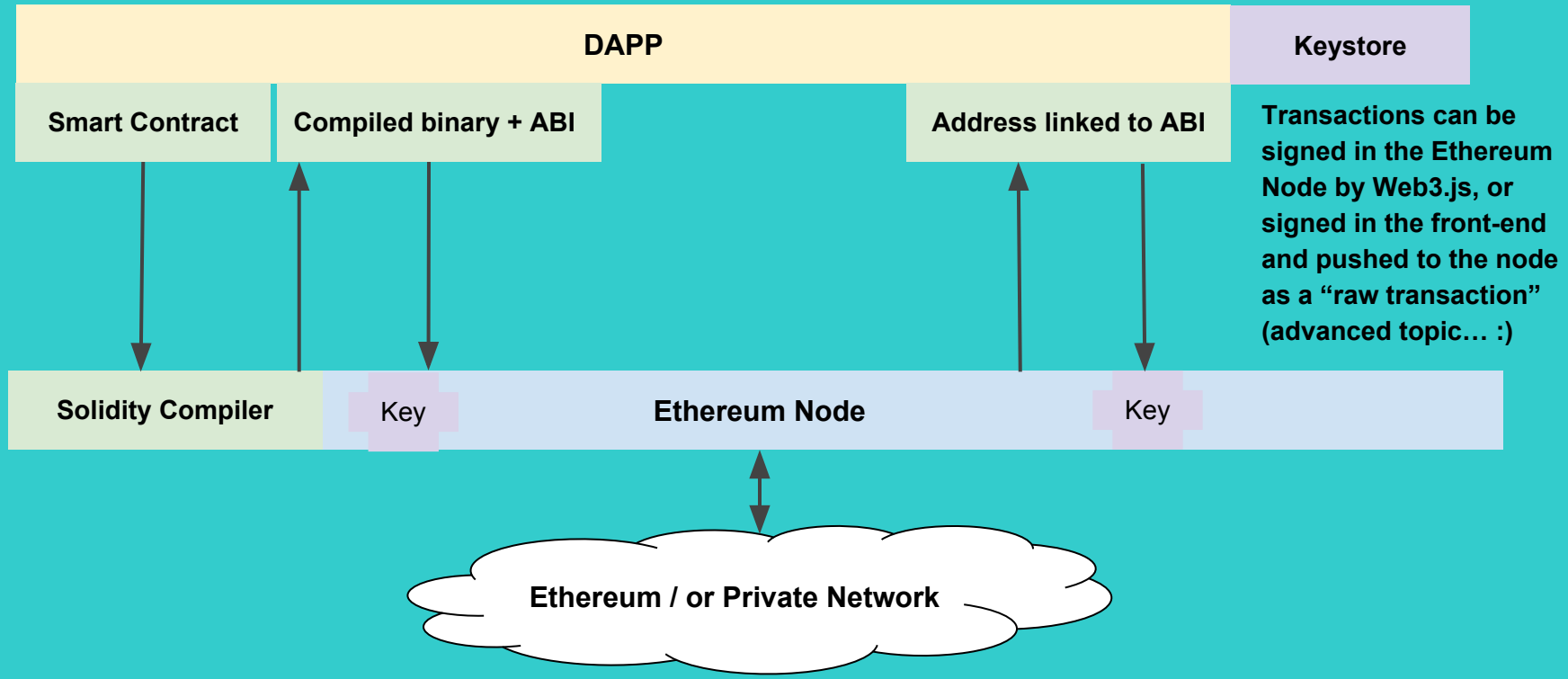
Mocha and Chai

JavaScript, SASS, ES6 and JSX



All Transactions* must be signed with keys and require gas

** any changes to state (writes, etc.)*



Transactions can be signed in the Ethereum Node by Web3.js, or signed in the front-end and pushed to the node as a "raw transaction" (advanced topic… :)

Installing and Using Truffle

$ npm -g install truffle

testrpc

```
$ npm install -g ethereumjs-testrpc
```

solc

https://github.com/ethereum/go-ethereum/wiki/Contract-Tutorial#using-an-online-compiler

New truffle project

```
$ mkdir newproj && cd newproj
$ truffle init
```

Creates this in newproj/

Compile smart contracts (do you have solc installed?)

```
$ truffle compile
```

Deploy smart contract to blockchain

Run an Ethereum node (like testrpc)

```
$ truffle deploy
```

Run truffle tests from test/ directory

```
$ truffle test
```

Run a server from localhost

```
$ truffle serve
```