# Onchain Offchain Programming Model

If you've followed along from theprevious tutorial , you'll have deployed a simple contract which outlines twoonchain andoffchain functions for a Suapp on SUAVE. It should look like this:

// SPDX-License-Identifier: Unlicense pragma

solidity

^ 0.8.8 ;

contract

MyFirstSuapp

{ function

onchain ( )

public

{ }

function

offchain ( )

public

pure

returns

( bytes

memory )

{ / *This is where you will write all your compute-heavy, off-chain logic to be done in a Kettle!* return abi . encodeWithSelector ( this . onchain . selector ) ; } } We'd now like to interact with our deployed contract. Thespell tool we used to deploy the contract can also be used to send confidential compute requests. Using the address your contract was deployed to, you can usespell to call any function you like.

In this case, we'll get our contract to run "offchain" computation (even though we have yet to define exactly what that should be). From the root of yoursuapp directory we first created, run:

suave-geth spell conf-request < your_contract_address

'offchain()' If you builtsuave-geth from source, you may need to specify the full path:

./ < path_to_suave-geth

/build/bin/suave-geth conf-request < your_contract_address

'offchain()' It should print a result like this to your console:

INFO [ 03-26 | 10 :21:30.148 ] Running with local devchain settings INFO [ 03-26 | 10 :21:30.148 ] Contract at address 0xFcdbc6055c5C36dBc326F308ed74fA8cB00771a6 INFO [ 03-26 | 10 :21:30.148 ] Sending offchain confidential compute request kettle = 0xB5fEAfbDD752ad52Afb7e1bD2E40432A485bBB7F INFO [ 03-26 | 10 :21:30.153 ] Hash of the result onchain transaction hash = 0xa38ef6d191f273056ec967f1308752ed48a45376b059c43be3b5d2f47ef5f25f INFO [ 03-26 | 10 :21:30.153 ] Waiting for the transaction to be mined .. . INFO [ 03-26 | 10 :21:30.256 ] Transaction mined status = 1

# blockNum

6 As ouroffchain() function doesn't do anything yet, there isn't much to see here. That said, you have already deployed your first contract, and sent your first confidential request to it! We are well on the way to building our first functional SUAPP and understanding a new paradigm for public blockchains...

## Try to Extend Your Contract

Now, for a Suapp which defines something like an orderflow auction, we want our offchain component to emit information about user trades for searchers to construct backruns, as well as information about successful backruns merged with user trades for builders to construct blocks.

So, let's try add an event and see if we can emit it from ouroffchain() function:

// SPDX-License-Identifier: Unlicense // SPDX-License-Identifier: Unlicense pragma

solidity

^ 0.8.8 ;

contract

MyFirstSuapp

{ event

OffchainEvent ( uint256 num ) ;

function

onchain ( )

public

{ }

function

offchain ( )

public

returns

( bytes

memory )

{ emit

OffchainEvent ( 1 ) ; emit

OffchainEvent ( 2 ) ;

return abi . encodeWithSelector ( this . onchain . selector ) ; } } Now, let's compile and redeploy this contract, and then cast the same spell as we did above to see what happens:

forge build suave-geth spell deploy MyFirstSuapp.sol:MyFirstSuapp suave-geth spell conf-request < your_new_contract_address

'offchain()' The logs printed to your console should still look the same:

INFO [ 03-26 | 10 :30:27.896 ] Running with local devchain settings INFO [ 03-26 | 10 :30:27.896 ] Contract at address 0xB6BbB7dccc69CDc5BCE294aeBeD89684b339E3e4 INFO [ 03-26 | 10 :30:27.896 ] Sending offchain confidential compute request kettle = 0xB5fEAfbDD752ad52Afb7e1bD2E40432A485bBB7F INFO [ 03-26 | 10 :30:27.900 ] Hash of the result onchain transaction hash = 0x3e884e80e455bca7fe0438529729e74e7592c5a2766c811e5751e14e3f98d82a INFO [ 03-26 | 10 :30:27.900 ] Waiting for the transaction to be mined .. . INFO [ 03-26 | 10 :30:28.004 ] Transaction mined status = 1

# blockNum

8 What's going on here? Well, we can't emit the results of offchain computation directly onchain, because that leaks whatever happened in the computation, which is precisely what we want to avoid doing.

## Using SUAVE-STD

If we want the relevant part of the results of our offchain computation to result in things happening onchain, we need to be a little more clever with how we write our contracts.

In order to make this easy, we maintain auseful library calledSUAVE-STD which enables you to do anything from emitting logs (like we're trying to do in this tutorial), to making arbitrary http calls, using Chat GPT in your contracts, encoding and decoding JSON, RLP encoding/decoding transactions, and doing various other commonly useful things.

First, you'll need to add and commit the changes you've made already before we can installSUAVE-STD :

git

add

.

&&

git commit -m

"my first suapp" Now you can install our friendly helper library in yoursuapp directory:

forge install flashbots/suave-std You can import the basicSuapp.sol contract from theSUAVE-STD library to help with emitting logs. This comes along with a modifier calledemitOffchainLogs which will enable you to emit any logs specified in offchain functions from the onchain function selected in thereturn statement. To see this in action, modifyMyFirstSuapp.sol to look like this:

// SPDX-License-Identifier: Unlicense pragma

solidity

^ 0.8.13 ;

import

"suave-std/Suapp.sol" ;

contract

MyFirstSuapp

is Suapp { event

OffchainEvent ( uint256 num ) ;

function

onchain ( )

public emitOffchainLogs { }

function

offchain ( )

public

returns

( bytes

memory )

{ emit

OffchainEvent ( 1 ) ; emit

OffchainEvent ( 2 ) ;

return abi . encodeWithSelector ( this . onchain . selector ) ; } } You'll need to compile, redeploy and cast another spell to your new contract:

forge build suave-geth spell deploy MyFirstSuapp.sol:MyFirstSuapp suave-geth spell conf-request < your_new_contract_address

'offchain()' Now, you should see a result like this printed to your console:

INFO [ 03-26 | 10 :48:38.628 ] Running with local devchain settings INFO [ 03-26 | 10 :48:38.628 ] Contract at address 0x2D09719D6f4fA3AAe6b43328eB150a6902491713 INFO [ 03-26 | 10 :48:38.628 ] Sending offchain confidential compute request kettle = 0xB5fEAfbDD752ad52Afb7e1bD2E40432A485bBB7F INFO [ 03-26 | 10 :48:38.633 ] Hash of the result onchain transaction hash = 0x25965c7866aa81489c0de0ae9928c3f3e7d9e8f85288aca1c9839e672a5bf009 INFO [ 03-26 | 10 :48:38.633 ] Waiting for the transaction to be mined .. . INFO [ 03-26 | 10 :48:38.736 ] Transaction mined status = 1

# blockNum

10 INFO [ 03-26 | 10 :48:38.753 ] Logs emitted in the onchain transaction numLogs = 2 INFO [ 03-26 | 10 :48:38.753 ] Log emitted name = OffchainEvent ( uint256 )

# num

1 INFO [ 03-26 | 10 :48:38.753 ] Log emitted name = OffchainEvent ( uint256 )

# num

2 Congratulations! You have deployed a contract, sent it various confidential requests, and begun learning how to useSUAVE-STD to handle the requirements of programming Suapps with both onchain and offchain components!

Edit this page Previous Deploy a C
Confidential Store