Dec 31, 2022 5 min read

Smart contract creation cost

Updated: Oct 14, 2023

Smart contract creation cost can be anywhere from $10 to $2,000 assuming Ether costs between $1,500 to $2,000. The biggest factors are 1) Ethereum price, 2) the size of the compiled contract (in bytes), 3) the current gas price on the Ethereum network.

There are a total of six components that determine the amount of gas required to deploy the contract. The cost of the gas in dollar terms depends on market conditions and network conditions. We will work out all of these numbers in this article.

If you want to estimate the deployment cost conveniently, use our smart contract deployment cost calculator.

Smart contract deployment cost

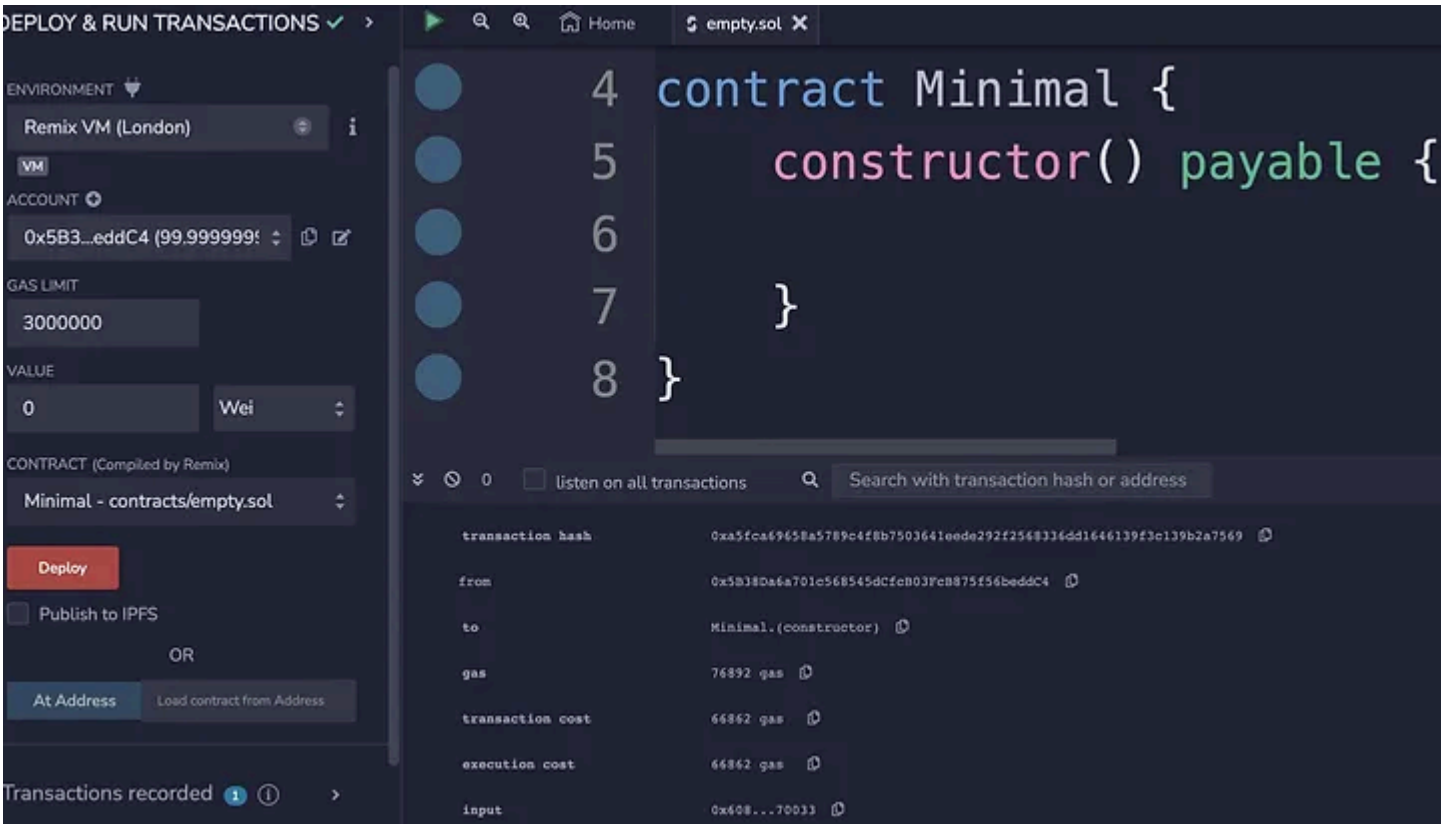
1. **The 21,000 gas that all Ethereum transactions must pay**
2. **A fixed cost of 32,000 gas for creating a new contract**
3. **22,100 for each storage variable set**
4. **4 gas for each zero byte in the transaction data 16 gas for each non-zero byte in the transaction.**
5. **The cost to execute each bytecode during the initialization**
6. **200 gas per byte of deployed bytecode**

Let's use an example of a deploying a minimal Solidity contract in Remix

```
pragma solidity 0.8.7;

contract Minimal {
    constructor() payable {

    }
}
```



smart contract creation gas cost

Note that the deployment cost according to remix was 66,862. We will break down this cost in this article.

We have made the constructor payable and set the optimizer to 200 runs. This has the effect of making the smart contract smaller.

Let's add it up

```
21,000 gas | deployment
32,000 gas | creation
Total: 53,000
```

We still have 13,862 gas to account for

Transaction bytecode gas cost (tx.data)

The transaction bytecode was

```
0x
6080604052603f806011600039600f3fe6080604052600080fdfea2646970667
```

```
358221220c5cad0aa1e64e2ca6a6cdf28a25255a8ebbf3cdd5ea0b8e4129a3c83
c4fbb72a64736f6c63430008070033
```

Each hex par is a byte, so let's add spaces to make it more readable.
To split it up like that, we can use the following python code

```
import itertools

# note that we manually removed the "0x" at the beginning
s =
"6080604052603f8060116000396000f3fe6080604052600080fdfea264697066
7358221220c5cad0aa1e64e2ca6a6cdf28a25255a8ebbf3cdd5ea0b8e4129a3c8
3c4fbb72a64736f6c63430008070033"
s = " ".join(["".join(group) for group in
itertools.zip_longest(s[::2], s[1::2])])

print(s)
```

We get

```
60 80 60 40 52 60 3f 80 60 11 60 00 39 60 00 f3 fe 60 80 60 40 52
60 00 80 fd fe a2 64 69 70 66 73 58 22 12 20 c5 ca d0 aa 1e 64 e2
ca 6a 6c df 28 a2 52 55 a8 eb bf 3c dd 5e a0 b8 e4 12 9a 3c 83 c4
fb b7 2a 64 73 6f 6c 63 43 00 08 07 00 33
```

Each non-zero byte costs 16 gas, and each zero byte (00) costs 4 gas.
To count them, we can use this python one-liner:

```
s = "60 80 60 40 52 60 3f 80 60 11 60 00 39 60 00 f3 fe 60 80 60
40 52 60 00 80 fd fe a2 64 69 70 66 73 58 22 12 20 c5 ca d0 aa 1e
64 e2 ca 6a 6c df 28 a2 52 55 a8 eb bf 3c dd 5e a0 b8 e4 12 9a 3c
83 c4 fb b7 2a 64 73 6f 6c 63 43 00 08 07 00 33"

# non-zero bytes
print(len(list(filter(lambda x: x != '00', s.split(' ')))))

# zero bytes
print(len(list(filter(lambda x: x == '00', s.split(' ')))))
```

We have 75 non-zero bytes and 5 zero bytes. The math works out to 75 x 16 + 5 x 4 = 1220 gas

```
21,000 gas | deployment
32,000 gas | creation
 1,220 gas | bytecode cost
Total: 54,220 gas.
```

We have 12,642 gas to account for to bring the total cost to 66,862.

Deployment code

Let's look at the bytecode again

60 80 60 40 52 60 3f 80 60 11 60 00 39 60 00 f3 fe 60 80 60 40 52 60 00 80 fd fe a2 64 69 70 66 73 58 22 12 20 c5 ca d0 aa 1e 64 e2 ca 6a 6c df 28 a2 52 55 a8 eb bf 3c dd 5e a0 b8 e4 12 9a 3c 83 c4 fb b7 2a 64 73 6f 6c 63 43 00 08 07 00 33

The parts in bold are the deployment code. The first part is the initialization code. We need to multiply each of the deployment code by 200 gas to get the cost. This has a higher cost than the bytecode cost above because this is stored in the Ethereum state.

Let's do that in python again

```
deployment_code = '60 80 60 40 52 60 00 80 fd fe a2 64 69 70 66
73 58 22 12 20 c5 ca d0 aa 1e 64 e2 ca 6a 6c df 28 a2 52 55 a8 eb
bf 3c dd 5e a0 b8 e4 12 9a 3c 83 c4 fb b7 2a 64 73 6f 6c 63 43 00
08 07 00 33'

print(len(deployment_code.split(' ')))
# 63
```

63 x 200 = 12,600 gas
So here is the breakdown so far

```
21,000 gas | deployment
32,000 gas | creation
 1,220 gas | bytecode cost
12,600 gas | deployed bytecode
Total: 66,820
```

Almost there! We are 42 gas short of our 66,862 target.

Bytecode execution gas cost

We also need to factor in the actual execution of the initialization bytecode.

60 80 60 40 52 60 3f 80 60 11 60 00 39 60 00 f3 fe

We can translate that to a more convenient format using the evm playground tool.

```
PUSH1 0x80 | 3 gas
PUSH1 0x40 | 3 gas
MSTORE     | 12 gas
PUSH1 0x3f | 3 gas
DUP1       | 3 gas
PUSH1 0x11 | 3 gas
PUSH1 0x00 | 3 gas
CODECOPY   | 9 gas
PUSH1 0x00 | 3 gas
RETURN     | 0 gas
INVALID    | not executed
```

And the total is 42, as expected. These gas costs were obtained by running the remix debugger.

smart contract deployment gas debugging

And we are done, we have accounted for each component of the deployment of a smart contract.

So here is the final breakdown

```
21,000 gas | deployment
32,000 gas | creation
 1,220 gas | bytecode cost
12,600 gas | deployed bytecode
    42 gas | deployment execution cost

Total: 66,862 gas
```

Note that if we set storage variables in the constructor, the cost would be higher. We'd have to pay 22,100 gas for each variable set. But to keep this walkthrough manageable, we have omitted that case.

If you want to become a solidity ninja and be able to account for gas costs fluently, sign up for our Solidity coding bootcamp, which is the first blockchain bootcamp in our series of web3 bootcamps. See our article on solidity gas optimization to learn how to reduce deployment costs. All the numbers for gas cost obtained here are from the Ethereum Yellow Paper.

Translating to dollars

To turn units of "gas" into dollars, the formula is

gas x gas per gwei x price of ether ÷ 1 billion.

Gas per gwei can be obtain from sites like ethgasstation or etherscan. In our example, assuming Eth costs $1,000 and the price of gas is 20 gwei, the cost in dollars would be

66,862 x 20 x 1000 ÷ 1 billion = $1.34

Again, our calculator is here: smart contract deployment cost calculator