

Adding a puzzle

We're going to make a new puzzle, which means we need to provide the smart contract with a set of clues and info about the answers.

Of course, we'll not be sending the answers to the smart contract, otherwise everyone could see. We will, however, send details about each clue, including:

- The clue number
- Whether it's a down or across clue
- The coordinates (x and y position)
- The length of the clue. (How many letters)

Essentially, we're going to tell the smart contract enough information for an empty puzzle like this:

(Note that we aren't showing the human-readable clues in the above screenshot, but we will provide that as well.)

Building and deploying

Let's use the same steps we learned from the first chapter:

Art by [herogranada.near](#)

Navigate to the `contract` directory, then run the build script for your system:

```
./build.sh
```

If following from the previous chapter, you'll likely have a subaccount already created. For the purpose of demonstration, we're calling the subaccount (where we deploy the contract) `crossword.friend.testnet` and the parent account is thus `friend.testnet`.

Let's delete the subaccount and recreate it, to start from a blank slate.

Art by [3one9.near](#)

Here's how to delete and recreate the subaccount using NEAR CLI:

Delete the subaccount and send remaining balance to `friend.testnet`

```
near delete crossword.friend.testnet friend.testnet
```

Create the subaccount again

```
near create-account crossword.friend.testnet --masterAccount friend.testnet
```

Deploy, calling the "new" method with the parameter for `owner_id`

```
near deploy crossword.friend.testnet --wasmFile res/crossword_tutorial_chapter_2.wasm --initFunction new --initArgs '{"owner_id": "crossword.friend.testnet"}'
```

Now we're ready to construct our new crossword puzzle and add it via the `new_puzzle` method. Let's start with the clues for this new puzzle.

The clues

We're going to use these clues below for our improved puzzle. The `Answer` column will not get sent to the smart contract when we call `new_puzzle`.

Number Answer Clue (x, y) coords length 1 paras NFT market on NEAR that specializes in cards and comics. (1, 1) 5 2 rainbowbridge You can move assets between NEAR and different chains, including Ethereum, by visiting _____.app (0, 2) 13 3 mintbase NFT market on NEAR with art, physical items, tickets, and more. (9, 1) 8 4 yoctonear The smallest denomination of the native token on NEAR. (3, 8) 9 5 cli You typically deploy a smart contract with the NEAR ____ tool. (5, 8) 3 The x and y coordinates have their origin in the upper-left side of the puzzle grid, and each row and column start at 0.

Solution hash

Let's derive the sha256 hash using an [easy online tool](#) (there are many other offline methods as well) to discover the solution hash:

d1a5cf9ad1adefe0528f7d31866cf901e665745ff172b96892693769ad284010

Add the puzzle

Add a new puzzle using NEAR CLI with this long command, replacing `crossword.friend.testnet` with your subaccount:

```
near call crossword.friend.testnet new_puzzle '{ "solution_hash":  
"d1a5cf9ad1adefe0528f7d31866cf901e665745ff172b96892693769ad284010", "answers": [ { "num": 1, "start": { "x": 1, "y": 1  
, "direction": "Down", "length": 5, "clue": "NFT market on NEAR that specializes in cards and comics." }, { "num": 2, "start": {  
"x": 0, "y": 2 }, "direction": "Across", "length": 13, "clue": "You can move assets between NEAR and different chains,  
including Ethereum, by visiting .app" }, { "num": 3, "start": { "x": 9, "y": 1 }, "direction": "Down", "length": 8, "clue": "NFT  
market on NEAR with art, physical items, tickets, and more." }, { "num": 4, "start": { "x": 3, "y": 8 }, "direction": "Across",  
"length": 9, "clue": "The smallest denomination of the native token on NEAR." }, { "num": 5, "start": { "x": 5, "y": 8 },  
"direction": "Down", "length": 3, "clue": "You typically deploy a smart contract with the NEAR     tool." } ] }' --accountId  
crossword.friend.testnet Note that our contract name and the account we're calling this from are both crossword.friend.testnet  
. That's because we added a check at the top of new_puzzle to make sure the predecessor is the owner_id.
```

Now our smart contract has information about this second crossword puzzle.

Let's explore how to make our frontend have a login button and truly turn this into a decentralized app (dApp) [Edit this page](#)
Last updated on Oct 5, 2023 by omahs Was this page helpful? Yes No

[Previous Actions and sending NEAR](#) [Next Access keys and login 1/2](#)