XRP LEDGER

Band Oracle Workshop

What is an Oracle?

- Oracles connect blockchain with external data.
- They ensure accurate, tamper-proof information.
- Use Cases:
  - Price feeds,
  - Weather data
  - Event outcomes
  - Etc

This workshop we will explore Band Oracle

- For this workshop, we'll focus on price feed oracles, specifically Band Oracle.
- Band Oracle provides reliable, scalable, and cross-chain price data.
- Band Oracle is currently deployed on the XRPL EVM Devnet and is providing limited price feed data.
- This can enable the development of multiple defi primitives that are dependent on outside price data.

Go to the Band Oracle contract

- Explorer: https://explorer.xrplevm.org/
- CA: 0xdE2022A8aB68AE86B0CD3Ba5EFa10AaB859d0293
- Click on the Contract tab
- Click on Read Contract
- We will be working with the getReferenceData and getReferenceDataBulk functions.
- Currently the only supported price feeds on devnet are XRP, BTC & ETH

Example: Get a single price feed

7. getReferenceData

_base* (string)

XRP

_quote* (string)

USD

Read

r (tuple)

```
[ getReferenceData method response ]
[
  r (tuple[uint256,uint256,uint256]) :
      (uint256) : 499790972828598700
      (uint256) : 1723045450
      (uint256) : 1724278373
]
```

Example: Get multiple price feeds at once

8. getReferenceDataBulk

_bases* (string[])

BTC

XRP

ETH

_quotes* (string[])

USD

USD

USD

Read

tuple[]

```
[ getReferenceDataBulk method response ]
[
  struct IStdReference.ReferenceData[] (tuple[uint256,uint256,uint256][]) :
      (uint256) : 5.6019399896237565e+22,1723045450,1724278303
      (uint256) : 499790972828598700,1723045450,1724278303
      (uint256[]) : 2.399248544722519e+21,1723045450,1724278303
]
```

Band Oracle
Integration

Setting up

- We will be working with Foundry
  - Please visit the foundry docs for installation instructions
  - https://book.getfoundry.sh/getting-started/installation
- Clone this repo: https://github.com/hazardcookie/Band-Oracle-Foundry-Workshop
- Create a fresh metamask wallet and
- Connect to the XRPL EVM Sidechain Devnet
- Fund the wallet with devnet XRP and bridge to the sidechain
- Export the private key from metamask and save it in a .env file in your repo

Metamask Setup

Setting up Metamask

Add a custom network using the details below:

Network name

XRPL EVM Sidechain Devnet

New RPC URL

https://rpc-evm-sidechain.xrpl.org/

Chain ID

1440002

Currency symbol

XRP

Ticker symbol verification data is currently unavailable, make sure that the symbol you have entered is correct. It will impact the conversion rates that you see for this network

Block explorer URL (Optional)

https://explorer.xrplevm.org/

Cancel | Save

Getting Devnet XRP

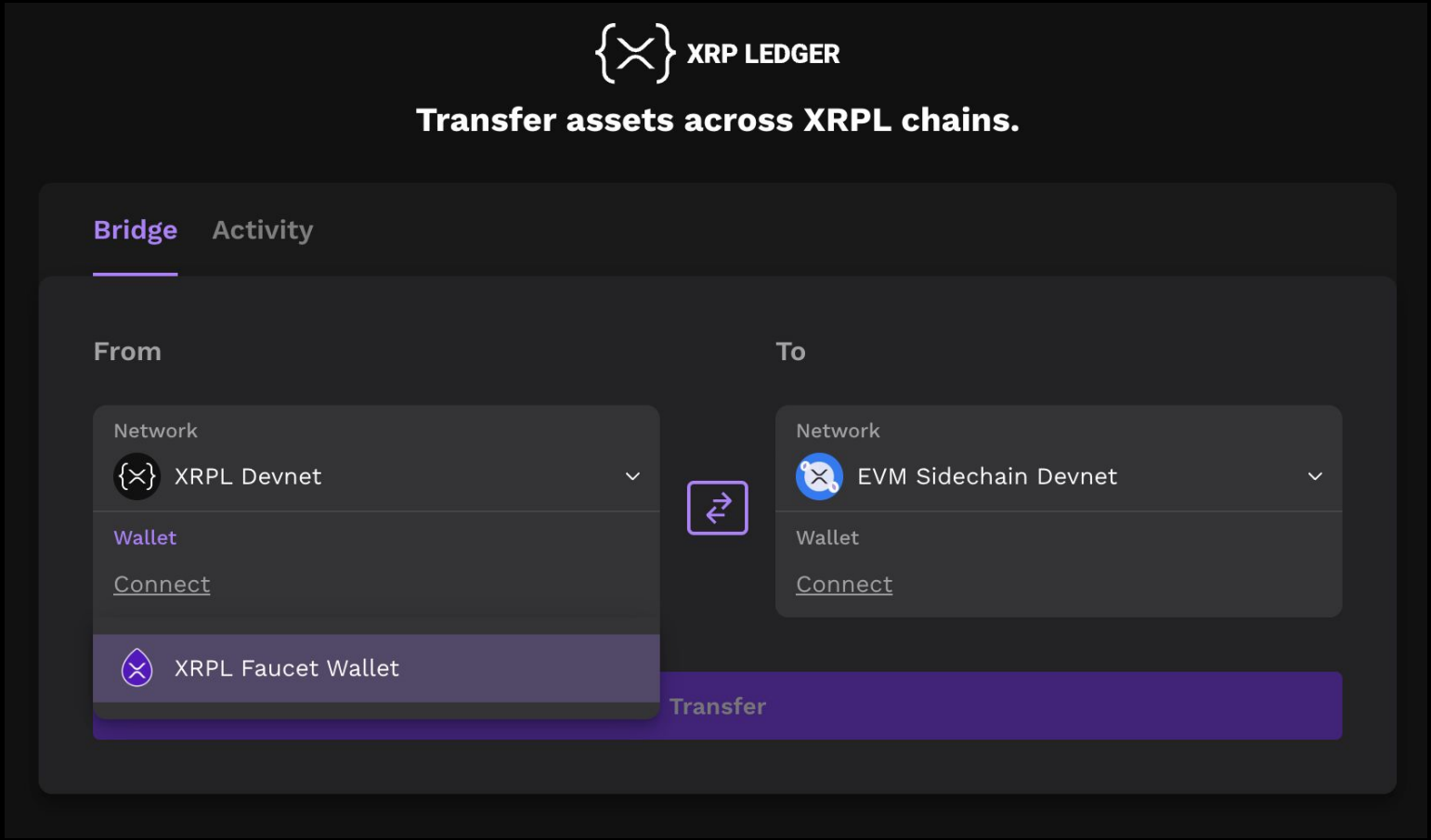
Getting Devnet XRP Checklist

- Go to the combined bridge & faucet website:

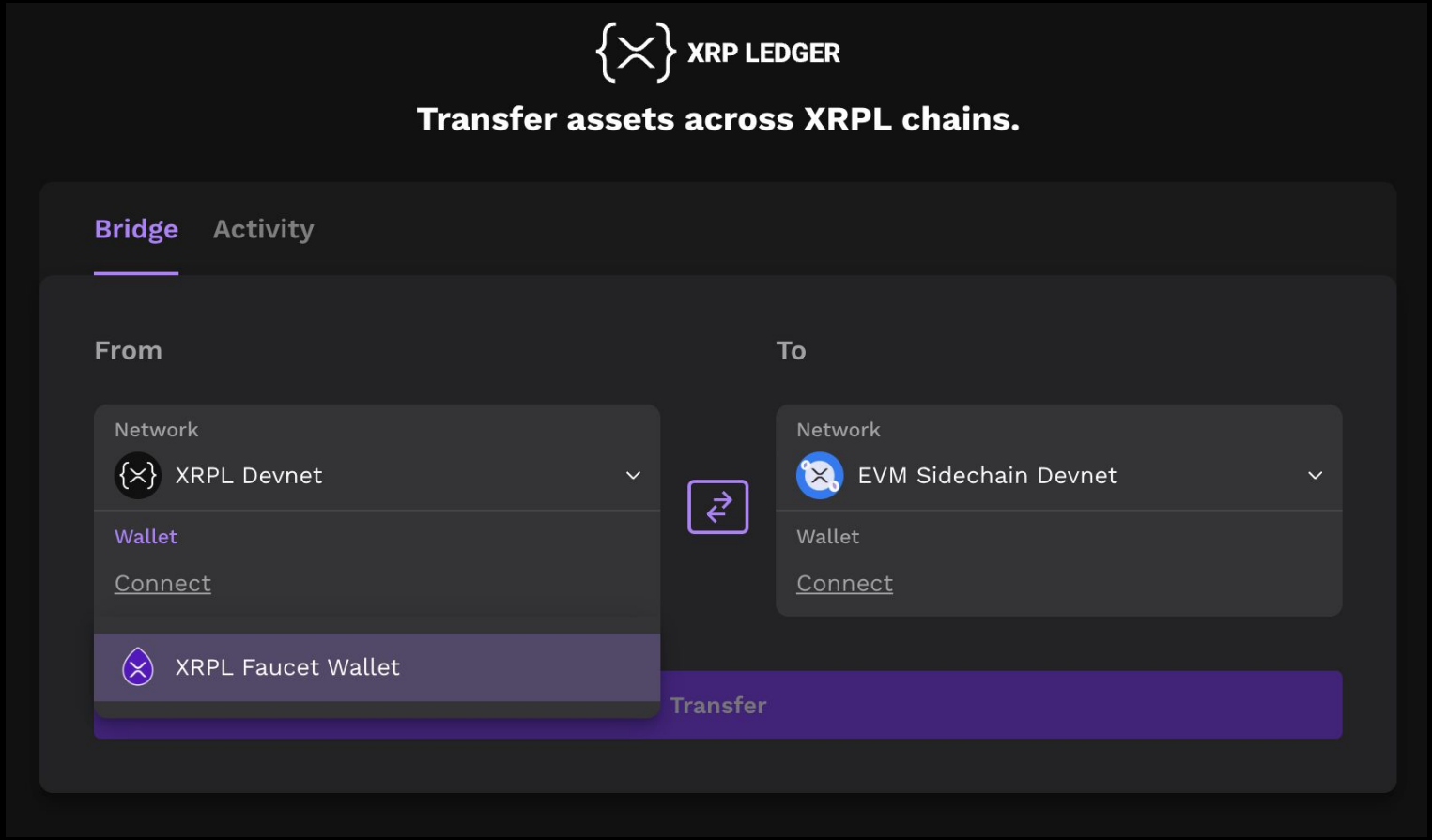
  https://bridge.xrplevm.org

- Ensure Metamask is installed
- Create a fresh Metamask account
  - We will be extracting private keys later on
  - Do not use a wallet that contains real funds
- Fund the XRPL Devnet faucet wallet
- Connect the Metamask wallet
- Bridge XRP from the faucet wallet to the Metamask wallet

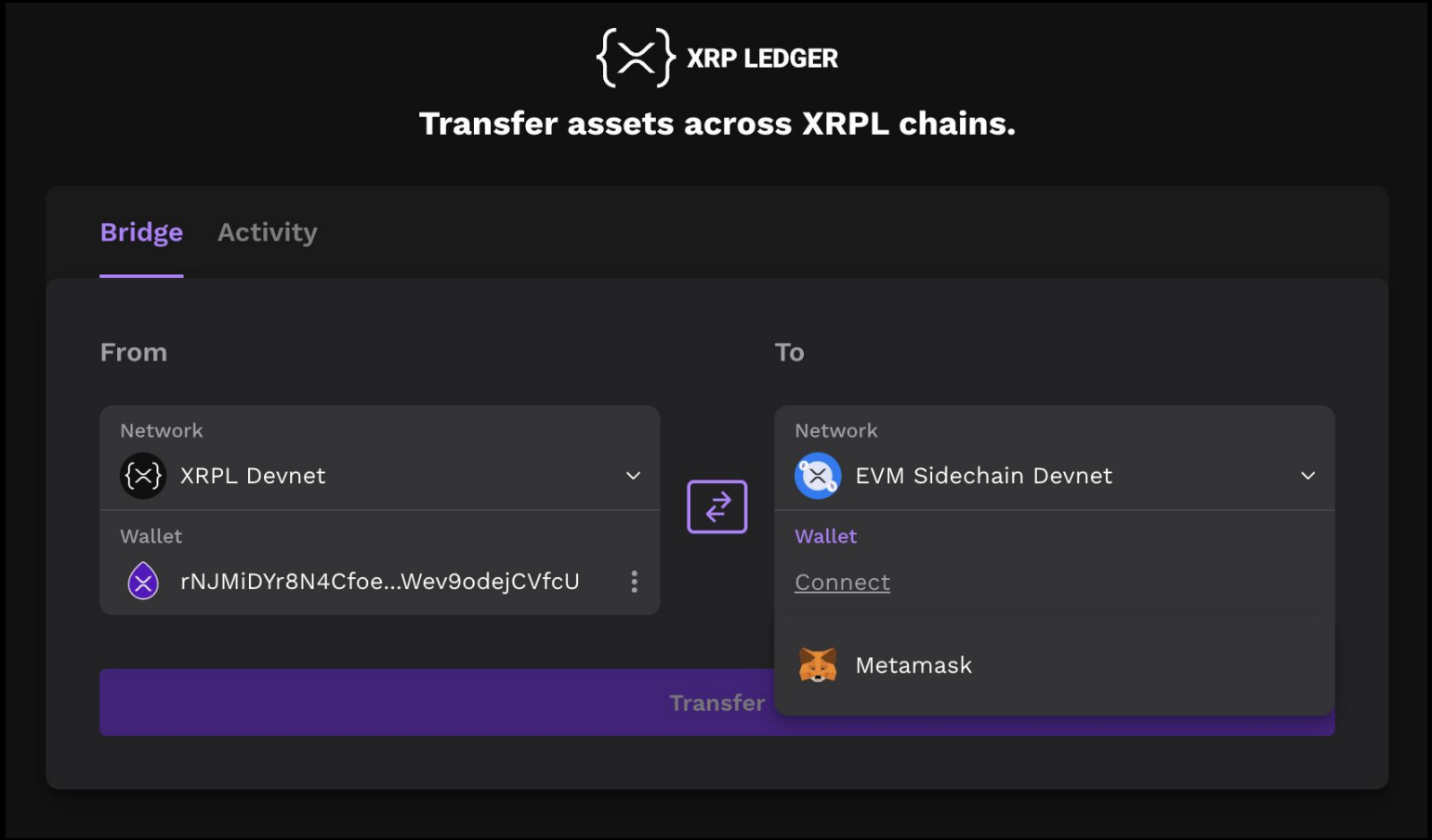
Click on *Connect* - Then press XRPL Faucet Wallet



Click on *Connect* - Then press XRPL Faucet Wallet



Click on *Connect* - Then press Metamask



Bridge over some XRP

XRP LEDGER

Transfer assets across XRPL chains.

Bridge | Activity

From

Network
XRPL Devnet

Wallet
rNJMiDYr8N4Cfoe...Wev9odejCVfcU

To

Network
EVM Sidechain Devnet

Wallet
0x37604322a9D88...95eB00054D81d

You send

Amount
85 XRP

Send max: 85 XRP

You receive

Amount
85 XRP

Bridge Transfer Fee	~ 5 XRP
Estimated time of arrival	~ 30 seconds - 3 minutes

Transfer

Bridge over some XRP

XRP LEDGER

Transfer assets across XRPL chains.

Your Transaction has been sent

Origin transaction hash

E0AA32DFFFD374B21630BDDCE9863AA5987C37FDCAED29ADCE537302E22AE5FF

From Address

XRPL Devnet rNJMiDYr8N4CfoeUEmoimWev9odejCVfcU

To Address

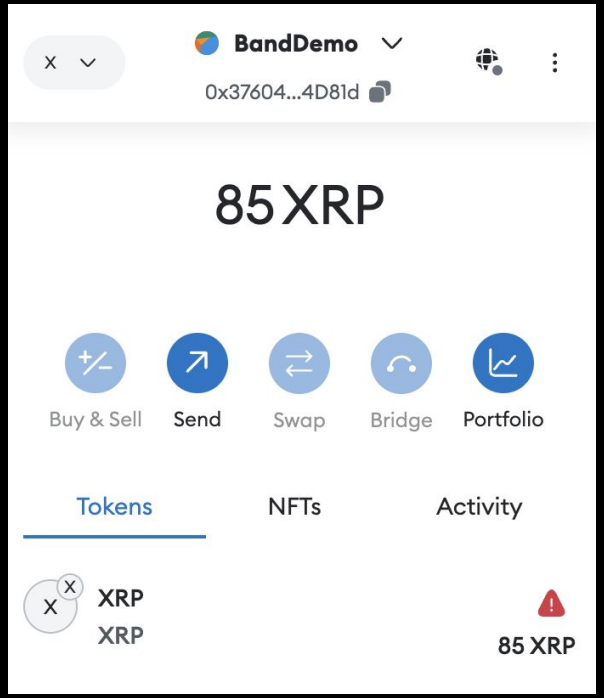
EVM Sidechain Devnet 0x37604322a9D88B13D614E35DF3995eB00054D81d

Receive

85 **XRP**

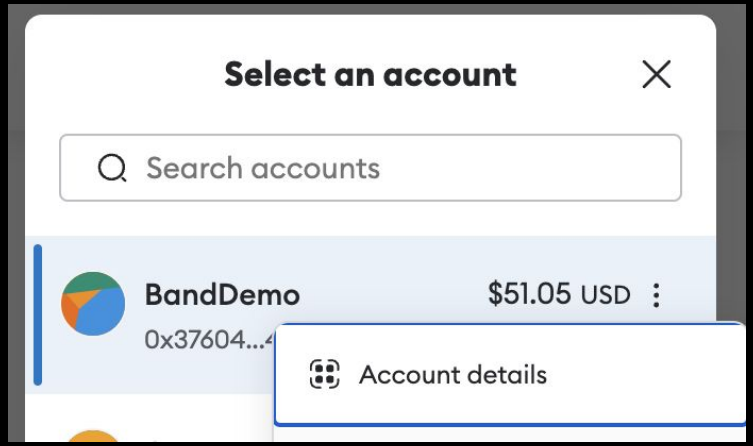
Done

Soon you should see your XRP in your MetaMask Wallet



Extracting the private key

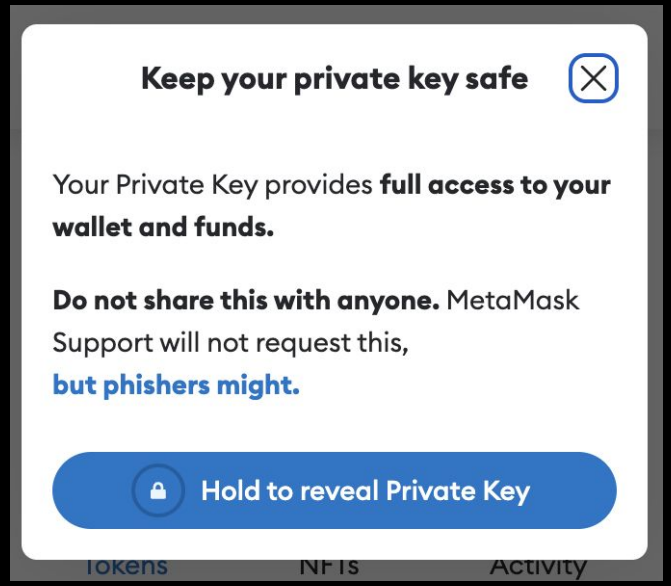
Step 1: Select your account and click account details



Extracting the private key (part 1)

Step 2:

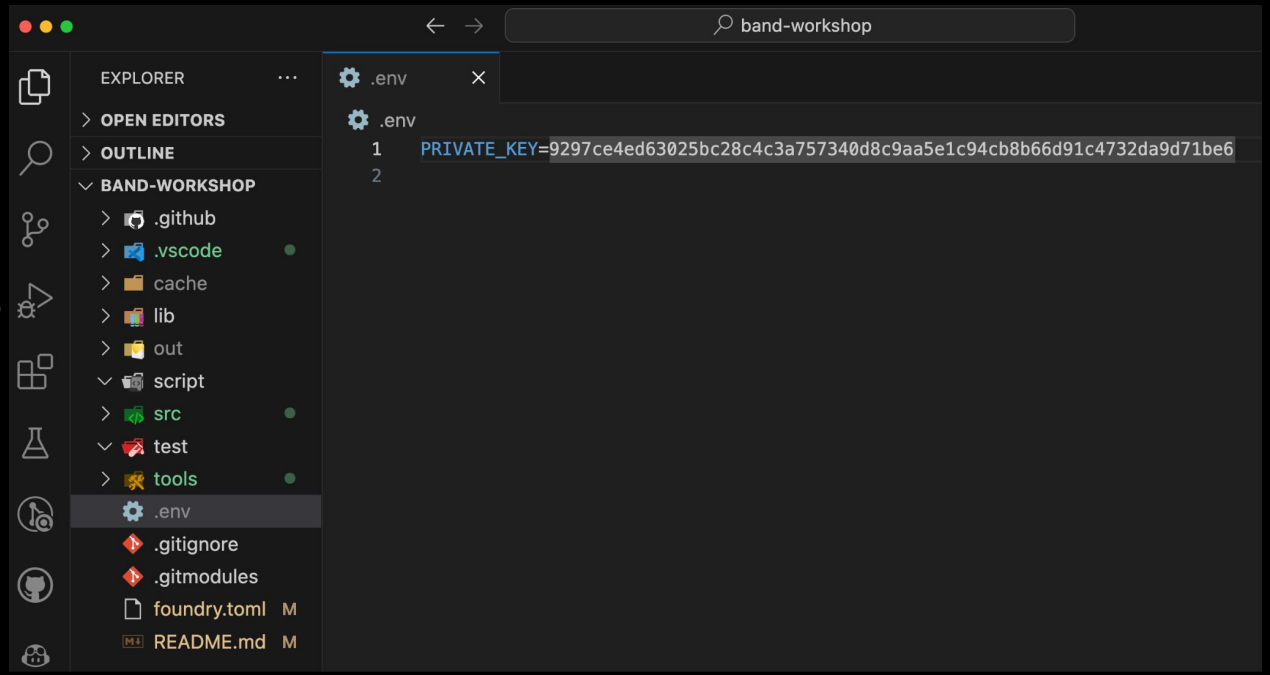
- Click show private key
- Enter password
- Press and hold to reveal
- Copy private key to clipboard
- Paste into .env file (see part 2)



Extracting the private key

Step 2:

- Paste into .env file
- This is so you don't commit them to github by mistake
- We will need this key later to deploy the contract



Contract Deployment

Contract Deployment Checklist

- Clone the Github repo
- Assuming Foundry is installed, jump into the project folder
- Extract the private key from metamask
- Run the cli command to deploy the example contract
- Run the cli command to verify the contract on the explorer

Deploy the Contract

- Make sure to include your private key in the deploy cli command
- Copy the cli command into your terminal and run it
- Grab your private key from your .env file and pass it through the private-key flag
  - Do not save this command with your keys in it to your readme file
  - DO NOT COMMIT KEYS TO GITHUB

```
● (base) ➜  band-workshop git:(master) ✗ forge create --rpc-url https://rpc-evm-sidechain.xrpl.org \
    --constructor-args 0xdE2022A8aB68AE86B0CD3Ba5EFa10AaB859d0293 \
    --private-key 9[redacted]6 \
    src/BandWorkshop.sol:BandWorkshop
[⠊] Compiling...
[⠊] Compiling 2 files with Solc 0.8.26
[⠊] Solc 0.8.26 finished in 306.37ms
Compiler run successful!
Deployer: 0x86Db6b33e7f76733aDC7071910EEfa61Ef62440c
Deployed to: 0x5deDf28FE20896E63304C643Af8c3A38f561e267
Transaction hash: 0x182e153b022d39243eb9901bf72b65065a65ddf2fe0816695b4526b1001cf92a
```

View the contract on the explorer

Contract details

0x5deDf28FE20896E63304C643Af8c3A38f561e267

Creator	0x86...440c at txn 0x18...f92a
Balance	0 XRP ($0)
Transactions	0
Gas used	0
Last balance update	10657434

Transactions 1 | Token transfers 0 | Tokens 0 | Internal txns 0 | Coin balance history | Contract

Contract creation code

Verify & publish

0x6080604052348015600f57600080fd5b506040516107e03803806107e083398101604081905260 2c916050565b6000805460016001600160a01b0319166001600160a01b0392909216919091179055607e565b600060208284031215606157600080fd5b81516001600160a01b0381168114607757600080fd5b9392505050565b61075380610 08d6000396000f3fe6080604052348015610010576000080fd5b506004361061004157600035 60e01c806321a78f68146100465780632 9d5453314610076578063d0f34da14610096575b600080fd5b6000546100599060016001600160a01b031681565b604051600160a01b03909116815260200 15b604051810390390f35b61008961008436600461 03f2565b6100cb565b604051610060d919061045b565b6100a96100a436600461049e565b6101e7565b604080518251815260208084015190820152918101519082015260600161006d565b6000805460405163e42a071b60e01b815260609291600160a01b03169063e42a071b90610100908790879060040161059f565b600060405180830381865afa15801561011d573d6000803e3d6000fd5b505050506040513d6000823e601f3d908101601f191682016040526101459190810190610626565b905060000815167ffffffffffffffff811115610163576101636102855 65b604051908082528060200260200182016040528015610 18c57816020016020820280368337019050 5b5090506000 5b82518110156101de57828181518110610 1ad576101ad6106c6565b602002602001015160000151828281518110610 1cb576101cb6106c6565b602090810291909101015260010161019256 5b509493505050565b61020b604051806060016040528060008152602001600081526020016000815250 90565b600054604051631955 56f360e21b81526001600160a01b03909116906365555bcc90

Deployed ByteCode

0x6080604052348015610010576000080fd5b506004361061004157600035 60e01c806321a78f68146100465780632 9d5453314610076578063d0f34da14610096575b600080fd5b6000546100599060016001600160a01b031681565b604051600160a01b03909116815260200 15b604051810390390f35b61008961008436600461 03f2565b6100cb565b604051610060d919061045b565b6100a96100a436600461049e565b6101e7565b604080518251815260208084015190820152918101519082015260600161006d565b6000805460405163e42a071b60e01b815260609291600160a01b03169063e42a071b90610100908790879060040161059f565b600060405180830381865afa15801561011d573d6000803e3d6000fd5b505050506040513d6000823e601f3d908101601f191682016040526101459190810190610626565b905060000815167ffffffffffffffff811115610163576101636102855 65b604051908082528060200260200182016040528015610 18c57816020016020820280368337019050 5b5090506000 5b82518110156101de57828181518110610 1ad576101ad6106c6565b602002602001015160000151828281518110610 1cb576101cb6106c6565b602090810291909101015260010161019256 5b509493505050565b61020b604051806060016040528060008152602001600081526020016000815250 90565b600054604051631955 56f360e21b81526001600160a01b03909116906365555bcc9061023d908690869060040161006dc565b606060405180830381865afa15801561025a573d6000803e3d6000fd5b505050506040513d601f19601f8201168201806040525081019061027e9190610701565b9392505050565b634e487b7160e01b600052604160045260246000fd5b604051601f8201601f1916810167ffffffffffffffff811182821017156102c457610

Contract Verification

- As we can see the contract is not readable on the explorer
- To fix this we must verify our contract to the explorer api
- Run the cli command provided in the Readme file
- Switch out the contract address with the address of the contract you deployed
- **Note:** If you haven't changed the the name of your contract, you might not need to verify
  - This is because i have verified this exact bytecode already
  - If you change the name of your contract pre-deployment, it will generate new bytecode and you can verify a fresh contract

Verify the Contract

- Copy the “deployed to” contract address output from your terminal
- Paste this into the cli command to verify the contract and run it

```
Deployer: 0x86Db6b33e7f76733aDC7071910EEfa61Ef62440c
Deployed to: 0x5deDf28FE20896E63304C643Af8c3A38f561e267
Transaction hash: 0x182e153b022d39243eb9901bf72b65065a65ddf2fe0816695b4526b1001cf92a
(base) ➜  band-workshop git:(master) ✗ forge verify-contract  --chain-id 1440002 --verifier=blockscout \
--verifier-url=https://explorer.xrplevm.org/api \
0x5deDf28FE20896E63304C643Af8c3A38f561e267  src/BandWorkshop.sol:BandWorkshop
Start verifying contract `0x5deDf28FE20896E63304C643Af8c3A38f561e267` deployed on 1440002

Submitting verification for [src/BandWorkshop.sol:BandWorkshop] 0x5deDf28FE20896E63304C643Af8c3A38f561e267.
Submitted contract for verification:
        Response: `OK`
        GUID: `5dedf28fe20896e63304c643af8c3a38f561e26766c68218`
        URL: https://explorer.xrplevm.org/address/0x5dedf28fe20896e63304c643af8c3a38f561e267
```

View the contract on the explorer

Contract details

0x5deDf28FE20896E63304C643Af8c3A38f561e267

Contract name	BandWorkshop
Creator	0x86...440c at txn 0x18...f92a
Balance	0 XRP ($0)
Transactions	0
Gas used	0
Last balance update	10657451

Transactions 1 | Token transfers 0 | Tokens 0 | Internal txns 0 | Coin balance history | Contract

Code | Read contract

Contract Source Code Verified (Exact Match)

Contract name	BandWorkshop	Compiler version	
EVM Version	Paris	Optimization enal	
Optimization runs	200	Verified at	
Contract file path	src/BandWorkshop.sol		

Constructor Arguments

0x000000000000000000000000de2022a8ab68ae86b0cd3ba5efa10aab859d0293

Arg [0] _ref (address): 0xde2022a8ab68ae86b0cd3ba5efa10aab859d0293

Contract source code (Solidity)

BandWorkshop.sol

src > BandWorkshop.sol

```solidity
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.26;

import {IStdReference} from "./interfaces/IStdReference.sol";

/// @title BandWorkshop
/// @notice A simple contract that demonstrates how to use the Band Standard Reference contract
/// @author @hazardcookie
contract BandWorkshop {
    IStdReference public ref;

    /// @param _ref The address of the Band Standard Reference contract
    constructor(IStdReference _ref) {
        ref = _ref;
    }

    /// @notice Get the price of a base/quote pair
    /// @param _symbol The symbol of the base asset
    /// @param _base The symbol of the quote asset
    /// @return The price data for the given base/quote pair
    function getPrice(string memory _symbol, string memory _base)
        external
        view
        returns (IStdReference.ReferenceData memory)
    {
        return ref.getReferenceData(_symbol, _base);
    }

```

EXPLORER SEARCH

- src
  - interfaces
    - IStdReference.sol
  - BandWorkshop.sol

Test out your contract!

Transactions 1 | Token transfers 0 | Tokens 0 | Internal txns 0 | Coin balance history | Contract

Code | Read contract

Disconnected | Connect wallet

Contract information

Expand all | Reset

1. getBulkPrice

_bases* (string[])

XRP

ETH

BTC

_quotes* (string[])

USD

USD

USD

Read

uint256[]

[getBulkPrice method response]
[
uint256[] (uint256[]) : 499790972828598700,2.399248544722519e+21,5.6019399896237565e+22
]

XRPL EVM Sidechain - Band Oracle Workshop

https://github.com/hazardcookie/Band-Oracle-Foundry-Workshop