

{×}

# XRPL EVM Sidechain

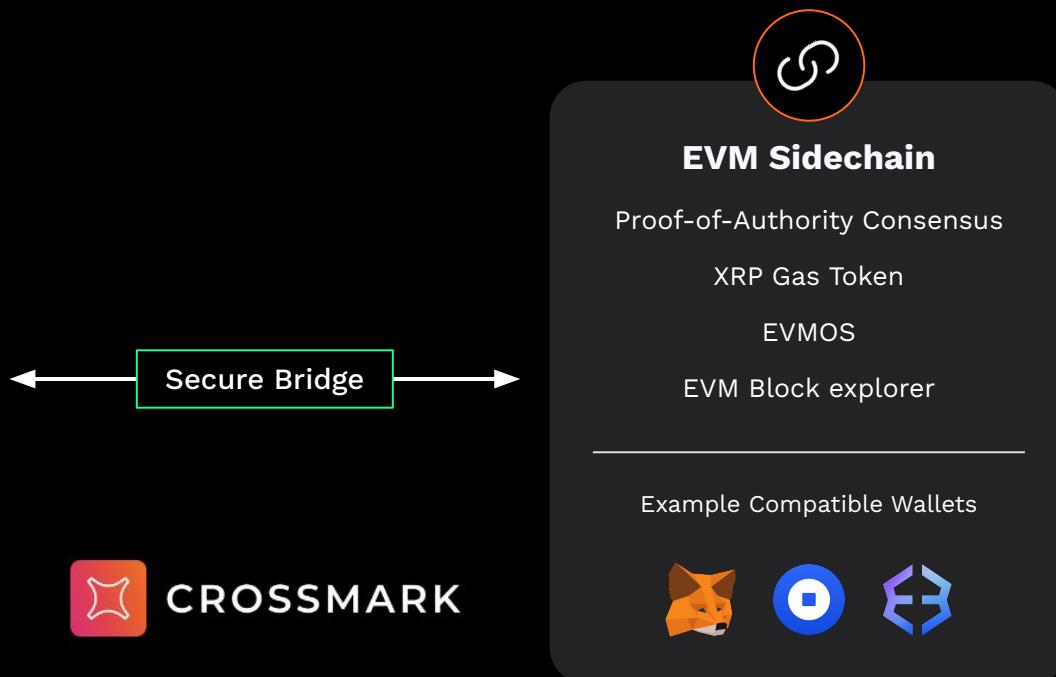
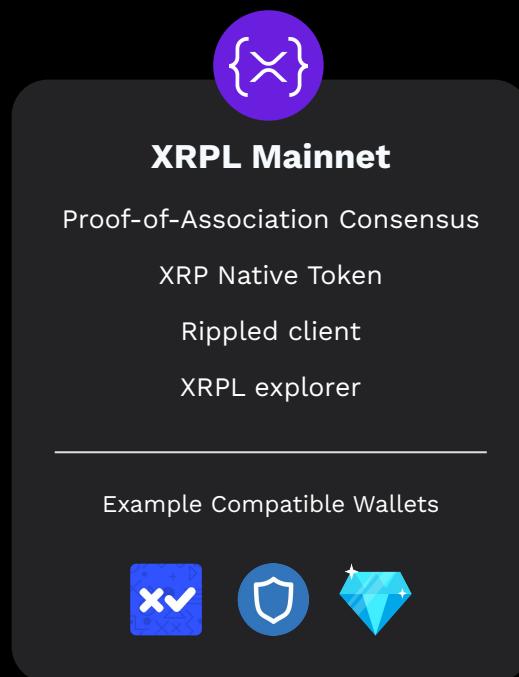


**Hazard Cookie**  
Developer Advocate  
RippleX

# Intro to the EVM Sidechain

# What is the “EVM Sidechain”

The EVM Sidechain enables the ability to interact or deploy smart contracts written in Solidity with a secure bridge to XRPL Mainnet



# EVM Compatibility on different blockchains

---

Blockchain Ecosystem	EVM Compatibility Solution/Project
Ethereum	Native
Solana	Neon
Polkadot	Moonbeam
Cosmos	Evmos
Polygon	zkEVM
BNB Chain	BNB Smart chain
Avalanche	Avalanche C-chain
XRPL	<b>EVM Sidechain</b>

# Network Details

---

- Consensus
  - Powered by CometBFT
  - Tendermint fork
  - Byzantine fault tolerance engine
  - 1 block finalization time
- Execution
  - Cosmos SDK
  - EVMOS
- Validators
  - POA Cosmos SDK

# Why a EVM Sidechain for XRPL?

---

GM Pools		PRICE ↴	TVL ↴	LIQUIDITY ↴	NET RATE / 1 H	UTILIZATION ↴
 XRP/USD	▼	\$0.56489	\$1,787,608.91	\$2,193,088.81	-0.0018% / +0.0018%	12.36%

# Why a EVM Sidechain for XRPL?

---

## PancakeSwap XRP Stats



## XRPL All Tokens Stats



# Benefits of EVM Sidechain for Developers

---

## Interact with XRPL

EVM applications can interact with XRPL via the bridge

## Faster Go Live

EVM applications can be launched without the need of an amendment

## Use App As-Is

Any contract written for Ethereum/EVM can be used as-is without code changes

## EVM ecosystem

XRPL community will have access to more EVM apps and developers who build on the sidechain

# Dev Tooling

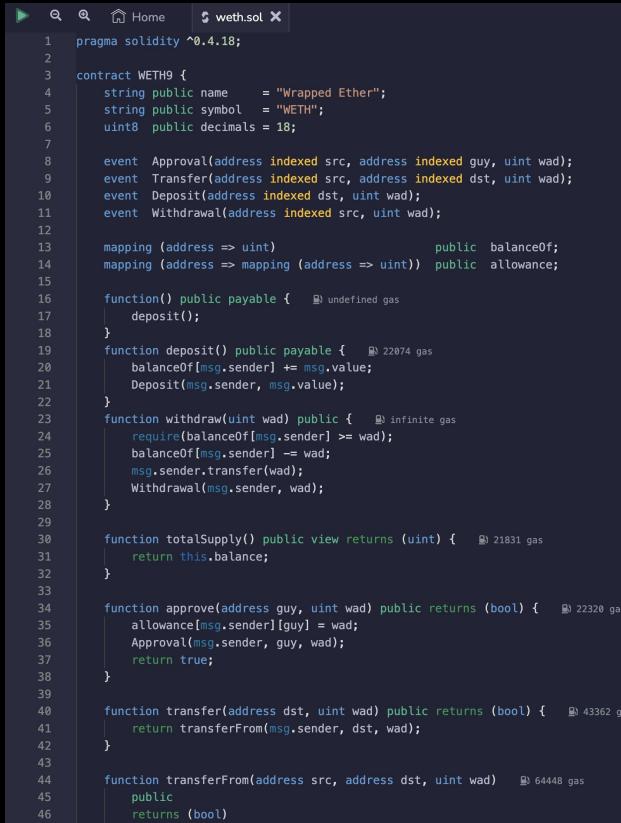
---

## Languages

- Solidity
- Vyper
- Huff
- Yul
- Assembly

## Frameworks

- Remix
- Hardhat
- Foundry



A screenshot of a code editor displaying a Solidity smart contract named `weth.sol`. The code defines a `WETH9` contract with various functions for depositing Ether, withdrawing Ether, approving transfers, and transferring tokens between addresses. The editor interface includes tabs for Home and weth.sol, and a toolbar with icons for search, refresh, and file operations.

```
1 pragma solidity ^0.4.18;
2
3 contract WETH9 {
4     string public name    = "Wrapped Ether";
5     string public symbol = "WETH";
6     uint8  public decimals = 18;
7
8     event Approval(address indexed src, address indexed guy, uint wad);
9     event Transfer(address indexed src, address indexed dst, uint wad);
10    event Deposit(address indexed dst, uint wad);
11    event Withdrawal(address indexed src, uint wad);
12
13    mapping (address => uint)           public balanceOf;
14    mapping (address => mapping (address => uint)) public allowance;
15
16    function() public payable {   gas;
17        deposit();
18    }
19    function deposit() public payable {   gas 22074
20        balanceOf[msg.sender] += msg.value;
21        Deposit(msg.sender, msg.value);
22    }
23    function withdraw(uint wad) public {   gas infinite
24        require(balanceOf[msg.sender] >= wad);
25        balanceOf[msg.sender] -= wad;
26        msg.sender.transfer(wad);
27        Withdrawal(msg.sender, wad);
28    }
29
30    function totalSupply() public view returns (uint) {   gas 21831
31        return this.balance;
32    }
33
34    function approve(address guy, uint wad) public returns (bool) {   gas 22320
35        allowance[msg.sender][guy] = wad;
36        Approval(msg.sender, guy, wad);
37        return true;
38    }
39
40    function transfer(address dst, uint wad) public returns (bool) {   gas 43362
41        return transferFrom(msg.sender, dst, wad);
42    }
43
44    function transferFrom(address src, address dst, uint wad)   gas 64448
45        public
46        returns (bool)
```



# Connecting it all together!

## How will it connect to mainnet?

---

A~~X~~ELAR

# What is “Axelar”

---

**Interoperability Platform:** Axelar acts as a bridge between chains, enabling token transfers and general message passing.

## Bridging Process:

- **Outbound:** Locks XRPL assets, issues wrapped assets on other chains.
- **Inbound:** Burns tokens on other chains, releases locked assets on XRPL.

## Security:

- Secured by a multisig XRPL account
- Managed by top 32 Axelar validators.

## Key Components:

- **On-Chain:** XRPL multisig account, Gateway, Voting Verifier, Multisig Prover.
- **Off-Chain:** Relayer processes (Event Monitor, TX Broadcaster, Axelar Validator).

# Why a EVM Sidechain for XRPL?

#	DEXes	Pairs	Price	Volume	Confidence	Liquidity	Liquidity score
1	► KLAYswap	OXRP/KLAY	\$0.5267	\$46.98K	High	\$1.4M	436
2	abbit PancakeSwap v2 (BS C)	XRP/WBNB	\$0.5637	\$39.04K	High	\$576.78K	377
3	abbit PancakeSwap v3 (B SC)	XRP/ETH	\$0.5638	\$97.21K	High	\$405.88K	354
4	► KLAYswap	OXRP/OETH	\$0.5256	\$3.67K	High	\$232.89K	317
5	pink Uniswap v3 (BSC)	XRP/USDT	\$0.5630	\$35.11K	High	\$121.52K	273
6	abbit PancakeSwap v3 (B SC)	XRP/USDT	\$0.5634	\$42.13K	High	\$64.49K	231
7	purple THENA FUSION	XRP/WBNB	\$0.5636	\$18.8K	High	\$64.12K	231
8	blue Biswap v2	XRP/WBNB	\$0.5627	\$1.13K	High	\$52.67K	218
9	► KLAYswap	OXRP/USDT	\$0.5272	\$1.81K	High	\$47.06K	210

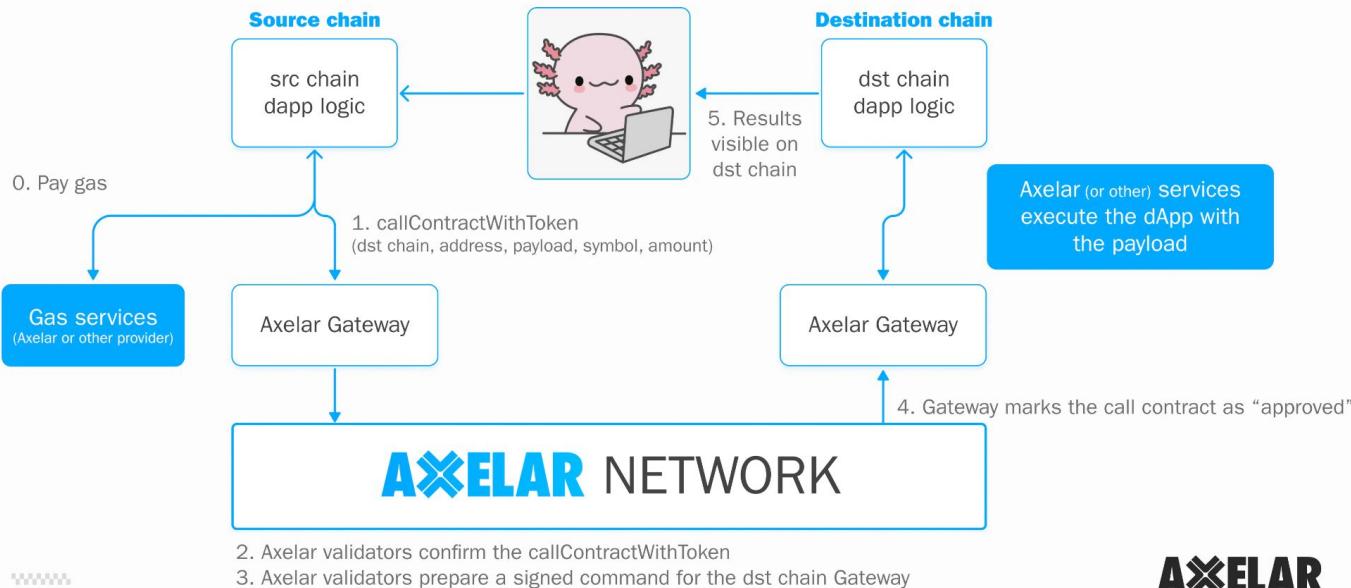
# How it works

---



# What is GMP?

## General Message Passing



# Example of Axelar GMP Smart Contract

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import { AxelarExecutable } from '../AxelarExecutable.sol';
import { IAxelarGateway } from '../../../../../interfaces/IAxelarGateway.sol';

contract ExecutableSample is AxelarExecutable {
    string public sourceChain;
    string public sourceAddress;
    string public message;
    string public tokenDenom;
    uint256 public tokenAmount;

    event Executed(string _sourceChain, string _sourceAddress, string _message, string _tokenSymbol,
        uint256 _tokenAmount);

    constructor(address gateway_) AxelarExecutable(gateway_) {
    }

    function _execute(
        string calldata sourceChain_,
        string calldata sourceAddress_,
        bytes calldata payload_
    ) internal override {
        bytes memory gmpPayload;
        (tokenDenom, tokenAmount, gmpPayload) = abi.decode(payload_, (string, uint256, bytes));
        (message) = abi.decode(gmpPayload, (string));
        sourceChain = sourceChain_;
        sourceAddress = sourceAddress_;

        emit Executed(sourceChain, sourceAddress, message,
            IAxelarGateway(gateway).tokenSymbol(tokenDenom), tokenAmount);
    }
}
```

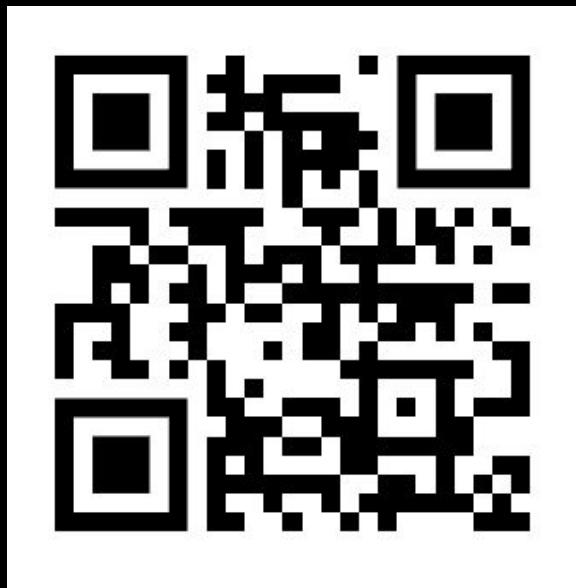
# Example of GMP Call From XRPL

```
● ● ●  
  
import * as xrpl from "xrpl";  
  
const XRPL_RPC_URL = "wss://s.altnet.rippletest.net:51233";  
async function gmp() {  
    const client = new xrpl.Client(XRPL_RPC_URL);  
    await client.connect();  
  
    // const user = xrpl.Wallet.fromSeed(SEED); // Read XRPL wallet seed from environment or generate  
and fund new wallet:  
    const user = xrpl.Wallet.generate();  
    await client.fundWallet(user);  
  
    const paymentTx: xrpl.Transaction = {  
        TransactionType: "Payment",  
        Account: user.address,  
        Amount: "1",  
        Destination: "rfEf91bLxrTVC76vw1W3Ur8Jk4Lwujskmb",  
        SigningPubKey: "",  
        Flags: 0,  
        Fee: "30",  
        Memos: [  
            {  
                Memo: {  
                    MemoData: "143669292488bd98a0F14F1c73829572f2c25773", // the 'ExecutableSample'  
contract  
                MemoType: Buffer.from("destination_address").toString('hex').toUpperCase(),  
            },  
            {  
                Memo: {  
                    MemoData: Buffer.from("ethereum").toString('hex').toUpperCase(),  
                    MemoType: Buffer.from("destination_chain").toString('hex').toUpperCase(),  
                },  
            {  
                Memo: {  
                    MemoData: "df031b281246235d0e8c8254cd731ed95d2caf4db4da67f41a71567664a1fae8", //  
keccak256(abi.encode(gmpPayload)), in this example, keccak256(abi.encode(['string'], ['Just transferred  
XRP to Ethereum!']))  
                MemoType: Buffer.from("payload_hash").toString('hex').toUpperCase(),  
            },  
        ],  
    };  
  
    const signed = user.sign(await client.autofill(paymentTx));  
    console.log(signed);  
    await client.submitAndWait(signed.tx_blob);  
    await client.disconnect();  
}  
  
gmp();
```



Join the Discord to get Started on Devnet

---



# Useful Axelar <> XRPL Docs

---

Axelar XRPL Bridge Docs



Axelar XRPL GMP Docs





# 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/>
- Contract Address:  
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**

**New RPC URL**

**Chain ID ⓘ**

**Currency symbol**

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)**

[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

{ $\times$ } XRP LEDGER

Transfer assets across XRPL chains.

Bridge Activity

From To

Network Network

{ $\times$ } XRPL Devnet { $\times$ } EVM Sidechain Devnet

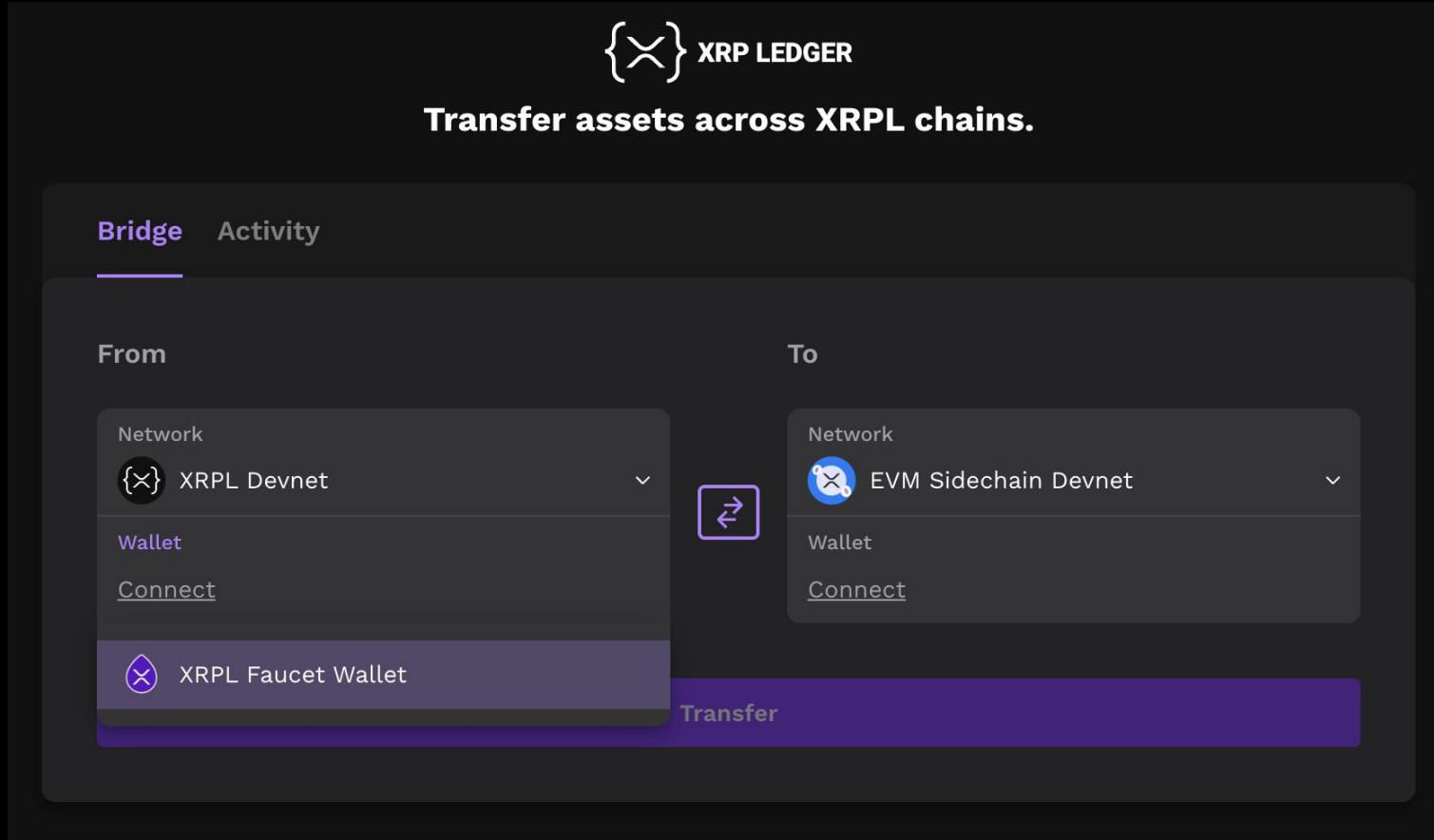
Wallet Wallet

[Connect](#) [Connect](#)

{ $\times$ } XRPL Faucet Wallet Transfer

{ $\times$ }

# Click on \*Connect\* - Then press XRPL Faucet Wallet



# Click on \*Connect\* - Then press Metamask

{X} XRP LEDGER

Transfer assets across XRPL chains.

Bridge Activity

From

Network {X} XRPL Devnet

Wallet {X} rNJM...Wev9odejCVfcU

To

Network {X} EVM Sidechain Devnet

Wallet Connect

Metamask

Transfer

{X}

# Bridge over some XRP

{X} XRP LEDGER

Transfer assets across XRPL chains.

**Bridge** Activity

**From**

Network {X} XRPL Devnet

Wallet rNjMiDYr8N4Cfoe...Wev9odejCVfcU

**To**

Network {X} 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



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

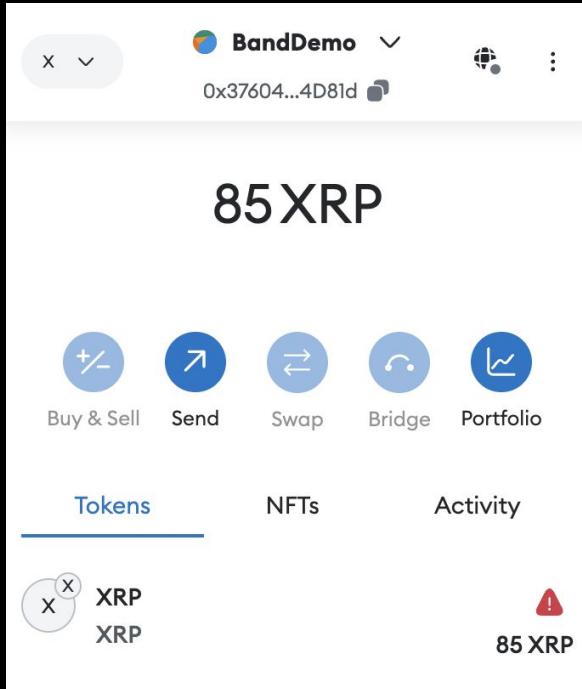
Receive

85 XRP

Done

# Soon you should see your XRP in your MetaMask Wallet

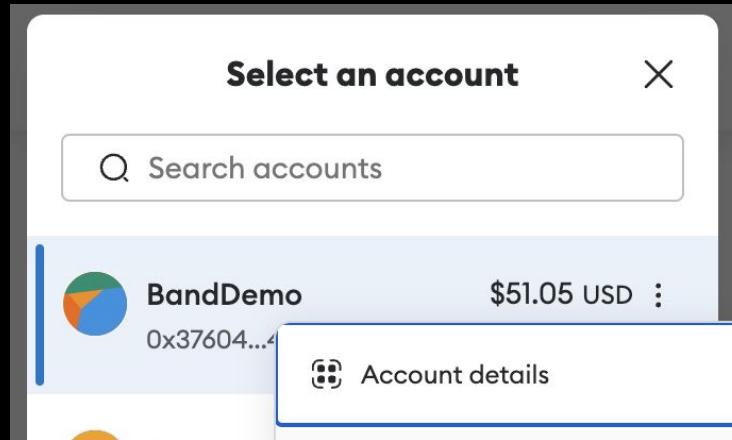
---



# Extracting the private key

---

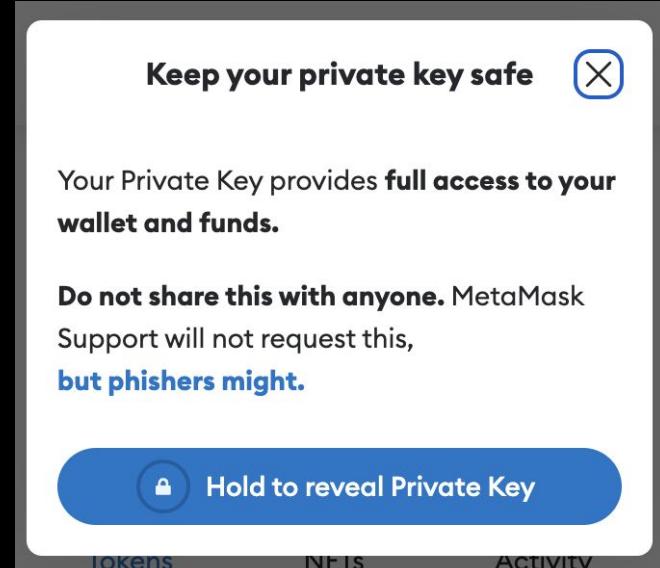
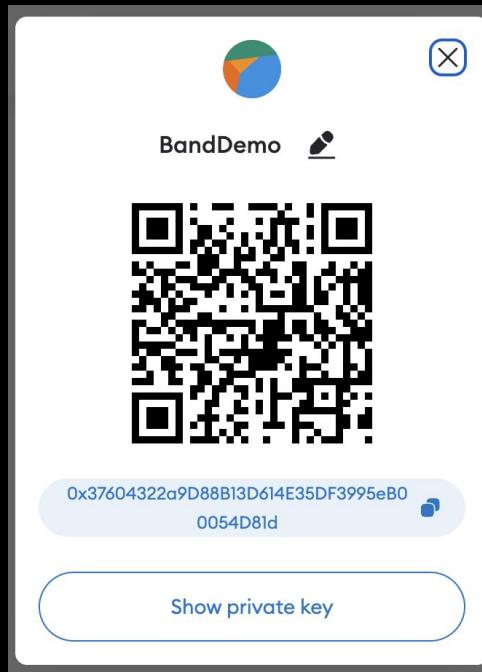
Step 1: Select your account and click account details



# Extracting the private key (part 1)

Step 2:

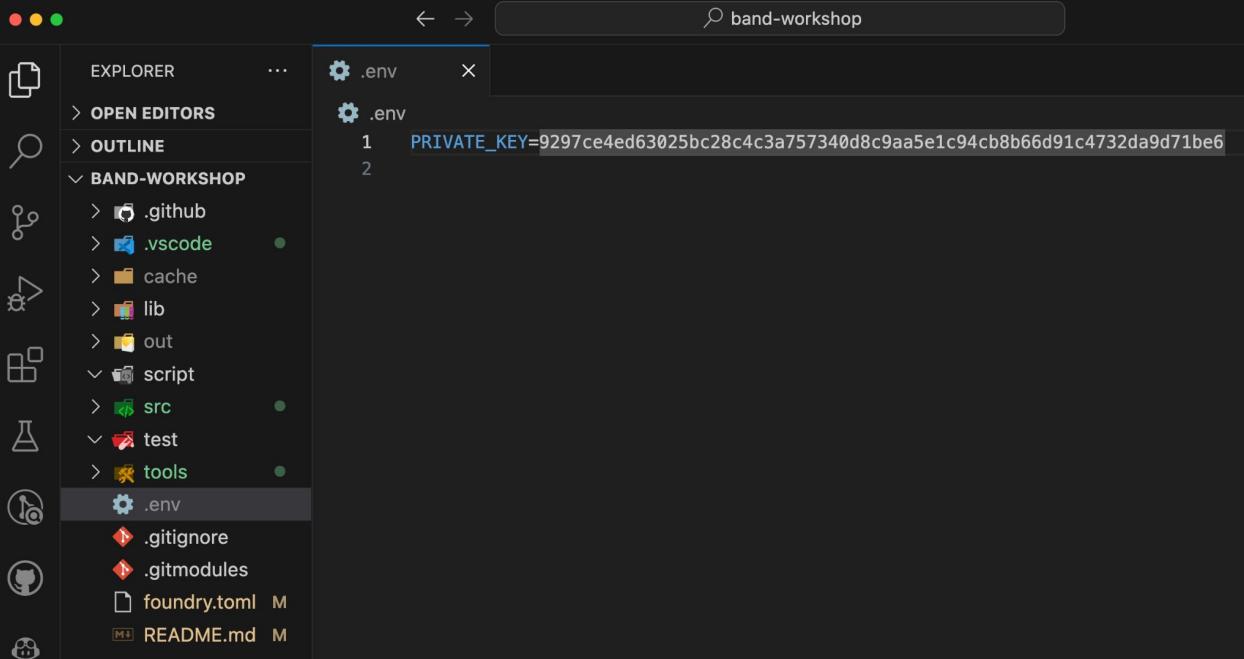
- Click show private key
- Enter password
- Press and hold to reveal
- Copy private key to clipboard
- Paste into .env file (see part 2)



# 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



A screenshot of the VS Code interface. The left sidebar shows a dark-themed Explorer view with various project files and folders. In the center, there is an Editor tab titled '.env' which contains the following code:

```
PRIVATE_KEY=9297ce4ed63025bc28c4c3a757340d8c9aa5e1c94cb8b66d91c4732da9d71be6
```

# 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 0xE2022A8aB68AE86B0CD3Ba5EFa10AaB859d0293 \
--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 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 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

0x000000000000000000000000000000de2022a8ab68ae86b0cd3ba5efa10aab859d0293

Arg [0] \_ref (address): 0xde2022a8ab68ae86b0cd3ba5efa10aab859d0293

## Contract source code (Solidity)

BandWorkshop.sol

```
src > BandWorkshop.sol
1 // SPDX-License-Identifier: UNLICENSED
2 pragma solidity ^0.8.26;
3
4 import {IStdReference} from "./interfaces/IStdReference.sol";
5
6 /// @title BandWorkshop
7 /// @notice A simple contract that demonstrates how to use the Band Standard Reference contract
8 /// @author @hazardcookie
9 contract BandWorkshop {
10     IStdReference public ref;
11
12     /// @param _ref The address of the Band Standard Reference contract
13     constructor(IStdReference _ref) {
14         ref = _ref;
15     }
16
17     /// @notice Get the price of a base-quote pair
18     /// @param _symbol The symbol of the base asset
19     /// @param _base The symbol of the quote asset
20     /// @return The price data for the given base-quote pair
21     function getPrice(string memory _symbol, string memory _base)
22         external
23         view
24         returns (IStdReference.ReferenceData memory)
25     {
26         return ref.getReferenceData(_symbol, _base);
27     }
28 }
```

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>