**Using Data Feeds on EVM Chains**

**NEW TO SMART CONTRACTS?**

This tutorial assumes that you know how to create and deploy basic smart contracts. If you are new to smart contract development, learn how to [Deploy Your First Smart Contract](https://docs.chain.link/getting-started/deploy-your-first-contract/) before you start this guide.

When you connect a smart contract to real-world services or off-chain data, you create a *hybrid smart contract*. For example, you can use Chainlink Data Feeds to connect your smart contracts to asset pricing data like the [ETH / USD feed](https://data.chain.link/eth-usd). These data feeds use the data aggregated from many independent Chainlink node operators. Each price feed has an on-chain address and functions that enable contracts to read pricing data from that address.

The code for reading Data Feeds is the same across all EVM-compatible blockchains and Data Feed types. You choose different types of feeds for different uses, but the request and response format are the same.

**Examine the sample contract**

This example contract obtains the latest price answer from the [BTC / USD feed](https://docs.chain.link/data-feeds/price-feeds/addresses) on the Sepolia testnet, but you can modify it to read any of the different [Types of Data Feeds](https://docs.chain.link/data-feeds#types-of-data-feeds).

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.7;

import "@chainlink/contracts/src/v0.8/interfaces/AggregatorV3Interface.sol";

contract PriceConsumerV3 {

AggregatorV3Interface internal priceFeed;

/\*\*

\* Network: Sepolia

\* Aggregator: BTC/USD

\* Address: 0x1b44F3514812d835EB1BDB0acB33d3fA3351Ee43

\*/

constructor() {

priceFeed = AggregatorV3Interface(

0x1b44F3514812d835EB1BDB0acB33d3fA3351Ee43

);

}

/\*\*

\* Returns the latest price.

\*/

function getLatestPrice() public view returns (int) {

// prettier-ignore

(

/\* uint80 roundID \*/,

int price,

/\*uint startedAt\*/,

/\*uint timeStamp\*/,

/\*uint80 answeredInRound\*/

) = priceFeed.latestRoundData();

return price;

}

}

[Open in Remix](https://remix.ethereum.org/#url=https://docs.chain.link/samples/PriceFeeds/PriceConsumerV3.sol)[What is Remix?](https://docs.chain.link/getting-started/conceptual-overview#what-is-remix)

The contract has the following components:

* The import line imports an interface named AggregatorV3Interface. Interfaces define functions without their implementation, which leaves inheriting contracts to define the actual implementation themselves. In this case, AggregatorV3Interface defines that all v3 Aggregators have the function latestRoundData. You can [see the complete code](https://github.com/smartcontractkit/chainlink/blob/develop/contracts/src/v0.8/interfaces/AggregatorV3Interface.sol) for the AggregatorV3Interface on GitHub.
* The constructor() {} initializes an interface object named priceFeed that uses AggregatorV3Interface and connects specifically to a proxy aggregator contract that is already deployed at 0x1b44F3514812d835EB1BDB0acB33d3fA3351Ee43. The interface allows your contract to run functions on that deployed aggregator contract.
* The getLatestPrice() function calls your priceFeed object and runs the latestRoundData() function. When you deploy the contract, it initializes the priceFeed object to point to the aggregator at 0x1b44F3514812d835EB1BDB0acB33d3fA3351Ee43, which is the proxy address for the Sepolia BTC / USD data feed. Your contract connects to that address and executes the function. The aggregator connects with several oracle nodes and aggregates the pricing data from those nodes. The response from the aggregator includes several variables, but getLatestPrice() returns only the price variable.

**Compile, deploy, and run the contract**

**CONFIGURE AND FUND YOUR WALLET**

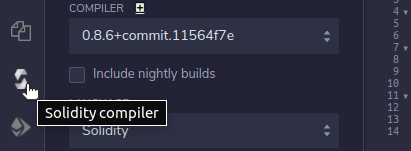
If you have not already configured your MetaMask wallet and funded it with testnet ETH, follow the instructions in [Deploy Your First Smart Contract](https://docs.chain.link/getting-started/deploy-your-first-contract#install-and-fund-your-metamask-wallet) to set that up. You can get testnet ETH at one of the available [Sepolia faucets](https://docs.chain.link/resources/link-token-contracts/" \l "sepolia-testnet).

Deploy the PriceConsumerV3 smart contract on the Sepolia testnet.

1. [Open the example contract](https://remix.ethereum.org/#url=https://docs.chain.link/samples/PriceFeeds/PriceConsumerV3.sol) in Remix. Remix opens and shows the contents of the smart contract.

[Open the contract in Remix](https://remix.ethereum.org/#url=https://docs.chain.link/samples/PriceFeeds/PriceConsumerV3.sol)

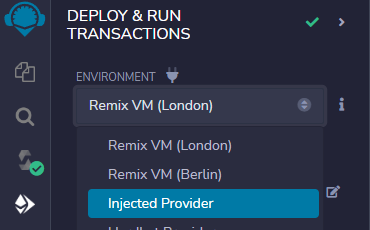
1. Because the code is already written, you can start the compile step. On the left side of Remix, click the **Solidity Compiler** tab to view the compiler settings.



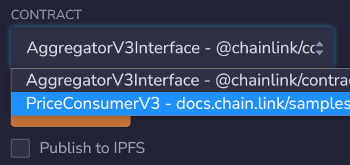
1. Use the default compiler settings. Click the **Compile PriceConsumerV3.sol** button to compile the contract. Remix automatically detects the correct compiler version depending on the pragma that you specify in the contract. You can ignore warnings about unused local variables in this example.



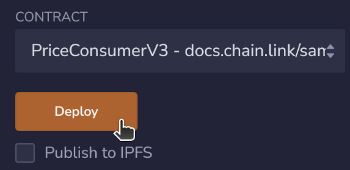
1. On the **Deploy** tab, select the **Injected Provider** environment. This contract specifically requires Web3 because it connects with another contract on the blockchain. Running in a JavaScript VM will not work.



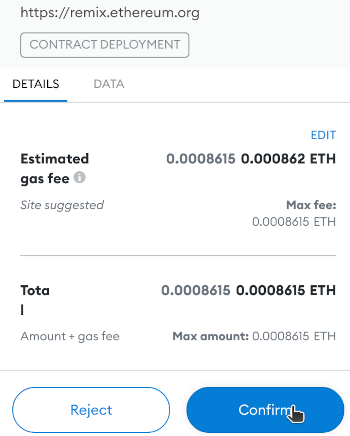
1. Because the example contract has several imports, Remix might select another contract to deploy by default. In the **Contract** section, select the PriceConsumerV3 contract to make sure that Remix deploys the correct contract.



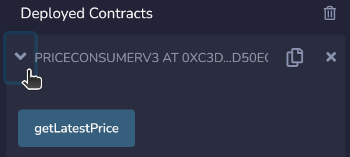
1. Click **Deploy** to deploy the contract to the Sepolia testnet. MetaMask opens and asks you to confirm payment for deploying the contract. Make sure MetaMask is set to the Sepolia network before you accept the transaction. Because these transactions are on the blockchain, they are not reversible.



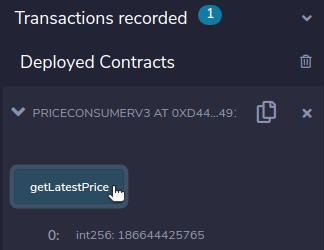
1. In the MetaMask prompt, click **Confirm** to approve the transaction and spend your testnet ETH required to deploy the contract.



1. After a few seconds, the transaction completes and your contract appears under the **Deployed Contracts** list in Remix. Click the contract dropdown to view its variables and functions.



1. Click **getLatestPrice** to show the latest price from the aggregator contract. The latest price appears just below the button. The returned price is an integer, so it is missing its decimal point.



You can run your own oracle networks that provide data to smart contracts similar to the AggregatorV3Interface, but first, you should learn how to configure your contracts to pay oracles using LINK tokens. Follow the [Generate Random Numbers](https://docs.chain.link/getting-started/intermediates-tutorial/) to learn how.