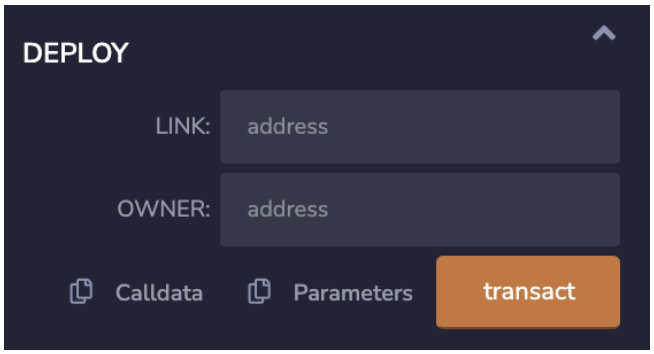
A. ORACLE CONTRACT:

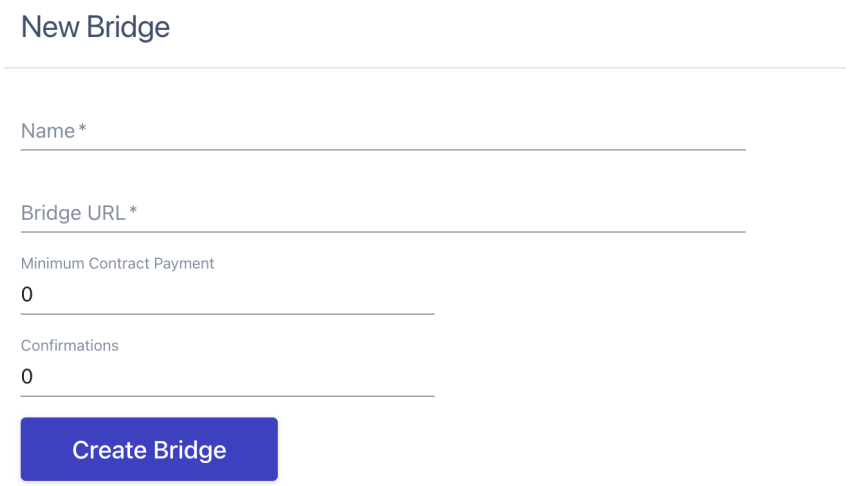
```
// SPDX-License-Identifier: MIT
pragme solidity ^0.7.6
import "@chainlink/contracts/src/v0.7/Operator.sol
```

While deploying this contract, need to give the “Link” as “0x1a55174123E992a3f8cd492924c5701546f7E7D7” for Polygon, “0x33f4212b027e22af7e6ba21fc572843c0d701cd1” for Apothem and for “Owner” , the wallet address from which you are going to deploy.



Once the contract is deployed, note down the oracle address.

B. BRIDGE CREATION:



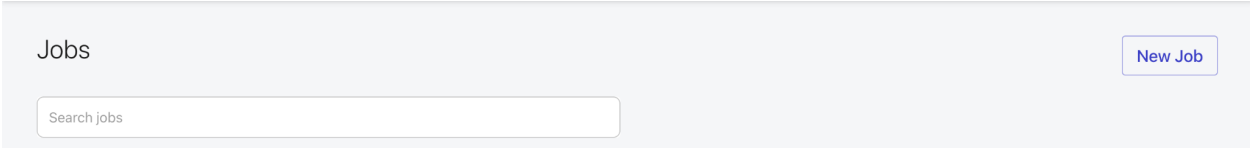
Name: <Bridge_Name>
Bridge URL: http://<IP_address>:<port_number>

NOTE: The <port_number>, should match your adapter port number.

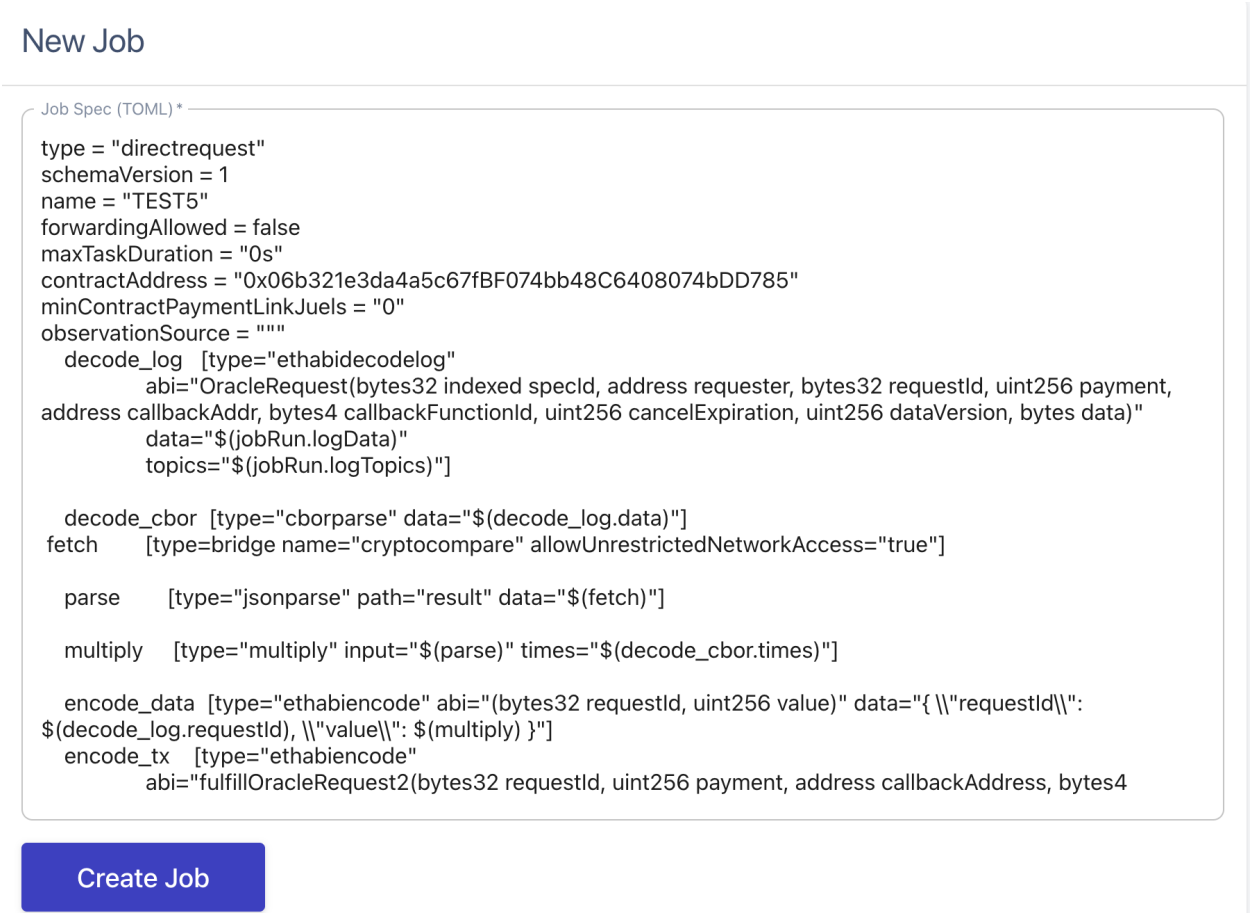
##

C. JOB SUBMISSION:

1. In the Jobs Section, click on the “New Job” button.



2. Now, submit the below **Job Spec** and press the “Create job” button.



Job Spec:

```
type = "directrequest"
schemaVersion = 1
name = "TEST5"
forwardingAllowed = false
maxTaskDuration = "0s"
```

```
contractAddress = "0x06b321e3da4a5c67fBF074bb48C6408074bDD785" //Oracle Address which you deployed.
minContractPaymentLinkJuels = "0"
observationSource = """
decode_log [type="ethabidecodelog" abi="OracleRequest(bytes32 indexed specId, address requester, bytes32 requestId, uint256 payment, address callbackAddr, bytes4 callbackFunctionId, uint256 cancelExpiration, uint256 dataVersion, bytes data)" data="$(jobRun.logData)" topics="$(jobRun.logTopics)"]

decode_cbor [type="cborparse" data="$(decode_log.data)"]

fetch [type=bridge name="cryptocompare" allowUnrestrictedNetworkAccess="true"]

parse [type="jsonparse" path="result" data="$(fetch)"]

multiply [type="multiply" input="$(parse)" times="$(decode_cbor.times)"]

encode_data [type="ethabiencode" abi="(bytes32 requestId, uint256 value)" data="{ \\"requestId\\": $(decode_log.requestId), \\"value\\": $(multiply) }"]

encode_tx [type="ethabiencode" abi="fulfillOracleRequest2(bytes32 requestId, uint256 payment, address callbackAddress, bytes4 callbackFunctionId, uint256 expiration, bytes calldata data)"
data="{\\"requestId\\": $(decode_log.requestId), \\"payment\\": $(decode_log.payment), \\"callbackAddress\\": $(decode_log.callbackAddr), \\"callbackFunctionId\\": $(decode_log.callbackFunctionId), \\"expiration\\": $(decode_log.cancelExpiration), \\"data\\": $(encode_data)}" ]

submit_tx [type="ethtx" to="0x06b321e3da4a5c67fBF074bb48C6408074bDD785" data="$(encode_tx)"] //paste the Oracle Address which you deployed in ‘to’ field

decode_log -> decode_cbor -> fetch -> parse -> multiply -> encode_data -> encode_tx -> submit_tx
"""
```

3. Once the Job has been submitted successfully, copy the External Job ID and remove the hyphen(‘-’) from the job ID

ID	Name	Type	External Job ID
5	TEST5	Direct Request	53034a3f-0999-49f7-8793-7a96fa3e9a32

##

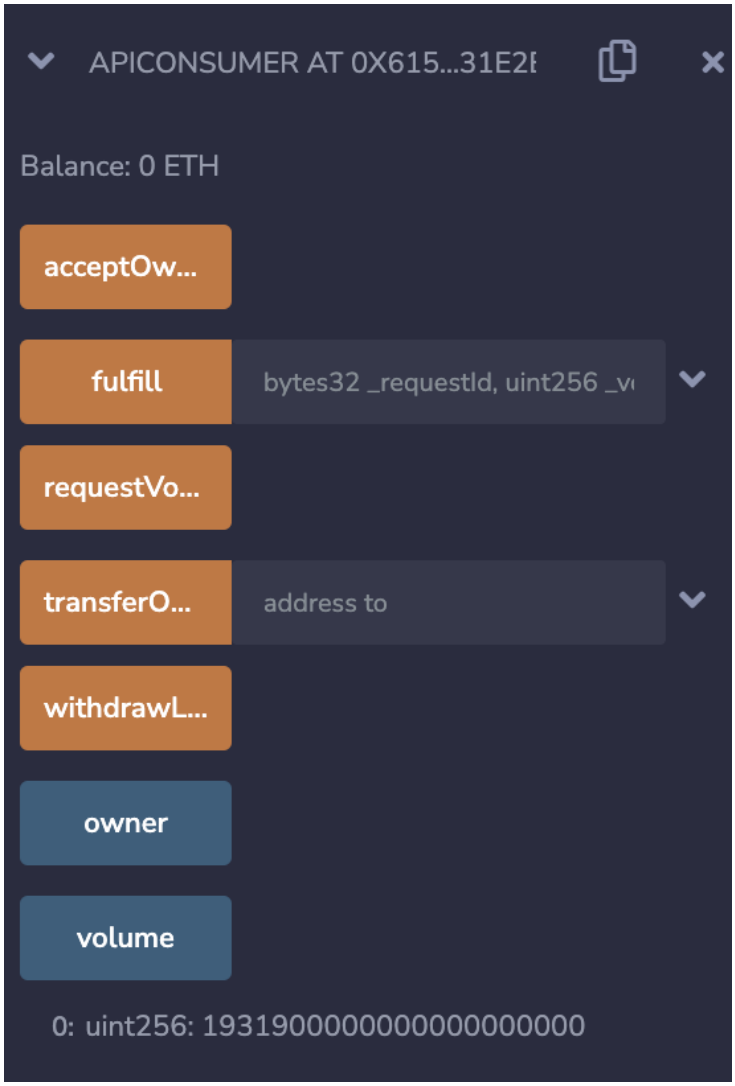
D. CONSUMER CONTRACT

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.7;

import "@chainlink/contracts/src/v0.8/ChainlinkClient.sol";
import "@chainlink/contracts/src/v0.8/ConfirmedOwner.sol";

/**
* Request testnet LINK and ETH here: https://faucets.chain.link/
* Find information on LINK Token Contracts and get the latest ETH and LINK
faucets here: https://docs.chain.link/docs/link-token-contracts/
*/

/**
* THIS IS AN EXAMPLE CONTRACT WHICH USES HARDCODED VALUES FOR CLARITY.
* THIS EXAMPLE USES UN-AUDITED CODE.
* DO NOT USE THIS CODE IN PRODUCTION.
*/

contract APIConsumer is ChainlinkClient, ConfirmedOwner {
using Chainlink for Chainlink.Request;

uint256 public volume;
bytes32 private jobId;
uint256 private fee;

event RequestVolume(bytes32 indexed requestId, uint256 volume);

/**
* @notice Initialize the link token and target oracle
*
* Sepolia Testnet details:
* Link Token: 0x779877A7B0D9E8603169DdbD7836e478b4624789
* Oracle: 0x6090149792dAAeE9D1D568c9f9a6F6B46AA29eFD (Chainlink DevRel)
* jobId: ca98366cc7314957b8c012c72f05aeeb
*
*/
constructor() ConfirmedOwner(msg.sender) {
setChainlinkToken(0x1a55174123E992a3f8cd492924c5701546f7E7D7);//Link address as
mentioned in 'A'
setChainlinkOracle(0x06b321e3da4a5c67fBF074bb48C6408074bDD785);//Oracle address
```

```
jobId = "53034a3f099949f787937a96fa3e9a32";//Job ID as stored in 'C' JOB
SUBMISSION
fee = (0.001 * 1000000000000000000) / 10;
}

/**
* Create a Chainlink request to retrieve API response, find the target
* data, then multiply by 1000000000000000000 (to remove decimal places from
data).
*/
function requestVolumeData() public returns (bytes32 requestId) {
Chainlink.Request memory req = buildChainlinkRequest(
jobId,
address(this),
this.fulfill.selector
);

// Set the URL to perform the GET request on
// req.add(
// "get",
// "https://min-api.cryptocompare.com/data/pricemultifull?fsyms=ETH&tsyms=USD"
// );
req.add(
"get",
"https://min-api.cryptocompare.com/data/price?fsym=ETH&tsyms=USD"
);

// Set the path to find the desired data in the API response, where the
response format is:
// {"RAW":
// {"ETH":
// {"USD":
// {
// "VOLUME24HOUR": xxx.xxx,
// }
// }
// }
// }
// request.add("path", "RAW.ETH.USD.VOLUME24HOUR"); // Chainlink nodes prior to
1.0.0 support this format
//req.add("path", "RAW.ETH.USD.VOLUME24HOUR"); // Chainlink nodes 1.0.0 and
later support this format
```

```
//req.add("path", "USD"); // Chainlink nodes 1.0.0 and later support this
format
//req.add("path", "Result,data,result"); // Chainlink nodes 1.0.0 and later
support this format

// Multiply the result by 1000000000000000000 to remove decimals
int256 timesAmount = 10 ** 18;
req.addInt("times", timesAmount);

// Sends the request
return sendChainlinkRequest(req, fee);
}

/**
* Receive the response in the form of uint256
*/
function fulfill(
bytes32 _requestId,
uint256 _volume
) public recordChainlinkFulfillment(_requestId) {
emit RequestVolume(_requestId, _volume);
volume = _volume;
}

/**
* Allow withdraw of Link tokens from the contract
*/
function withdrawLink() public onlyOwner {
LinkTokenInterface link = LinkTokenInterface(chainlinkTokenAddress());
require(
link.transfer(msg.sender, link.balanceOf(address(this))),
"Unable to transfer"
);
}
}
```

Once the contract is deployed, fund the contract with 0.1 PLI and click on ‘requestVolumeData’ & wait for few minutes and then click on ‘volume’ to get the ETH-USD value.


APICONSUMER AT 0X615...31E2E
Balance: 0 ETH
acceptOw...
fulfill
bytes32 _requestId, uint256 _v
requestVo...
transferO...
address to
withdrawL...
owner
volume
0: uint256: 1931900000000000000000