How to Run a Watcher

By the end of this guide, you should know how to run a Watcher on your local machine, on VPS, or a bare-metal server. The guide is useful for enterprise clients who want to integrate with the OMG Network.

Goals

You should use this guide if you need to accomplish one of the following goals:

- Rely on a personal Watcher to verify transactions and receive network's events
- Host a redundant Watcher node to secure the network
- Be an active participant of the network

Prerequisites

1. Basic knowledge of Linux and blockchain technology
2. A laptop/PC, a Linux-based VPS, or a bare-metal server
3. A fully synced Ethereum client

Ethereum client is required to synchronize transactions on the OMG Network with the Ethereum Network. The easiest way to have a full ETH client is to use one of the Ethereum infrastructure providers. Our team has tried Infura and Geth but other providers may work too.

Supported Platforms

The Watcher has been tested on the following environments:

- Ubuntu 16.04
- Ubuntu 18.04
- Alpine 3.11
- macOS 11.0.0 (local usage only)

> Note, it might be possible to run a Watcher on other environments. Above are provided the systems that have been tested.

Minimum Hardware Requirements

The following hardware is required to run a Watcher:

- Storage: 8GB SSD
- CPU: 1 CPU Core with at least 2.2 GHz
- RAM: 4GB
- Bandwidth: 20 Mbps

> The requirements are based on the network's load in Q3 2020. It is recommended to use hardware with higher performance to avoid a potential increase in transaction volume.

Costs

The costs of running a Watcher include the following components:

- A full Ethereum node (local or ETH provider)
- VPS, bare-metal server, or local machine that matches the minimum hardware requirements
- DevOps setup and maintenance fee

Setup

Before installing Watcher service on a server or your local machine, you should have a few things setup:

1. Log in to Your Server

If you're planning to install a Watcher on a remote server (VPS or bare-metal), make sure to have an active session open before proceeding with the guide.

The configuration process takes a significant amount of time and may require help from your DevOps team. This step is fully covered in the Manage VPS guide.

You can log in using the following command from your terminal or command prompt:

```
ssh $USER@$REMOTE_SERVER -p $PORT
```

- `$USER` - the name of the user with root privileges used to log into the remote server. Default: root.
- `$REMOTE_SERVER` - an IP address of your remote server.
- `$PORT` - a port used to connect to the server. Default: 22.

2. Check TCP ports

Make sure you have don't have any of the services running on one of the following ports: 7434, 7534, 5432. The ports are used as follows:

- 7434: `watcher`, a light-weight Watcher to ensure the security of funds deposited into the childchain
- 7534: `watcher_info`, a convenient and performant API to the child chain data
- 5432: `postgres`, a PostgreSQL database that stores transactions and contains data needed for challenges and exits

You can use `lsof`, `netstat` or other alternatives to check occupied ports:

```
sudo lsof -i -n -P | grep LISTEN
```

If you found one of the ports is already in use, kill the process the port is occupied with as follows:

```
sudo lsof -t -i:$PORT
sudo kill -9 $PID
```

- `$PORT` - a port to clear from other processes
- `$PID` - process ID listening on a defined port

If you're already using Docker, verify these ports are free too:

```
docker ps
```

3. Update Packages

If you're using a Linux-based system, make sure to update your packages:

```
sudo apt-get update
```

Install

Running a Watcher locally is recommended for testing purposes only. For production, you should use VPS or a bare-metal server. This allows increasing uptime, reducing latency, and configuring advanced security measures for your instance.

Bare-metal (Ubuntu) | Docker Compose (Ubuntu) | Docker Compose (macOS)

This method shows how to install and run a Watcher from a bare-metal release of the `elixir-omg` repository. It is a recommended approach for production use. You can run it on VPS, dedicated, or bare-metal server on one of the supported environments.

1. Install Dependencies

1.1 Install Erlang and Elixir

The current implementation is built with Erlang and Elixir and uses `asdf` to manage multiple runtime versions. Asdf relies on several libraries that you may need to install first:

```
sudo apt-get install libssl-dev make automake autoconf libncurses5-dev gcc unzip
```

If you've used these libraries before, you may proceed to `asdf` installation as follows:

```
git clone https://github.com/asdf-vm/asdf.git ~/.asdf --branch v0.7.4
echo -e '\n. $HOME/.asdf/asdf.sh' >> ~/.bash_profile
echo -e '\n. $HOME/.asdf/completions/asdf.bash' >> ~/.bash_profile
source ~/.bash_profile

asdf plugin-add elixir https://github.com/asdf-vm/asdf-elixir.git
asdf plugin-add erlang https://github.com/asdf-vm/asdf-erlang.git

asdf install erlang 22.3
asdf install elixir 1.10.2
asdf global erlang 22.3
asdf global elixir 1.10.2

mix do local.hex --force, local.rebar --force
```

1.2 Install PostgreSQL

PostgreSQL is required to store Watcher's data, such as deposits, transactions, exits, and byzantine events.

```
apt-get install postgresql postgresql-contrib

sudo -u postgres psql <<EOF
  CREATE USER omisego_dev WITH CREATEDB ENCRYPTED PASSWORD 'omisego_dev';
  CREATE DATABASE omisego_dev OWNER 'omisego_dev';
EOF
```

2. Compile the Watcher

2.1 Clone `elixir-omg` Repository

```
git clone https://github.com/omgnetwork/elixir-omg.git
```

2.2 Setup the Project

```
cd elixir-omg
sh bin/setup
```

> This step may take up to 30 minutes.

2.3 Build the Watcher

```
make install-hex-rebar
make build-watcher_info-prod
```

3. Configure the Watcher

3.1 Configure an Environment File

There are several environmental variables the Watcher uses to run its service. To configure them, run the following command in your terminal:

```
echo "export ETHEREUM_NETWORK=MAINNET
export ETH_NODE=geth
export ETHEREUM_RPC_URL=$ETHEREUM_RPC_URL
export CHILD_CHAIN_URL=https://childchain.mainnet.v1.omg.network
export AUTHORITY_ADDRESS=0x22405c1782913fb676bc74ef54a60727b0e1026f
export TXHASH_CONTRACT=0x1c29b67acc33eba0d26f52a1e4d26625f52b53e6fbb0a4db915aeb052f7ec849
export CONTRACT_ADDRESS_PLASMA_FRAMEWORK=0x0d4c1222f5e839a911e2053860e45f18921d72ac
export DATABASE_URL=postgres://omisego_dev:omisego_dev@localhost:5432/omisego_dev
export ETHEREUM_EVENTS_CHECK_INTERVAL_MS=8000
export ETHEREUM_STALLED_SYNC_THRESHOLD_MS=300000
export ETHEREUM_BLOCK_TIME_SECONDS=15
export EXIT_PROCESSOR_SLA_MARGIN=5520
export EXIT_PROCESSOR_SLA_MARGIN_FORCED=TRUE
export NODE_HOST=127.0.0.1
export HOSTNAME=localhost
export PORT=7534
export DB_PATH=./data
export APP_ENV=local-development
export DD_DISABLED=true
export LOGGER_BACKEND=console" > ./env
```

- `$ETHEREUM_RPC_URL` - a full Ethereum node URL

Above are provided the values for `OMG NETWORK MAINNET BETA V1`. If you want to work with another environment, please refer to `environments`.

3.2 Apply Environment Variables

For the environment variables to take effect, run the command as follows:

```
source ./env
```

Note, these values live within your current shell's context. So you need to run this command again on system restart, exiting the shell, etc. To permanently set these values, check the following thread.

If you want to set up additional configurations, refer to deployment configuration document.

4. Install the Watcher

4.1 Create a Directory for Geth

```
mkdir data
```

4.2 Initialize Database

```
_build/prod/rel/watcher_info/bin/watcher_info eval "OMG.DB.ReleaseTasks.InitKeyValueDB.run()"
```

Example output:

```
2020-08-21 07:02:48.428 [info] module=OMG.DB.ReleaseTasks.InitKeyValueDB function=process/1 ·Creating database at "./dat
2020-08-21 07:02:48.534 [info] module=OMG.DB.ReleaseTasks.InitKeyValueDB function=init_kv_db/1 ·The database at "./data/
```

4.3 Migrate Database Tables

```
_build/prod/rel/watcher_info/bin/watcher_info eval "OMG.WatcherInfo.ReleaseTasks.InitPostgresqlDB.migrate()"
```

Example output:

```
2020-08-21 07:03:36.968 [info] module=Ecto.Migration.Runner function=log/2 ·create table blocks·
2020-08-21 07:03:37.008 [info] module=Ecto.Migration.Runner function=log/2 ·== Migrated 20180813131000 in 0.0s·
2020-08-21 07:03:37.081 [info] module=Ecto.Migration.Runner function=log/2 ·== Running 20180813131706 OMG.WatcherInfo.Re
2020-08-21 07:03:37.081 [info] module=Ecto.Migration.Runner function=log/2 ·create table transactions·
2020-08-21 07:03:37.115 [info] module=Ecto.Migration.Runner function=log/2 ·create index unq_transaction_blknum_txindex·
2020-08-21 07:03:37.125 [info] module=Ecto.Migration.Runner function=log/2 ·== Migrated 20180813131706 in 0.0s·
2020-08-21 07:03:37.140 [info] module=Ecto.Migration.Runner function=log/2 ·== Running 20180813133000 OMG.WatcherInfo.Re
2020-08-21 07:03:37.141 [info] module=Ecto.Migration.Runner function=log/2 ·create table ethevents·
2020-08-21 07:03:37.171 [info] module=Ecto.Migration.Runner function=log/2 ·== Migrated 20180813133000 in 0.0s·
2020-08-21 07:03:37.188 [info] module=Ecto.Migration.Runner function=log/2 ·== Running 20180813143343 OMG.WatcherInfo.Re
```

4.4 Run Ethereum Tasks

```
_build/prod/rel/watcher_info/bin/watcher_info eval "OMG.WatcherInfo.ReleaseTasks.EthereumTasks.run()"
```

Example output:

```
2020-08-21 07:06:57.631 [info] module=OMG.WatcherInfo.ExitConsumer function=init/1 ·Started OMG.WatcherInfo.ExitConsumer
2020-08-21 07:06:57.655 [info] module=OMG.WatcherInfo.ReleaseTasks.EthereumTasks function=run/0 ·Running Ethereum tasks.
2020-08-21 07:06:57.713 [info] module=OMG.WatcherInfo.ReleaseTasks.EthereumTasks.AddEthereumHeightToEthEvents function=r
```

5. Run the Watcher

You can start a Watcher as follows:

```
_build/prod/rel/watcher_info/bin/watcher_info start
```

But it's recommended to run this service in the background:

```
_build/prod/rel/watcher_info/bin/watcher_info daemon
```

Example output:

```
2020-08-21 08:17:33.195 [info] module=OMG.Watcher.BlockGetter function=handle_continue/2 ·Applied block: #2618000, from
2020-08-21 08:17:33.335 [info] module=OMG.Watcher.BlockGetter function=handle_continue/2 ·Applied block: #2619000, from
2020-08-21 08:17:33.465 [info] module=OMG.Watcher.BlockGetter function=handle_continue/2 ·Applied block: #2620000, from
2020-08-21 08:17:33.596 [info] module=OMG.Watcher.BlockGetter function=handle_continue/2 ·Applied block: #2621000, from
2020-08-21 08:17:33.721 [info] module=OMG.Watcher.BlockGetter function=handle_continue/2 ·Applied block: #2622000, from
2020-08-21 08:17:33.854 [info] module=OMG.Watcher.BlockGetter function=handle_continue/2 ·Applied block: #2623000, from
2020-08-21 08:17:33.986 [info] module=OMG.Watcher.BlockGetter function=handle_continue/2 ·Applied block: #2624000, from
2020-08-21 08:17:34.084 [info] module=OMG.WatcherInfo.DB.Block function=insert_from_pending_block/1 ·Block #2618000 pers
```

6. Update the Watcher

Frequently there will be updates that you need to apply to your Watcher. First, pull the changes from the repo as follows:

```
git pull
```

Then, repeat the steps starting from here.

Verify

To verify that you're fully synced, check the status of Watcher and Watcher Info:

Watcher Info

```
curl -X POST "http://$REMOTE_SERVER:7534/status.get"
```

- `$REMOTE_SERVER` - an IP address of your remote server. If you run the Watcher on your local machine, replace the value with the `localhost`.

Example output:

```
{
  "data": {
    "byzantine_events": [],
    "contract_addr": {
      "erc20_vault": "0x070cb1270a4b2ba53c81cef89d0fd584ed4f430b",
      "eth_vault": "0x3eed23ea148d356a72ca695dbce2fceb40a32ce0",
      "payment_exit_game": "0x48d7a6bbc428bca019a560cf3e8ea5364395aad3",
      "plasma_framework": "0x0d4c1222f5e839a911e2053860e45f18921d72ac"
    },
    "eth_syncing": false,
    "in_flight_exits": [],
    "last_mined_child_block_number": 2656000,
    "last_mined_child_block_timestamp": 1598017412,
    "last_seen_eth_block_number": 10704012,
    "last_seen_eth_block_timestamp": 1598018818,
    "last_validated_child_block_number": 2656000,
    "last_validated_child_block_timestamp": 1598017412,
    "services_synced_heights": [
      {
        "height": 10704012,
        "service": "block_getter"
      },
      {
        "height": 10703999,
        "service": "challenges_responds_processor"
      },
      {
        "height": 10704000,
        "service": "competitor_processor"
      },
      {
        "height": 10704002,
        "service": "depositor"
      },
      {
        "height": 10704000,
        "service": "exit_challenger"
      },
      {
        "height": 10704000,
        "service": "exit_finalizer"
      },
      {
        "height": 10704000,
        "service": "exit_processor"
      },
      {
        "height": 10703999,
        "service": "ife_exit_finalizer"
      },
      {
        "height": 10704000,
        "service": "in_flight_exit_processor"
      },
      {
        "height": 10703999,
        "service": "piggyback_challenges_processor"
      },
      {
        "height": 10704000,
        "service": "piggyback_processor"
      },
      {
        "height": 10704012,
        "service": "root_chain_height"
      }
    ]
  },
  "service_name": "watcher_info",
  "success": true,
  "version": "1.0.3+cbbc2dc"
}
```

Notice, the server may not respond until the following line appears in the `watcher_info` logs:

```
watcher_info_1   | 2020-05-30 06:13:36.445 [info] module=Phoenix.Endpoint.CowboyAdapter function=start_link/3 ·Running O
```

Copy

Watcher

```
curl -X POST "http://$REMOTE_SERVER:7434/status.get"
```

Copy

- `$REMOTE_SERVER` - an IP address of your remote server. If you run the Watcher on your local machine, replace the value with the `localhost`.

Example output:

```
{
  "data": {
    "byzantine_events": [],
    "contract_addr": {
      "erc20_vault": "0x070cb1270a4b2ba53c81cef89d0fd584ed4f430b",
      "eth_vault": "0x3eed23ea148d356a72ca695dbce2fceb40a32ce0",
      "payment_exit_game": "0x48d7a6bbc428bca019a560cf3e8ea5364395aad3",
      "plasma_framework": "0x0d4c1222f5e839a911e2053860e45f18921d72ac"
    },
    "eth_syncing": false,
    "in_flight_exits": [],
    "last_mined_child_block_number": 2656000,
    "last_mined_child_block_timestamp": 1598017412,
    "last_seen_eth_block_number": 10703998,
    "last_seen_eth_block_timestamp": 1598018636,
    "last_validated_child_block_number": 2656000,
    "last_validated_child_block_timestamp": 1598017412,
    "services_synced_heights": [
      {
        "height": 10703996,
        "service": "block_getter"
      },
      {
        "height": 10703984,
        "service": "challenges_responds_processor"
      },
      {
        "height": 10703986,
        "service": "competitor_processor"
      },
      {
        "height": 10703988,
        "service": "depositor"
      },
      {
        "height": 10703984,
        "service": "exit_challenger"
      },
      {
        "height": 10703984,
        "service": "exit_finalizer"
      },
      {
        "height": 10703984,
        "service": "exit_processor"
      },
      {
        "height": 10703984,
        "service": "ife_exit_finalizer"
      },
      {
        "height": 10703986,
        "service": "in_flight_exit_processor"
      },
      {
        "height": 10703984,
        "service": "piggyback_challenges_processor"
      },
      {
        "height": 10703986,
        "service": "piggyback_processor"
      },
      {
        "height": 10703998,
        "service": "root_chain_height"
      }
    ]
  },
  "service_name": "watcher",
  "success": true,
  "version": "1.0.3+cb41972"
}
```

Test

There are two ways to test that your Watcher is working properly:

1. Use `http://$REMOTE_SERVER:7534` as a `WATCHER_URL` value in your configs to make a transfer in your own or one of the OMG Network projects, such as omg-js samples.
2. Make a transaction or another operation using Watcher Info API.

> - `$REMOTE_SERVER` - an IP address of your remote server. If you run the Watcher on your local machine, replace the value with the `localhost`.