Manual Deployment Guide

Deploy Blockscout with a user-friendly UI and all microservices

This guide walks you through a new Blockscout deployment including the user-friendly UI frontend and installation of all microservices. If you'd prefer a more automated approach see thedocker-compose deployment page .

A. Prerequisites

Please familiarize yourself with thegeneral requirements ,db storage requirements ,JSON RPC requirements andClient setting requirements before installing Blockscout.

Minimum Local Hardware Requirements

- CPU: 4core / 8core
- RAM: 8GB / 16GB / 32GB
- DISK: 120gb or 500GB NVME SSD or Standart SSD
- OS: Linux, MacOS
-

Hosting Environment Hardware Requirements

If you are running Blockscout on a cloud server or other remote environment, see theHardware and Hosting Requirements

Software Dependencies

For Erlang/Elixir,asdf is recommended to install and set the appropriate versions. Note the supported versions for Erlang/Elixir/Node are specified in the.tool-versions file. Additional Instructions for setting up the environment are available forUbuntu andMacOS .

Dependency Mac Linux Erlang/OTP 25 brew install erlang Erlang Install Example Elixir 1.14.x brew install elixir Elixir Install Example Postgres 12, 13, 14 brew install postgresql Postgres Install Example Node.js 18.x.x brew install node Node.js Install Example Automake brew install automake Automake Install Example Libtool brew install libtool Libtool Install Example Inotify-tools Not Required Ubuntu -apt-get install inotify-tools GCC Compiler brew install gcc GCC Compiler Example GMP brew install gmp Install GMP Devel Make - sudo apt install make if Debian 9 G++ Compiler - sudo apt install g++ if Debian 9

B. Manual Deployment

The following guide contains 5 sections that cover a complete Blockcout installation.

1. Prepare the backend
2. Run microservices
3. Add the microservices integration to backend
4. Run the backend
5. Run the frontend
6.

1. Prepare the backend

1) Clone the repositorygit clone https://github.com/blockscout/blockscout blockscout-backend

2) Change directoriescd blockscout-backend

3) Provide DB URL with your usernameexport DATABASE_URL=postgresql://username:password@localhost:5432/blockscout

- Linux:
- Update the database username and password configuration
- Mac:
- Use logged-in user name and empty password (export DATABASE_URL=postgresql://username:@localhost:5432/blockscout
-)
- Optional:
- Change credentials inapps/explorer/config/test.exs
- for test envExample usage:
- Changing the default Postgres port from localhost:5432 iBoxen
- is installed.
-

You can check the regex pattern for the db url viahttps://regex101.com/ with the following regular expression:

```

Copy \w:*VV(?[a-zA-Z0-9_-])*:(?[a-zA-Z0-9-*#!%^&_.])?@(?(([a-zA-Z0-9]|[a-zA-Z0-9][a-zA-Z0-9-]*[a-zA-Z0-9]).)([A-Za-z0-9]|[A-Za-z0-9][A-Za-z0-9-]*[A-Za-z0-9])):(?\d+)*V(?[a-zA-Z0-9_-])*

```

? 4) Install Mix dependencies and compilemix do deps.get, local.rebar --force, deps.compile

5) Generate a new secret_key_base for the DBmix phx.gen.secret

6) Copy keybase and export as an env (for example)export SECRET_KEY_BASE=VTIB3uHDNbvrY0+60ZWgUoUBKDn9ppLR8MI4CpRz4/qLyEFs54ktJfaNT6Z221No

7) Export remainingenvironment variables as needed.

CLI basic example:

```

Copy export ETHEREUM_JSONRPC_VARIANT=geth export ETHEREUM_JSONRPC_HTTP_URL=http://localhost:8545 export API_V2_ENABLED=true export PORT=3001 # set for local API usage export COIN=yourcoin export COIN_NAME=yourcoinname export DISPLAY_TOKEN_ICONS=true

```

Notes:

- TheETHEREUM_JSONRPC_VARIANT
- will vary depending on your client (nethermind, geth etc)More information on client settings
- .
- If you're in production environment, please, setMIX_ENV=prod
- . The current default isMIX_ENV=dev
- which is a slower and less secure setting. However, for development purposes, unsetting or setting isMIX_ENV=dev
- is preferred.
- To configure “My Account
- ” section on the backend, seehttps://docs.blockscout.com/for-developers/configuration-options/my-account-settings
-

8) Compile the application:mix compile

9) If not already running, start Postgres:pg_ctl -D /usr/local/var/postgres start orbrew services start postgresql

Checkpostgres status :pg_isready 10) Create and migrate databasemix do ecto.create, ecto.migrate

If you are in dev environment and have run the application previously with a different blockchain, drop the previous database:mix do ecto.drop, ecto.create, ecto.migrate Be careful since this will delete all data from the DB. Don't execute it on production if you don't want to lose all of the data! 11) Install Node.js dependencies

Optional: If preferred, use npm ci rather than npm install to strictly follow all package versions in package-lock.json. cd apps/block_scout_web/assets; npm install &&

node_modules/webpack/bin/webpack.js --mode production; cd -

cd apps/explorer && npm install; cd -

12) Build static assets for deployment

mix phx.digest

13) Enable HTTPS in development. The Phoenix server only runs with HTTPS.

cd apps/block_scout_web; mix phx.gen.cert blockscout blockscout.local; cd -

14) Add blockscout and blockscout.local to your/etc/hosts

```

Copy 127.0.0.1 localhost blockscout blockscout.local

255.255.255.255 broadcasthost

::1 localhost blockscout blockscout.local

```

If using Chrome, Enablechrome://flags/#allow-insecure-localhost This completes the backend setup! Proceed to setup microservices.

1. Run Microservices

You will use Docker to run 4 Rust microservices: smart-contract verification, smart-contract sol2uml visualizer, sig-provider, and stats services. These add additional functionality to your instance once everything is connected.

Prerequisites

- Docker v20.10+
- Docker-compose 2.x.x+
-

Commands

1. Go to the docker-compose directorycd ./blockscout-backend/docker-compose
2. run docker-composedocker-compose -f microservices.yml up -d
3.

Stats

- Stats will be served fromhttp://localhost:8080/
- You can check, that service works by requestinghttp://localhost:8080/health
- . It should return{"status":"SERVING"}
-

sig-provider

- sig-provider will be athttp://localhost:8083/
- You can check, that service works by requestinghttp://localhost:8083/health
- . It should return{"status":"SERVING"}
-

Sc-visualizer

A visualizer for smart contracts.

- sc-visualizer will be lcoated athttp://localhost:8081/
- Check the visualizer service works by requesting thehttp://localhost:8081/health
- page - it should return{"status":"SERVING"}
- .
-

Sc-verifier

A separate smart contract verification service.

- sc-verifier will be athttp://localhost:8082/
- .
- Check that the sc-verifier service works by requestinghttp://localhost:8082/api/v2/verifier/solidity/versions
- page
-

it should return the list of compilers (click to see the sample response)```

Copy {"compilerVersions":
["v0.8.23+commit.f704f362","v0.8.22+commit.4fc1097e","v0.8.21+commit.d9974bed","v0.8.20+commit.a1b79de6","v0.8.19+commit.7dd6d404","v0.8.18+commit.87f61d96","v0.8.17+commit.8df45f5f","v0

``` * You can also use the Blockscout endpoint for smart-contract verification if you prefer (instructions in the integration section) *

To stop all microservices, rundocker-compose -f microservices.yml down To troubleshoot issues with a container, rundocker ps to check which containers are not starting.

Check logs withdocker logs visualizer -f

1. Add the microservices integration to the backend

Add the microservices env variables to the backend. Use the export command to add.

```

Copy export MICROSERVICE_SC_VERIFIER_ENABLED=true export MICROSERVICE_SC_VERIFIER_URL=http://localhost:8082/ export MICROSERVICE_VISUALIZE_SOL2UML_ENABLED=true
export MICROSERVICE_VISUALIZE_SOL2UML_URL=http://localhost:8081/ export MICROSERVICE_SIG_PROVIDER_ENABLED=true export
MICROSERVICE_SIG_PROVIDER_URL=http://localhost:8083/

```

The Blockscout team also provides an endpoint for smart-contract verification. To use, set the following for theMICROSERVICE_SC_VERIFIER envs ```

Copy export MICROSERVICE_SC_VERIFIER_ENABLED=true export MICROSERVICE_SC_VERIFIER_URL=https://eth-bytecode-db.services.blockscout.com/ export
MICROSERVICE_SC_VERIFIER_TYPE=eth_bytecode_db

```

This completes the microservices setup! Proceed to run the backend and frontend.

1. Run backend
2. Return to the blockscout-backend directory./blockscout-backend
3. Runmix phx.server
4.

The API will be available at http://localhost:3001/api/

1. Run frontend

The frontend can be added to the same high-level directory as the blockscout-backend or a different directory of your choice.

1. clone the blockscout frontend repositorygit clone https://github.com/blockscout/frontend blockscout-frontend
2. change directoriescd blockscout-frontend
3. create a .env file, for exampletouch .env
4. Add this minimal set of required env variables (additional variables are available here
5.)
6.

```

Copy NEXT_PUBLIC_API_HOST=localhost NEXT_PUBLIC_API_PORT=3001 NEXT_PUBLIC_API_PROTOCOL=http NEXT_PUBLIC_STATS_API_HOST=http://localhost:8080
NEXT_PUBLIC_VISUALIZE_API_HOST=http://localhost:8081 NEXT_PUBLIC_APP_HOST=localhost NEXT_PUBLIC_APP_PORT=3000 NEXT_PUBLIC_APP_INSTANCE=localhost
NEXT_PUBLIC_APP_ENV=development NEXT_PUBLIC_API_WEBSOCKET_PROTOCOL='ws'

```

1. install dependenciesyarn install
2. run frontendyarn dev
3.

Once completed, the frontend should be available at http://localhost:3000 Notes:

- To configue theMy Account
- section, you will add additional env variables on the frontend. Seehttps://github.com/blockscout/frontend/blob/main/docs/ENVS.md#my-account
- More info related to the frontend is available athttps://github.com/blockscout/frontend/blob/main/docs/CONTRIBUTING.md#local-development
-

Last updated1 month ago