Installation

Node Operator Checklist

Software Requirements

- Docker: Ensure that Docker is installed on your system. To download Docker, follow the instructions listedhere
- .
- Docker Compose: Make sure Docker Compose is also installed and properly configured. To download Docker Compose, follow the instructions listedhere
- .
- Linux Environment: EigenLayer is supported only on Linux. Ensure you have a Linux environment, such as Docker, for installation.* If you choose to install eigenlayer-cli using the Go programming language, ensure you have Go installed, version 1.21 or higher. You can find the installation guidehere
-
    - .

Checking for Requirements

On a native Linux system, you can use the lsb_release -a command to get information about your Linux distribution.

Check for Docker If you are not using a native Linux system and want to use EigenLayer, you can check if Docker is installed:

- Open a terminal or command prompt.
- Run the following command to check if Docker is installed and running:css

docker --version If Docker is installed and running, EigenLayer can be used within a Docker container, which provides a Linux environment.

By following these steps, you can determine if you have a suitable Linux environment for EigenLayer installation.

Installation

Install CLI using Binary

To download a binary for latest release, run:

curl -sSfL https://raw.githubusercontent.com/layr-labs/eigenlayer-cli/master/scripts/install.sh | sh -s The binary will be installed inside the~/bin directory.

To add the binary to your path, run:

export PATH=PATH:~/bin

Installing in custom location

To download the binary in a custom location, run:

curl -sSfL https://raw.githubusercontent.com/layr-labs/eigenlayer-cli/master/scripts/install.sh | sh -s -- -b

Install CLI using Go

Now we're going to install the eigenlayer-CLI using Go. The following command will install eigenlayer's executable along with the library and its dependencies in your system.

go install github.com/Layr-Labs/eigenlayer-cli/cmd/eigenlayer@latest To check if the GOBIN is not in your PATH, you can executeecho GOBIN from the Terminal. If it doesn't print anything, then it is not in your PATH. To add GOBIN to your PATH, add the following lines to your HOME/.profile:

export GOBIN=GOPATH/bin export PATH=GOBIN:PATH Changes made to a profile file may not apply until the next time you log into your computer. To apply the changes immediately, run the shell commands directly or execute them from the profile using a command such as source HOME/.profile.

Install CLI from Source

To pursue this installation method you need to have Go. Please ensure that you installed Go with a minimum version of 1.21here .

With this method, you generate the binary manually, downloading and compiling the source code.

git clone https://github.com/Layr-Labs/eigenlayer-cli.git cd eigenlayer-cli mkdir -p build go build -o build/eigenlayer cmd/eigenlayer/main.go or if you havemake installed:

git clone https://github.com/Layr-Labs/eigenlayer-cli.git cd eigenlayer make build The executable will be in the build folder.

In case you want the binary in your PATH (or if you used theGo method and you don't have GOBIN in your PATH), please copy the binary to /usr/local/bin:

Create and List Keys

ECDSA keypair corresponds to the operator Ethereum address and key for interacting with Eigenlayer. The BLS key is used for attestation purposes within the EigenLayer protocol. BLS key is used when you register an AVS to EigenLayer.

Create Keys

Generate encrypted ECDSA and BLS keys using the CLI:

eigenlayer operator keys create --key-type ecdsa [keyname] eigenlayer operator keys create --key-type bls [keyname] * [keyname] * - This will be the name

of the created key file. It will be saved as.ecdsa.key.json * or.bls.key.json * .

This will prompt a password which you can use to encrypt the keys. Keys will be stored in a local disk and will be shown once keys are created. It will also show the private key only once, so that you can back it up in case you lose the password or key file.

You can also create keys by piping your password to this command. This can help in automated key creation and will not prompt for password. This support got added inv0.6.2

echo "password" | eigenlayer operator keys create --key-type ecdsa [keyname]

Input Command

eigenlayer operator keys create --key-type ecdsa test The tool is requesting a password to encrypt the ECDSA private key for security purposes. The password input is hidden for security reasons.

Output

? Enter password to encrypt the ecdsa private key: ECDSA Private Key (Hex):
b3eba201405d5b5f7aaa9adf6bb734dc6c0f448ef64dd39df80ca2d92fca6d7b Please backup the above private key hex in safe place.

Key location: /home/ubuntu/.eigenlayer/operator_keys/test.ecdsa.key.json Public Key hex:
f87ee475109c2943038b3c006b8a004ee17bebf3357d10d8f63ef202c5c28723906533dccfda5d76c1da0a9f05cc6d32085ca1af8aaab5a28171474b1ad0aa68
Ethereum Address 0x6a8c0D554a694899041E52a91B4EC3Ff23d8aBD5

Import Keys

You can import existing ECDSA and BLS keys using the CLI, which are required for operator registration and other on-chain operations. This is useful if you already have an address which you want to use as your operator.

To import an ECDSA key, use the command:eigenlayer operator keys import --key-type ecdsa [keyname] [privatekey] .

To import a BLS key, use the command:eigenlayer operator keys import --key-type bls [keyname] [privatekey] .

- [keyname]
- is the name of the imported key file, and it will be saved asecdsa.key.json
- or.bls.key.json
- .
- privatekey
- is the private key of the key you wish to import.* For BLS keys, it should be a large number.
-
    - For ECDSA keys, it should be in hex format.

You can also import keys by piping your password to this command. This can help in automated key creation and will not prompt for password. This support got added inv0.6.2

echo "password" | eigenlayer operator keys import --key-type ecdsa [keyname] [privatekey]

Input Command

This part of the command tells the EigenLayer tool that you want to import a key.

eigenlayer operator keys import --key-type ecdsa test 6842fb8f5fa574d0482818b8a825a15c4d68f542693197f2c2497e3562f335f6

Output

This is a prompt asking you to enter a password to encrypt the ECDSA private key.

? Enter password to encrypt the ecdsa private key:* ECDSA Private Key (Hex):
6842fb8f5fa574d0482818b8a825a15c4d68f542693197f2c2497e3562f335f6 Please backup the above private key hex in safe place.

Key location: /home/ubuntu/.eigenlayer/operator_keys/test.ecdsa.key.json Public Key hex:
a30264c19cd7292d5153da9c9df58f81aced417e8587dd339021c45ee61f20d55f4c3d374d6f472d3a2c4382e2a9770db395d60756d3b3ea97e8c1f9013eb1bb
Ethereum Address 0x9F664973BF656d6077E66973c474cB58eD5E97E1 This will initiate a password prompt that you can use to encrypt the keys. The keys will be stored on your local disk and will be displayed after they are created.

The private key will also be shown only once, enabling you to create a backup in case you forget the password or lose the key file.

List Keys

This is the command you can use to retrieve a list of the keys you have created with the EigenLayer cli tool.

eigenlayer operator keys list When you run the Eigenlayer operator keys list command, it will display a list of all the keys that were generated using this specific command, along with their corresponding public keys.

This information can be useful for managing and identifying the keys you've created. Public keys are typically used for encryption, authentication, and verifying digital signatures.

Export keys

If you want to see the private key of the existing keys, you can use the below command. This will only work if your keys are in default location (~/.eigenlayer/operator_keys)

eigenlayer operator keys export --key-type ecdsa [keyname] This will also prompt for the password used to encrypt the key.

If your keys is not in the default location, you can give the full path to the key file using --key-path flag. You don't need to provide the key name in that case.

eigenlayer operator keys export --key-type ecdsa --key-path [path]

Fund ECDSA Wallet

Step 1: Follow the instructions inObtaining Testnet ETH to fund a web3 wallet with Goerli ETH.

Step 2: Sendat least 1 Goerli ETH to the “address” field referenced in your operator-config.yaml file. This ETH will be used to cover the gas cost for operator registration in the subsequent steps.

Operator Registration

Registration

Configuration Setup

You can create the config files needed for operator registration using the below command:

eigenlayer operator config create When prompted for operator address, make sure your operator address is same as the ecdsa key address you created/imported in key creation steps.

It will create two files:operator.yaml andmetadata.json After filling the details inmetadata.json , please upload this into a publicly accessible location and fill that url inoperator.yaml . A valid metadata url is required for successful registration. An example operator.yaml file is provided for your reference here:operator.yaml .

A public metadata url is required to register the operator. After updating yourmetadata.json file, you must upload it to a publicly accessible location and add that public url to youroperator.yaml file. You are also required to upload the image of the operator to a publicly accessible location and give the url in the metadata file. Operator registration only supports.png images for now.

warning Please ensure that themetadata.json file is hosted in a publicly accessible location. The metadata_url parameter in the operator configuration must be publicly accessible in order for registration to succeed. When using Github for hosting this file please ensure you link to the raw file (example), rather than the github repo URL (example). The EigenLayer CLI requires access to an Ethereum RPC node in order to post registration. Please plan to either leverage an RPC node provider or run your own local RPC node to reference in operator-config.yaml.

An example list of providers isavailable here for your reference. You may locate one usingChainlist Goerli .

Goerli Smart contract addresses

For operator registration in a Goerli environment, you need to set the Delegation Manager contract address as follows: The Operator CLI requires ECDSA keys for registration to EigenLayer. This is an Operator Ethereum address and key for interacting with Eigenlayer.

Delegation Manager:0x1b7b8F6b258f95Cf9596EabB9aa18B62940Eb0a8

EigenLayer Delegation Manager contract address

This will be provided by EigenLayer team

el_delegation_manager_address: 0x1b7b8F6b258f95Cf9596EabB9aa18B62940Eb0a8

Registration commands

This is the command you can use to register your operator.

Note: ECDSA key is required for operator registration. You may choose to eithercreate your own set of keys using the EigenLayer CLI (recommended for first time users) or import your existing keys (recommended for advanced users who already have keys created) as outlined in the previous section. eigenlayer operator register operator.yaml

Checking Status of Registration

This is the command you can use to inquire about the registration status of your operator.

eigenlayer operator status operator.yaml

Metadata Updates

This is the command you can use to make changes or updates to the metadata of your operator.

eigenlayer operator update operator.yaml Previous Introduction Next AVS Installation and Registration