Quickstart

The easiest way to get started with ethPM.

Installation

```
brew update
brew upgrade
brew tap ethpm/ethpm-cli
brew install ethpm-cli
```

Or in a fresh Python virtual environment.

```
pip install ethpm-cli
```

More installation details and options

Setting your environment

Infura (required)

ethPM currently uses Infura to talk to the blockchain, so you must provide an Infura API key to authenticate your requests. It's free and simple to sign up for a key. You can sign up for a key here and then use the following command to set your key as the environment variable `WEB3_INFURA_PROJECT_ID`.

```
export WEB3_INFURA_PROJECT_ID=abc123...xyz890
```

Wallet account (optional)

Setting a wallet account authentication is necessary to use commands that sign a transaction (i.e. deploying an ethPM registry, releasing an ethPM package to a registry). ethPM uses eth-keyfile to interact with your encrypted private keyfile and sign transactions. Follow the steps in the

README to generate your keyfile, and then use the following command to link your keyfile to ethPM.

```
ethpm auth --keyfile-path KEYFILE_PATH
```

Test that your keyfile has been properly stored.

```
ethpm auth
> Keyfile stored for address: 0x123abc...................
```

You can now use your keyfile's password with the flag `--keyfile-password` for any ethPM command, and it will be used to automatically sign any transactions.

Activate

The activate command is the simplest way to start interacting with ethPM. First, find a REGISTRY_URI for the package you want to activate, some popular registries can be found here. Then use the following command to "activate" the package using its ethpm URI. An IPython console will pop up in your terminal, automatically populated with all contract factories, deployments, and web3 instances, ready to use!

```
ethpm activate ethpm://packages.ethpm.eth/package@1.0.0
```

If you have authenticated a wallet account with `ethpm auth`, pass your password in with the `--keyfile-password` flag to automatically configure all contract factories, deployments and web3 instances to sign for your account.

```
ethpm activate ethpm://packages.ethpm.eth/package@1.0.0 --keyfile-password xxx
```

To instantly interact with any verified contract on Etherscan, use an Etherscan URI (though, this will require you setting you Etherscan API key to the environment variable: `ETHPM_CLI_ETHERSCAN_API_KEY`).

```
ethpm activate etherscan://0x123v3r1f13dc0ntractaddr3ss890:1
```

Install

The install command will install any ethPM package to a local `_ethpm_packages/` directory. Think of this directory like `node_modules/` in npm. The files are written to disk according to this scheme. By default, `ethpm install` will look for an `_ethpm_packages/` in the current working directory, but a specific `_ethpm_packages/` directory can be targeted if you pass in its path with the `--ethpm-dir` flag. If you want to install a package under an alias, you can use the `--alias` flag to do so. If you're installing an etherscan verified contract as a package, you **must** pass in `--package-name` and `--package-version` flags.

```
ethpm install ethpm://packages.ethpm.eth/package_name@1.0.0
```

List all installed packages.

```
ethpm list
```

Uninstall a package.

```
ethpm uninstall package_name
```

Create

To create your own ethPM package from local contracts requires compilation. If you don't have the Solidity Compiler installed on your machine, there are frameworks available to help with the compilation and automatically generate your ethPM package.

If you have the Solidity compiler installed on your machine, the best way to get started is with the manifest wizard. The wizard expects a project directory with the following structure.

- project/ - contracts/
    - xxx.sol
    - yyy.sol

Pass in a path to your project directory under the `--project-dir` flag. The wizard will attempt to compile these contracts using the available `solc` on your machine. The available `solc` version on your machine must be sufficient for compiling the project contracts. After compiliation, the

CLI will start the manifest wizard for complete package details.

```
ethpm create wizard --project-dir /path/to/project
```

Registry

ethPM packages are recorded on-chain using package registries. There is no central registry, and everybody who wants to release a package needs to deploy a registry on which they control what packages are released. In the CLI, there is a registry store to manage the different registies that you choose to interact with. If you want to store a registry under an alias, you can use the `--alias` flag to do so

```
ethpm registry list
```

```
ethpm registry add ethpm://ens.ethpm.eth:1 --alias my_favorite_registry
```

```
ethpm registry remove [URI_OR_ALIAS]
```

Active registries are used as the de-facto registry to release an ethPM package to. You can change the active registry with the following command.

```
ethpm registry activate [URI_OR_ALIAS]
```

Deploy

To deploy your own package registry, the following command is available. {link to code} This requires authentication via `ethpm auth`. Once deployed, you can check out your fresh registry on the ethPM explorer.

```
ethpm registry deploy --alias my_favorite_registry chain-id 1 --keyfile-password xxx
```

Release

To release a package to a registry is simple with the cli. First, make sure that the registry you want to release on is the active registry. You can confirm this with the `ethpm registry list` command.

```
ethpm release --package-name my_pkg --version 1.0.0 --manifest-uri ipfs://Qm... --keyfile-password
xxx
```

Now your brilliant smart contract ideas are available for the world to use!