

SSV SDK

A TypeScript SDK for interacting with the SSV (Secret Shared Validator) network, enabling distributed validator operations on Ethereum.

Installation

```
pnpm install ssv-sdk ssv-keys viem
# or
npm install ssv-sdk ssv-keys viem
# or
yarn add ssv-sdk ssv-keys viem
```

Usage

Initialize the SDK

```
import { SSVSDK } from 'ssv-sdk'

// Initialize with basic configuration
const sdk = new SSVSDK({
  chain: 'mainnet', // or holesky
  private_key: '0x.....',
})
```

API Interactions

```
const operators = await sdk.api.getOperators({
  operatorIds: ['220', '221', '223', '224'],
})

const nonce = await sdk.api.getOwnerNonce({
  owner: '0x',
})
```

Cluster management

```
// Get cluster balance
import { getClusterSnapshot } from '@utils/queries'
import { SSVSDK } from 'ssv-sdk'

// Initialize with basic configuration
const sdk = new SSVSDK({
  chain: 'mainnet', // or holesky
  private_key: '0x.....',
})

// Get Cluster
const cluster = await sdk.api.getCluster({
  id: '',
})

const balance = await sdk.contract.write({
  cluster: getClusterSnapshot(cluster),
  clusterOwner: '0x',
  operatorIds: operators.map((o) => BigInt(o.id)),
})
```

Cluster Management

```
import { parseEther } from 'viem'

await sdk.clusters.deposit({
  id: '...',
  amount: parseEther('1.5'),
  options: {
    approve: true, // Automatically triggers token approval transaction if the allowance is low
  },
})
```

Environment Setup for Testing

To run tests, you'll need to set up your environment variables. Create a `.env` file in the root directory of your project using the provided `.env.example` as a template:

```
# Copy the example env file
cp .env.example .env
```

The `.env` file should contain the following variables:

```
PRIVATE_KEY=your_private_key_here
OWNER_ADDRESS=your_owner_address_here
```

Make sure to never commit your actual `.env` file to version control. The `.env.example` file serves as a template showing which variables are required.