@push.rocks/smartssh

A library for setting up SSH configuration quickly and painlessly.

- readme.md for @push.rocks/smartssh

readme.md for @push.rocks/smartssh

setup SSH quickly and in a painless manner

Install

To begin using `@push.rocks/smartssh` in your project, you'll need to install it via npm or yarn. You can do so by running one of the following commands:

```
npm install @push.rocks/smartssh --save
```

or

```
yarn add @push.rocks/smartssh
```

Usage

`@push.rocks/smartssh` is a powerful package designed to simplify SSH configurations, key management, and interaction in a Typescript environment, using ESM syntax. This guide will cover how to utilize the primary functionalities provided by the package.

Setting Up an SSH Instance

An SSH instance represents your SSH configurations, including the keys and the SSH directory. Here's how to create an instance:

```
import { SshInstance } from '@push.rocks/smartssh';

const mySshInstance = new SshInstance({
  sshDirPath: '/path/to/.ssh', // Optional: specify SSH directory path
  sshSync: true, // Optional: keep the instance in sync with the SSH directory automatically
});
```

Working with SSH Keys

SSH keys can be managed using the `SshKey` class. You can add, remove, or retrieve keys from your SSH instance.

```
import { SshKey } from '@push.rocks/smartssh';

// Creating a new SSH key
const mySshKey = new SshKey({
  host: 'github.com', // Hostname
  private: 'privateKeyString', // Private key string
  public: 'publicKeyString', // Optional: public key string
  authorized: false // Optional: Is this key authorized? Defaults to false
});

// Adding the SSH key to the instance
mySshInstance.addKey(mySshKey);

// Getting an SSH key by host
const githubKey = mySshInstance.getKey('github.com');

// Removing an SSH key by instance
mySshInstance.removeKey(githubKey);
```

Syncing Keys with the File System

`@push.rocks/smartssh` makes it easy to synchronize your SSH keys with the file system, keeping your actual SSH configuration and your program state in alignment.

```
// To write the current state to the SSH directory
mySshInstance.writeToDisk();

// To read and synchronize the state from the SSH directory
mySshInstance.readFromDisk();
```

Advanced Key Management

- **Encoding and Decoding**: Keys can be encoded in `base64` for easier environment variable storage.
- **Key Type Detection**: The package can detect and handle private, public, or both keys present scenarios (`duplex`).

- **Custom SSH Directory**: Support for custom SSH directory locations.
- **Automatic Syncing**: Optionally keep the SSH instance automatically synced with the SSH directory on modifications.

Comprehensive Example

Below is a comprehensive example demonstrating SSH instance creation, adding a new SSH key, and syncing with the filesystem.

```
import { SshInstance, SshKey } from '@push.rocks/smartssh';

async function setupSsh() {
  // Initialize the SSH instance
  const sshInstance = new SshInstance({
    sshDirPath: '/custom/path/to/.ssh',
    sshSync: true,
  });

  // Create a new SSH key
  const newSshKey = new SshKey({
    host: 'my.custom.server.com',
    private: 'myPrivateKeyInBase64',
    public: 'myPublicKeyInBase64',
  });

  // Add the new key to the instance
  sshInstance.addKey(newSshKey);

  // Optionally, write to disk immediately
  sshInstance.writeToDisk();
}

// Running the SSH setup
setupSsh().then(() => {
  console.log('SSH setup complete.');
}).catch((error) => {
  console.error('SSH setup failed:', error);
});
```

This guide should provide a robust start to managing SSH configurations using `@push.rocks/smartssh`. Whether for individual projects or shared across a team, this package offers a streamlined approach to handling SSH keys, config synchronization, and more, all within a TypeScript project.

License and Legal Information

Trademarks

Company Information

Task Venture Capital GmbH
Registered at District court Bremen HRB 35230 HB, Germany

For any legal inquiries or if you require further information, please contact us via email at hello@task.vc.