@apiclient.xyz/hetznercloud

Documentation for @apiclient.xyz/hetznercloud

readme.md for @apiclient.xyz/hetznercloud

An unofficial API client for the Hetzner Cloud API

Install

You can install the `@apiclient.xyz/hetznercloud` package via npm:

```
npm install @apiclient.xyz/hetznercloud
```

Or using yarn:

```
yarn add @apiclient.xyz/hetznercloud
```

Usage

The `@apiclient.xyz/hetznercloud` package provides a modern approach to interact with the Hetzner Cloud API. Below are some detailed examples demonstrating the usage of this package for different scenarios like managing accounts, servers, volumes, and firewalls.

Initial Setup

To begin using the package, you need to import it and initialize your Hetzner account with an API token:

```
import { HetznerAccount, HetznerServer, Volume, HetznerFirewall } from
'@apiclient.xyz/hetznercloud';

// Initialize Hetzner account
const myHetznerAccount = new HetznerAccount('yourHetznerApiToken');
```

Managing Servers

Creating a Server

You can create a new server using the `createServer` method in the `HetznerAccount` class. Provide the necessary options such as name, type, location, labels, and optional user data:

```
const newServer = await myHetznerAccount.createServer({
  name: 'my-server',
  type: 'cpx31',
  location: 'nbg1',
  labels: {
    purpose: 'test'
  },
  userData: '#!/bin/bash\necho Hello from your new server!'
});

console.log('New server details:', newServer.data);
```

Listing Servers

To list all servers in your account, you can use the `getServers` method:

```
const servers = await myHetznerAccount.getServers();
console.log('Current servers:', servers);
```

Filtering Servers by Labels

You can filter servers based on specific labels:

```
const filteredServers = await myHetznerAccount.getServersByLabel({ purpose: 'test' });
console.log('Filtered servers:', filteredServers);
```

Deleting a Server

To delete a server, call the `delete` method on the server instance:

```
const serverToDelete = filteredServers[0]; // Example, choose the first filtered server
await serverToDelete.delete();
console.log('Server deleted successfully');
```

Managing Volumes

Creating a Volume

To create a new volume, use the `Volume.create` method, passing in the necessary options:

```
const newVolume = await Volume.create(myHetznerAccount, {
  name: 'my-volume',
  size: 10, // Size in GB
  location: 'nbg1',
  labels: {
    purpose: 'test-volume'
  },
  server: newServer // Attach the volume to a specific server (newServer in this case)
});

console.log('New volume details:', newVolume.data);
```

Listing Volumes

You can list all volumes in your account with the `getVolumes` method:

```
const volumes = await Volume.getVolumes(myHetznerAccount);
console.log('Current volumes:', volumes);
```

Filtering Volumes by Labels

To filter volumes based on specific labels:

```
const filteredVolumes = await Volume.getVolumesByLabel(myHetznerAccount, { purpose: 'test-volume' });
console.log('Filtered volumes:', filteredVolumes);
```

Deleting a Volume

To delete a volume, call the `delete` method on the volume instance:

```
const volumeToDelete = filteredVolumes[0]; // Example, choose the first filtered volume
await volumeToDelete.delete();
console.log('Volume deleted successfully');
```

Managing Firewalls

Creating a Firewall

To create a new firewall, use the `create` method in the `HetznerFirewall` class:

```
const newFirewall = await HetznerFirewall.create(myHetznerAccount, {
  name: 'my-firewall',
  labels: {
    purpose: 'test-firewall'
  },
  rules: [
    {
      direction: 'in',
      protocol: 'tcp',
      port: '80',
      source_ips: ['0.0.0.0/0', '::/0']
    },
    {
      direction: 'in',
      protocol: 'tcp',
      port: '443',
      source_ips: ['0.0.0.0/0', '::/0']
    }
  ]
});

console.log('New firewall details:', newFirewall.data);
```

Listing Firewalls

To list all firewalls in your account:

```
const firewalls = await HetznerFirewall.getFirewalls(myHetznerAccount);
console.log('Current firewalls:', firewalls);
```

Filtering Firewalls by Labels

To filter firewalls based on specific labels:

```
const filteredFirewalls = await HetznerFirewall.getFirewallsByLabel(myHetznerAccount, {
purpose: 'test-firewall' });
console.log('Filtered firewalls:', filteredFirewalls);
```

Deleting a Firewall

To delete a firewall, call the `delete` method on the firewall instance:

```
const firewallToDelete = filteredFirewalls[0]; // Example, choose the first filtered firewall
await firewallToDelete.delete();
console.log('Firewall deleted successfully');
```

Example: Full Lifecycle Management

Here is a complete example that demonstrates the full lifecycle of creating, listing, filtering, and deleting servers, volumes, and firewalls:

```
import { HetznerAccount, HetznerServer, Volume, HetznerFirewall } from
'@apiclient.xyz/hetznercloud';

// Initialize Hetzner account
const myHetznerAccount = new HetznerAccount('yourHetznerApiToken');

// Step 1: Create a new server
const newServer = await myHetznerAccount.createServer({
  name: 'my-server',
  type: 'cpx31',
  location: 'nbg1',
  labels: { purpose: 'test' },
  userData: '#!/bin/bash\necho Hello from your new server!'
});
console.log('New server created:', newServer.data);

// Step 2: Create a new volume and attach it to the server
const newVolume = await Volume.create(myHetznerAccount, {
  name: 'my-volume',
  size: 10, // GB
  location: 'nbg1',
  labels: { purpose: 'test-volume' },
```

```
  server: newServer
});
console.log('New volume created:', newVolume.data);

// Step 3: Create a new firewall and attach rules to it
const newFirewall = await HetznerFirewall.create(myHetznerAccount, {
  name: 'my-firewall',
  labels: { purpose: 'test-firewall' },
  rules: [
    { direction: 'in', protocol: 'tcp', port: '80', source_ips: ['0.0.0.0/0', '::/0'] },
    { direction: 'in', protocol: 'tcp', port: '443', source_ips: ['0.0.0.0/0', '::/0'] }
  ]
});
console.log('New firewall created:', newFirewall.data);

// Step 4: List all servers, volumes, and firewalls
const servers = await myHetznerAccount.getServers();
console.log('All servers:', servers);

const volumes = await Volume.getVolumes(myHetznerAccount);
console.log('All volumes:', volumes);

const firewalls = await HetznerFirewall.getFirewalls(myHetznerAccount);
console.log('All firewalls:', firewalls);

// Step 5: Filter servers, volumes, and firewalls by labels
const filteredServers = await myHetznerAccount.getServersByLabel({ purpose: 'test' });
console.log('Filtered servers:', filteredServers);

const filteredVolumes = await Volume.getVolumesByLabel(myHetznerAccount, { purpose: 'test-
volume' });
console.log('Filtered volumes:', filteredVolumes);

const filteredFirewalls = await HetznerFirewall.getFirewallsByLabel(myHetznerAccount, {
purpose: 'test-firewall' });
console.log('Filtered firewalls:', filteredFirewalls);

// Step 6: Cleanup - Delete created resources
for (const server of filteredServers) {
  await server.delete();
```

```
  console.log('Server deleted:', server.data);
}

for (const volume of filteredVolumes) {
  await volume.delete();
  console.log('Volume deleted:', volume.data);
}

for (const firewall of filteredFirewalls) {
  await firewall.delete();
  console.log('Firewall deleted:', firewall.data);
}
```

This comprehensive example demonstrates how you can manage your Hetzner resources using the `@apiclient.xyz/hetznercloud` package efficiently and effectively. undefined