

## Part 2: Interacting with JumpCloud

### Task 1: User and Group Management

**Objective:** Create users, groups, and associations using JavaScript.

#### 1. Create Two Users

**Script:**

```
const fetch = require('node-fetch'); // Only needed in Node.js

const API_KEY = 'API_KEY_HERE'; // Replace with your JumpCloud API key
const users = [
  {
    username: 'superman',
    email: 'superman@testing.com',
    firstname: 'Super',
    lastname: 'Man',
    activated: true
  },
  {
    username: 'batman',
    email: 'batman@testing.com',
    firstname: 'Bat',
    lastname: 'Man',
    activated: true
  }
];

async function createUser(user) {
  const response = await
fetch('https://console.jumpcloud.com/api/systemusers', {
  method: 'POST',
  headers: {
    'Accept': 'application/json',
    'Content-Type': 'application/json',
    'x-api-key': API_KEY
  },
  body: JSON.stringify(user)
});
```

```

const data = await response.json();
if (response.ok) {
  console.log(`User created and activated: ${user.username}`);
} else {
  console.error(`Failed to create ${user.username}:`, data);
}
}

async function createUsers() {
  for (const user of users) {
    await createUser(user);
  }
}

createUsers();

```

#### Result:

- superman@testing.com - UserID: 680f5df3decf82d9aa4acad7
  - batman@testing.com - UserID: 680f5df3507894ce192ac2f4
- 

## 2. Create User Group

#### Script:

```

const fetch = require('node-fetch'); // Required for making HTTP
requests in Node.js

const API_KEY = 'API_KEY_HERE'; // Replace with your JumpCloud API
key

// Function to create a user group
async function createUserGroup() {
  const response = await
fetch('https://console.jumpcloud.com/api/v2/usergroups', {
  method: 'POST',
  headers: {
    'Accept': 'application/json',
    'Content-Type': 'application/json',
    'x-api-key': API_KEY
  },

```

```

    body: JSON.stringify({
      name: 'TestUsers' // Name of the user group
    })
  });

  const data = await response.json();

  if (response.ok) {
    console.log('User group "TestUsers" created successfully:',
data);
  } else {
    console.error('Failed to create user group:', data);
  }
}

// Call the function to create the user group
createUserGroup();

```

#### Result:

- UserGroup Name: TestUser
  - ID: 680f620d2a99290001477467
- 

### 3. Bind Users to Group

#### Script:

```

const fetch = require('node-fetch'); // Required for making HTTP requests
in Node.js

const API_KEY = 'API_KEY'; // Replace with your JumpCloud API key
const GROUP_ID = '680f620d2a99290001477467'; // Replace with the actual
user group ID

// Array of user IDs to add to the group
const userIds = [
  '680f5df3decf82d9aa4acad7', // Replace with the first user's ID
  '680f5df3507894ce192ac2f4' // Replace with the second user's ID
];

// Function to bind users to a user group

```

```

async function bindUsersToGroup() {
  // Loop through the user IDs and send requests to bind each user
  for (let userId of userIds) {
    const response = await
fetch(`https://console.jumpcloud.com/api/v2/usergroups/${GROUP_ID}/members`
, {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
    'x-api-key': API_KEY
  },
  body: JSON.stringify({
    id: userId,      // The user ID to add
    op: 'add',       // Operation to add the user
    attributes: {},
    type: 'user'     // Type is 'user' since we're adding a user
  })
});

    const data = await response.json();
    if (response.ok) {
      console.log(`User ${userId} added to group ${GROUP_ID}`);
    } else {
      console.error(`Failed to add user ${userId} to group ${GROUP_ID}:`,
data);
    }
  }
}

// Call the function to bind users
bindUsersToGroup();

```

---

#### 4. Associate User to System

**Username:** superman

**UserID:** 680f5df3decf82d9aa4acad7

**SystemID:** 680f7699b209d8c423cfaee1

### Script:

```
const fetch = require('node-fetch'); // Required for making HTTP requests
in Node.js

const API_KEY = 'API_KEY_HERE'; // Replace with your JumpCloud API key
const SYSTEM_ID = '680f7699b209d8c423cfaee1'; // Replace with the actual
system ID
const USER_ID = '680f5df3decf82d9aa4acad7'; // Replace with the actual user
ID

// Function to associate a user with the system
async function addUserToSystem() {
  const response = await
fetch(`https://console.jumpcloud.com/api/v2/systems/${SYSTEM_ID}/associatio
ns`, {
  method: 'POST',
  headers: {
    'Accept': 'application/json',
    'Content-Type': 'application/json',
    'x-api-key': API_KEY
  },
  body: JSON.stringify({
    attributes: {
      sudo: {
        enabled: true,
        withoutPassword: false
      }
    },
    op: 'add',          // Operation to add the user
    type: 'user',       // Type is 'user' since we're associating a user
    id: USER_ID        // The actual user ID to add
  })
});

const data = await response.json();

if (response.ok) {
  console.log(`User ${USER_ID} added to system ${SYSTEM_ID}`);
} else {
  console.error(`Failed to add user ${USER_ID} to system ${SYSTEM_ID}:`,
data);
}
```

```
}  
  
// Call the function to add user to the system  
addUserToSystem();
```

## Task 2: LDAP Configuration

**Objective:** Configure LDAP and query users.

#/ldap/ldap\_server/63638c5fdb9edc000171f185/details

Alerts P \



JumpCloud LDAP

Details User Groups Users

**i** Multi-Factor Authentication can be enforced on your selection of LDAP Applications. Learn More: [Configuring MFA for LDAP](#).

### LDAP Instance

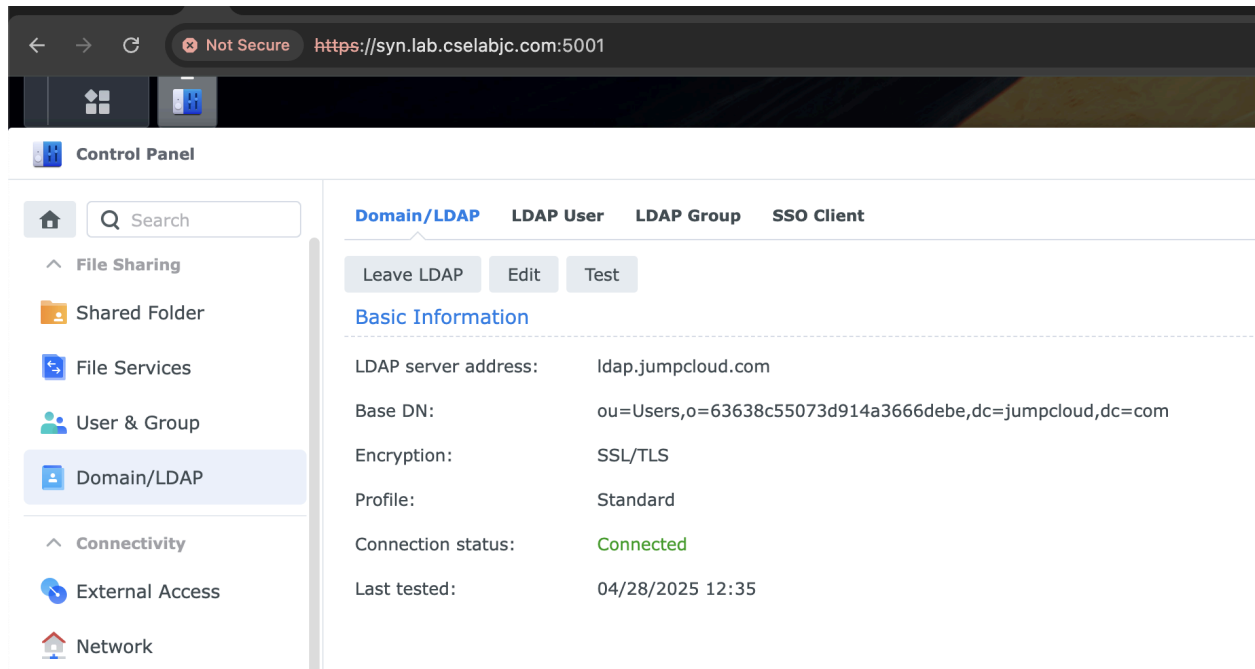
ORG DN

o=63638c55073d914a3666debe,dc=jumpcloud,dc=com

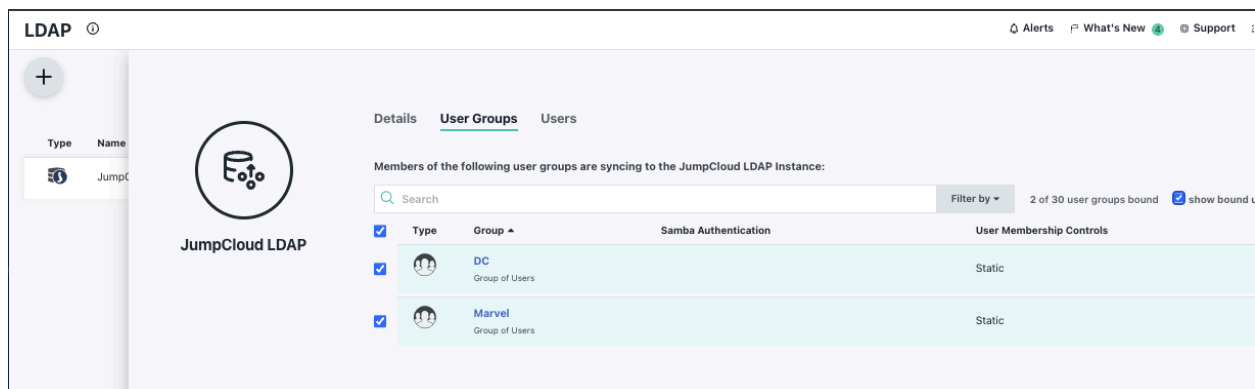
**i** To learn how to easily add, disable, and update user accounts. Read our [Knowledge Base article](#)


### LDAP Configuration

☐ Configure Samba Authentication **i**



Add two user groups to your LDAP directory, each user group containing at least 2 unique users






Marvel

[Details](#)
[Users](#)
[Device Groups](#)
[Applications](#)
[RADIUS](#)
[Directories](#)

Marvel has the following members:

<input checked="" type="checkbox"/>	User State	Name ▲	Email
<input checked="" type="checkbox"/>	Active	Man, Spider spiderman	spiderman@testing.com
<input checked="" type="checkbox"/>	Active	Man, Iron ironman	ironman@testing.com

D



DC

[Details](#)
[Users](#)
[Device Groups](#)
[Applications](#)
[RADIUS](#)
[Directories](#)

DC has the following members:

<input checked="" type="checkbox"/>	User State	Name ▲	Email
<input checked="" type="checkbox"/>	Active	Man, Bat batman	batman@testing.com
<input checked="" type="checkbox"/>	Active	Man, Super superman	superman@testing.com

## 1. LDAP Search (All Users)

Command:

```
ldapsearch -H ldaps://ldap.jumpcloud.com.com \
  -D "uid=superman,ou=Users,o=63638c55073d914a3666debe,dc=jumpcloud,dc=com" \
  -W \
  -b "ou=Users,o=63638c55073d914a3666debe,dc=jumpcloud,dc=com" \
  "(objectClass=inetOrgPerson)"
```



## Output:

```
# spiderman, Users, 63638c55073d914a3666debe, jumpcloud.com
dn: uid=spiderman,ou=Users,o=63638c55073d914a3666debe,dc=jumpcloud,dc=com
uidNumber: 5118
gidNumber: 5118
homeDirectory: /home/spiderman
mail: spiderman@testing.com
givenName: Spider
cn: Spider Man
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: shadowAccount
objectClass: posixAccount
objectClass: jumpcloudUser
uid: spiderman
loginShell: /bin/bash
sn: Man
jcLdapAdmin: TRUE
memberOf: cn=Marvel,ou=Users,o=63638c55073d914a3666debe,dc=jumpcloud,dc=com

# ironman, Users, 63638c55073d914a3666debe, jumpcloud.com
dn: uid=ironman,ou=Users,o=63638c55073d914a3666debe,dc=jumpcloud,dc=com
gidNumber: 5119
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: shadowAccount
objectClass: posixAccount
objectClass: jumpcloudUser
uidNumber: 5119
loginShell: /bin/bash
homeDirectory: /home/ironman
mail: ironman@testing.com
givenName: Iron
sn: Man
cn: Iron Man
uid: ironman
jcLdapAdmin: TRUE
memberOf: cn=Marvel,ou=Users,o=63638c55073d914a3666debe,dc=jumpcloud,dc=com
```

```
# batman, Users, 63638c55073d914a3666debe, jumpcloud.com
dn: uid=batman,ou=Users,o=63638c55073d914a3666debe,dc=jumpcloud,dc=com
uid: batman
gidNumber: 5117
homeDirectory: /home/batman
mail: batman@testing.com
givenName: Bat
cn: Bat Man
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: shadowAccount
objectClass: posixAccount
objectClass: jumpcloudUser
loginShell: /bin/bash
sn: Man
uidNumber: 5117
jcLdapAdmin: TRUE
memberOf: cn=DC,ou=Users,o=63638c55073d914a3666debe,dc=jumpcloud,dc=com
```

```
# superman, Users, 63638c55073d914a3666debe, jumpcloud.com
dn: uid=superman,ou=Users,o=63638c55073d914a3666debe,dc=jumpcloud,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: shadowAccount
objectClass: posixAccount
objectClass: jumpcloudUser
uid: superman
uidNumber: 5116
homeDirectory: /home/superman
sn: Man
gidNumber: 5116
loginShell: /bin/bash
mail: superman@testing.com
givenName: Super
cn: Super Man
jcLdapAdmin: TRUE
memberOf: cn=DC,ou=Users,o=63638c55073d914a3666debe,dc=jumpcloud,dc=com
```

## 2. LDAP Search (Specific Group - Email/First/Last Name)

Command:

```
ldapsearch -H ldaps://ldap.jumpcloud.com \  
  -D "uid=superman,ou=Users,o=63638c55073d914a3666debe,dc=jumpcloud,dc=com" \  
  -W \  
  -b "ou=Users,o=63638c55073d914a3666debe,dc=jumpcloud,dc=com" \  
  "(&(objectClass=inetOrgPerson)(memberOf=cn=DC,ou=users,o=63638c55073d914a3666debe,dc=jumpcloud,dc=com))" \  
  mail givenName sn
```

Output:

```
# batman, Users, 63638c55073d914a3666debe, jumpcloud.com  
dn: uid=batman,ou=Users,o=63638c55073d914a3666debe,dc=jumpcloud,dc=com  
mail: batman@testing.com  
givenName: Bat  
sn: Man  
  
# superman, Users, 63638c55073d914a3666debe, jumpcloud.com  
dn: uid=superman,ou=Users,o=63638c55073d914a3666debe,dc=jumpcloud,dc=com  
sn: Man  
mail: superman@testing.com  
givenName: Super
```