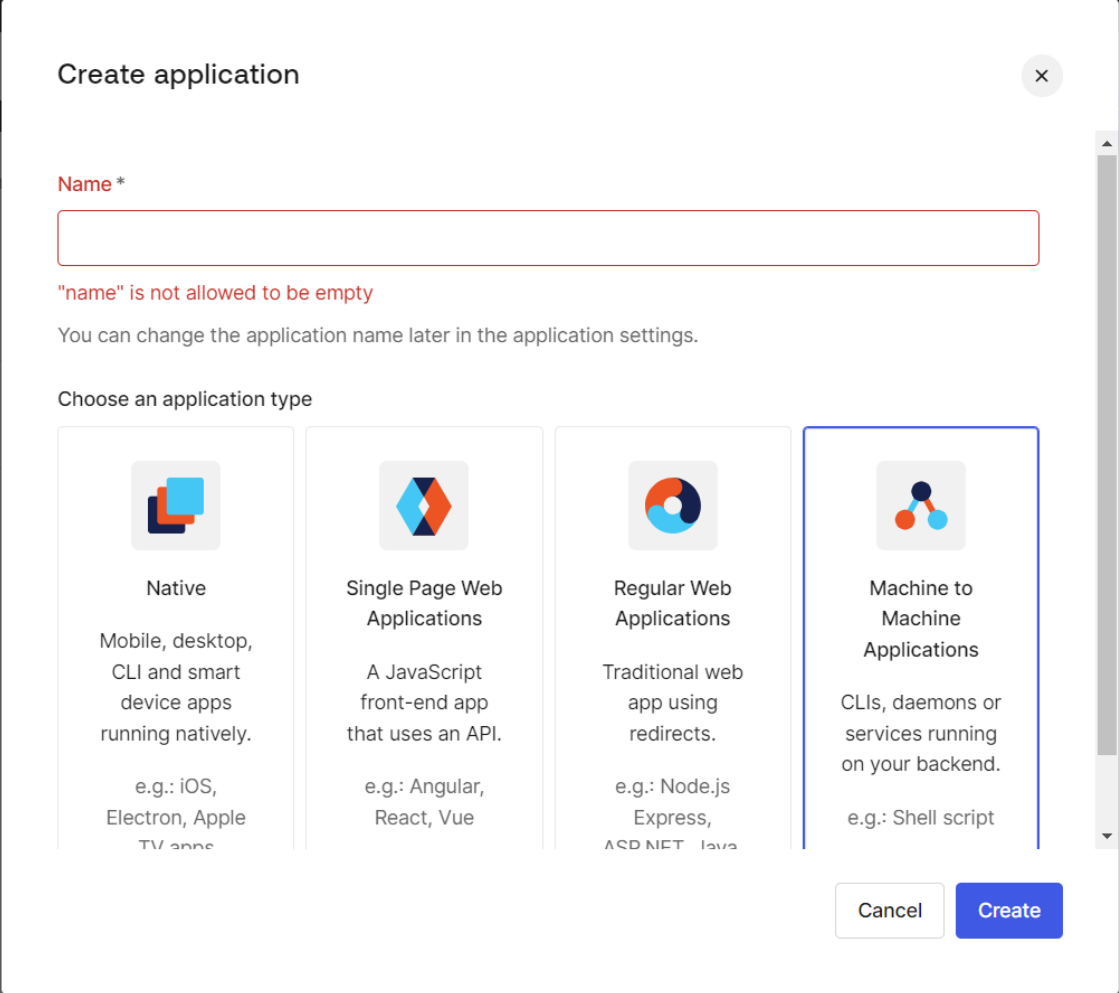


## Documentation on how to manage users using Auth0.

1. Set up an Auth0 account and create an application.

I have created an auth account and created a Machine to Machine application called Cybyte\_Task1



The image shows a screenshot of the Auth0 'Create application' dialog box. The dialog has a title bar with a close button (X). Below the title, there is a 'Name' field with a red asterisk indicating it is required. The field is empty, and a red error message below it states: '"name" is not allowed to be empty'. A note below the error message says: 'You can change the application name later in the application settings.' Below this, there is a section titled 'Choose an application type' with four options, each with an icon and a description. The 'Machine to Machine Applications' option is selected, indicated by a blue border around its card. At the bottom right, there are 'Cancel' and 'Create' buttons.





**Create application**

**Name \***

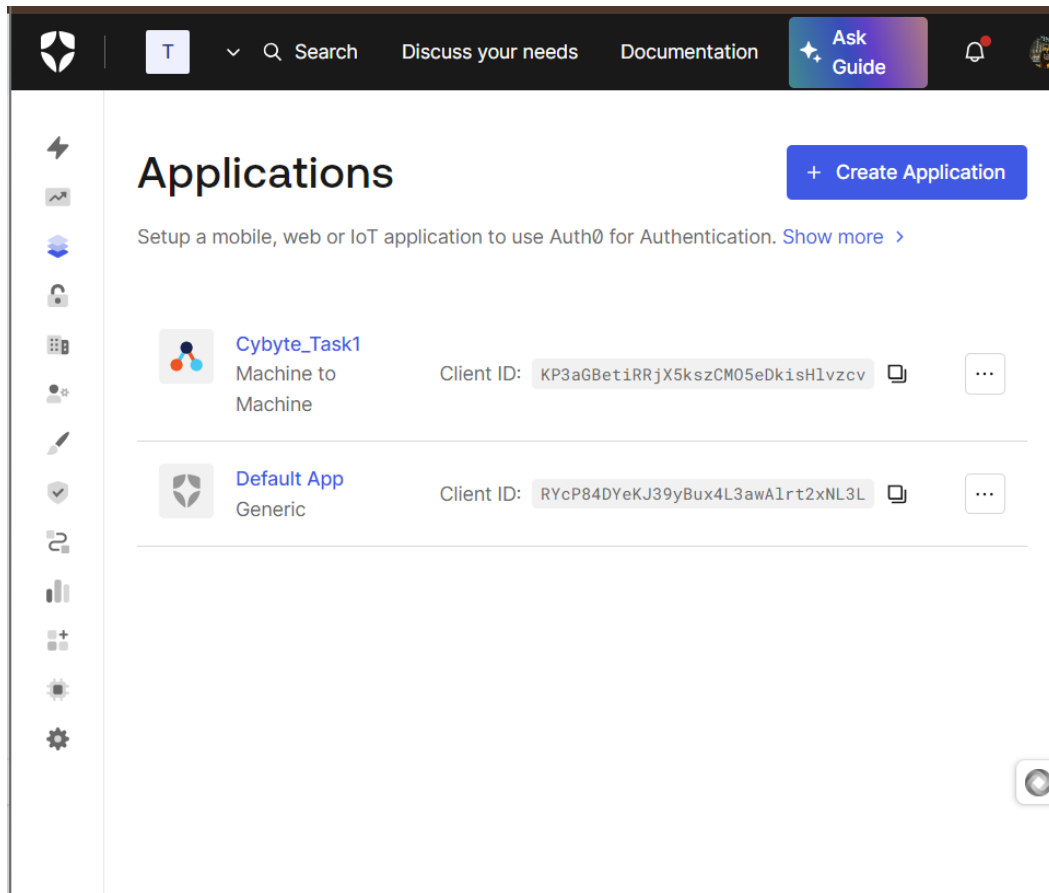
"name" is not allowed to be empty

You can change the application name later in the application settings.

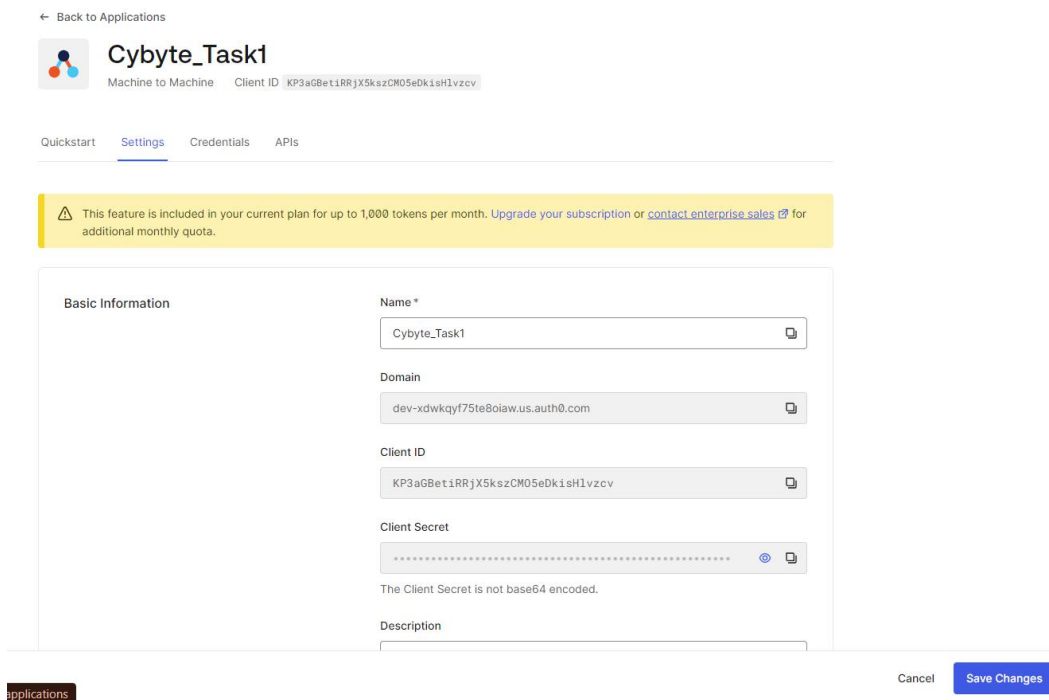
**Choose an application type**

 <b>Native</b> Mobile, desktop, CLI and smart device apps running natively. e.g.: iOS, Electron, Apple TV apps	 <b>Single Page Web Applications</b> A JavaScript front-end app that uses an API. e.g.: Angular, React, Vue	 <b>Regular Web Applications</b> Traditional web app using redirects. e.g.: Node.js, Express, ASP.NET, Java	 <b>Machine to Machine Applications</b> CLIs, daemons or services running on your backend. e.g.: Shell script
---	--	---	--

**Cancel** **Create**



These are the information which I will be using in further to create users



## 2. Use the Auth0 Management API to create users.

here I have written a function to get the access token

```
// Function to get Auth0 Management API access token
export async function getAuth0AccessToken() {
  try {
    const response = await axios.post(
      `https://${AUTH0_DOMAIN}/oauth/token`,
      {
        client_id: AUTH0_CLIENT_ID,
        client_secret: AUTH0_CLIENT_SECRET,
        audience: AUTH0_API_AUDIENCE,
        grant_type: 'client_credentials',
        scope: 'create:users'
      },
      { headers: { 'Content-Type': 'application/json' } }
    );
    return response.data.access_token;
  } catch (error) {
    console.error('Error getting Auth0 access token:', error);
    throw error;
  }
}
```

Using this token, I written a function to create the user by authentication the token

```
// Function to create a user in Auth0
export async function createUser(userData) {
  let token;
  try {
    const token = await getAuth0AccessToken();

    const response = await axios.post(
      `https://${AUTH0_DOMAIN}/api/v2/users`,
      userData,
      { headers: { Authorization: `Bearer ${token}`, 'Content-Type': 'application/json' } }
    );
    // console.log(response.data);
    console.log(token);
    return response.data;
  } catch (error) {
    console.log(token);
    console.error('Error creating user in Auth0:', error.response?.data || error.message);

    // console.error('Error creating user in Auth0:', error);
    throw error;
  }
}
```

3. Ensure you can authenticate and authorize users to interact with both databases.

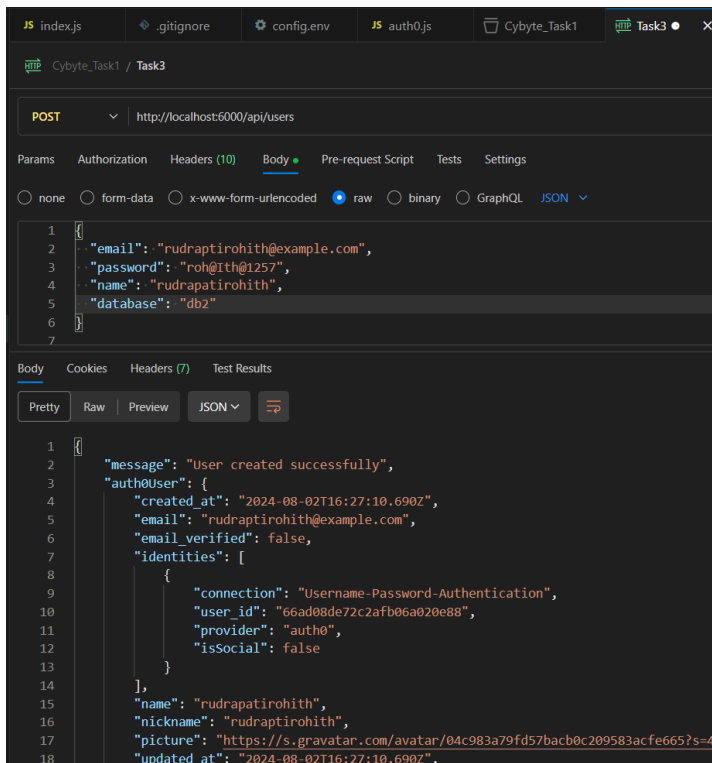
Now when I post req in `http://localhost:6000/api/users` I could insert to db1 by creating new user if I gave db1 as database in postman and I could insert to db2 by creating new user if I gave db2 as database in postman.

```
// Route to create a user in Auth0 and insert into a database
app.post('/api/users', async (req, res) => {
  const { name, email, password, database } = req.body;

  try {
    const user = await createUser({
      connection: 'Username-Password-Authentication',
      email,
      password,
      name
    });

    const db = database === 'db1' ? db1 : db2;
    await db.query('INSERT INTO users (name, email, password) VALUES (?, ?, ?)', [name, email, password]);

    res.status(201).json({ message: 'User created successfully', auth0User: user });
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});
```



I have created users one had inserted the data in the body to db1 and one had inserted into body2 in my 2<sup>nd</sup> post req.

this was my first post api call worked on db1 by creating user

```
mysql> USE DB1
Database changed
mysql> select * from users
-> ;
```

id	name	email	password
1	Amit Sharma	amit.sharma@example.com	AmitSharma@1257
2	Priya Singh	priya.singh@example.com	PriyaSingh@1257
3	Rahul Verma	rahul.verma@example.com	RahulVerma@1257
4	Neha Gupta	neha.gupta@example.com	NehaGupta@1257
5	John Smith	john.smith@example.com	securepassword
6	John Doe	johndoe@example.com	john@doe@1257

```
6 rows in set (0.01 sec)
```

this is my 2<sup>nd</sup> post api call worked on db2 by creating new user using the token

```
mysql> USE DB2
Database changed
mysql> select * from users;
```

id	name	email	password
1	Vikas Patel	vikas.patel@example.com	VikasPatel@1257
2	Sana Khan	sana.khan@example.com	SanaKhan@1257
3	Rohit Mehta	rohit.mehta@example.com	RohitMehta@1257
4	Anjali Reddy	anjali.reddy@example.com	AnjaliReddy@1257
5	Rohith	rohithh@example.com	rohu@th@1257
6	Rohith12	rohith@example.com	roh@Ith@1257
7	rudrapatirohith	rudraptirohith@example.com	roh@Ith@1257

```
7 rows in set (0.00 sec)
```

The users which I have created in auth0 are

Users

+ Create User

An easy to use UI to help administrators manage user identities including password resets, creating and provisioning, blocking and deleting users.

Show more >

Q Search for users

Search by: User

✕ Reset

Name	Connection	Logins	Latest Login ▾
<div>RU</div> <div>rudrapatirohith</div> <div>rudraptirohith@example.com</div>	Username-Password-Authenti...	0	never
<div>RO</div> <div>Rohith12</div> <div>rohith@example.com</div>	Username-Password-Authenti...	0	never
<div>RO</div> <div>Rohith</div> <div>rohithh@example.com</div>	Username-Password-Authenti...	0	never
<div>JD</div> <div>John Doe</div> <div>johndoe@example.com</div>	Username-Password-Authenti...	0	never