Netbird

Eine Alternative zu ZeroTier einer Alternative zu klassischen VPNs. Es ermöglicht Overlay-Netzwerke, Peer-2-Peer Verbindungen zwischen den Clients, die Verbindungen basieren auf Wireguard, sodass es auf dessen Stärken (z. B. Linux Kernel Integration) profitiert. Als Self-Hosted kostenlos und 100% OpenSource.

Übersicht

Bei Netbird handelt es sich um ein OpenSource Netzwerk-Overlay. Es ist vollständig OpenSource und kann kostenlos selbst gehosted werden. Für die VPN-Verbindungen wird Wireguard als Basis genutzt, sodass es von den Vorteilen profitieren kann, wie z. B. stabile Verbindungen, hohe Performance und Linux Kernel Integration. Im Gegensatz zu den klassischen VPNs werden die Clients in Netzwerken miteinander verbunden und können direkt miteinander kommunizieren, sodass nicht der gesamte Traffic durch den VPN-Server fließen muss. Es können verschiedene Regeln definiert werden und die Netzwerk lassen sich trennen und verbinden, sodass jederzeit festgelegt werden kann, welcher Clients untereinander kommunizieren dürfen. Diese Regeln werden zentral vom Server aus gesteuert und von den Clients ruck zuck übernommen.

Der Quelltext sowie diverse Dokumentation und einige Einblicke können im Netbird GitHub eingesehen werden.

Neben der selbst gehosteten Variante bietet Netbird auch einen Cloud Dienst an: Netbird Homepage.

Server Installation

Die Installation des Netbird Servers, über diesen können die Clients ihre Einstellungen beziehen und miteinander verbunden werden. Dieser muss als einziger Teilnehmer öffentlich erreichbar und auf einer festen Domäne sein.

Im folgenden wird die Anleitung der offiziellen Dokumentation von Netbird verwendet: Netbird Advanced Guide

Code von GitHub laden

Bevor der Code heruntergeladen wird, empfiehlt es sich einen Ordner zu wählen, in dem später die Konfigurationsdateien und Daten gespeichert werden sollen. Anschließend in das Verzeichnis wechseln.

```
mkdir /pfad/zu/netbird
cd /pfad/zu/netbird
```

Dann laden wir uns Netbird samt Konfiguration von GitHub.

```
#!/bin/bash
# Die URL des Repositorys in einer Variable speichern
REPO="https://github.com/netbirdio/netbird/"
# Hiermit wird die neuste Version (latest) ermittelt und als Variable gesetzt
LATEST_TAG=$(basename $(curl -fs -o/dev/null -w %{redirect_url} ${REPO}releases/latest))
# Die Version ausgeben
echo $LATEST_TAG
# Das Repositiory von GitHub herunterladen
git clone --depth 1 --branch $LATEST_TAG $REPO
```

Es werden nicht alle heruntergeladenen Daten benötigt, also schieben wir uns die benötigten Dateien direkt passend ins Verzeichnis und löschen den Rest.

```
mv netbird/infrastructure_files/* .
rm -r netbird/
```

Identitätsanbieter konfigurieren

Der Identitätsanbieter kümmert sich um die Authentifizierung und Autorisierung der Benutzer. Hier setzt Netbird auf bereits etablierte Anbieter, die OpenSource und kostenlos für Self-Hosting zur Verfügung stehen. In diesem Fall wird Zitadel gewählt.

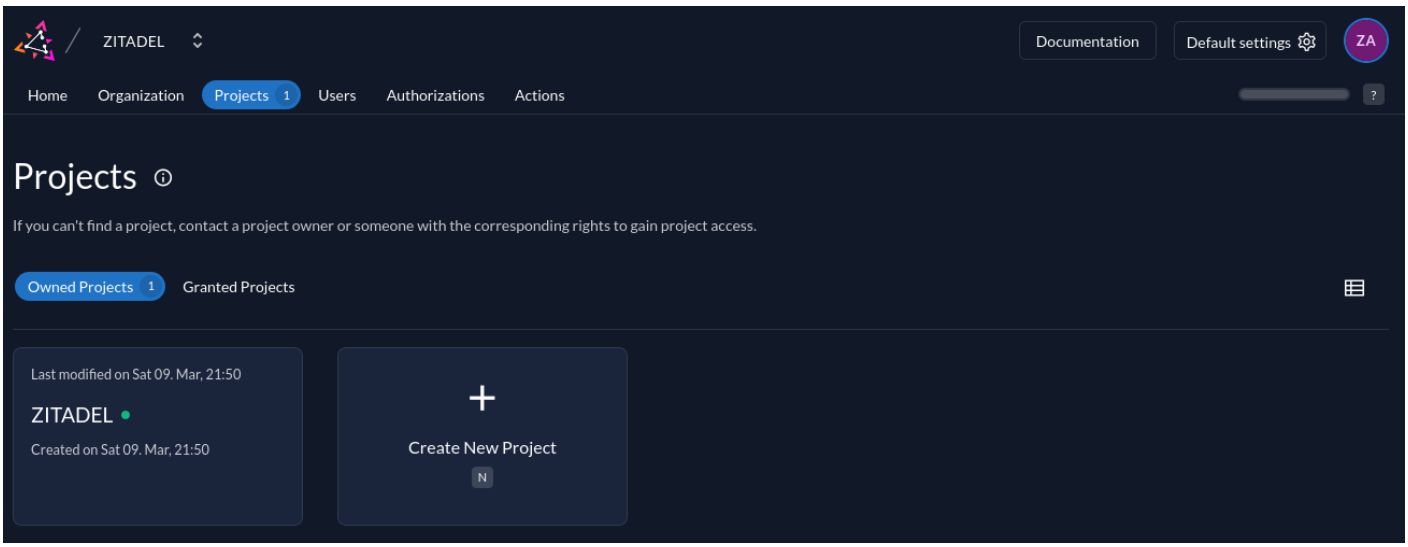
>

> Anstelle von Zitadel könnten auch Keycloak oder Authentik verwendet werden. Alle 3werden offiziell von Netbird unterstützt. Siehe hierzu: Identity Providers

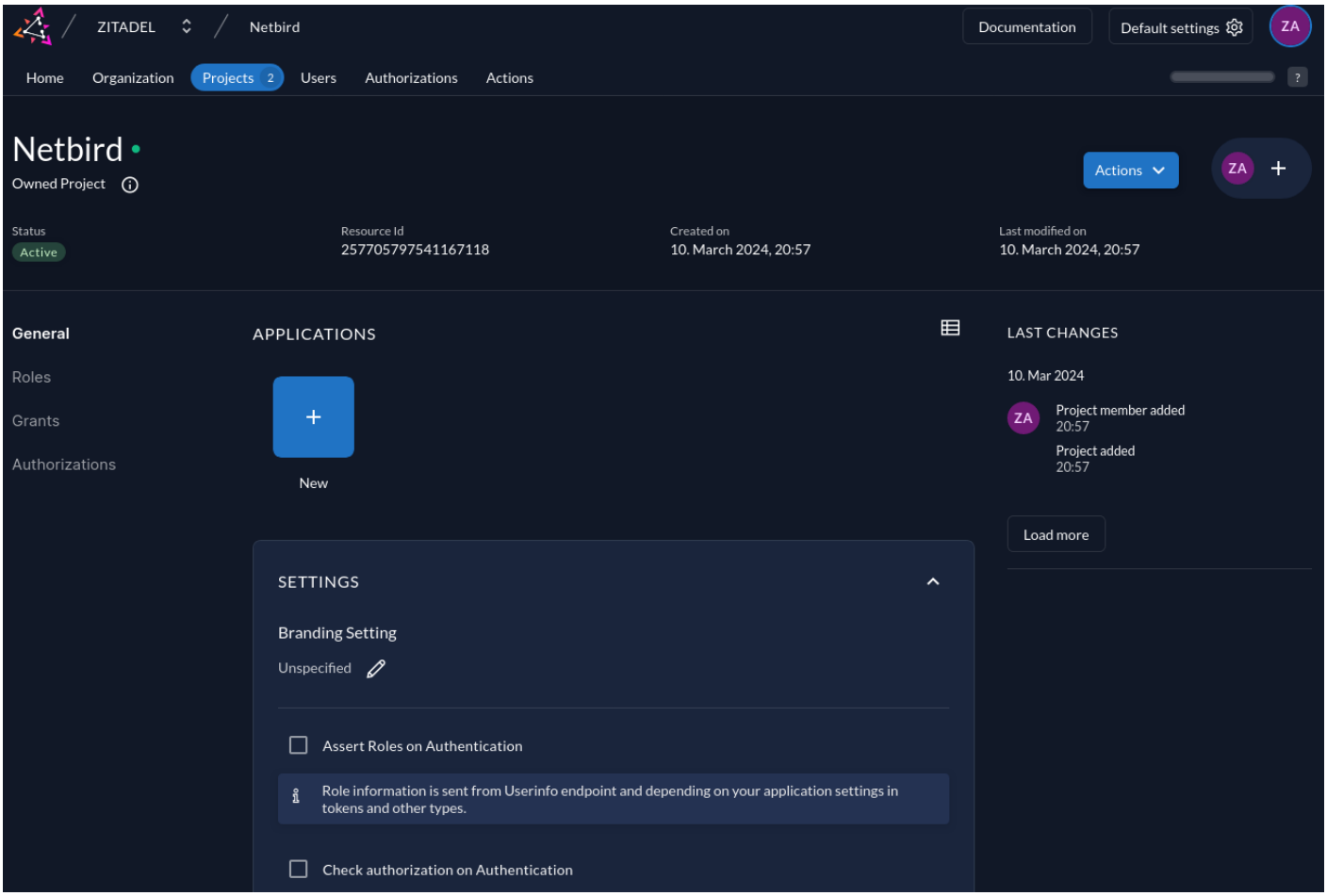
Im folgenden wird davon ausgegangen, dass Zitadel bereits eingerichtet wurde und ausgeführt wird.

Eine Anleitung, wie Zitadel mit Docker bereitgestellt werden kann ist hier: Buch Zitadel

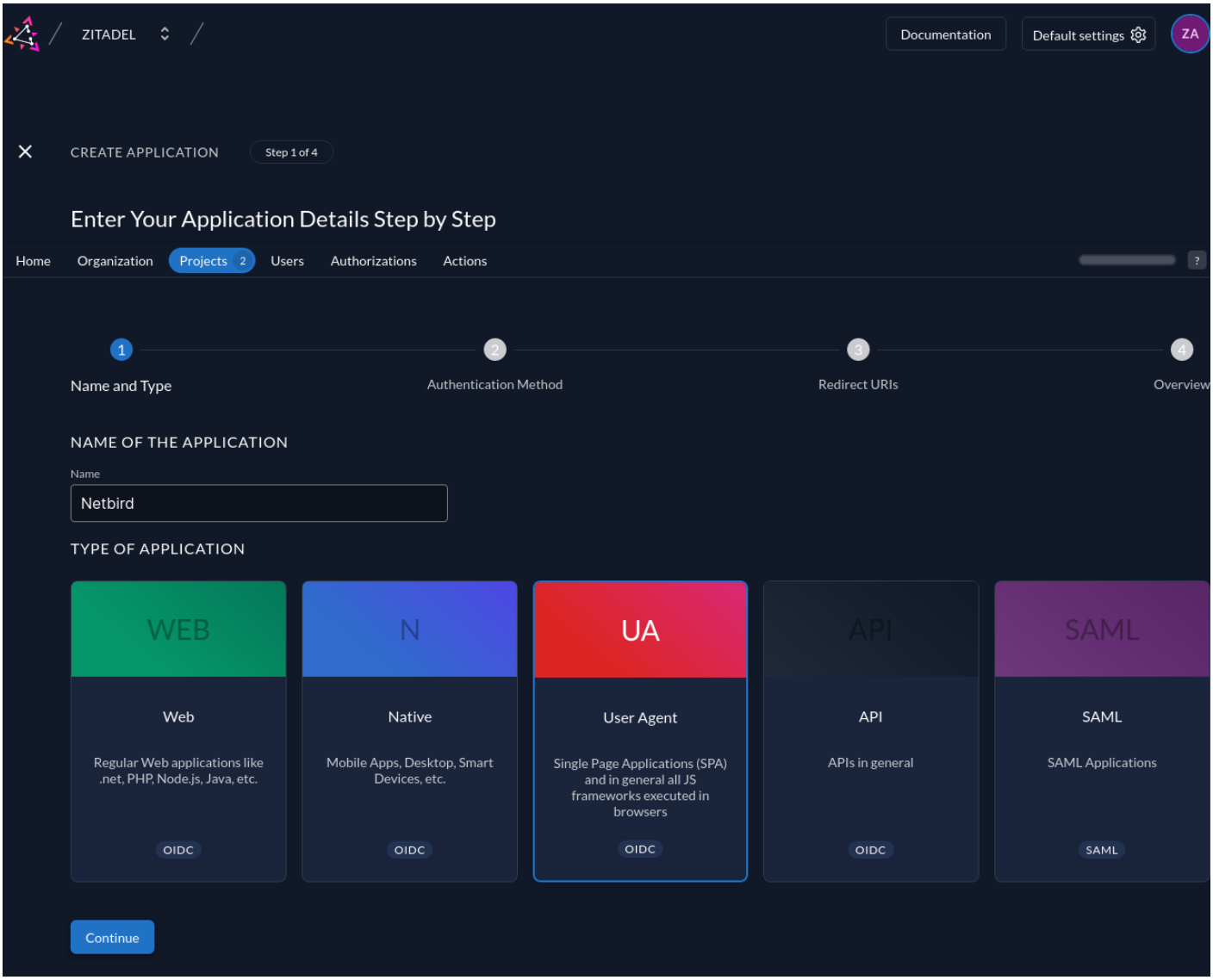
Zunächst wird ein neues Projekt erstellt. In der oberen Menüleiste auf *Projects* klicken und dann auf *Create New Project*.



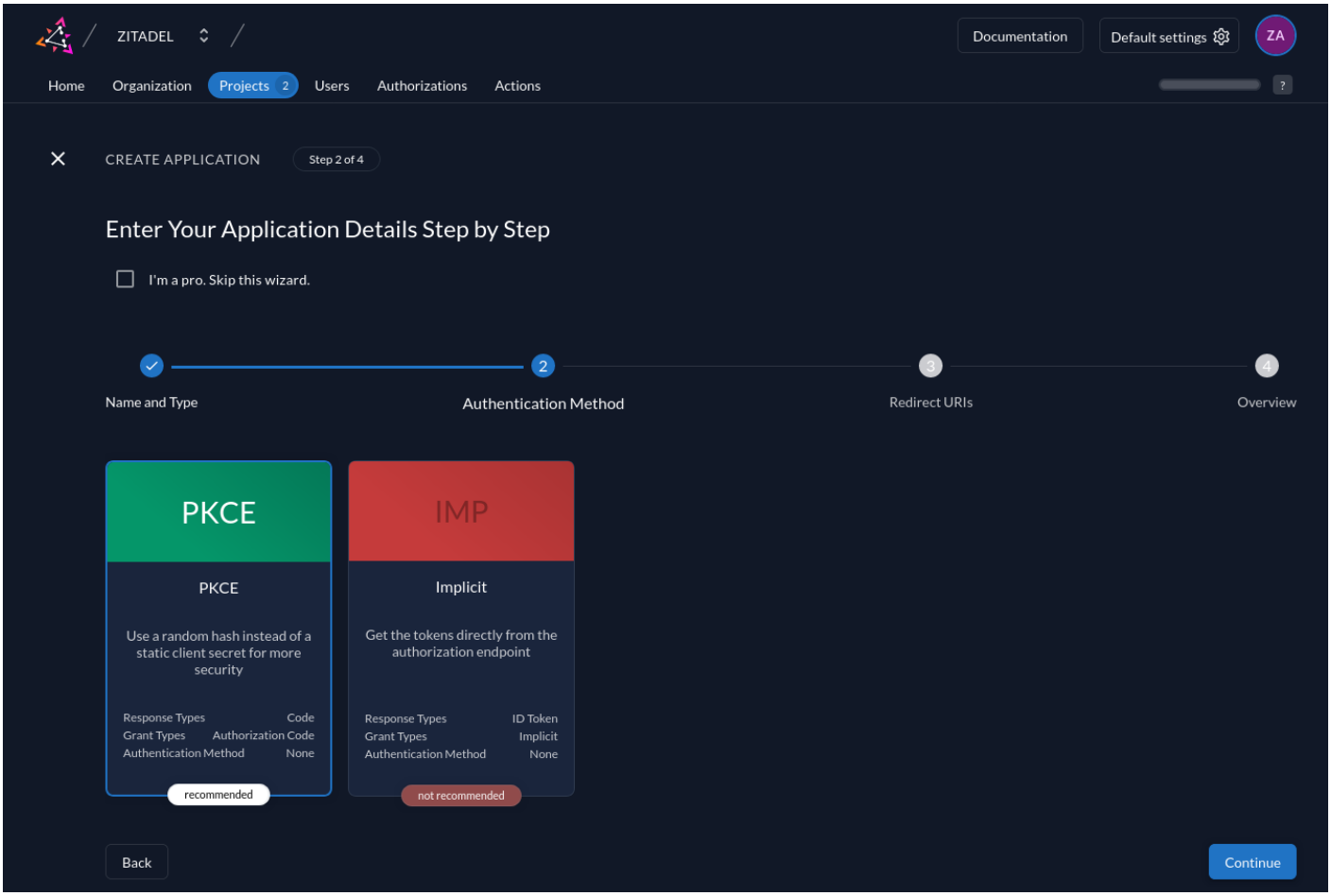
Es wird nur ein Name für das Projekt abgefragt, hier z. B. Netbird eingeben, es kann selbst ein passender Name eingegeben werden. Nachdem der Name bestätigt wurde, erstellt Zitadel das Projekt und leitet direkt zu diesem weiter. Die Seite sieht in etwa wie folgt aus.



Hier wird nun unter *Applications* auf *New* geklickt, um eine neue Applikation anzulegen. Nun öffnet sich ein Dialog, um die Applikation zu erstellen. Als erstes wird ein Name vergeben und der Typ ausgewählt. Der Name kann frei gewählt werden und der Typ ist auf *User Agent* festzulegen. Die Konfiguration mit *Continue* bestätigen.



Danach wird der Typ noch spezifiziert, hier wird *PKCE* ausgewählt und wieder mit *Continue* bestätigt.



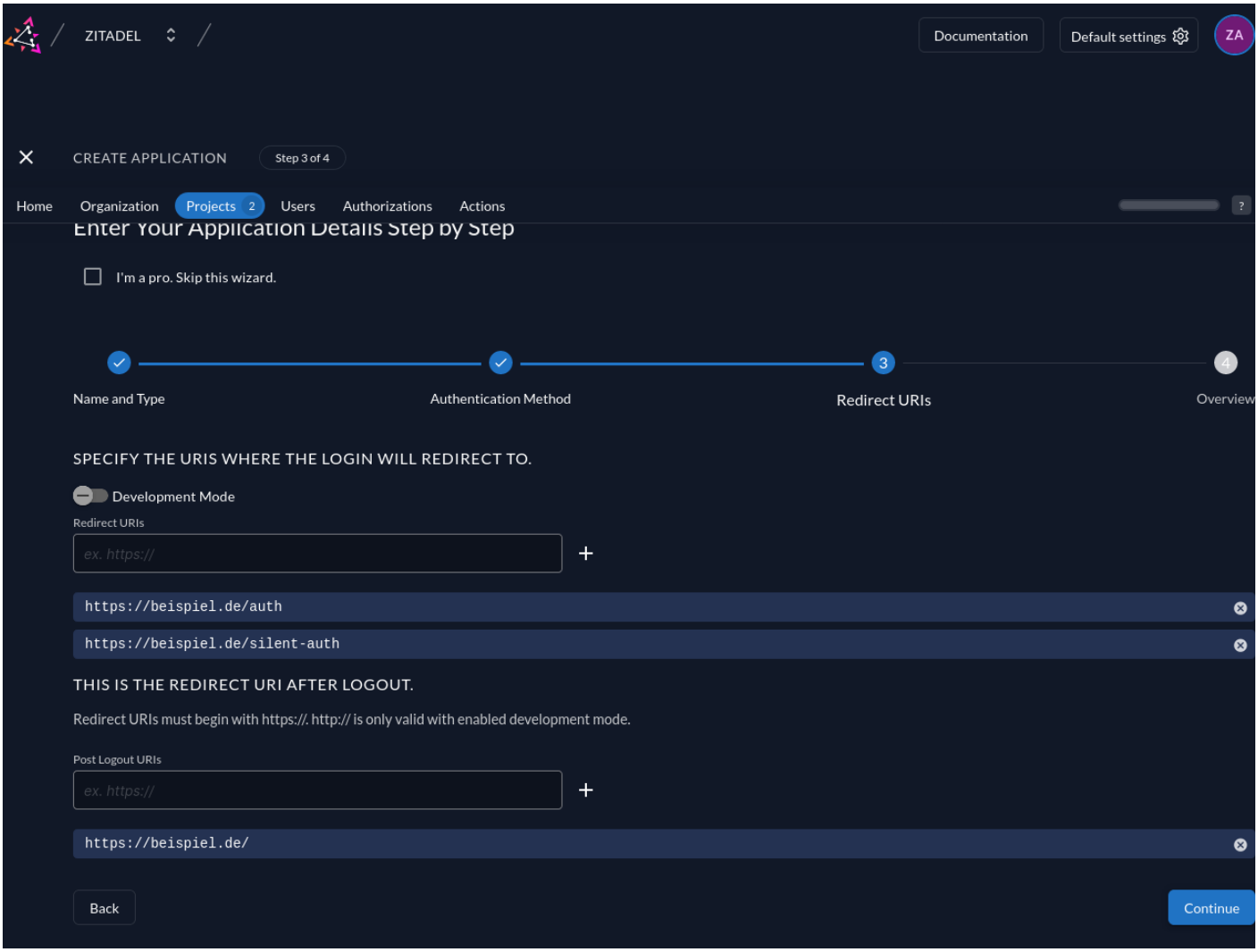
Im nächsten Dialog sind die Weiterleitungs-Adressen sowie die Logout-Adresse anzugeben.

Unter dem Punkt *Specify the URIs where the login will redirect to.* sind die folgenden beiden Adressen einzugeben (die Domäne durch die eigene ersetzen, Eingabe durch Klick auf das Pluszeichen bestätigen):

- https://beispiel.de/auth
- https://beispiel.de/silent-auth

Unter dem Punkt *This is the redirect URI after logout.* ist die folgende Adresse zu hinterlegen:

- https://beispiel.de/

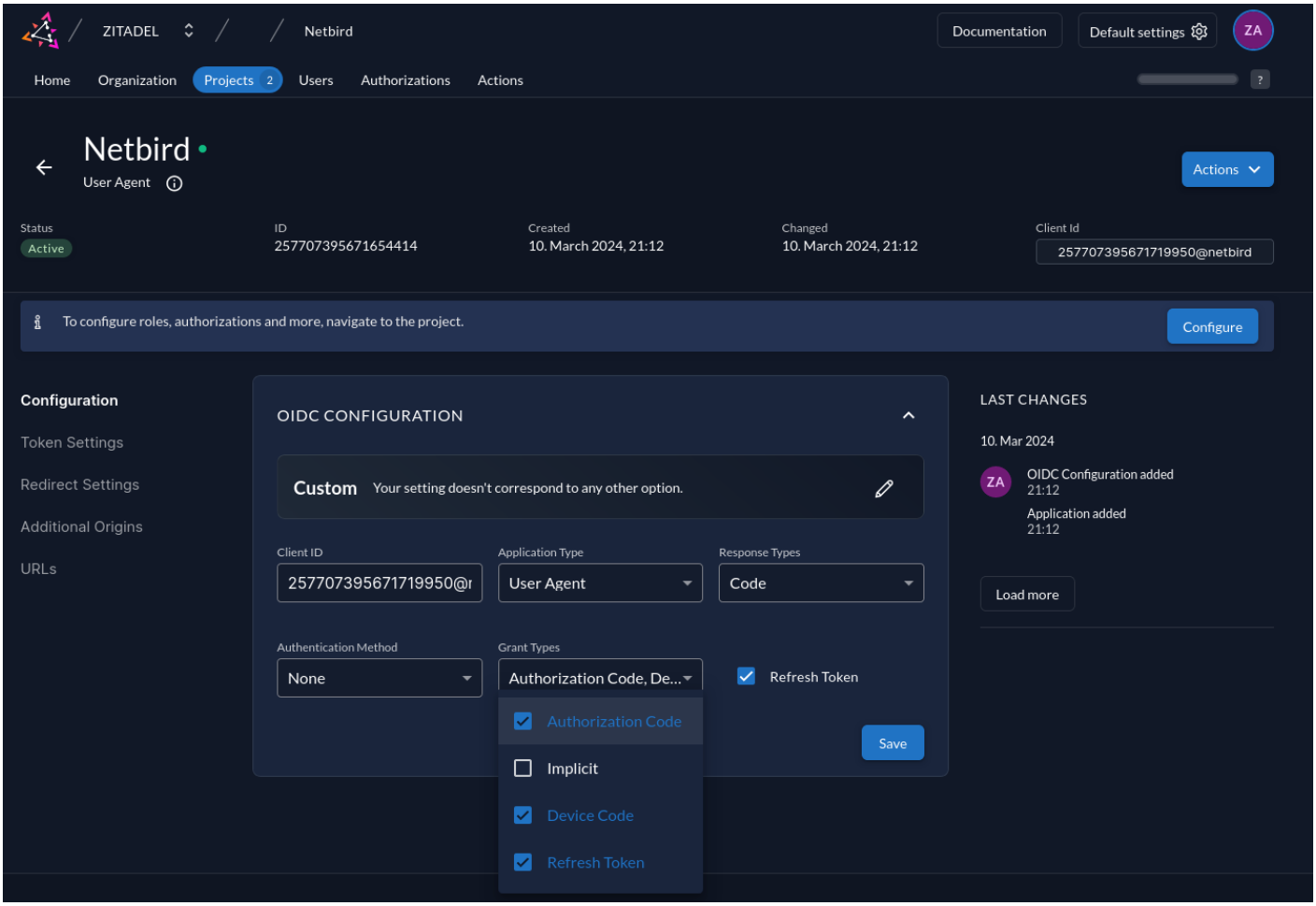


Abschließend wird die gesamte Konfiguration nochmal angezeigt. Diese gilt es zu bestätigen und danach ist die App eingerichtet. Das nun erzeugt, nur einmalig angezeigt Secret muss nicht gespeichert werden. Das Fenster kann einfach mit *Close* geschlossen werden.

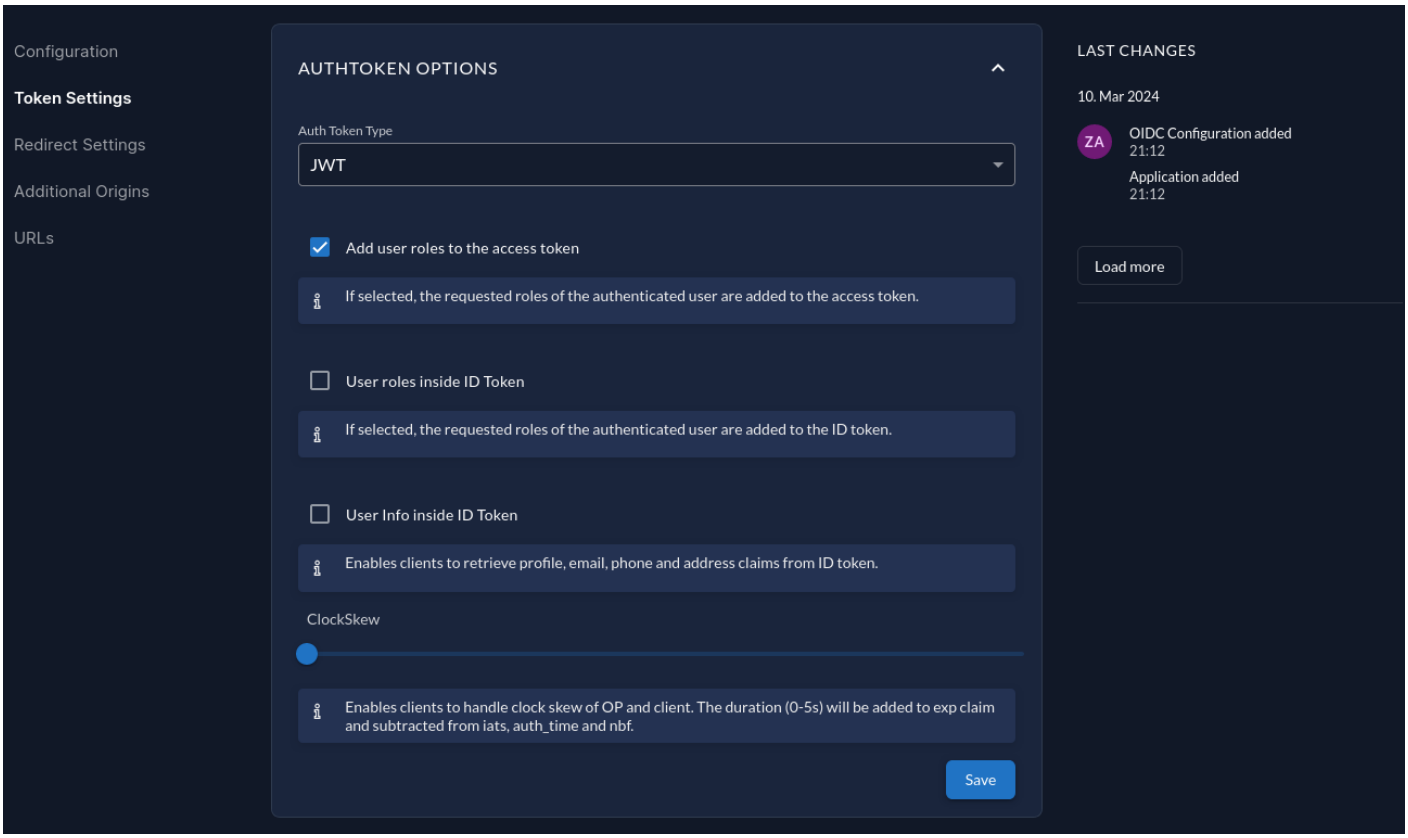
Nun erscheint noch die OIDC Konfiguration. Bei dieser müssen unter *Grant Tyoes* die folgenden Optionen angehakt werden:

- Authorization Code
- Device Code
- Refresh Token

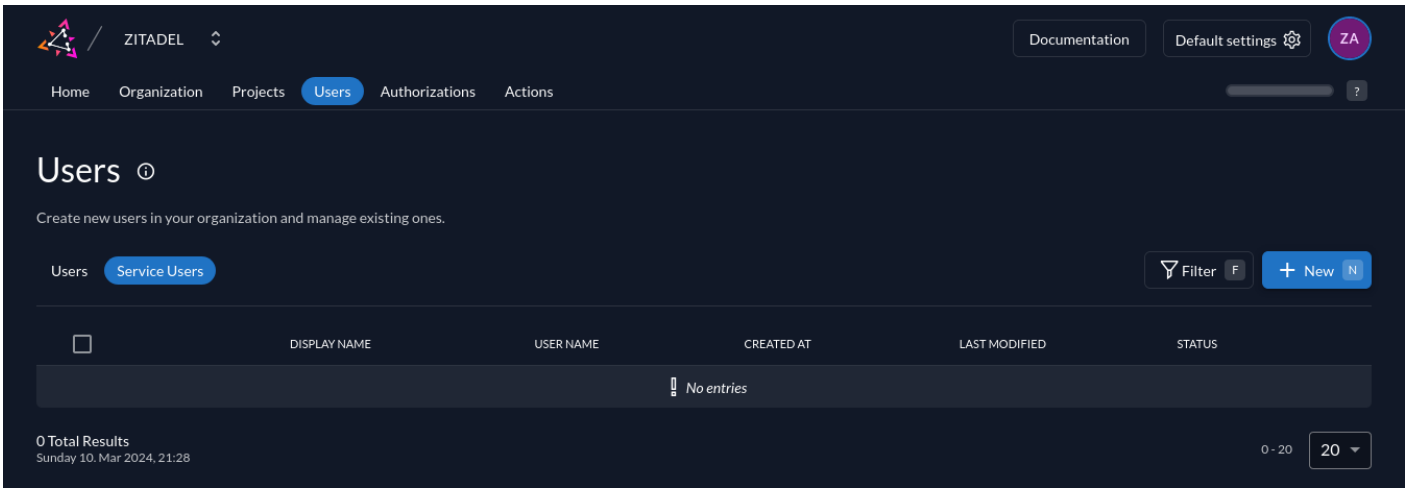
Nachdem das ganze mit *Save* bestätigt wird, am besten direkt die *Client ID* kopieren, da sie später noch benötigt wird.



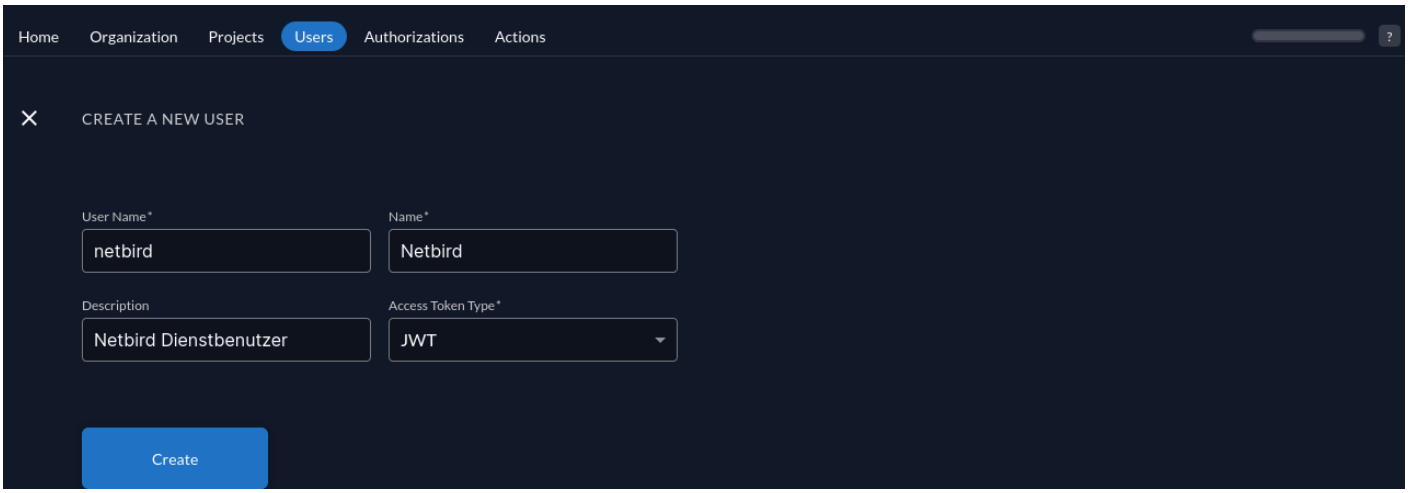
Im selben Fenster wird nun links in der Menüleiste auf *Token Settings* geklickt. Hier wird nun zuerst der *Auth Token Type* auf *JWT* geändert. Außerdem wird die Checkbos bei *Add user roles to the access token* angehakt. Die Einstellung wird mit *Save* gespeichert.



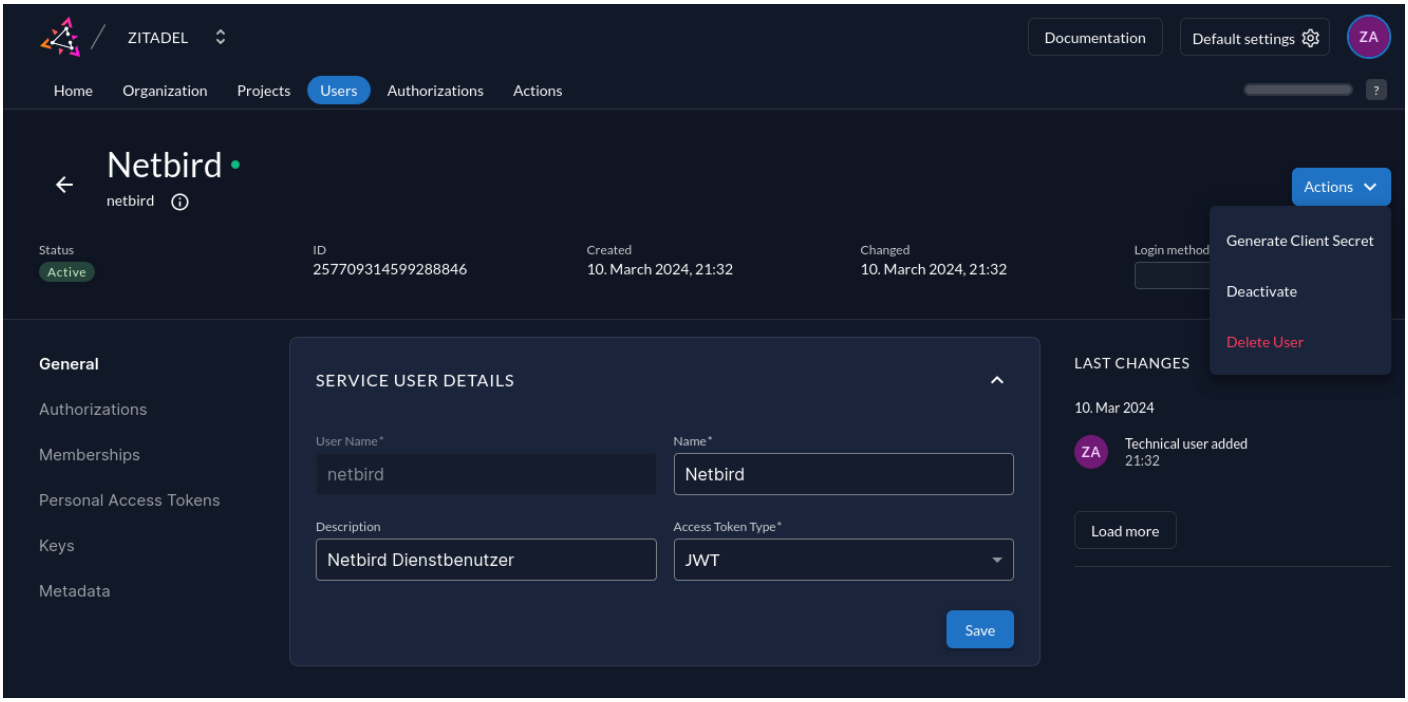
Nachdem Projekt und App konfiguriert sind, wird ein Dienstbenutzer für Netbird angelegt. Hierzu in der oberen Menüleiste *Users* auswählen, dann auf *Service Users* wechseln und auf *New* klicken. Durch die Auswahl der *Service Users* müssen z. B. keine Daten wie E-Mail oder Tel. eingegeben werden, da diese bei Dienstbenutzern nicht benötigt werden.



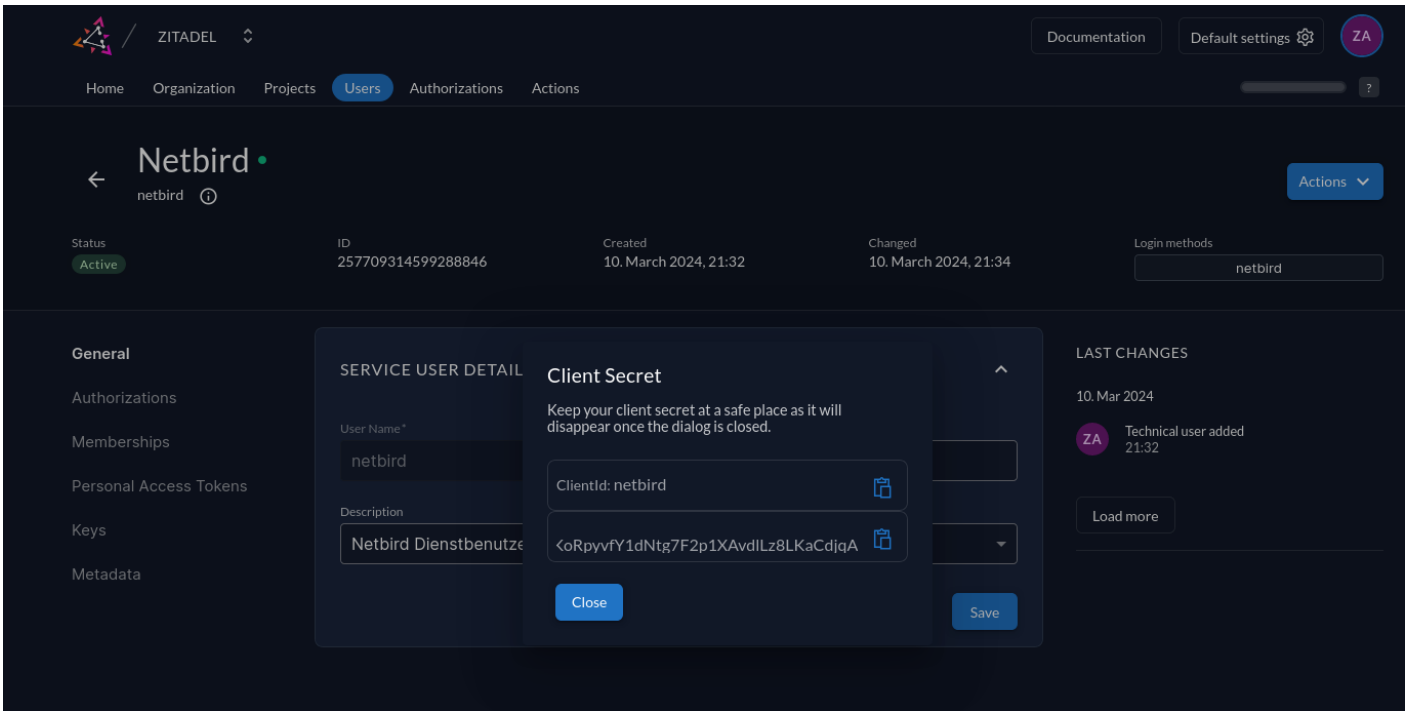
Die Felder können beliebig ausgefüllt werden, nur der *Access Token Type* muss auf *JWT* festgelegt werden.



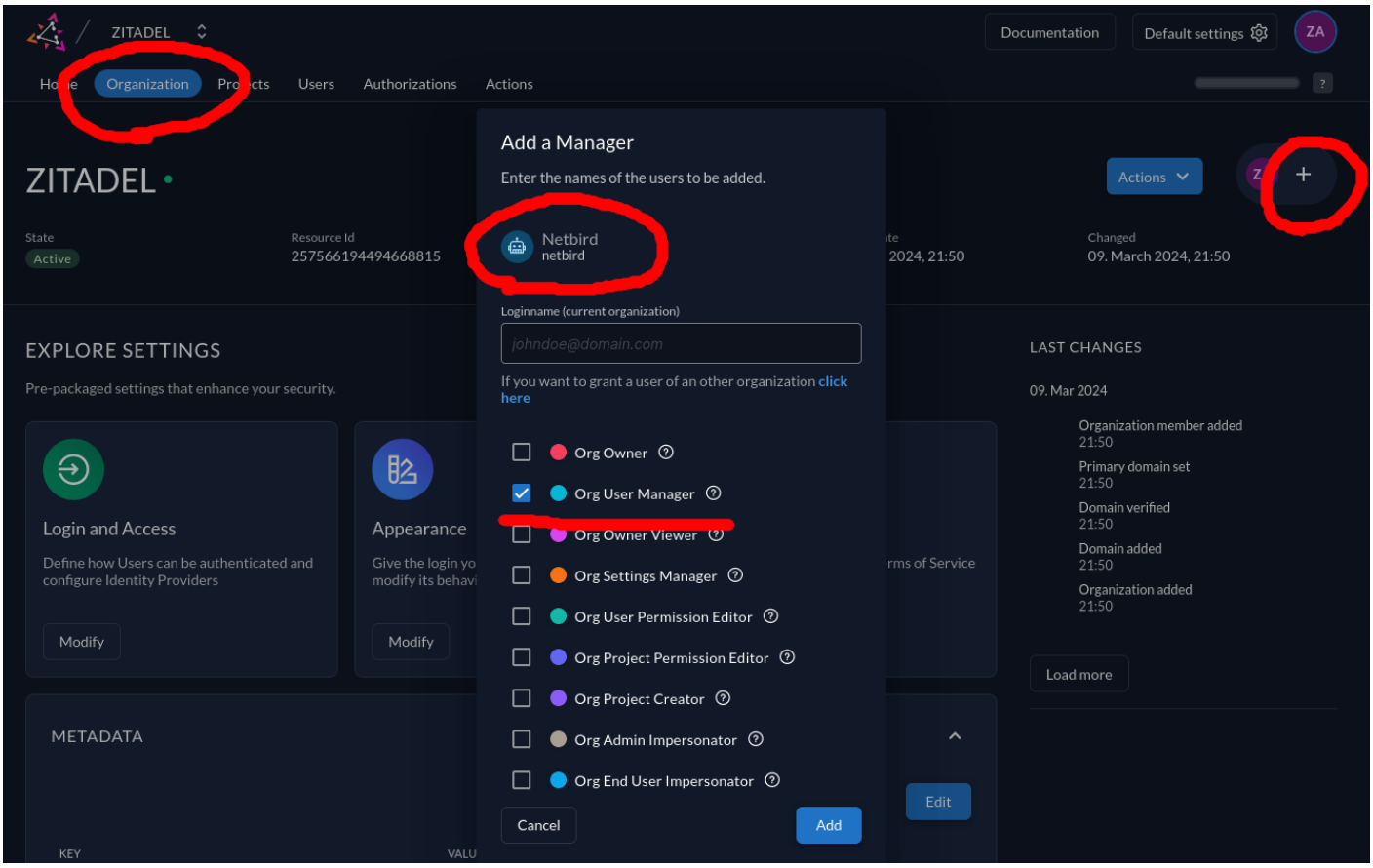
Nach einem Klick auf *Create* wird der Benutzer erstellt und geöffnet. In diesem Menü wird nun rechts oben unter *Actions* auf *Generate Client Secret* geklickt.



Daraufhin wird das *Client Secret* erzeugt und angezeigt. Dieses lässt sich später nie wieder anzeigen, als muss es sicher gespeichert werden. Es kann zwar jederzeit neu erzeugt werden, jedoch wird dadurch das vorherige Client Secret automatisch gelöscht.



Zum Schluss muss dem Dienstbenutzer nur noch die Rolle zum Verwalten von Benutzern zugewiesen weden. Dazu oben in der Menüleiste auf *Organizations* klicken, dann auf das Pluszeichen rechts oben neben *Actions*. Nun in das Suchfeld den Namen des Dienstbenutzers eingeben und diesen auswählen. Anschließend die Rolle *Org User Manager* zuweisen und das ganze mit *Add* bestätigen.



Konfigurationsdateien vorbereiten

Im nächsten Schritt werden die Konfigurationsdateien angepasst. Hier werden die Einstellungen für unsere Umgebung gesetzt.

> Bei dieser Installation wird davon ausgegangen, dass bereits ein Reverse Proxy eingesetzt wird, sodass der Port 443 nicht verfügbar ist und stattdessen der Reverse Proxy sowie Netbird entsprechend konfiguriert werden müssen. Darauf geht die Anleitung später noch ein.

Die Datei setup.env definiert die Parameter, mit welchem später das Setup konfiguriert wird. Hierfür hat Netbird eine kommentierte Vorlage bereitgestellt, die im aktuellen Ordner liegt: setup.env.example

Am besten wird die einfach umbenannt und dann angepasst.

```
mv setup.env.example setup.env
# Optional: Direkt im Terminal im Texteditor nano öffnen:
nano docker-compose.yml.tmpl
```

Für die Zitadel Konfiguration sind die folgenden Parameter in der *setup.env* anzupassen.

```
NETBIRD_AUTH_OIDC_CONFIGURATION_ENDPOINT="https://zitadel.beispiel.de/.well-known/openid-configuration"
NETBIRD_USE_AUTH0=false
NETBIRD_AUTH_CLIENT_ID="<CLIENT_ID>"
NETBIRD_AUTH_SUPPORTED_SCOPES="openid profile email offline_access api"
NETBIRD_AUTH_AUDIENCE="<CLIENT_ID>"
NETBIRD_AUTH_REDIRECT_URI="/auth"
NETBIRD_AUTH_SILENT_REDIRECT_URI="/silent-auth"

NETBIRD_AUTH_DEVICE_AUTH_PROVIDER="hosted"
NETBIRD_AUTH_DEVICE_AUTH_CLIENT_ID="<CLIENT_ID>"
NETBIRD_AUTH_DEVICE_AUTH_AUDIENCE="<CLIENT_ID>"

NETBIRD_MGMT_IDP="zitadel"
NETBIRD_IDP_MGMT_CLIENT_ID="netbird"
NETBIRD_IDP_MGMT_CLIENT_SECRET="<CLIENT_SECRET>"
NETBIRD_IDP_MGMT_EXTRA_MANAGEMENT_ENDPOINT="https://zitadel.beispiel.de/management/v1"
NETBIRD_MGMT_IDP_SIGNKEY_REFRESH=true
```

Konfigurationskript ausführen

Nachdem nun die Konfigurationsdateien vorbereitet wurden, muss noch das Skript ausgeführt werden, welches auf Basis der Konfigurationsdatei die eigentlichen Modifikationen vornimmt.

```
# Für das Skript wird jq benötigt, dieses ggf. mit folgendem Befehl nachinstallieren
sudo apt -y install jq
# Das eigentliche Skript ausführen
./configure.sh
```

> Nach der Ausführung des Skripts wird ein neuer Ordner *artifacts* erstellt, in welchem alle notwendigen Konfigurationen abgelegt wurden. Einfach in diesen Ordner wechseln und die Docker Compose Konfiguration mit `sudo docker compose up -d` ausführen.

Reverse Proxy konfigurieren

Prinzipiell lässt sich Netbird hinter verschiedenen Reverse Proxys einsetzen. Es muss also kein eigener Server nur für Netbird eingerichtet werden.

Die offizielle Dokumentation empfiehlt den Reverse Proxy Traefik und bietet hierfür bereits eine angepasste Docker Compose Konfiguration an.

Aufgrund der Einfachheit, wird im folgenden die notwendige Konfiguration beschrieben, um Netbird hinter dem Reverse Proxy Traefik einzusetzen.

Zuerst sollte Netbird wie in Server Installation beschrieben eingerichtet werden.

Anschließend liegen im Ordner `infrastructure_files` 2 Docker Compose Dateien. In der Installationsanleitung wird die `docker-compose.yml.tmpl` angepasst. Für die Traefik Variante gibt es eine extra Compose Konfiguration, welche `docker-compose.yml.tmpl.traefik` heißt. Diese Datei wird einfach in `docker-compose.yml.tmpl` umbenannt. Die andere Compose Datei kann bei dem Vorgang einfach überschrieben oder vorher gelöscht werden.

Nun ggf. noch alle anderen Konfigurationsparameter wie beschrieben anpassen und danach das Konfigurationsskript ausführen, was alles in der Server Installation beschrieben ist.

Bevor die Konfiguration unter *artifacts* direkt gestartet wird, wie in der anderen Anleitung beschrieben, ist ggf. vorher noch die Netzwerke der hinter Traefik zu veröffentlichen Containers anzupassen. Hierfür wird die Docker Compose Konfiguration wie folgt angepasst (einige Bereiche sind ausgelassen, um den Fokus auf die zu ändernden Bereiche zu setzen).

```
version: "3"
services:
  #UI dashboard
  dashboard:
    # [...]
    networks:
      - traefik
  # Signal
  signal:
    # [...]
    networks:
      - traefik
  # Management
```

```
  management:
    # [...]
    networks:
      - traefik
 # [...]
networks:
  traefik:
    external: true
```

Der Name des Netzwerks muss ggf. angepasst werden und sollte der selbe sein, wie der von Traefik.

Nun kann die Konfiguration einfach gestartet werden.

> Da der letzte Container, in der Konfiguration oben nicht gezeigt, keinem Netzwerk angehört, sondern direkt über das Netzwerk des Hosts läuft, müssen hierfür 2 Firewall Regeln erstellt werden. Sofern die Uncomplicated Firewall (ufw) eingesetzt wird, können einfach die beiden folgenden Befehle ausgeführt werden.

```
sudo ufw allow from any to any port 3478
sudo ufw allow 49152:65535/udp
```