

VERSION 1.0 // FEBRUAR 2026
TICKET.ARMESSA.DE

Inhaltsverzeichnis

TEIL 1: INSTALLATION	
Installation Übersicht	3
Voraussetzungen	3
Schritt 1: Server vorbereiten	4
Schritt 2: MongoDB installieren	4
Schritt 3: Python & Dependencies	5
Schritt 4: Discord Bot einrichten	5
Schritt 5: Web Panel einrichten	6
Schritt 6: Nginx & HTTPS	7
Schritt 7: Supervisor & Autostart	8
Schritt 8: Firewall & Sicherheit	9
Schritt 9: Discord Developer Portal	9
Schritt 10: Erster Start & Test	10
TEIL 2: BENUTZERVERWALTUNG	
Benutzerverwaltung Übersicht	11
Rollen & Berechtigungen	11
Benutzer anlegen	13
Benutzer verwalten	14
API-Zugang & Authentifizierung	15
ANHANG	
Troubleshooting	16
Backup & Recovery	16

Installation Übersicht

Das Control Center X Ultra besteht aus drei Hauptkomponenten, die auf dem Server eingerichtet werden müssen:

Discord Bot

Python-basierter Bot mit discord.py, verbindet sich mit dem Discord Server

Web Panel

FastAPI Backend + React Frontend Dashboard unter ticket.armesa.de

MongoDB

Datenbank für Tickets, Benutzer, Events, Metriken und Audit-Logs

Die Installation erfolgt auf einem **Debian 12** Server mit Root-Zugriff. Die geschätzte Installationszeit beträgt ca. **20–30 Minuten**.

Voraussetzungen

Folgendes wird benötigt bevor die Installation beginnen kann:

- ✓ **Debian 12 VPS** mit mindestens 1 GB RAM und 10 GB Speicher
- ✓ **Root-Zugriff**(SSH als root oder sudo-Rechte)
- ✓ **Domain/Subdomain**(z.B. ticket.armesa.de) mit DNS A-Record auf die Server-IP
- ✓ **Discord Bot Token** aus dem Discord Developer Portal
- ✓ **Discord Server IDs**(Guild ID, Channel IDs, Rollen IDs)

⚠️ Wichtig: Stelle sicher, dass der DNS A-Record für die Subdomain bereits auf die IP-Adresse des Servers zeigt, bevor du mit Schritt 6 (HTTPS) beginnst.

Schritt 1: Server vorbereiten

Verbinde dich per SSH mit dem Server und aktualisiere das System:

SSH-VERBINDUNG & SYSTEM-UPDATE

```
# Per SSH verbinden
ssh root@DEINE_SERVER_IP

# System aktualisieren
apt update && apt upgrade -y

# Basis-Pakete installieren
apt install -y \
  curl wget git gnupg2 \
  build-essential gcc \
  ufw \
  software-properties-common
```

Installationsverzeichnis anlegen:

VERZEICHNISSTRUKTUR

```
mkdir -p /opt/ccx-ultra/{bot,web,web/static,logs,backups,transcripts}
cd /opt/ccx-ultra
```

Schritt 2: MongoDB installieren

MongoDB 7.0 als Datenbank installieren und starten:

```
MONGODB 7.0 AUF DEBIAN 12

# GPG Key importieren
curl -fsSL https://www.mongodb.org/static/pgp/server-7.0.asc | \
gpg --dearmor -o /usr/share/keyrings/mongodb-server-7.0.gpg

# Repository hinzufügen
echo "deb [ signed-by=/usr/share/keyrings/mongodb-server-7.0.gpg ] \
http://repo.mongodb.org/apt/debian bookworm/mongodb-org/7.0 main" | \
tee /etc/apt/sources.list.d/mongodb-org-7.0.list

# Installieren
apt update
apt install -y mongodb-org

# Starten und Autostart aktivieren
systemctl start mongod
systemctl enable mongod

# Status prüfen
systemctl status mongod
```

✓ MongoDB läuft nun auf localhost:27017. Für die Produktionsumgebung ist keine zusätzliche Konfiguration nötig, da nur lokaler Zugriff besteht.

Schritt 3: Python & Dependencies

PYTHON SETUP

```
# Python installieren
apt install -y python3 python3-pip python3-venv python3-dev

# Virtual Environment erstellen
python3 -m venv /opt/ccx-ultra/venv

# Aktivieren
source /opt/ccx-ultra/venv/bin/activate

# Dependencies installieren
pip install --upgrade pip
pip install \
    fastapi uvicorn \
    motor pymongo \
    bcrypt pyjwt "python-jose[cryptography]" \
    python-dotenv \
    discord.py aiohttp \
    python-multipart \
    sse-starlette
```

Schritt 4: Discord Bot einrichten

Die Bot-Dateien auf den Server kopieren:

```
BOT-DATEIEN KOPIEREN

# bot.py auf den Server übertragen (von deinem lokalen PC)
scp bot.py root@DEINE_SERVER_IP:/opt/ccx-ultra/bot/
```

Konfigurationsdatei erstellen:

```
/OPT/CCX-ULTRA/BOT/CONFIG.ENV

# Discord Bot Token (aus Developer Portal)
DISCORD_BOT_TOKEN=DEIN_BOT_TOKEN_HIER

# Discord Server ID
DISCORD_GUILD_ID=1407623359365120090

# Channel IDs
TICKET_CHANNEL_ID=1469120567532847225
TRANSCRIPT_LOG_CHANNEL_ID=1470212303260483817
TICKET_CATEGORY_ID=1469120434115969064

# Support Rollen IDs (komma-getrennt)
SUPPORT_ROLE_IDS=368510579511656449,372850494445453314,1006197315566051388

# Web Panel API URL (lokaler Zugriff)
API_URL=http://localhost:8001/api

# Ticket-Einstellungen
MAX_TICKETS_PER_USER=3
SLA_FIRST_RESPONSE_MINUTES=30
AUTO_CLOSE_INACTIVE_HOURS=48
```

⚠ Niemals den Bot-Token öffentlich teilen oder in Git committen! Nutze `.gitignore` für alle `.env` Dateien.

Schritt 5: Web Panel einrichten

Backend und Frontend auf den Server übertragen:

```
WEB PANEL DATEIEN

# server.py auf den Server kopieren
scp server.py root@DEINE_SERVER_IP:/opt/ccx-ultra/web/

# React Frontend bauen (auf deinem lokalen PC)
cd frontend
yarn build

# Build-Ordner auf den Server kopieren
scp -r build/* root@DEINE_SERVER_IP:/opt/ccx-ultra/web/static/
```

Konfigurationsdatei erstellen:

```
/OPT/CCX-ULTRA/WEB/.ENV

# MongoDB Verbindung (lokal)
MONGO_URL=mongodb://localhost:27017
DB_NAME=ccx_ultra

# Sicherheit (generiere einen einzigartigen Key!)
JWT_SECRET=GENERIERE_EINEN_SICHEREN_KEY
CORS_ORIGINS=https://ticket.armesa.de

# Discord Bot Token (für Webhook-Authentifizierung)
DISCORD_BOT_TOKEN=DEIN_BOT_TOKEN_HIER

# Discord IDs
DISCORD_GUILD_ID=1407623359365120090
TICKET_CHANNEL_ID=1469120567532847225
TRANSCRIPT_LOG_CHANNEL_ID=1470212303260483817
TICKET_CATEGORY_ID=1469120434115969064
SUPPORT_ROLE_IDS=368510579511656449,372850494445453314,1006197315566051388
```

Einen sicheren JWT-Secret generieren:

SICHEREN KEY GENERIEREN

```
python3 -c "import secrets; print(secrets.token_hex(32))"
```

Schritt 6: Nginx & HTTPS

NGINX INSTALLIEREN

```
apt install -y nginx certbot python3-certbot-nginx
```

Nginx-Konfiguration erstellen:

```
/ETC/NGINX/SITES-AVAILABLE/TICKET.ARMESSA.DE

# Rate Limiting Zone
limit_req_zone $binary_remote_addr zone=ccx_limit:10m rate=10r/s;

server {
    listen 80;
    server_name ticket.armesa.de;

    # Security Headers
    add_header X-Frame-Options "SAMEORIGIN" always;
    add_header X-Content-Type-Options "nosniff" always;
    add_header X-XSS-Protection "1; mode=block" always;

    # Gzip Kompression
    gzip on;
    gzip_vary on;
    gzip_types text/plain text/css application/json
               application/javascript text/xml;

    # API Reverse Proxy
    location /api/ {
        limit_req zone=ccx_limit burst=20 nodelay;
        proxy_pass http://127.0.0.1:8001;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_buffering off; # SSE Support
        proxy_cache off;
        proxy_read_timeout 86400s;
    }

    # Frontend (React Build)
    location / {
        root /opt/ccx-ultra/web/static;
        try_files $uri $uri/ /index.html;
    }
}
```

NGINX AKTIVIEREN & HTTPS

```
# Seite aktivieren
ln -sf /etc/nginx/sites-available/ticket.armesa.de /etc/nginx/sites-enabled/
rm -f /etc/nginx/sites-enabled/default
nginx -t && systemctl reload nginx

# HTTPS Zertifikat erstellen
certbot --nginx -d ticket.armesa.de \
--email admin@armesa.de \
--agree-tos --non-interactive

# Renewal prüfen
certbot renew --dry-run
```

Schritt 7: Supervisor & Autostart

```
SUPERVISOR INSTALLIEREN

apt install -y supervisor

/ETC/SUPERVISOR/CONF.D/CCX-ULTRA.CONF

[program:ccx-webpanel]
command=/opt/ccx-ultra/venv/bin/uvicorn server:app --host 0.0.0.0 --port 8001 --workers 2
directory=/opt/ccx-ultra/web
user=root
autostart=true
autorestart=true
startsecs=5
stderr_logfile=/opt/ccx-ultra/logs/webpanel.err.log
stdout_logfile=/opt/ccx-ultra/logs/webpanel.out.log
stderr_logfile_maxbytes=10MB
stdout_logfile_maxbytes=10MB
environment=PATH="/opt/ccx-ultra/venv/bin"

[program:ccx-bot]
command=/opt/ccx-ultra/venv/bin/python bot.py
directory=/opt/ccx-ultra/bot
user=root
autostart=true
autorestart=true
startsecs=5
stderr_logfile=/opt/ccx-ultra/logs/bot.err.log
stdout_logfile=/opt/ccx-ultra/logs/bot.out.log
stderr_logfile_maxbytes=10MB
stdout_logfile_maxbytes=10MB
environment=PATH="/opt/ccx-ultra/venv/bin"
```

```
SUPERVISOR STARTEN
```

```
supervisorctl reread
supervisorctl update
supervisorctl status
```

Schritt 8: Firewall & Sicherheit

```
UFW FIREWALL KONFIGURIEREN
```

```
ufw allow 22/tcp # SSH
ufw allow 80/tcp # HTTP (Redirect)
ufw allow 443/tcp # HTTPS
ufw --force enable
ufw status verbose
```

 MongoDB (Port 27017) und der Webserver (Port 8001) sind **nicht** von außen erreichbar. Der Zugriff erfolgt ausschließlich über Nginx auf Port 443.

Schritt 9: Discord Developer Portal

Im **Discord Developer Portal** (discord.com/developers/applications) folgende Einstellungen vornehmen:

Privileged Gateway Intents (Pflicht)

- ✓ **PRESENCE INTENT** – Für Online-Status-Erkennung und Auto-Ping
- ✓ **SERVER MEMBERS INTENT** – Für Mitglieder-Zugriff und Rollenprüfung
- ✓ **MESSAGE CONTENT INTENT** – Für Transcript-Generierung

Bot Permissions

BERECHTIGUNG	ZWECK
Manage Channels	Ticket-Channels erstellen/bearbeiten
Read Messages	Channels lesen
Send Messages	Nachrichten in Tickets senden
Manage Messages	Nachrichten verwalten/löschen
Embed Links	Rich Embeds senden
Attach Files	Transcripts hochladen
Read Message History	Transcripts generieren
Use Slash Commands	/ticket-panel Befehl

OAuth2 Einladungs-URL

BOT EINLADEN (SCOPES: BOT, APPLICATIONS.COMMANDS)

https://discord.com/api/oauth2/authorize?client_id=YOUR_CLIENT_ID&permissions=8&scope=bot%20applications.commands

⚠️ Rollen-Position: Die Bot-Rolle muss in den Server-Einstellungen über allen Support-Rollen positioniert sein!

Schritt 10: Erster Start & Test

SYSTEM STARTEN UND PRÜFEN

```
# Alle Dienste starten
supervisorctl start all

# Status prüfen
supervisorctl status
# ccx-bot      RUNNING    pid 1234, uptime 0:00:05
# ccx-webpanel RUNNING    pid 1235, uptime 0:00:05

# API testen
curl https://ticket.armesa.de/api/health
# {"status":"ok","service":"Control Center X Ultra"}

# Web Panel öffnen: https://ticket.armesa.de
# Login: admin / admin123

# Im Discord Server den Ticket-Panel erstellen:
# /ticket-panel (als Admin im Bot-Channel ausführen)
```

⚠️ Sofort nach dem ersten Login: Ändere das Standard-Admin-Passwort! Erstelle neue Benutzer mit sicheren Passwörtern.

Benutzerverwaltung & Rechte

Benutzerverwaltung Übersicht

Das System verwendet ein rollenbasiertes Zugangssystem mit drei Berechtigungsstufen. Benutzer werden über das Web Panel oder die API erstellt und verwaltet.

 Die Benutzerverwaltung ist **nur für Admins** zugänglich. Navigiere zu **Settings** in der Seitenleiste.

Rollen & Berechtigungen

Es gibt drei Rollen mit unterschiedlichen Berechtigungen:

Admin

VOLLZUGRIFF

- ✓ Dashboard & KPIs einsehen
- ✓ Alle Tickets einsehen & bearbeiten
- ✓ Tickets claimen, schließen, eskalieren
- ✓ Notizen & Priorität ändern
- ✓ Analytics & SLA Daten einsehen
- ✓ Support-Ranking einsehen
- ✓ Audit-Log einsehen
- ✓ Benutzer anlegen & löschen
- ✓ Settings-Seite Zugang
- ✓ Bot-Konfiguration einsehen

Support

TICKET-BEARBEITUNG

- ✓ Dashboard & KPIs einsehen
 - ✓ Alle Tickets einsehen & bearbeiten
 - ✓ Tickets claimen, schließen, eskalieren
 - ✓ Notizen & Priorität ändern
 - ✓ Analytics & SLA Daten einsehen
 - ✓ Support-Ranking einsehen
 - ✓ Audit-Log einsehen
-  Keine Benutzerverwaltung
-  Kein Zugang zu Settings

Viewer

NUR LESEN

✓ Dashboard & KPIs einsehen

✓ Tickets einsehen (nur lesen)

✓ Analytics & SLA einsehen

✓ Support-Ranking einsehen

🔒 Keine Ticket-Aktionen

🔒 Kein Audit-Log Zugang

🔒 Keine Benutzerverwaltung

🔒 Kein Zugang zu Settings

Benutzer anlegen

Methode 1: Über das Web Panel (empfohlen)

- 1 Als **Admin** einloggen
- 2 In der Seitenleiste auf **Settings** klicken
- 3 Button **Add User** oben rechts klicken
- 4 Im Dialog ausfüllen: **Username**, **Passwort**, **Rolle**
- 5 Button **Create User** klicken

Methode 2: Über die API

BENUTZER PER API ERSTELLEN

```
# 1. Als Admin einloggen und Token erhalten
TOKEN=$(curl -s -X POST https://ticket.armesa.de/api/auth/login \
-H "Content-Type: application/json" \
-d '{"username":"admin","password":"admin123"}' \
| python3 -c "import sys,json; print(json.load(sys.stdin)['token'])")\n\n# 2. Neuen Support-Benutzer erstellen
curl -X POST https://ticket.armesa.de/api/admin/users \
-H "Content-Type: application/json" \
-H "Authorization: Bearer $TOKEN" \
-d '{
    "username": "sarah_support",
    "password": "sicheres_passwort_123",
    "role": "support"
}'\n\n# 3. Neuen Viewer erstellen
curl -X POST https://ticket.armesa.de/api/admin/users \
-H "Content-Type: application/json" \
-H "Authorization: Bearer $TOKEN" \
-d '{
    "username": "chef_viewer",
    "password": "anderes_sicheres_passwort",
    "role": "viewer"
}'
```

🔒 Passwörter werden automatisch mit **bcrypt** gehasht gespeichert. Klartext-Passwörter werden niemals in der Datenbank abgelegt.

Benutzer verwalten

Alle Benutzer anzeigen

In der **Settings**-Seite siehst du eine Tabelle aller Benutzer mit Username, Rolle und Erstellungsdatum.

Benutzer löschen

- 1 In der Benutzertabelle unter **Settings**
- 2 Auf das Papierkorb-Icon neben dem Benutzer klicken
- 3 Bestätigung im Dialog

⚠ Der **admin**-Standardbenutzer kann nicht über die UI gelöscht werden. Dies ist ein Sicherheitsmechanismus.

Benutzer per API löschen

BENUTZER PER API LÖSCHEN

```
# Alle Benutzer auflisten
curl -s https://ticket.armesa.de/api/admin/users \
-H "Authorization: Bearer $TOKEN" | python3 -m json.tool

# Benutzer löschen
curl -X DELETE https://ticket.armesa.de/api/admin/users/USER_ID \
-H "Authorization: Bearer $TOKEN"
```

Passwort zurücksetzen

PASSWORT PER MONGODB ZURÜCKSETZEN

```
# Neuen Passwort-Hash generieren
python3 -c "import bcrypt; print(bcrypt.hashpw(b'neues_passwort', bcrypt.gensalt()).decode())"

# MongoDB Shell öffnen
mongosh

# Hash in DB aktualisieren
use ccx_ultra
db.panel_users.updateOne(
  { username: "sarah_support" },
  { $set: { password_hash: "HASH_VON_OBEN_HIER" } }
)
```

API-Zugang & Authentifizierung

Jeder API-Zugriff erfordert einen gültigen JWT-Token im Authorization-Header:

AUTHENTIFIZIERUNGSABLAUF

```
# 1. Login → Token erhalten
curl -X POST https://ticket.armesa.de/api/auth/login \
-H "Content-Type: application/json" \
-d '{"username":"DEIN_USER","password":"DEIN_PASSWORD"}'

# Antwort:
# { "token": "eyJhbGci...", "user": {"username": "...", "role": "admin"} }

# 2. Token in allen Anfragen verwenden
curl https://ticket.armesa.de/api/kpi \
-H "Authorization: Bearer DEIN_TOKEN"

# 3. Aktuellen Benutzer prüfen
curl https://ticket.armesa.de/api/auth/me \
-H "Authorization: Bearer DEIN_TOKEN"
```

⌚ Tokens sind **24 Stunden** gültig. Danach muss ein neuer Token per Login angefordert werden.

Troubleshooting & Backup

Troubleshooting

Bot startet nicht

BOT-FEHLER PRÜFEN

```
tail -50 /opt/ccx-ultra/logs/bot.err.log

# Häufige Ursachen:
# - Ungültiger Token      → config.env prüfen
# - Fehlende Intents       → Developer Portal aktivieren
# - Python-Modul fehlt    → venv aktivieren + pip install
```

Web Panel nicht erreichbar

WEB PANEL DEBUGGEN

```
supervisorctl status ccx-webpanel
curl http://localhost:8001/api/health
nginx -t
systemctl status nginx
certbot certificates
```

MongoDB-Probleme

MONGODB DEBUGGEN

```
systemctl status mongod
tail -50 /var/log/mongodb/mongod.log
mongosh --eval "db.adminCommand({serverStatus: 1}).ok"
```

Backup & Recovery

BACKUP (AUTOMATISCH TÄGLICH UM 3:00 UHR)

```
# Manuelles Backup
/opt/ccx-ultra/backup.sh

# Backup-Inhalt:
# /opt/ccx-ultra/backups/YYYYMMDD_HHMMSS/
#   mongodb/      ← MongoDB Dump
#   .env           ← Web Panel Config
#   config.env    ← Bot Config

# Automatisches Backup prüfen
crontab -l | grep backup

# Aus Backup wiederherstellen
mongorestore --db ccx_ultra \
  /opt/ccx-ultra/backups/DATUM/mongodb/ccx_ultra/

# Nur die letzten 7 Backups werden behalten
```