Deployment-Handbuch – restaurant.bibabau.ch (Debian 13)

Dieses Handbuch beschreibt alle Schritte, die über eine minimal installierte Debian-13-Installation hinaus erforderlich sind, um die Produktionsumgebung für restaurant.bibabau.ch aufzusetzen: Docker/Compose, Projekt-Setup, Nginx + TLS, FastAPI/Frontend, SMTP mit Mailcow, PostgreSQL, lokale & Offsite-Backups (Backblaze B2 via restic), Raspberry‑Pi‑Kitchen‑Agent, Tests & Troubleshooting.

# Voraussetzungen (bereits vorhanden / Annahmen)

• Debian 13 (minimal) läuft, SSH-Zugang mit sudo-Rechten ist möglich.

• DNS: restaurant.bibabau.ch zeigt per A/AAAA auf diesen Server (Port 80/443 erreichbar).

• Mailcow-Instanz unter mail.bibabau.ch aktiv (SMTP-Konto no-reply@bibabau.ch vorhanden).

• Du arbeitest mit einem Benutzerkonto, das in die Gruppe 'sudo' gehört.

# System-Tools & Firewall (optional, falls noch nicht gesetzt)

Aktualisieren und Basis-Tools installieren:

sudo apt update && sudo apt -y upgrade  
sudo apt -y install ca-certificates curl gnupg lsb-release git ufw

Zeitzone (Europe/Zurich) sicherstellen:

sudo timedatectl set-timezone Europe/Zurich

UFW-Firewall (Ports 22, 80, 443 öffnen):

sudo ufw allow OpenSSH  
sudo ufw allow 80  
sudo ufw allow 443  
sudo ufw enable  
sudo ufw status

# Docker & Docker Compose installieren

Docker aus dem offiziellen Repository installieren:

sudo install -m 0755 -d /etc/apt/keyrings  
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg  
sudo chmod a+r /etc/apt/keyrings/docker.gpg  
  
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/debian $(. /etc/os-release && echo $VERSION\_CODENAME) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null  
  
sudo apt update  
sudo apt -y install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin  
  
sudo systemctl enable --now docker  
sudo usermod -aG docker $USER  
newgrp docker <<<' '  
  
docker version  
docker compose version

# Projektverzeichnis & Dateien bereitstellen

Projektordner anlegen und Inhalte aus deinem Repo/Canvas auf den Server bringen:

mkdir -p ~/asia-restaurant && cd ~/asia-restaurant  
# Falls lokal vorhanden: scp -r asia-restaurant/\* user@server:~/asia-restaurant/  
# Oder git init && Dateien gemäß Canvas-Struktur anlegen

# .env für restaurant.bibabau.ch ausfüllen

Erstelle die Datei .env im Projektroot. Ersetze STRONG\_\* und Secrets durch sichere Werte:

TZ=Europe/Zurich  
API\_HOST=0.0.0.0  
API\_PORT=8000  
  
POSTGRES\_USER=asiar  
POSTGRES\_PASSWORD=STRONG\_DB\_PASSWORD  
POSTGRES\_DB=asiar  
POSTGRES\_HOST=db  
POSTGRES\_PORT=5432  
DB\_URL=postgresql+psycopg://asiar:STRONG\_DB\_PASSWORD@db:5432/asiar  
  
JWT\_SECRET=CHANGE\_TO\_LONG\_RANDOM\_SECRET  
JWT\_ALG=HS256  
JWT\_EXPIRES\_MIN=43200  
  
VITE\_API\_URL=https://restaurant.bibabau.ch/api  
VITE\_WS\_URL=wss://restaurant.bibabau.ch  
  
VITE\_MATOMO\_URL=https://matomo.bibabau.ch/  
VITE\_MATOMO\_SITE\_ID=1  
  
MATOMO\_URL=https://matomo.bibabau.ch/matomo.php  
MATOMO\_SITE\_ID=1  
MATOMO\_TOKEN=  
  
# SMTP (Mailcow)  
SMTP\_HOST=mail.bibabau.ch  
SMTP\_PORT=587  
SMTP\_USER=no-reply@bibabau.ch  
SMTP\_PASS=STRONG\_MAIL\_PASSWORD  
SMTP\_STARTTLS=1  
SMTP\_FROM="Asia Restaurant <no-reply@bibabau.ch>"  
KITCHEN\_EMAIL=kueche@bibabau.ch  
HEALTH\_MAIL\_TOKEN=SET\_RANDOM\_TOKEN  
  
# Backups (lokal)  
BACKUP\_SCHEDULE\_CRON=0 3 \* \* \*  
BACKUP\_RETENTION\_DAYS=14  
BACKUP\_DIR\_HOST=./backups  
  
# Offsite Backups – Backblaze B2 via S3 API  
OFFSITE\_SCHEDULE\_CRON=30 3 \* \* \*  
RESTIC\_REPOSITORY=s3:https://s3.eu-central-003.backblazeb2.com/DEIN-BUCKET/pfad  
AWS\_ACCESS\_KEY\_ID=B2\_KEY\_ID  
AWS\_SECRET\_ACCESS\_KEY=B2\_APP\_KEY  
RESTIC\_PASSWORD=STRONG\_RESTIC\_PASSWORD  
RESTIC\_KEEP\_LAST=7  
RESTIC\_KEEP\_DAILY=14  
RESTIC\_KEEP\_WEEKLY=8  
RESTIC\_KEEP\_MONTHLY=12  
OFFSITE\_INCLUDE=/backups

mkdir -p backups

# Produktions-Build & Nginx mit TLS (Let’s Encrypt)

Frontend builden und als statische Dateien bereitstellen (Volume füllen):

# Im Projektroot  
docker build -t asia-frontend-prod:latest -f frontend/Dockerfile.prod frontend  
CID=$(docker create asia-frontend-prod:latest)  
docker run --rm -v frontend\_dist:/dest --volumes-from $CID alpine sh -c "cp -r /\* /dest/"  
docker rm $CID

Web (HTTP) vorab starten, damit ACME-Challenge beantwortet werden kann:

docker compose -f docker-compose.prod.yml up -d web

Zertifikat für restaurant.bibabau.ch holen:

docker run --rm -v $(pwd)/nginx:/etc/nginx/conf.d:ro -v webroot:/var/www/certbot -v letsencrypt:/etc/letsencrypt certbot/certbot certonly --webroot -w /var/www/certbot -d restaurant.bibabau.ch --agree-tos -m admin@bibabau.ch --no-eff-email

Gesamten Stack starten (HTTPS aktiv):

docker compose -f docker-compose.prod.yml up -d --build

# Demo-Menü, SMTP-Health, Matomo prüfen

Demo-Menü anlegen (einmalig):

curl -X POST https://restaurant.bibabau.ch/api/dev/seed

SMTP-Health-Test (ersetzt dein Token und Zielpostfach):

curl -X POST "https://restaurant.bibabau.ch/api/health/mail?to=dein@mail.tld" -H "X-Health-Token: $(grep HEALTH\_MAIL\_TOKEN .env | cut -d= -f2)"

Matomo: Prüfe, ob Pageviews/E-Commerce-Orders auftauchen (Adblocker deaktivieren).

# Raspberry Pi 4 – Kitchen-Agent installieren

Auf dem Pi vorbereiten und Agent als Service einrichten:

sudo apt update && sudo apt -y install python3 python3-pip git  
pip3 install websockets RPi.GPIO  
  
# pi\_agent.py nach /home/pi kopieren (aus deinem Repo)  
# scp pi\_agent/pi\_agent.py pi@raspberrypi:/home/pi/  
  
sudo tee /etc/systemd/system/kitchen-agent.service >/dev/null <<'EOF'  
[Unit]  
Description=Kitchen WebSocket Agent  
After=network-online.target  
  
[Service]  
ExecStart=/usr/bin/python3 /home/pi/pi\_agent.py  
Restart=always  
User=pi  
Environment=KITCHEN\_WS=wss://restaurant.bibabau.ch/ws/kitchen  
Environment=SIGNAL\_PIN=18  
Environment=PRINT\_TICKETS=0  
#Environment=PRINTER\_NAME=KitchenPrinter  
  
[Install]  
WantedBy=multi-user.target  
EOF  
  
sudo systemctl daemon-reload  
sudo systemctl enable --now kitchen-agent  
systemctl status kitchen-agent --no-pager

GPIO 18 (BCM) für LED/Buzzer nutzen. Druck aktivieren: PRINT\_TICKETS=1 und CUPS-Druckername setzen.

# Backups – lokal (pg\_dump) & Rotation

Ersten manuellen Lauf anstoßen und prüfen, dass Files entstehen:

docker compose -f docker-compose.prod.yml run --rm db-backup /app/backup.sh run  
ls -lah backups/

Automatik: täglich 03:00, Retention 14 Tage (über .env).

# Offsite-Backups – Backblaze B2 (restic, verschlüsselt)

Offsite-Service starten und ersten Upload testen:

docker compose -f docker-compose.prod.yml up -d --build offsite-backup  
docker compose -f docker-compose.prod.yml run --rm offsite-backup /bin/sh /app/offsite.sh run

Snapshots anzeigen und Restore testen (in lokalen Ordner ./restore):

docker compose -f docker-compose.prod.yml run --rm offsite-backup restic snapshots  
  
mkdir -p restore  
docker run --rm -e RESTIC\_REPOSITORY="$RESTIC\_REPOSITORY" -e RESTIC\_PASSWORD="$RESTIC\_PASSWORD" -e AWS\_ACCESS\_KEY\_ID="$AWS\_ACCESS\_KEY\_ID" -e AWS\_SECRET\_ACCESS\_KEY="$AWS\_SECRET\_ACCESS\_KEY" -v $(pwd)/restore:/restore restic/restic:latest restore latest --target /restore

Anschließend den gewünschten Dump (\*.dump.gz) mit pg\_restore einspielen (siehe Restore-Guide).

# Funktionstests (Ende-zu-Ende)

1) Frontend öffnen: https://restaurant.bibabau.ch

2) Registrieren/Einloggen → Bestellung durchführen (Abholzeit ≥ 30 Min).

3) Küche: Neue Bestellung erscheint in der Kitchen-UI; Pi signalisiert (LED/Buzzer).

4) E-Mail: Kunde & Küche erhalten Mails (Mailcow-Logs prüfen, falls nicht).

5) Matomo: E-Commerce-Order sichtbar nach Checkout.

# Troubleshooting – häufige Probleme & Lösungen

API 502/404:

docker compose -f docker-compose.prod.yml logs -f api web

WebSocket funktioniert nicht:

• Prüfe Nginx-/ws/-Block, Ports/Firewall, Browser-Konsole.

E-Mails kommen nicht an:

• SMTP-Creds/STARTTLS, DKIM/SPF/DMARC, rDNS, Spam-Ordner.

Matomo leer:

• VITE\_MATOMO\_\* prüfen; Adblocker deaktivieren.

Abholzeit abgelehnt:

• Server erzwingt now+30m (Europe/Zurich); Client-Uhrzeit ggf. falsch.