# Tazat Setup Guide

From a brand-new server to your own restaurant app on a phone. No programming needed: you copy a command, paste it, and check the result.

### What you will do

1. **Part 1, the server (about 45 minutes).** You rent a small online computer, put Tazat on it (with Docker, or without) and open your admin panel in the browser.
2. **Part 2, the mobile app (about 1 to 2 hours the first time).** You tell the app the address of your server, give it your own name and icon, and build the file you upload to Google Play and the App Store.

### What you need before you start

- The Tazat package you downloaded and unzipped. It has a server folder (it contains `nest-backend`) and the app folder `tazat_restaurant`.
- A **domain name** you own (for example `eat-with-us.com`). Mobile apps only talk to `https://` addresses, so this is required.
- About **10 to 15 USD a month** for the server.
- For the app: a computer (Windows, Mac or Linux). An iPhone app can only be built on a **Mac**. Publishing costs 25 USD once (Google Play) and 99 USD a year (Apple).

## Words explained

| Word | What it means here |
|---|---|
| Server | A computer that is always on, somewhere on the internet. Tazat lives there. |
| Terminal | A window where you type (or paste) a command and press Enter. On Windows use **PowerShell**; on Mac use the **Terminal** app. |
| Command | A line you copy from a black box on this page. Use the **Copy** button, paste, press Enter. Wait until it finishes (you get your prompt back). |

| | |
|---|---|
| Domain / DNS | The name people type (`eat-with-us.com`). DNS is the setting that says which server that name belongs to. |
| HTTPS | The padlock. It keeps logins private. Free, and set up automatically in Part 1 step 7. |
| Docker | A program that runs Tazat and its database in ready-made boxes, so you install almost nothing yourself. Optional: the guide also shows the way without it. |
| Admin panel | Your control page in the browser, at `https://your-domain/admin`. You manage restaurants, users and the look of the app there. |

**GOOD TO KNOW**

Replace every `your-domain.com` and `YOUR_SERVER_IP` in this guide with your own. Everything else you can paste exactly as it is.

## PART 1 The server

1

### Rent a server

Any provider works (Hetzner, DigitalOcean, Vultr, Hostinger VPS, AWS Lightsail). Choose these settings:

- **Type:** VPS / Cloud server (not "shared hosting" or "cPanel": Tazat does not run there).
- **System:** Ubuntu 24.04 (or 22.04).
- **Size:** 2 GB memory, 1 or 2 CPUs, at least 20 GB disk.
- **Login:** choose "password" if asked. Write down the password.

When the server is ready, the provider shows its **IP address**, four numbers like `203.0.113.25`. This is `YOUR_SERVER_IP`.

2

## Point your domain at the server

Log in where you bought the domain and open its **DNS settings**. Add one record:

| Type | Name / Host | Value |
|---|---|---|
| A | `eat` (gives `eat.your-domain.com`) or `@` for the main domain | YOUR_SERVER_IP |

Save. It usually works within 10 minutes, sometimes a few hours. Check at dnschecker.org: type your address, choose **A**, and see your IP.

**TIP**

A sub-address such as `eat.your-domain.com` keeps your normal website untouched. This guide uses `eat.your-domain.com`.

3

## Connect to the server

Open the Terminal (Mac) or PowerShell (Windows) and type this. Replace the IP with yours, press Enter, type `yes` if it asks, then the password. You will not see the password while you type; that is normal.

```
ssh root@YOUR_SERVER_IP
```

When the line starts with `root@` you are inside the server. **Every command in steps 4, 6 and 7 is typed here.**

4

## Install what Tazat needs

First, for everyone: install a few small helpers, and open the web doors of the server (80 and 443) while keeping your own connection open. Paste this:

```
apt-get update && apt-get install -y curl unzip caddy ufw
ufw allow OpenSSH && ufw allow 80 && ufw allow 443 && ufw --force enable
```

**CAREFUL**

Run the `ufw` line exactly as written. It keeps your own connection open (`OpenSSH`) while closing the doors you do not need.

### IF YOU USE DOCKER

Docker is the program that runs Tazat and its database for you. Paste both lines, one after the other. The first takes 1 to 3 minutes.

```
curl -fsSL https://get.docker.com | sh
```

```
docker compose version
```

The second line must print a version such as `Docker Compose version v2.x`.

### IF YOU DO NOT USE DOCKER

**a) Install the database and Node.js.** The first line installs PostgreSQL with map support (PostGIS). The next two install Node.js 22, the program Tazat runs on. This takes a few minutes.

```
apt-get install -y build-essential openssl postgresql postgis
curl -fsSL https://deb.nodesource.com/setup_22.x | bash -
apt-get install -y nodejs
node -v
```

The last line must print `v22` followed by numbers. If it prints a smaller number, stop and see If something goes wrong.

**b) Create the database.** Paste the whole block. It makes a database called `tazat` with its own random password, and saves that password in the file `/root/tazat-db-password.txt` (you need it in step 6):

```
DBPASS=$(openssl rand -hex 16)
echo "$DBPASS" > /root/tazat-db-password.txt
chmod 600 /root/tazat-db-password.txt
sudo -u postgres psql -c "CREATE ROLE tazat LOGIN PASSWORD '$DBPASS';"
sudo -u postgres createdb -O tazat tazat
for e in postgis pg_trgm citext unaccent btree_gist; do sudo -u postgres
psql -d tazat -c "CREATE EXTENSION IF NOT EXISTS $e;"; done
```

You should see `CREATE ROLE` once and `CREATE EXTENSION` five times. Any line saying `ERROR` means a step did not work: see If something goes wrong.

5

### Upload Tazat to the server

**Easiest way, with a free program:**

1. Install FileZilla (free).
2. Top bar: **Host** `sftp://YOUR_SERVER_IP`, **Username** `root`, your password, **Port** `22`, then **Quickconnect**.
3. On the right side (the server) open the `/root` folder. On the left (your computer) find the Tazat **server folder**. Make a `.zip` of it first (right-click, "Compress"), then drag the zip to the right side. Do not upload the `tazat_restaurant` app folder; the server does not need it.

Back in the terminal window, unpack it. Use your zip's real name instead of `tazat-server.zip`:

```
mkdir -p /opt/tazat && unzip -o ~/tazat-server.zip -d /opt/tazat
ls /opt/tazat
```

The list must show `nest-backend`, `admin-web`, `vendor-web` and `locales`. If it shows only one folder, the zip had an extra folder inside; look in it with `ls /opt/tazat/*` and use that folder in the next step instead of `/opt/tazat`.

6

## Start Tazat

IF YOU USE DOCKER

This one command sets everything up: database, passwords, the server itself. Use your own address (it must start with `https://`):

```
cd /opt/tazat/nest-backend
chmod +x install.sh
./install.sh https://eat.your-domain.com
```

It takes **5 to 10 minutes** the first time while it builds (lots of text scrolls by; that is normal). When it is done it prints **"Ready."**

**WANT SAMPLE RESTAURANTS TO LOOK AROUND?**

Use `DEMO=1 ./install.sh https://eat.your-domain.com` instead. You can delete the samples later in the admin panel.

**IF IT STOPS WITH AN ERROR**

Read the last lines. "port is already allocated" means something else on the server uses port 3000. See If something goes wrong.

IF YOU DO NOT USE DOCKER

Four small parts. Do them in order. Use your own address (it must start with `https://`) in part a).

**a) Write the settings file.** It holds your address, the database password you saved in step 4 and fresh random secrets.

```
cd /opt/tazat/nest-backend
URL=https://eat.your-domain.com
grep -Ev
'^(NODE_ENV|APP_BASE_URL|MEDIA_BASE_URL|JWT_ACCESS_SECRET|JWT_REFRESH_SECRE
.env.example > .env
cat >> .env <<EOF
NODE_ENV=production
APP_BASE_URL=$URL
MEDIA_BASE_URL=$URL/media/
COOKIE_SECURE=true
TRUST_PROXY=true
```

```
CORS_ORIGINS=*
SWAGGER_ENABLED=false
DATABASE_URL=postgresql://tazat:$(cat /root/tazat-db-
password.txt)@localhost:5432/tazat?schema=public
JWT_ACCESS_SECRET=$(openssl rand -hex 32)
JWT_REFRESH_SECRET=$(openssl rand -hex 32)
APP_ENCRYPTION_KEY=$(openssl rand -base64 32)
EOF
chmod 600 .env
```

Nothing is printed when it works. Keep this file private and back it up: `APP_ENCRYPTION_KEY` protects the payment keys you save later.

**b) Build the two web panels.** This takes 3 to 6 minutes and prints a lot of text.

```
cd /opt/tazat/admin-web && npm ci && npm run build
cd /opt/tazat/vendor-web && npm ci && VITE_BASE=/vendor/ npm run build
```

Each build ends with a line containing `built in`.

**c) Build the server and create the tables.**

```
cd /opt/tazat/nest-backend
npm ci
npm run build
npm run db:migrate
npm run db:seed
```

`npm run db:migrate` ends with "All migrations have been successfully applied". For sample restaurants to look around, run `SEED_SAMPLE=true npm run db:seed` instead of the last line.

**d) Run Tazat all the time (also after a restart of the server).**

```
useradd --system --home-dir /opt/tazat --shell /usr/sbin/nologin tazat
chown -R tazat:tazat /opt/tazat
cat > /etc/systemd/system/tazat.service <<'EOF'
[Unit]
Description=Tazat
After=network.target postgresql.service

[Service]
User=tazat
```

```
WorkingDirectory=/opt/tazat/nest-backend
ExecStart=/usr/bin/node dist/main.js
Restart=always
RestartSec=3

[Install]
WantedBy=multi-user.target
EOF
systemctl daemon-reload
systemctl enable --now tazat
sleep 8
curl -s http://127.0.0.1:3000/health
```

The last line must print a short message containing `"status":"ok"`. If it prints nothing, look at the log: `journalctl -u tazat -n 50 --no-pager`.

7

### Switch on HTTPS (the padlock)

Paste this whole block. Put your own address in the second line:

```
cat > /etc/caddy/Caddyfile <<'EOF'
eat.your-domain.com {
  reverse_proxy localhost:3000
}
EOF
systemctl reload caddy
```

Wait about a minute (the padlock is created automatically), then open `https://eat.your-domain.com/health` in your browser. You should see a short message with `"status":"ok"` or `"up"`.

**IF THE PAGE DOES NOT OPEN**

The domain from step 2 is probably not pointing at your server yet. Wait ten minutes and try again.

8

## Activate Tazat with your purchase code

Tazat must be activated once on each new server. Until you do, every page (the admin panel, the vendor portal and the app) shows a page asking for your purchase code. That is expected, not an error. Follow these four small steps:

**8.1 Get your purchase code.** Sign in at codecanyon.net/downloads. Next to Tazat click **Download**, then choose **License certificate & purchase code**. Open the file or the page: the code is a line like `xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`. Copy it.

**8.2 Open the activation page.** In your browser type your own address followed by `/rtk/sync`, for example `https://eat.your-domain.com/rtk/sync`. You should see a small form with a **Purchase Code** box. If you see something else, check step 7 (the padlock) first.

**8.3 Fill in the form. Username:** your Envato username, the name you sign in with on Envato (not your e-mail). **Purchase code:** paste the code from 8.1, with no spaces before or after.

**8.4 Press the button and wait.** It can take a few seconds. When you see **"Sync successful"**, Tazat is activated. Now continue with step 9. You never have to do this again on this server.

GOOD TO KNOW

One purchase code works on one server at a time. If you later move Tazat to a new server, release the code on the old one first (see `documentation/13-activation.md`). Your server needs normal internet access for this step. If a message appears instead of "Sync successful", open the help answers at the end of this page.

9

### Create your admin account

1. Open `https://eat.your-domain.com/admin` (after the activation in step 8).
2. The first time, a **setup page** appears. Enter your app name, your name, your e-mail and a password of **at least 12 characters**.
3. Submit. You are in. **The setup page disappears after this**, so do it right away and save the password in a password manager.

Restaurant owners sign in at `https://eat.your-domain.com/vendor`.

Then open **System** in the left menu. It lists the checks. A green mark is fine; anything marked *Attention* tells you what to fix.

10

### Make it yours (no app rebuild needed)

Open **Branding** in the admin panel: app name, logo, colours, default language, and the links to your privacy policy and terms. The mobile app picks these up by itself.

Now add your categories and restaurants (or let restaurant owners register in the vendor portal and approve them in the admin panel).

**PART 1 IS FINISHED.**

Back up regularly. Your provider's "snapshot" or "backup" option is the simplest way: switch it on in the provider's website.

## PART 2 The mobile app

Do this on your own computer, not on the server. It needs Part 1 finished, because the app must know your server's address.

1

## Install the tools

| Tool | Why | Where |
|---|---|---|
| Flutter | Builds the app | docs.flutter.dev/get-started/install: pick your computer, follow the page |
| Android Studio | Android parts and the phone simulator | developer.android.com/studio |
| Xcode (Mac only) | iPhone version | The Mac App Store |
| Python 3 | Used once, to set your app name | python.org/downloads (Macs already have it) |

When they are installed, open a terminal and run this. It checks everything and tells you in plain words what is still missing:

```
flutter doctor
```

Fix each red line it shows (it explains how). Android and, on a Mac, iOS must be ticked. If you only plan to publish for Android, the iOS line may stay red.

2

## Open the app project and download its parts

In the terminal go to the app folder. Use the real place where you unzipped it (tip: type `cd` with a space, then drag the `tazat_restaurant` folder into the terminal window):

```
cd path/to/tazat_restaurant
flutter pub get
```

The second line downloads everything the app needs and ends with "Got dependencies".

3

## Tell the app where your server is

Open the file `env/live.json` in any text editor (Notepad, TextEdit, VS Code). It has a line named `API_BASE_URL`. Replace the address inside the quotes with your own server address, with `https://` and **no slash at the end**. Your own address is the same one you opened in step 7 of Part 1. Save the file.

**IMPORTANT**

The address must start with `https://`. The finished app refuses to start without it, so it can never silently talk to the wrong server.

4

## Add your Google Maps key

The app shows restaurants on a map and needs your own free key.

1. Go to console.cloud.google.com, create a project, and switch on billing (Google gives a monthly free amount; normal use stays inside it).
2. **APIs & Services → Library:** enable **Maps SDK for Android** and **Maps SDK for iOS**.
3. **APIs & Services → Credentials → Create credentials → API key.** Copy the key.
4. In the app folder, find `android/secrets.properties.example`, make a copy named `android/secrets.properties`, open it and put your key after `MAPS_API_KEY=`.
5. Do the same on a Mac for iPhone: copy `ios/Flutter/Secrets.xcconfig.example` to `ios/Flutter/Secrets.xcconfig` and put your key after `MAPS_API_KEY=`.

**LATER, FOR SAFETY**

In Google Cloud, edit the key and restrict it to your app's id (the one you choose in the next step) so nobody else can use it.

5

### Give the app your own name, id and icon

The **app id** is a unique name for your app in the stores, written like a reversed website, for example `com.yourcompany.eats` (lower case, no spaces). You can never change it after publishing, so choose carefully. One command sets the id and the name shown under the icon on Android and iPhone:

```
python3 tool/rename_app.py com.yourcompany.eats "Your App Name"
```

On Windows, if `python3` is not found, use `python` instead.

**Icon:** replace the two pictures `assets/images/app_icon.png` and `assets/images/app_icon_foreground.png` with your own (square, 1024 × 1024 pixels, same file names), then run:

```
dart run flutter_launcher_icons
```

Logo, colours and the app name shown *inside* the app are set in the admin panel (Branding), not here.

6

### Try it on a phone first

**Android phone:** Settings → About phone → tap "Build number" 7 times → Developer options → switch on **USB debugging**. Plug the phone into your computer and allow the question on the phone.

**iPhone (Mac):** plug it in, trust the computer on the phone, and in Xcode sign in once with your Apple ID (Xcode → Settings → Accounts).

Check that your computer sees the phone, then start the app:

```
flutter devices
flutter run --dart-define-from-file=env/live.json
```

The first start takes a few minutes. Then the app opens on the phone and shows the restaurants from your server. Add a restaurant in the admin panel and check it appears.

7

## Build the Android app for Google Play

**A. Make your signing key (once, keep it safe forever).** It proves the app is yours. If you lose it you can never update the app. Run the command, answer the questions, and remember the password you choose. It creates the file `my-release-key.jks` in your home folder:

```
keytool -genkey -v -keystore ~/my-release-key.jks -keyalg RSA -keysize 2048
-validity 10000 -alias upload
```

**B. Tell the app about it.** Create a new text file `android/key.properties` with these four lines (use your real password and the full path to the file; on Windows use forward slashes, `C:/Users/you/my-release-key.jks`):

```
storePassword=YOUR_PASSWORD
keyPassword=YOUR_PASSWORD
keyAlias=upload
storeFile=/full/path/to/my-release-key.jks
```

**C. Build.**

```
flutter build appbundle --release --dart-define-from-file=env/live.json
```

The file to upload to Google Play is `build/app/outputs/bundle/release/app-release.aab`. In the Play Console: Create app → Production → Create release → upload that file. Google then asks for the store text, screenshots and a privacy policy link.

**WANT A FILE TO INSTALL DIRECTLY ON A PHONE?**

Run `flutter build apk --release --dart-define-from-file=env/live.json`. The file is `build/app/outputs/flutter-apk/app-release.apk`.

**BACK UP**

Copy `my-release-key.jks` and your password somewhere safe, away from this computer (cloud drive or USB stick).

8

### Build the iPhone app for the App Store (Mac only)

1. Join the Apple Developer Program (99 USD a year; approval can take a day or two).
2. In a terminal: `open ios/Runner.xcworkspace`. In Xcode click **Runner** (left), tab **Signing & Capabilities**, tick **Automatically manage signing** and choose your **Team**. The Bundle Identifier is the id you chose in step 5.
3. Create the app on appstoreconnect.apple.com (My Apps → + → New App) using the same Bundle ID.
4. Build:

```
flutter build ipa --release --dart-define-from-file=env/live.json
```

5. The result is in `build/ios/ipa`. Upload it with Apple's free **Transporter** app (Mac App Store): drag the file in and press **Deliver**. After a few minutes it appears in App Store Connect → TestFlight, where you can test it, then submit it for review.

9

### Check before you submit to the stores

- Admin panel → **Branding**: **Privacy Policy** and **Terms** links are filled in. Both stores require them.
- In the app: Profile → **Delete account** works. Apple and Google require it, and it is already built in.
- Test the finished app on a real phone in light and dark mode.
- Prepare store text and screenshots (phone-size pictures of the app).

**PART 2 IS FINISHED.**
Your restaurant app is ready for review.

## Optional extras (do them later)

| Extra | What it does | Where to read |
|---|---|---|

| | | |
|---|---|---|
| E-mail | Sends sign-up confirmation and password-reset mails. Without it those mails are only written to a log. | `documentation/01-installation.md` → "E-mail" |
| Push messages | Send a notification to every phone from the admin panel (needs a free Firebase project). | `documentation/08-push-notifications.md` |
| Payments for restaurant owners | Stripe, PayPal, Razorpay or bank transfer for listing fees. | `documentation/12-billing-and-payments.md` |
| More languages | The admin panel and the app offer English, French, German, Spanish, Portuguese, Japanese, Korean, Russian, Indonesian and Arabic. Each person picks theirs in the language menu. | Nothing to install |
| Updating Tazat | Back up first. Upload the new version over the old one in `/opt/tazat` (keep your `.env`). **Docker:** run `./install.sh` again in `nest-backend`. **Without Docker:** repeat parts b) and c) of step 6 (the builds and `npm run db:migrate`), then `systemctl restart tazat`. Your data stays. | `documentation/01-installation.md` → "Updating" |

## If something goes wrong

▼ **Every page asks for a purchase code**

Tazat is not activated yet. Open `https://eat.your-domain.com/rtk/sync` and follow step 8 of Part 1. The app and the admin panel start working right after "Sync successful".

▼ **"First Deactivate from already activated Machine."**

This purchase code is already used on another server. Release it there first (`documentation/13-activation.md`, "Moving Tazat to another server"), then activate again here.

▼ **"Invalid purchase code" or "Sync failed"**

Check the code for typos or extra spaces and that the username is your Envato username, not your e-mail. "Sync failed" can mean the server cannot reach the internet: try again in a few minutes.

▼ **Without Docker: "node -v" shows a number below 22**

The NodeSource line did not run or failed. Run these two lines again and check `node -v`:

```
curl -fsSL https://deb.nodesource.com/setup_22.x | bash -
apt-get install -y nodejs
```

▼ **Without Docker: "ERROR: could not open extension control file" or "permission denied to create extension"**

The map support is not installed. Run `apt-get install -y postgresql postgis`, then paste the database block of step 4 again (if it says the role already exists, that is fine; run only the `for e in …` line).

▼ **Without Docker: Tazat does not start or the health check prints nothing**

Read the log:

```
journalctl -u tazat -n 50 --no-pager
```

"Invalid environment configuration" means a line in `/opt/tazat/nest-backend/.env` is missing or wrong: repeat step 6 part a). "password authentication failed" means the database password in `.env` does not match `/root/tazat-db-password.txt`. After any fix: `systemctl restart tazat`.

### ▼ Without Docker: the web panels show "Not found" or a blank page

Part b) of step 6 (the two builds) did not finish. Run them again, then `systemctl restart tazat`. The server must be restarted after every build.

### ▼ With Docker: "port is already allocated" while installing

Another program already uses port 3000 or 5432. Install on other ports, and use the same new port in the Caddy block of step 7 (replace `3000`):

```
PORT=8080 POSTGRES_PORT=5433 ./install.sh https://eat.your-domain.com
```

### ▼ The browser says the site is not secure or does not open

Your domain does not point at the server yet (step 2), or the server's web doors are shut. Check the domain at dnschecker.org, and run `ufw status` on the server: ports 80 and 443 must say ALLOW.

### ▼ I cannot sign in to the admin panel

You must use the `https://` address you gave to `install.sh`. If you opened it with the plain IP address or `http://`, cookies are refused. Use the https address.

### ▼ The setup page does not appear at /admin

An admin account already exists. Sign in with it. If you forgot the password, use "Forgot password" on the sign-in page (needs e-mail set up), or ask the seller for the reset steps.

### ▼ I want to see what the server is doing

**With Docker**, on the server:

```
cd /opt/tazat/nest-backend
docker compose --profile app logs --tail 100 api
```

**Without Docker:** `journalctl -u tazat -n 100 --no-pager`

Send those lines to support if you need help.

▼ **The app opens but shows no restaurants**

First check that restaurants exist and are approved in the admin panel. Then check that `API_BASE_URL` in `env/live.json` is exactly your address and that you built with `--dart-define-from-file=env/live.json`. Restaurants are listed in the language the user picked, so a person using French sees restaurants that have a French name; add that language when you create the restaurant.

▼ **The map is grey or empty in the app**

The Maps key is missing, mistyped, or the two Maps SDKs are not enabled in Google Cloud (step 4 of Part 2). Fix it, then run the build again.

▼ **"flutter" is not recognised**

Close the terminal and open a new one after installing Flutter. If it still fails, repeat the PATH step on the Flutter installation page.

▼ **Android build fails with "keystore" or "signing"**

Check the four lines in `android/key.properties`: the password, `keyAlias=upload` and the full path of the `.jks` file.

---

Need more detail? The `documentation` folder of your package has a longer technical chapter for each topic. Support: see `SUPPORT.md`.