# Technical documentation – **PONG**

**Table of Contents**

## 1. Project Overview

**PONG** is a web-based version of the classic Pong game with both single-player and real-time multiplayer modes.
Users can:

- Register and log in
- Create and join game rooms
- Play online in real time

## 2. Architecture

- **Frontend**: React + Vite
- **Backend**: Node.js + Express + Socket.IO
- **Database**: PostgreSQL
- **Proxy**: Nginx Proxy Manager
- **Cloudflare Tunnel**: Secures all traffic and exposes app to network
- **Hosting**: Raspberry Pi 4 with Ubuntu Server 24.04
- **Management**: Portainer.io for container orchestration

# 3. Backend

**Technologies**

- Node.js + Express
- PostgreSQL
- Socket.IO
- bcrypt (password hashing)
- JWT (authorization)

**All scripts and configuration exists in server.js file.**

**Imports and express configuration:**

```
const express = require('express');
const { Client } = require('pg');
const bcrypt = require('bcrypt');
const { nanoid } = require('nanoid');
const cors = require('cors');
const jwt = require('jsonwebtoken');

const PORT = 3000;
const KEY = 'Key for hashing';
//for security purposes JWT key should be moved to environment variable
const app = express();
app.use(cors({
        origin: '*',
        methods: ['GET', 'POST'],
        credentials: true
}))
.use(express.json())
```

## Database Configuration:

```
const pgClient = new Client({
        user: 'user',
        host: 'ip address of rPi’,
        database: 'pong',
        password: 'password',
        port: 5432,
//user and password also should be moved to environment variable
});
pgClient.connect()
        .then(() => console.log('PostgreSQL connected'))
        .catch(error => console.error('PostgreSQL connection error', error));
```

## HTTP Endpoints:

### /create - For user registration via POST:

```
app.post('/create', async (request, response) => {
        const {username, password} = request.body;
        try{
                const result = await pgClient.query(
                        'SELECT COUNT(*) FROM users WHERE username = $1',
                        [username]
                );
                if(result.rows[0].count === '0'){
                        const hashedPassword = await bcrypt.hash(password, 10);
                        await pgClient.query(
                                'INSERT INTO users (username, password) VALUES ($1, $2)',
                                [username, hashedPassword]
                        );
                        const token = jwt.sign({name: username}, KEY, {expiresIn: '1h'});
                        response.status(201).json({token});
                }
                else{
                        return response.status(409).json({message: 'User with this name exists'});
                }
        }
        catch(error){
                response.status(400).json({error: error.message});
        }
});
```

### /login – For user logging via POST:

```
app.post('/login', async (request, response) => {
    const {username, password} = request.body;
    try {
        const result = await pgClient.query(
            'SELECT password FROM users WHERE username = $1',
            [username]
        );
        if (result.rows.length === 0) {
            return response.status(401).json({error: 'This username doesnt exists'});
        }
        const dbPassword = result.rows[0].password;
        const isMatch = await bcrypt.compare(password, dbPassword)
        if (isMatch) {
            const token = jwt.sign({name: username}, KEY, {expiresIn: '1h'});
            response.status(200).json({token});
        }
        else {
            response.status(401).json({error: 'Invalid password'});
        }
    }
    catch (error) {
        response.status(500).json({error: error.message});
    }
});
```

## /verify – For verifying JWT via GET:

```
app.get('/verify', (request, response) => {
    const auth = request.headers['authorization'];
    const token = auth && auth.split(' ')[1];
    if (!token) return response.sendStatus(401);
    jwt.verify(token, KEY, (error, decoded) => {
        if (error) return response.sendStatus(403);
        response.status(200).json({name: decoded.name});
    });
});
```

**For both /create and /login endpoints JSON input looks the same:**

```
{
  "username": "user",
  "password": "pass"
}
```

**/verify gets JWT via HTTP GET header:**

```
headers:{
        ‘Authorization’: ‘Bearer TOKEN_JWT’
}
```

# WebSocket (Socket.IO):

## JWT authentication for WebSocket server access:

```
io.use((socket, next) => {
        const token = socket.handshake.auth.token;
        if (!token) return next(new Error('Authentication error: Token required'));
        jwt.verify(token, KEY, (error, decoded) => {
                if (error) return next(new Error('Authentication error: Invalid token'));
                next();
        });
});
```

## Real time events:

## create_room – for room creation:

```
socket.on('create_room', (username) => {

        const roomId = nanoid(6).toUpperCase();
        socket.join(roomId);
        socket.roomId = roomId;
        roomUsers.set(roomId, { player1: socket, p1name: username, p1y: 400});
        socket.emit('room_created',{ role: 'right', roomId: roomId });
});
```

## join_room – for joining existing room and starting game:

```
socket.on('join_room', (roomId, username) => {
        const room = io.sockets.adapter.rooms.get(roomId);
        const users = roomUsers.get(roomId);
        if ( room && room.size === 1 ){
                socket.join(roomId);
                socket.roomId = roomId;
                users.player2 = socket;
                users.p2name = username;
                users.p2y = 400;
                users.playAgain = 0;
                socket.emit('room_joined', { role: 'left'});
                io.to(roomId).emit('start_game', {p1: users.p1name, p2: users.p2name});
                gameLoop(roomId, socket);
        }
});
```

## move– for sending to other player paddle movement:

```
socket.on('move' , (data) => {
	const users = roomUsers.get(data.roomId);
	if(!users) return;
	if (data.role === 'right' && typeof data.paddleY === 'number'){
		users.p1y = data.paddleY;
	}
	else if(data.role === 'left' && typeof data.paddleY === 'number'){
		users.p2y = data.paddleY;
	}
	socket.to(socket.roomId).emit('opp_move', data.pageY);
});
```

## disconnect – for ending game and deleting room if opponent disconnects:

```
socket.on('disconnect', () => {
	const roomId = socket.roomId;
	const room = roomUsers.get(roomId);
	if (!room) return;
	socket.to(roomId).emit('opponent_disconnected');
	roomUsers.delete(roomId);
});
```

## play_again – for handling request for rematch:

```
socket.on('play_again', () => {
	const roomId = socket.roomId;
	const room = roomUsers.get(roomId);
	room.playAgain += 1;
	if(room.playAgain === 2){
		room.playAgain = 0;
		io.to(roomId).emit('start_game', {p1: room.p1name, p2: room.p2name});
		gameLoop(roomId, socket);
	}
});
```

# Game logic:

```
async function gameLoop (roomId) {
    const score = { p1: 0, p2: 0 };
    const ball = { x: 900, y: 400 }; //middle of pitch
    const directions = [
        { dx: 10, dy: 10 },
        { dx: 10, dy: -10 },
        { dx: -10, dy: 10 },
        { dx: -10, dy: -10 }
    ];
    let ballV = directions[Math.floor(Math.random() * 4)];
    //direction randomized as px per frame

    const intervalId = setInterval(async () => {
        ball.x += ballV.dx;
        ball.y += ballV.dy;

        if (ball.y <= 0 || ball.y >= 755) {
            ballV.dy *= -1;
        }
        if (ball.x <= 10 || ball.x >= 1765) {
            const status = await checkIfLost(roomId, ball);
            switch(status){
                case 0: // ball bounced off paddle
                    ballV.dx *= -1;
                    ball.x += ballV.dx;
                    break;
                case 1: //p1 lost
                    score.p2++;
                    io.to(roomId).emit('score', score);
                    ball.x = 900;
                    ball.y = 400;
                    ballV = directions[Math.floor(Math.random() * 4)];
                    await new Promise(res => setTimeout(res, 1000));
                    break;
                case 2: //p2 lost
                    score.p1++;
                    io.to(roomId).emit('score', score);
                    ball.x = 900;
                    ball.y = 400;
                    ballV = directions[Math.floor(Math.random() * 4)];
                    await new Promise(res => setTimeout(res, 1000));
                    break;
            }
        }

        if (score.p1 >= 10 || score.p2 >= 10) {
            io.to(roomId).emit('game_over', score);
            clearInterval(intervalId);
            return;
        }

        io.to(roomId).emit('ball_move', { x: ball.x, y: ball.y, score });
    }, 1000 / 60);   //60 FPS
}

function checkIfLost (roomId, ball) {
    return new Promise((resolve) => {
        const room = roomUsers.get(roomId);
        if (!room) {
            return resolve(0);
        }
        if (ball.x <= 10) { //checks if ball is on p1 side
            const paddleY = room.p1y;
            if (ball.y < paddleY || ball.y > paddleY + 80) { //checks if ball is on paddle
                resolve(1);
            }
            else {
                resolve(0);
            }
        }
        else if (ball.x >= 1765) { //checks if ball is on p2 side
            const paddleY = room.p2y;
            if (ball.y < paddleY || ball.y > paddleY + 80) {  //checks if ball is on paddle
                resolve(2);
            }
             else {
                resolve(0);
            }
        }
        else {
            resolve(0);
        }
    });
}
```

# 4. Frontend

**Technologies**

- React
- Vite
- Socket.IO-client
- TailwindCSS

## File structure:

```
├── docker-compose.yml
├── package.json
├── vite.config.js
├── src/
│   ├── App.jsx
│   ├── main.jsx
│   └── components/
│       ├── LandingPage.jsx
│       ├── Welcome.jsx
│       ├── WaitingRoom.jsx
│       ├── RoomCreator.jsx
│       ├── Game1.jsx
│       └── Game2.jsx
├── index.html
└── index.css
```

# Components

## App.jsx:

## - Controls main views and passes functions to switch between them.

```
const [view, setView] = useState('landing');
const [gameMode, setGameMode] = useState(null);
const viewManagement = () =>{
        switch (view) {
                case 'start':
                        return <Welcome
                                        onButtonClick={(mode) => {
                                                setView('waitingRoom');
                                                setGameMode(mode);
                                }}/>
                case 'landing':
                        return <LandingPage
                                        onLogin={() => setView('start')}
                                />;
                case 'waitingRoom':
                        return <WaitingRoom
                                        view={gameMode}
                                        onButtonClick={setView}
                                />;
        }
}
```

## LandingPage.jsx:

## - Main functions:

- **UseEffect checking if active JWT exists in local storage and log in if verified**

```
useEffect(() => {

        const token = localStorage.getItem('token');

        if (token){
                fetch('https://backend-pong.konradito.win/verify', {
                        method: 'GET',
                        headers: { Authorization: `Bearer ${token}` }
                })
                .then(async response => {
                        if (response.status === 200) {
                                const data = await response.json();
```

```
                    localStorage.setItem('username', data.name)
                    onLogin();
                }
                else {
                    localStorage.removeItem('token');
                }
            })
            .catch(() => localStorage.removeItem('token'));
    }
}, []);
```

- **Login via POST /login and save JWT in local storage**

```
const logIn = async (event) => {
    event.preventDefault();
    const credentials = {
        username: event.target.username.value,
        password: event.target.password.value
    };
    try{
        const response = await fetch ('https://backend-pong.konradito.win/login', {
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify(credentials)
        });
        if(response.status === 200){
            const data = await response.json();
            localStorage.setItem('token', data.token);
            localStorage.setItem('username', credentials.username);
            console.log('User logged');
            onLogin(); //activate case ‘start’ in App.jsx
        }
        else{
            const data = await response.json();
            console.error('Error:', data.message || data.error);
            alert('Error:', data.message || data.error);
        }
    }
    catch (error) {
        console.error('Connection error:', error);
        alert('Connection error:', error);
    }
}
```

- **Register via POST /create and save JWT in local storage**

```
const register = async (event) => {
	event.preventDefault();
	if (event.target.password.value === event.target.retypedPassword.value){ //form requires 
retyping password
		const credentials = {
			username: event.target.username.value,
			password: event.target.password.value
		};
		try{
			const response = await fetch ('https://backend-pong.konradito.win/create', {
				method: 'POST',
				headers: { 'Content-Type': 'application/json' },
				body: JSON.stringify(credentials)
			});
			if(response.status === 201){
				const data = await response.json();
				localStorage.setItem('token', data.token);
				localStorage.setItem('username', credentials.username);
				console.log('User created');
				onLogin(); //activate case 'start' in App.jsx
			}
			else{
				const data = await response.json();
				console.error('Error:', data.message || data.error);
				alert('Error:', data.message || data.error);
			}
		}
		catch (error) {
			console.error('Connection error:', error);
			alert('Connection error:', error)
		}
	}
	else{
		alert('Passwords didnt match');
	}
}
```

**Welcome.jsx:**
**- Welcome screen with game mode selection (single or multiplayer).**

**WaitingRoom.jsx:**

**- if single-player mode picked → launches Game1 component.**

**- if multiplayer mode picked → launches RoomCreator component.**

**Game1.jsx:**
**- Single-player (offline) mode:**

- **Game starts instant by useEffect**

```
useEffect(() => {
    if(game){
        window.addEventListener('mousemove', changePaddlePosition);
        let animationFrameId;
        const update = () => {
            paddleRef.current.style.top = `${paddleY.current}px`
            changeBallPosition();
            ballRef.current.style.right = `${ballXY.current.x}px`;
            ballRef.current.style.top = `${ballXY.current.y}px`;
            animationFrameId = requestAnimationFrame(update);
        };
        animationFrameId = requestAnimationFrame(update);
        return () => {
            cancelAnimationFrame(animationFrameId);
            window.removeEventListener("mousemove", changePaddlePosition);
        }
    }
}, [game]);
```

- **Mouse movement triggers changePaddlePosition function that change paddle coordinates ref**
- **changeBallPosition function is triggered every frame.**

- **If ball coordinate y is near top or bottom wall, ball dy (movement speed in y axis) is multiplied by -1.**
- **Same thing is happening if ball x is near left or right wall but it also checks if paddle y coordinate matches ball y and only if it is matching, dx is multiplied by -1.**

```
const changeBallPosition = () => {
        const x = ballXY.current.x;
        const y = ballXY.current.y;
        if(x <= 10) {
                if(checkIfLost()){
                        return;
        }
        ballV.current.dx *= -1;
        }
        else if(x >= 1765) {
                ballV.current.dx *= -1;
                setCounter(prevCounter => prevCounter + 1);
        }
        ballXY.current.x += ballV.current.dx;
        if (y >= 755){
                ballV.current.dy *= -1;
        }
        else if(y <= 0){
                ballV.current.dy *= -1;
        }
        ballXY.current.y += ballV.current.dy;
}
```

```
const checkIfLost = () => {
        if (ballXY.current.y < paddleY.current
        ||
        ballXY.current.y > paddleY.current + 80)
        {
                setGame(false);
                return true;
        }
        return false;
}
```

## RoomCreator.jsx:

## - Creates, joins and starts a game room via Socket.IO.

```
const createRoom = () => {
        connectSocket();
        socketRef.current.emit('create_room',localStorage.getItem('username'));
};

const joinRoom = () => {
        connectSocket();
        socketRef.current.emit('join_room', roomId.toUpperCase(), localStorage.getItem('username'));
};


const connectSocket = () => {
        const token = localStorage.getItem('token');
        const socket = io("https://backend-pong.konradito.win", {
                auth: {
                        token: token
                }
        });
        socketRef.current = socket;

        socket.on('room_created', (data) => {
                setRoomId(data.roomId);
                setRole(data.role);
                setCreatedRoom(true);
        })

        socket.on('room_joined', (data) => {
                setRole(data.role);
        })

        socket.on('start_game', (data) => {
                setPlayers({p1: data.p1, p2: data.p2});
                setGameStarted(true);
        })
};
```

**- Handles room code, player roles, game start.**

```
const handleCode = (event) => {
        if (event.target.value.length <= 6){
                setRoomId(event.target.value);
        }
}

const handleEnter = (event) => {
        if (event.key === 'Enter') {
                if (event.target.value.length === 6) {
                        joinRoom();
                }
                else{
                        alert('Room ID must be at least 6 characters long');
                }
        }
}
```

**- Starts Game2 component when both players are ready (gameStarted variable is true).**

**Game2.jsx:**

**- Two-player (online) mode:**

- **Real-time sync via Socket.IO.**

```
UseEffect(() => {

        socket.current.on('connect', () => {

                console.log("Connected to wss")

        });

        socket.current.on("opp_move", (y) => {
                if(role === 'left') {
                        if (y - 140 > 700) {
                                paddleRightRefY.current = 700;
                        }
                        else if (y - 140 < 0) {
                                paddleRightRefY.current = 0;
                }
                else {
                    paddleRightRefY.current = y - 140;
                }
            }
            else if(role === 'right') {
                if (y - 140 > 700) {
                    paddleLeftRefY.current = 700;
                }
                else if (y - 140 < 0) {
                    paddleLeftRefY.current = 0;
                }
                else {
                    paddleLeftRefY.current = y - 140;
                }
            }
        });
        socket.current.on('ball_move', (data) => {
            const x = data.x;
            const y = data.y;
            ballRef.current.style.right = `${x}px`;
            ballRef.current.style.top = `${y}px`;
        });
        socket.current.on('score', (data) => {
            setScore(data);
        })
        socket.current.on('game_over', (data) => {
            setGame(false);
            setScore(data);
            if (role === 'left') {
                setDidWon(data.p2 > data.p1);
            }
            else {
                setDidWon(data.p2 < data.p1);
            }
        })
        socket.current.on('opponent_disconnected', () => {
            setOpponentIsPresent(false);
        })

        socket.current.on('start_game', (data) => {
            setScore({p1: 0, p2: 0});
            setPlayers({p1: data.p1, p2: data.p2});
            setGame(true);
            setPlayAgainButton(true);
```

```
                setOpponentIsPresent(true);
        });

}, []);
```

- **Synchronizes paddle and ball movement.**

```
const changePaddlePosition = (event) => {
        if(role==='right') {
                if (event.pageY - 140 > 700) {
                        paddleRightRefY.current = 700;
                }
                else if (event.pageY - 140 < 0) {
                        paddleRightRefY.current = 0;
                }
                else {
                        paddleRightRefY.current = event.pageY - 140;
                }
                socket.current.emit('move', {roomId: roomId, role: role, paddleY: 
paddleRightRefY.current, pageY: event.pageY});
        }
        else if(role==='left') {
                if (event.pageY - 140 > 700) {
                        paddleLeftRefY.current = 700;
                }
                else if (event.pageY - 140 < 0) {
                        paddleLeftRefY.current = 0;
                }
                else {
                        paddleLeftRefY.current = event.pageY - 140;
                }
                socket.current.emit('move', {roomId: roomId, role: role, paddleY: 
paddleLeftRefY.current, pageY: event.pageY});
        }
}
```

## TailwindCSS

Besides basic style for menu and game components, app requires 1800x800 resolution because of game field size, so if browser window is smaller, it displays warning

```
@media (max-width: 1799px), (max-height: 799px) {
	main,
	.ball,
	.paddle,
	.counter,
	.menuBox,
	.youLostBox {
		display: none;
	}

	.tooSmall {
		display: flex;
	}
}
```

```
<div className={
	'tooSmall h-full w-full hidden justify-center items-center text-center text-white text-6xl'
}>
	<div>
		<h1>
			Window is too small.
		</h1>
		<h2>
			To play pong you need at least: 1800×800.
		</h2>
	</div>
</div>
```

## 5. Docker and deployment

All apps where deployed using portainer.io.

docker-compose.yml files:

Nginx Proxy Manager:

```
version: ‘3.8’

services:
  app:
    image: 'jc21/nginx-proxy-manager:latest'
    restart: unless-stopped
    ports:
      - '80:80'
      - '443:443'
      - '81:81'
    volumes:
      - ./data:/data
      - ./letsencrypt:/etc/letsencrypt
```

Cloudflared:

```
version: ‘3.8’
services:
  cloudflared:
    image: wisdomsky/cloudflared-web:2025.2.1@sha256:2c7ad1c94f56db004f587d4e9f71aa4d5b541e564dda973cab51458266c2c3a3
    entrypoint: /bin/sh -c "node /var/app/backend/app.js"
    environment:
      - EDGE_IP_VERSION=auto
      - METRICS_ENABLE=false
      - METRICS_PORT=60123
      - NODE_VERSION=18.20.7
      - PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
      - PROTOCOL=auto
      - VERSION=2025.2.1
      - WEBUI_PORT=14333
      - YARN_VERSION=1.22.22
    restart: unless-stopped
```

Backend:

docker-compose.yml:

```
version: ‘3.8’
services:
	backend:
		build: .
		container_name: pong_backend
		ports:
			- "3000:3000"
		environment:
			- NODE_ENV=production
		labels:
			- com.casaplatform.app=pong_backend
		restart: unless-stopped
```

Dockerfile

```
FROM node:20-alpine

WORKDIR /app

COPY package*.json ./

RUN npm install --omit=dev

COPY . .

EXPOSE 3000

CMD ["npm", "start"]
```

Frontend:

docker-compose.yml:

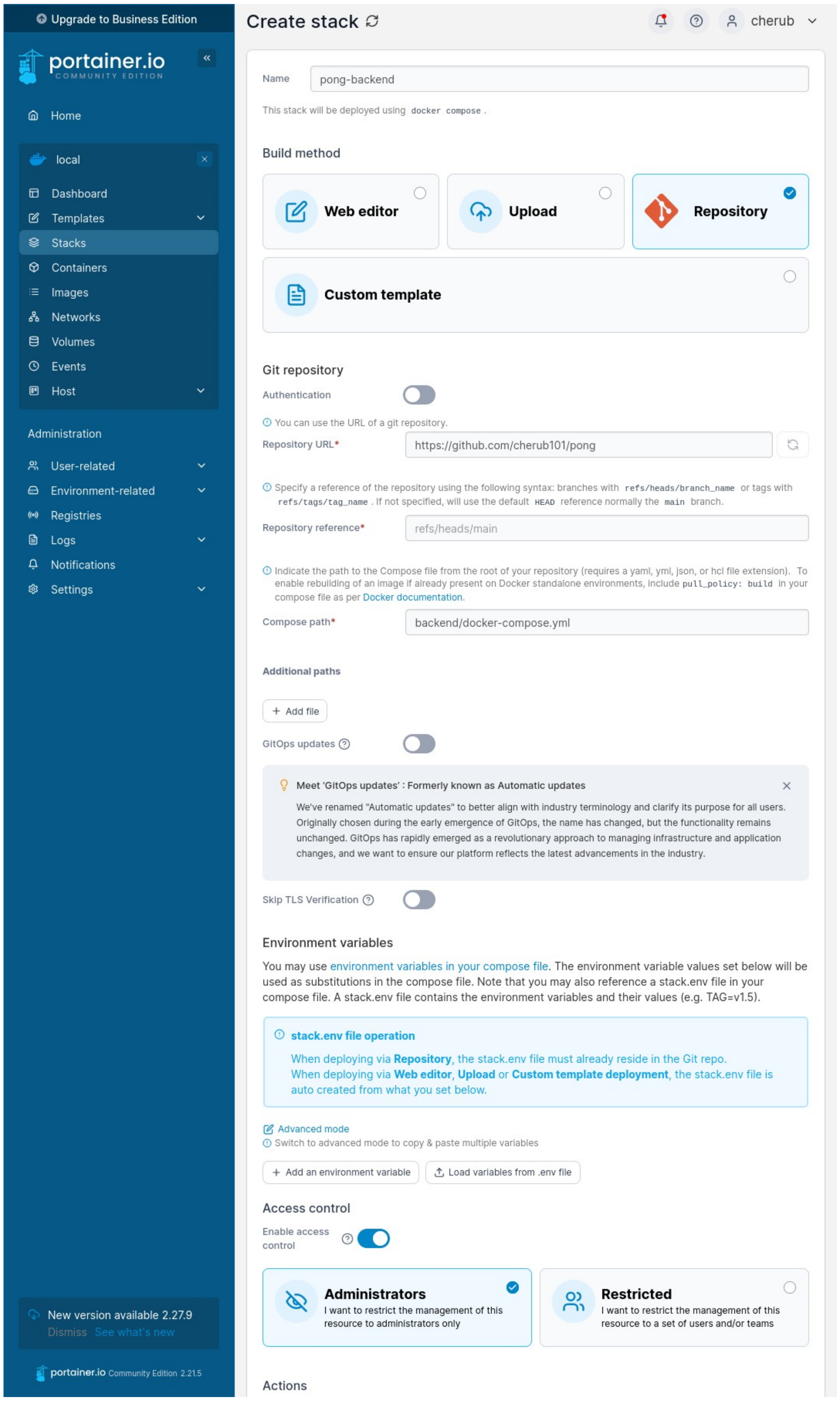
```
version: "3.8"
services:
        pong-react-app:
                build: .
                ports:
                        - "3001:80"
                environment:
                        - NGINX_VERSION=1.29.0
                        - NJS_VERSION=0.9.0
                deploy:
                        resources:
                                limits:
                                memory: 1846M
                restart: unless-stopped
```

Dockerfile

```
FROM node:20-alpine AS build
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build
FROM nginx:alpine
COPY --from=build /app/dist /usr/share/nginx/html
EXPOSE 3001
CMD ["nginx", "-g", "daemon off;"]
```

NPM and Cloudflared were directly composed by script but frontend and backend were downloaded from git.

Example for backend (for frontend, remember to change compose path to frontend/docker-compose.yml):



Create stack

Name: pong-backend

This stack will be deployed using `docker compose`.

Build method: Repository

Git repository

Authentication: off

You can use the URL of a git repository.

Repository URL*: https://github.com/cherub101/pong

Specify a reference of the repository using the following syntax: branches with `refs/heads/branch_name` or tags with `refs/tags/tag_name`. If not specified, will use the default `HEAD` reference normally the `main` branch.

Repository reference*: refs/heads/main

Indicate the path to the Compose file from the root of your repository (requires a yaml, yml, json, or hcl file extension). To enable rebuilding of an image if already present on Docker standalone environments, include `pull_policy: build` in your compose file as per Docker documentation.

Compose path*: backend/docker-compose.yml

Additional paths

GitOps updates: off

Meet 'GitOps updates' : Formerly known as Automatic updates

We've renamed "Automatic updates" to better align with industry terminology and clarify its purpose for all users. Originally chosen during the early emergence of GitOps, the name has changed, but the functionality remains unchanged. GitOps has rapidly emerged as a revolutionary approach to managing infrastructure and application changes, and we want to ensure our platform reflects the latest advancements in the industry.

Skip TLS Verification: off

Environment variables

You may use environment variables in your compose file. The environment variable values set below will be used as substitutions in the compose file. Note that you may also reference a stack.env file in your compose file. A stack.env file contains the environment variables and their values (e.g. TAG=v1.5).

**stack.env file operation**

When deploying via **Repository**, the stack.env file must already reside in the Git repo.
When deploying via **Web editor**, **Upload** or **Custom template deployment**, the stack.env file is auto created from what you set below.

Switch to advanced mode to copy & paste multiple variables

Access control

Enable access control: on

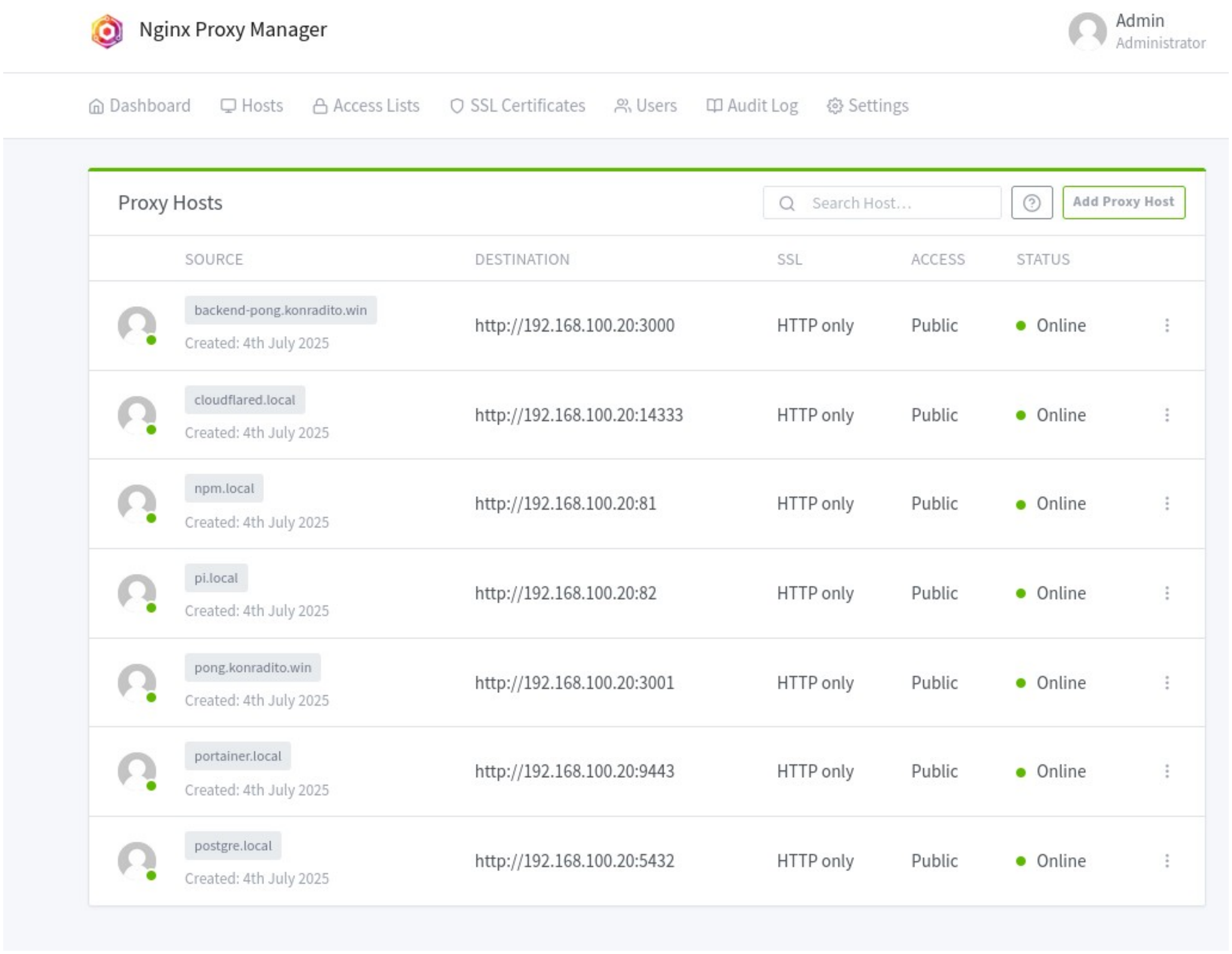
[✓ **Administrators** I want to restrict the management of this resource to administrators only]

**Restricted** I want to restrict the management of this resource to a set of users and/or teams

Actions

# 6. Port forwarding and DNS configuration

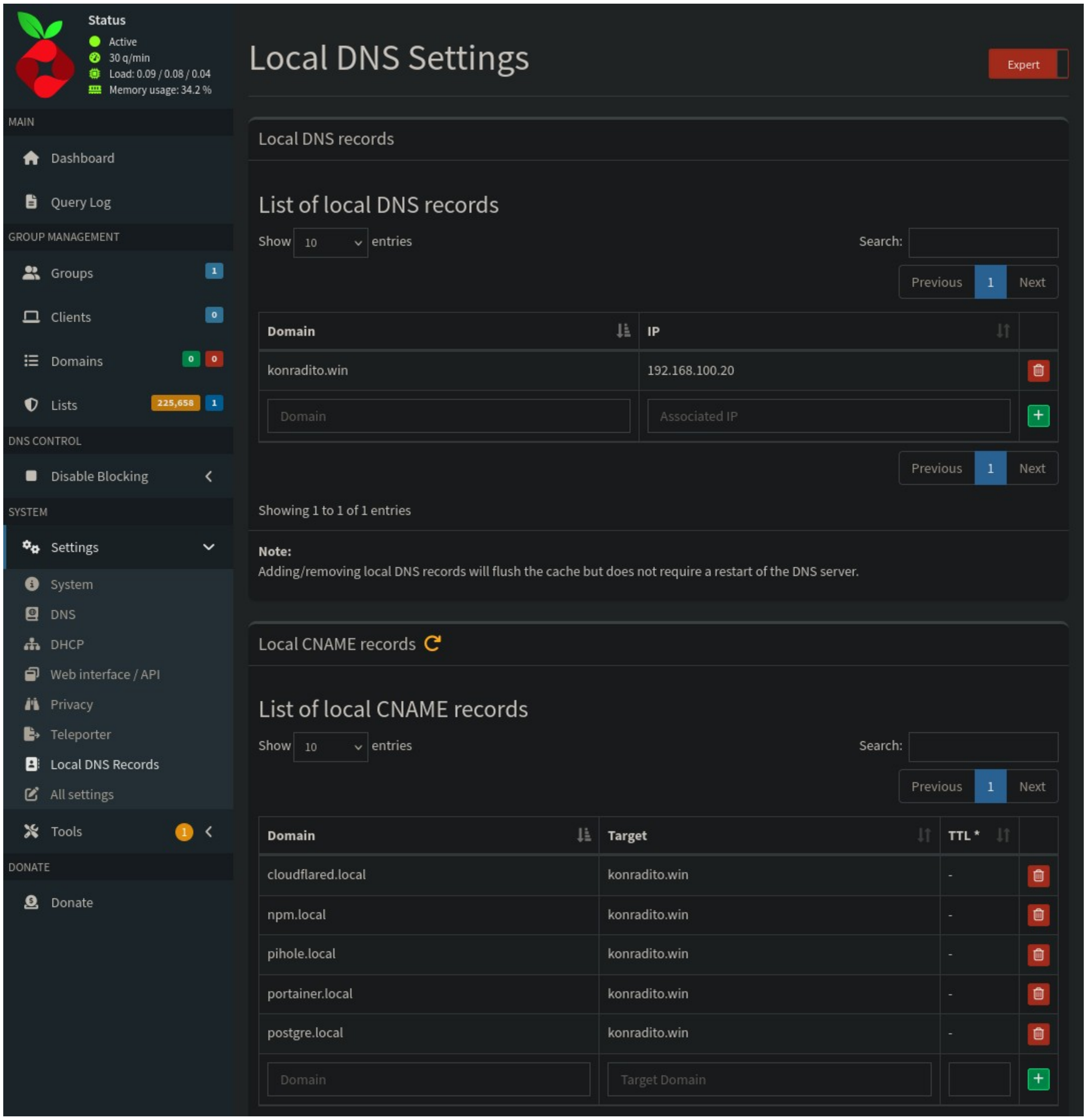
## Nginx proxy manager:



| SOURCE | DESTINATION | SSL | ACCESS | STATUS |
|---|---|---|---|---|
| backend-pong.konradito.win Created: 4th July 2025 | http://192.168.100.20:3000 | HTTP only | Public | ● Online |
| cloudflared.local Created: 4th July 2025 | http://192.168.100.20:14333 | HTTP only | Public | ● Online |
| npm.local Created: 4th July 2025 | http://192.168.100.20:81 | HTTP only | Public | ● Online |
| pi.local Created: 4th July 2025 | http://192.168.100.20:82 | HTTP only | Public | ● Online |
| pong.konradito.win Created: 4th July 2025 | http://192.168.100.20:3001 | HTTP only | Public | ● Online |
| portainer.local Created: 4th July 2025 | http://192.168.100.20:9443 | HTTP only | Public | ● Online |
| postgre.local Created: 4th July 2025 | http://192.168.100.20:5432 | HTTP only | Public | ● Online |

Names .local where configured in pi-hole DNS configuration

## Pi-hole:



Status
Active
30 q/min
Load: 0.09 / 0.08 / 0.04
Memory usage: 34.2 %

# Local DNS Settings

Local DNS records

List of local DNS records

Show 10 entries

| Domain | IP |
|---|---|
| konradito.win | 192.168.100.20 |

Showing 1 to 1 of 1 entries

**Note:**
Adding/removing local DNS records will flush the cache but does not require a restart of the DNS server.

Local CNAME records

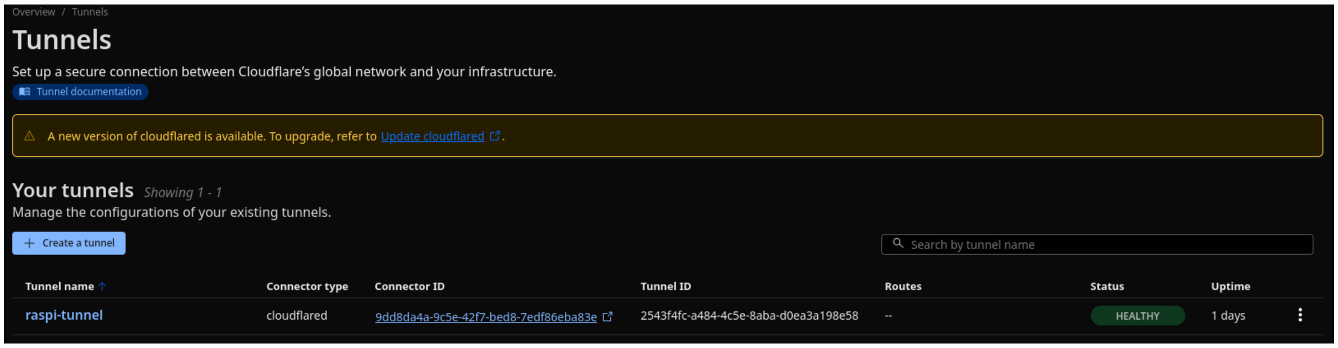
List of local CNAME records

Show 10 entries

| Domain | Target | TTL * |
|---|---|---|
| cloudflared.local | konradito.win | - |
| npm.local | konradito.win | - |
| pihole.local | konradito.win | - |
| portainer.local | konradito.win | - |
| postgre.local | konradito.win | - |

Backend and frontend dns records where added to cloudflare while creating tunnel

## Cloudflare:



Tunnels

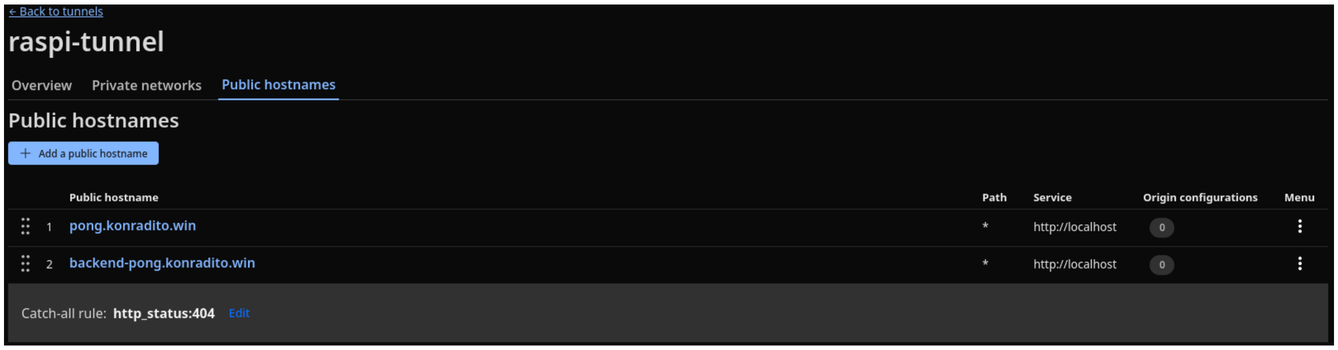
Set up a secure connection between Cloudflare's global network and your infrastructure.

A new version of cloudflared is available. To upgrade, refer to Update cloudflared.

Your tunnels *Showing 1 - 1*

Manage the configurations of your existing tunnels.

| Tunnel name | Connector type | Connector ID | Tunnel ID | Routes | Status | Uptime |
|---|---|---|---|---|---|---|
| raspi-tunnel | cloudflared | 9dd8da4a-9c5e-42f7-bed8-7edf86eba83e | 2543f4fc-a484-4c5e-8aba-d0ea3a198e58 | -- | HEALTHY | 1 days |



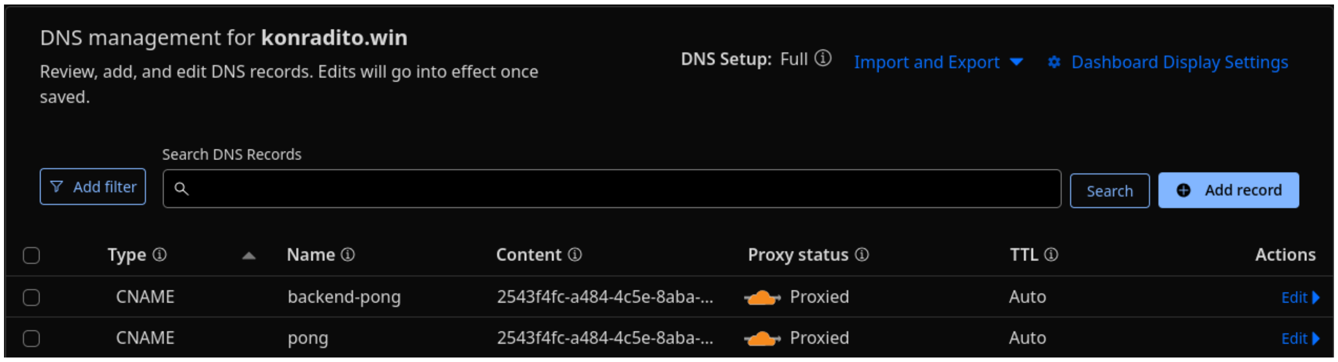
raspi-tunnel

Public hostnames

| | Public hostname | Path | Service | Origin configurations | Menu |
|---|---|---|---|---|---|
| 1 | pong.konradito.win | * | http://localhost | 0 | |
| 2 | backend-pong.konradito.win | * | http://localhost | 0 | |

Catch-all rule: **http_status:404** Edit

When public hostname is created, the CNAME record pointing to tunnel is automatically created:



DNS management for **konradito.win**

Review, add, and edit DNS records. Edits will go into effect once saved.

DNS Setup: Full

Search DNS Records

| Type | Name | Content | Proxy status | TTL | Actions |
|---|---|---|---|---|---|
| CNAME | backend-pong | 2543f4fc-a484-4c5e-8aba-... | Proxied | Auto | Edit |
| CNAME | pong | 2543f4fc-a484-4c5e-8aba-... | Proxied | Auto | Edit |

## 7. Final Notes

This project was developed as part of a student initiative to explore real-time multiplayer applications using modern web technologies. It showcases:

- A functional web-based Pong game with single-player and real-time multiplayer modes
- User authentication using JWT and bcrypt
- Real-time communication with Socket.IO and WebSocket
- A complete Dockerized deployment pipeline using Nginx Proxy Manager, Cloudflare Tunnel, and Portainer
- Responsive and scalable frontend architecture with React, Vite, and TailwindCSS

---

This is the first version of the project.
It is currently in an active development stage, and several enhancements are planned, including:

- Matchmaking and public lobby system
- ELO rating for multiplayer and a scoreboard for single-player mode
- Improved security (e.g., rate limiting, input validation)
- Power-ups to enhance gameplay dynamics
- Improved styling and animations for a better user experience

---

Author:

Developed and maintained by cherub