



W3C Web of Things Community Meetup - Edge Computing

Script and Deploy WoT Applications to the Serverless Edge

Arturo Romero (CEO)
Miguel Romero (CTO)

15.12.2022



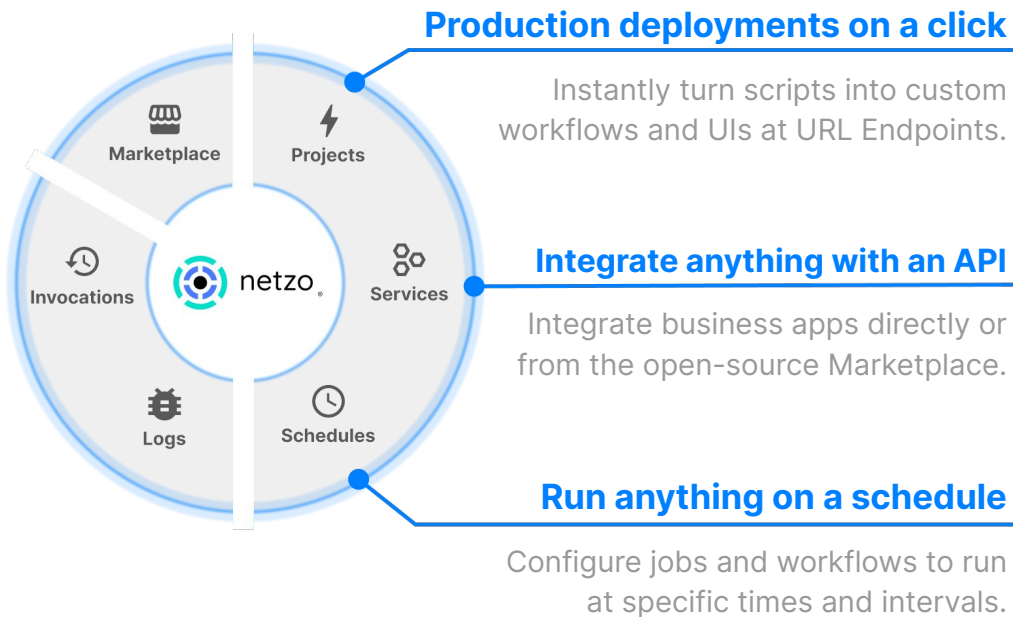
```
...land/x/sift@0.6.0/mod.ts";
...ts"
...es.tsx"
...deno-gfm
...//deno.land/x/gfm/mod.ts";
...json" assert { type: 'json' };

...v.get('DENO_DEPLOYMENT_ID')
...mentId.replace("dev-", "").split('-')
...ps://${deploymentId}.netzo.io`

...access-control-allow-origin': '*' }

...handler(req: Request, _connInfo, params): Promise<Response> {
...= new URL(req.url)
...templateId = url.pathname.split('/')[2] // templateId is always second
...ie.log(templateId)
...ch (req.method) {
case 'GET': {
  if (templateId) {
    return jsx(<HtmlTemplate page={templateId} />, { headers })
  }
}
else {
  const markdownResponse = await fetch(`https://api.netzo.io/projects/${id}/readme.md`)
  const readme = await markdownResponse.text()
  const body = render(readme, projectURL);
  const html = `
    <!DOCTYPE html>
    <html lang="en">
      <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <style>
          main {
            max-width: 800px;
            margin: 0 auto;
          }
          ${CSS}
        </style>
      </head>
      <body>
        <main data-color-mode="light" data-light-theme="light" data-dark-theme="dark" class="markdown-t
  `
}
```

Deploy JavaScript and TypeScript to URL endpoints instantly. Connect essential APIs, automate processes and build tools faster, without managing infrastructure.



Webhook Handling

Request Proxying

Data Processing

Admin Panels

Form Handling

Server-side
Rendering (with JSX)

KPI Dashboards

Request Scheduling
(Cron Jobs)

Workflow Automation

and more...

Instantly turn scripts into automations and UIs



Integrate anything with an API



Deploy instantly, skip the Devops



Code fast in the browser



Run anything on a schedule



Automate work and get more done



Empower developers to innovate

The screenshot displays the Netzo IDE interface. On the left, a sidebar shows a file explorer with a project named 'SSR with JSX' containing a file 'mod.tsx'. The main editor area shows the code for 'mod.tsx', which imports 'serve', 'renderToString', and 'Netzo' from Deno, and defines a handler function. On the right, a 'Preview' panel shows a GET request to 'https://(deploymentId).netzo.io' with a 'SEND' button. Below the preview, a 'CONFIGURATION' section shows 'Query', 'Headers', 'Body', and 'Variables' tabs. The 'Body' tab is active, showing a 200 success status and a preview of a grid of colored squares (green, purple, red, blue, yellow, orange). At the bottom left, the 'Imports' section shows 'netzo' as an imported module.



Sales and Marketing

Close more sales, faster by optimizing lead generation, prospect management and revenue generation



Production

Capture production process and inventory data to improve quality and supply chain controls



Operations

Keep data in sync across all teams and build portals and interfaces that enhance internal business operations



Internet of Things

Enhance process data, trigger workflows and augment operations through IoT device interactions



Accounting and Finance

Load and sync electronic invoice data with multiple systems to enhance reporting, file naming and storage.



IT

Increase cybersecurity and minimize disruptions through isolated functions and identify incidents immediately.

and more...

Workers: Custom HTTP endpoints for any use case



[Click Here](#)



URL imports



Env Variables



Native TS



TSX/JSX



Web Standards

```
/** @jsx h */
import { serve } from "https://deno.land/std@0.140.0/http/server.ts";
import { h } from "https://esm.sh/preact@10.5.15";
import { renderToString } from "https://esm.sh/preact-render-to-string@5.1.19?deps=preact@10.5.15";
import { Netzo } from "https://deno.land/x/netzo@v0.1.41/mod.ts";

const netzo = Netzo({ apiKey: Deno.env.get("NETZO_API_KEY") });

const { client } = await netzo.service(Deno.env.get("SERVICE_ID"));

async function handler(_req: Request): Promise<Response> {
  const photos = await client.photos.get({ albumId: 1 });

  const page = (
    <div>
      {photos.map(photo =>
        <img src={photo.thumbnailUrl} alt={photo.title} />
      )}
    </div>
  );

  const html = renderToString(page);

  return new Response(html, {
    headers: {
      "access-control-allow-origin": "*",
      "content-type": "text/html"
    },
  });
}

serve(handler);
```

Netzo SDK



API Integrations



SSR (server- side rendering)



HTTP Server to handle Requests



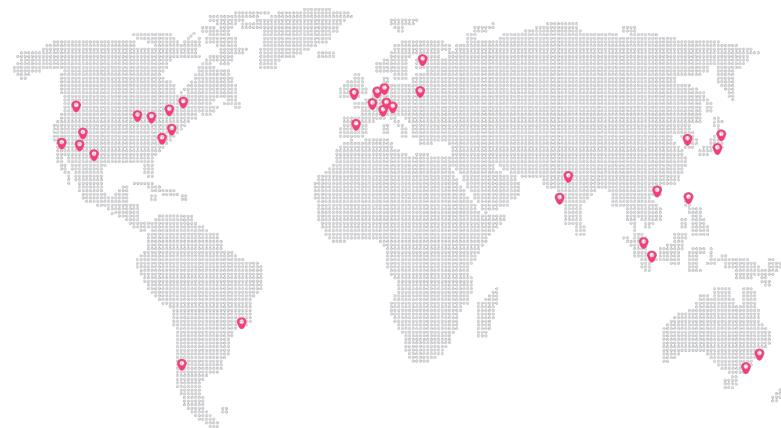
Runs locally as well: `deno run -A https://api.netzo.io/projects/:id/mod.tsx`

Scripting WoT applications at worker runtimes (serverless edge)

Worker Runtimes are the new standard for writing HTTP servers in JavaScript.

Netzo streamlines scripting and deployment of Web of Things applications, enabling users to bring ideas to production faster. For the Web of Things, scripts can be used to...

- **augment** existing WoT deployments
- **integrate** WoT entities with cloud services
- **proxy** WoT entities (act as HTTP Gateway)
- **process** data (aggregated, transform, etc.)
- **respond** to events (e.g. webhooks)
- **automate** interactions of WoT Things
- **schedule** recurring jobs



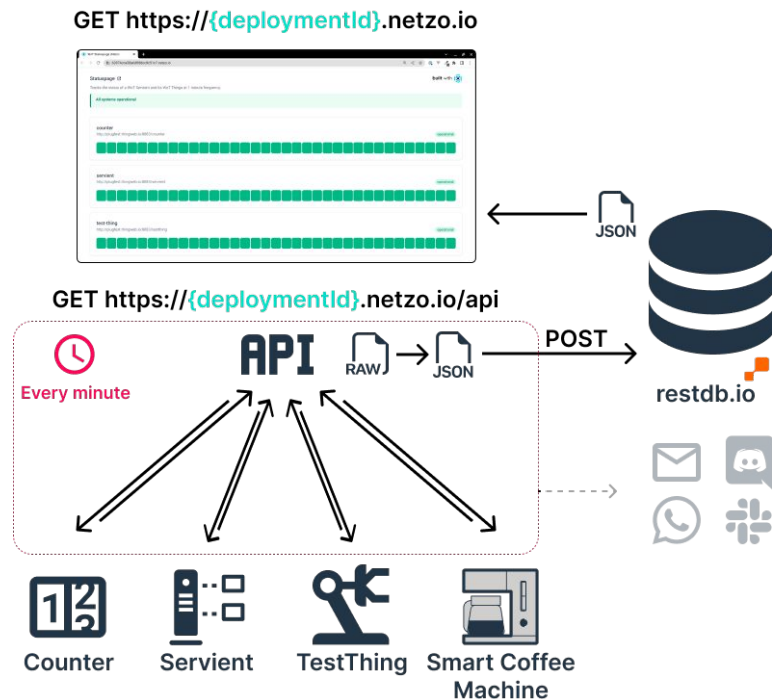
Demo 01: WoT statuspage for multiple WoT systems



[Click Here](#)

Goal: Provide monitoring capabilities to existing WoT Things and automate status reporting (e.g. notifications, email)

- `GET /` → serves UI
- `GET /api` → serves JSON data after fetching, aggregating and storing real-time state to DB
- **Process:** aggregate real-time state
- **Store:** state of WoT systems to DB
- **Alert:** when systems are down



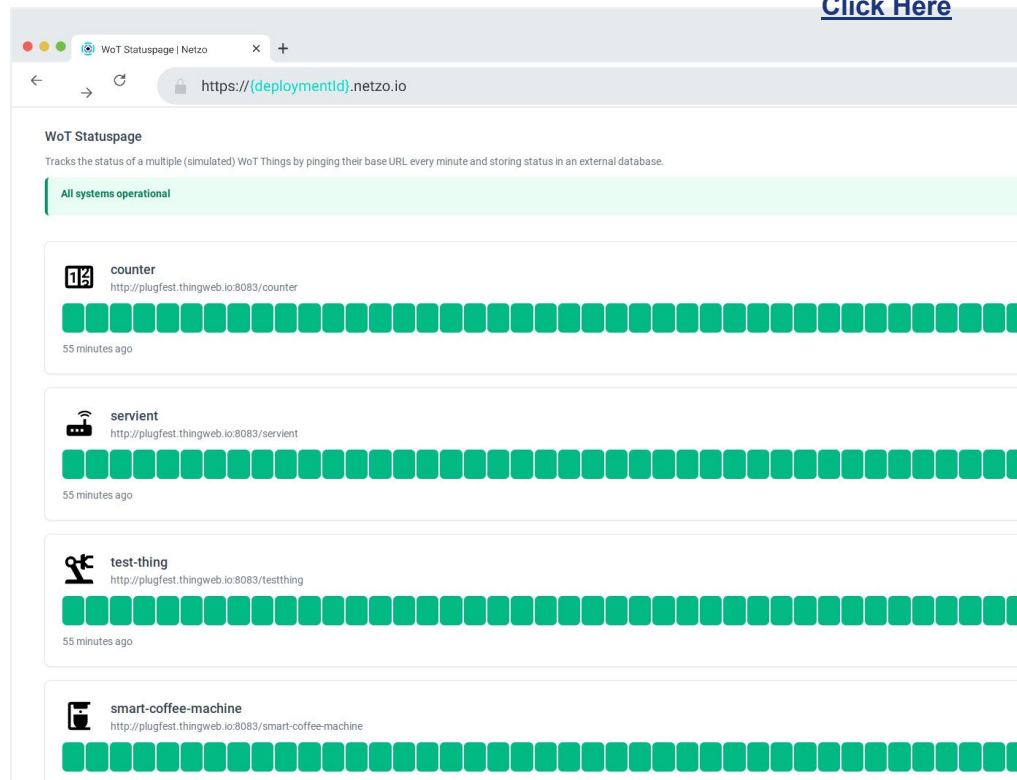
Demo 01: WoT statuspage for multiple WoT systems



[Click Here](#)

This demo showcases:

- **SSR** (server-side rendered) UI built with TSX (no build step)
- **Schedules**: polls the real-time state and persists it to an external database
- **API Integrations**: GETs real-time state and POSTs it to external database
- **Database**: persist data to external DB
- **Routes**: separate handlers for routes



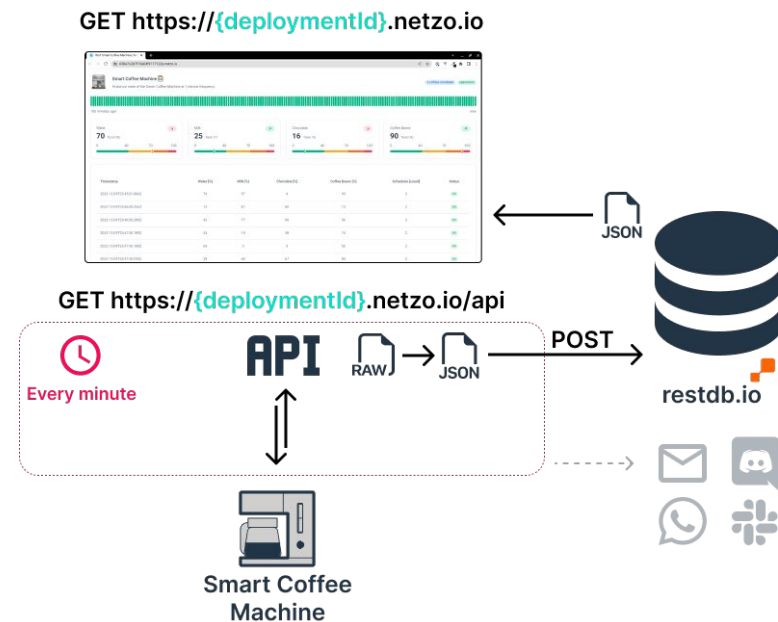
Demo 02: WoT dashboard for monitoring and control



[Click Here](#)

Goal: Persists historical state (time-series data) of a WoT Thing, augment it, provide a dashboard for monitoring and control and add reporting (e.g. via notifications, email)

- GET / → serves UI
- **Collect:** data from multiple sources
- **Process:** aggregate real-time stats
- **Visualize:** render information and insights in a actionable dashboard
- **Report:** notify to external services



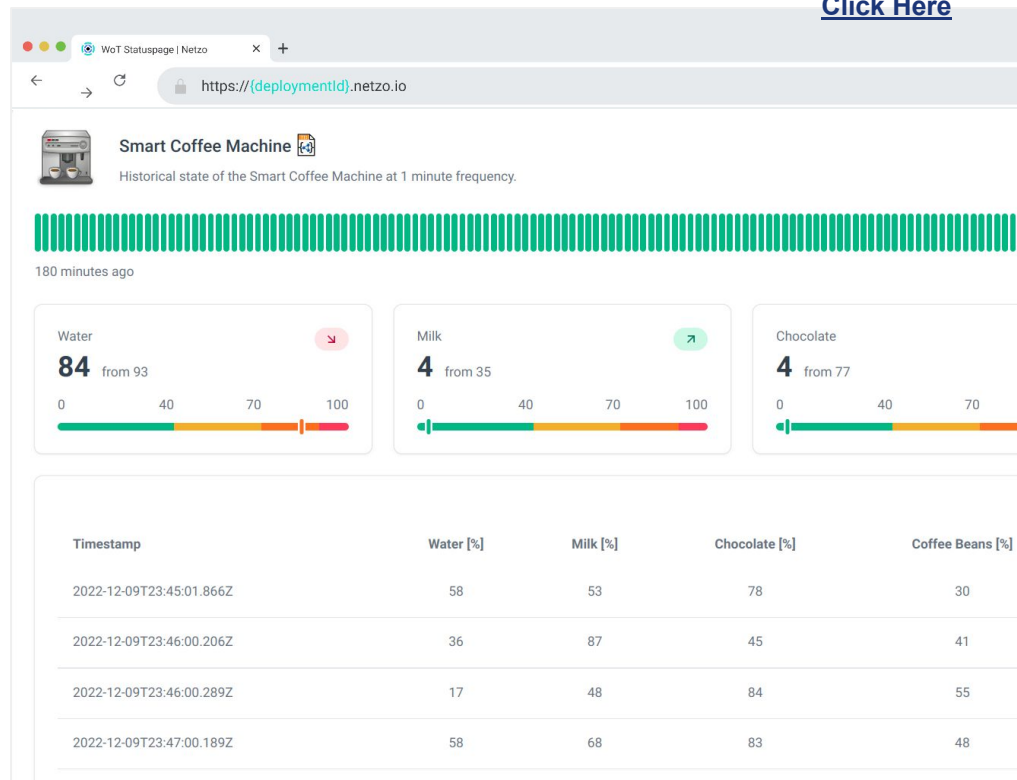
Demo 02: WoT dashboard for monitoring and control




[Click Here](#)

This demo showcases:

- **Environment Variables:** provides separate dev/prod environments
- **Static file hosting:** serves WoT Thing Description (JSON) at a Web URL
- **SSR** (server-side rendered) UI built with TSX (no build step)
- **Schedules:** polls the real-time state and persists it to an external database
- **API Integration:** GETs real-time state and POSTs it to external database
- **Database:** persist data to external DB



Marketplace of ready-made solutions

**Marketplace**

Marketplace items are ready-made solutions for specific use-cases. Anything you find here is public and you are free to fork into any or your Workspaces. When you fork an items, a copy is created so you can go ahead and make changes. Forked items will count normally for your Workspace usage.

TYPE

☒ All
☐ Services
☐ Projects

STATUS

☒ All
☐ Stable
☐ Beta
☐ Alpha
☐ Requested
☐ Deprecated

CATEGORY

☒ All
☐ Core
☐ Community
☐ Enterprise

Filter by name



Aa


Request



Contribute



i



admin analytics artificial-intelligence billing blog chart communications crm database energy example form framework google infrastructure integrations iot >



**ActiveCampaign**  Alpha
service
Service for the ActiveCampaign...
crm marketing



**Bar Chart with Billboard...**
project
An HTTP server that serves a ...
example chart



**Bigin**  Requested
service
Service for the Bigin API
crm



**Clarifai**  Alpha
service
Service for the Clarifai API
artificial-intelligence machine-learning



**Cloudflare**  Alpha
service
Service for the Cloudflare API
infrastructure



**Discord**  Alpha
service
Service for the Discord API
social communications



**Enode**  Requested
service
Service for the Enode API
iot energy solar



**Enphase**  Requested
service
Service for the Enphase API
iot energy solar



**Ergast F1**  Alpha
service
Service for the Ergast F1 API
mock



**Facturama**  Alpha
service
Service for the Facturama API
billing


**Fathom Analytics**  Alpha
service
Service for the Fathom Analyti...
analytics


**FaunaDB**  Alpha
service
Service for the FaunaDB Graph...
database


**Get Client IP Address** 
project
An HTTP server that responds ...
example json

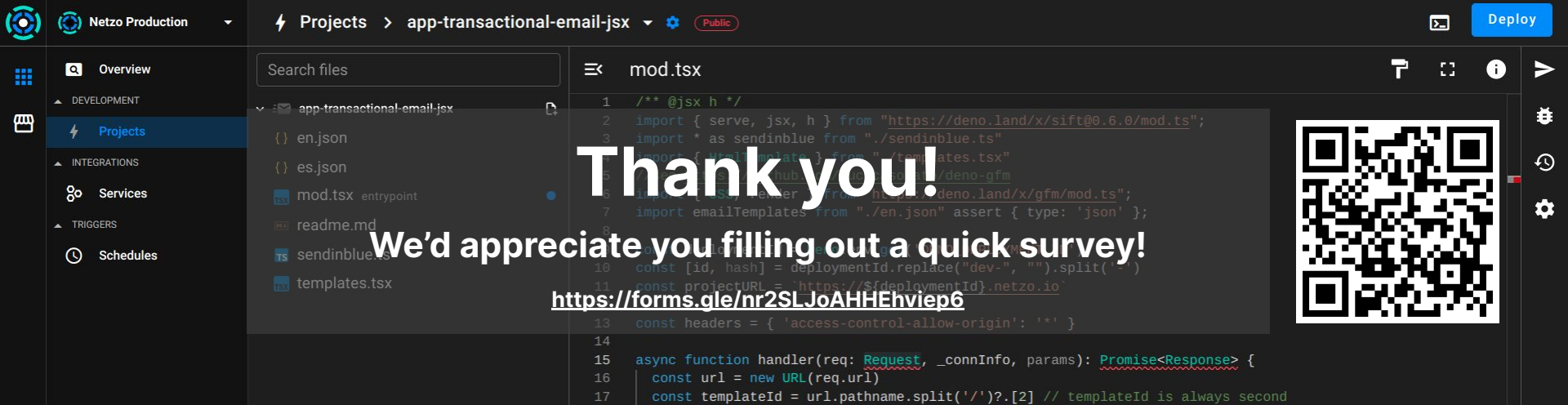
**GitHub**  Alpha
service
Service for the GitHub API
infrastructure

**Google AppSheet**  Alpha
service
Service for the Google AppShe...
productivity

**Google Sheets** Requested
service
Service for the Google Sheets ...
productivity google

**Handling Form Submis...**
project
An HTTP server that serves a ...
example form json

**Hashnode** Requested
service
Service for the Hashnode API
social blog



netzo

ROKAWARE SL

P. de la Castellana 89, Planta 8ª
28046 Madrid, España
netzo.io | hello@netzo.io | +34 910 601 536
Netzo © ROKAWARE SL

The information provided in this document has been obtained from sources we believe to be reliable. However, we cannot guarantee the accuracy or completeness of this information and assume no liability for it. Copies of the contents of this presentation, in particular videos, screenshots, printouts, copies or publications on electronic media, shall only be authorized with the written consent of ROKAWARE SL.