SunshineSpend Intern Quickstart — Autoreload Setup (PHP, Flask, Node, MySQL)

Updates: Aug 22, 2025

Last Updated: August 23, 2025

Access & Authentication

• Your private site: https://teamX.sunshinespend.com (X = 1..8).

• You will be prompted for Basic Auth. Username is your team name (e.g., team0X).

• SFTP/VS Code Remote-SSH: log in as your team user; change directory to /srv/groups/team0X and place files under your team folders below.

Project Layout

```
/srv/groups/teamXX/

  web/      # PHP (served directly by Apache)

  python/   # Flask app (served at /pyapp; chatbot at /bot → /pyapp/ask)

  node/     # Node app (served at /nodeapp)
```

How Your Apps Are Served

• PHP: files in /srv/groups/teamXX/web are executed by PHP-FPM.

• Flask: Gunicorn runs your Flask app on 127.0.0.1:900X (X = team number); Apache proxies it at /pyapp.

• Chatbot: POST /bot is forwarded to your Flask ~~/pyapp~~/ask endpoint.

• Node: Node runs on 127.0.0.1:910X; Apache proxies it at /nodeapp.

Autoreload (No Restarts Needed)

• Flask: Gunicorn is configured with --reload. Saving any file under /srv/groups/teamXX/python/ auto-reloads the app within ~1–2 seconds.

• Node: Node 20+ runs with --watch. Saving files under /srv/groups/teamXX/node/ auto-restarts the process.

• Just save your changes and refresh your browser. No admin restarts required.

MySQL — Your Team Database

• Database: teamXXdb   User: teamXXu   (ask the admin for your password)

• Host: localhost Charset: utf8mb4

PostgreSQL — Your Team's Vector Database (RAG support)

• Database: teamX   User: teamX   (ask the admin for your password)

PYTHON/FLASK/GUNICORN - Your Python/Flask Development Environment:

- Each team has its own Gunicorn server running on port 910X - where X is your team #
- Your team's Gunicorn server starts up using the application file /srv/groups/team0X/python/wsgi.py
- Your Gunicorn instance is configured to automatically restart when files in srv/groups/team0X/python/ change
- If you need your team's instance of Gunicorn manually restarted, please contact SunshineSpend
- It's been detected that Gunicorn is overriding Python's 'requests' library. So, you will receive an error if your Python scripts try to
- import requests

  and use requests' methods. The sample scripts show a way around this problem using urllib.requests - but there are other library work arounds too. There may be other Python libraries whose functionality is overridden as well.

- If you would like additional Python libraries added to your team's environment - contact SunshineSpend

https://teamX.sunshinespend.com/pyapp -- mapped to /srv/groups/team0X/python/wsgi.py

you should see: "Hello World! -- Flask is running. POST /ask with {'question':'...'}""

https://teamX.sunshinespend.com/bot -- mapped to /srv/groups/team0X/python/wsgi.py (POST / ask method)

you should see: "Method Not Allowed" (has to be an 'http post', not a 'get')

Instead, to test, invoke this URL using curl ('http post' with basic auth & JSON):

```
curl -u team0X:YOUR_TEAM'S_BASIC_AUTH_PASSWORD -X POST
https://teamX.sunshinespend.com/bot -H "Content-Type: application/json" -d '{"question":
"Find Dr. Abbott"}'
```

Note the -X above is a command line parameter indicator, not your team #

you should should see (something like):

```
{"answer":"Dr. Lisa Abbott (NPI 1609883701) is a Family Medicine physician located in Las Vegas,
NV.","used_context":true}
```

Your team's Python/Flask/GUnicorn logs can be found in:

/srv/groups/team0X/logs/ - where X is your team #

NODE.JS - Your Node Development Environment:

- Each team has its own Node.js server running on port 900X - where X is your team #
- Your team's Node.js server starts up using the configuration / application file /srv/groups/team0X/node/server.js
- Your Node.js instance is configured to automatically restart when files in srv/groups/team0X/node/ change
- If you need your team's instance of Node.js manually restarted, please contact SunshineSpend
- Puppeteer (a Node.js library) is installed and your Node scripts can use it to do things like log into remote web servers.
- If you would like additional Node.js libraries added to your team's environment - contact SunshineSpend

https://teamX.sunshinespend.com/nodeapp - where X is your team #

you should see: "hello from node"

PHP and OTHER WEB DOCUMENTS - Your PHP and Apache Development Environment:

==================================

.If you would like additional PHP libraries added to your team's environment - contact SunshineSpend

https://teamX.sunshinespend.com/ (defaults to index.php, which is in /srv/groups/team0X/web/)

you should see: php system-related info

https://teamX.sunshinespend.com/db_test.php - where X is your team #

you should see: "DB status: ok"

Your team's Apache logs can be found in:

/srv/groups/team0X/logs/ - where X is your team #

PHP (PDO) example — /srv/groups/teamXX/web/db_test.php

```
~~<?php~~

~~$dsn="mysql:host=localhost;dbname=teamXXdb;charset=utf8mb4";~~

~~try {~~

~~  $pdo=new PDO($dsn,"teamXXu","REPLACE_ME",[PDO::ATTR_ERRMODE=>PDO::ERRMODE_EXCEPTION]);~~

~~  $row=$pdo->query("SELECT 'ok' AS status")->fetch(PDO::FETCH_ASSOC);~~

~~  echo "DB status: ".htmlspecialchars($row['status']);~~
~~} catch(Throwable $e){ http_response_code(500); echo "DB error: ".htmlspecialchars($e->getMessage()); }~~
```

Flask (PyMySQL) snippet — inside wsgi.py

```
~~import pymysql~~

~~def db_ok():~~

~~    conn=pymysql.connect(host='localhost',user='teamXXu',password='REPLACE_ME',~~
~~                         database='teamXXdb',charset='utf8mb4')~~
~~    with conn, conn.cursor() as cur:~~
~~        cur.execute("SELECT 'ok'"); return cur.fetchone()[0]~~
```

Node (mysql2) route

```
~~# one time in your node folder:~~
~~# npm init -y && npm install mysql2~~


~~# add this to server.js~~
~~const mysql=require('mysql2/promise'); const url=require('url');~~
~~const http=require('http'); const port=Number(process.env.PORT||910X);~~
~~const server=http.createServer(async (req,res)=>{~~
~~  const u=url.parse(req.url,true);~~
~~  if(u.pathname==='/dbcheck'){~~
~~    try{~~
~~      const conn=await mysql.createConnection({host:'localhost',user:'teamXXu',password:'REPLACE_ME',database:'teamXXdb'});~~
~~      const [rows]=await conn.query("SELECT 'ok' AS status");~~
~~      res.writeHead(200,{'Content-Type':'application/json'});~~
~~      return res.end(JSON.stringify(rows[0]));~~
~~    }catch(e){ res.writeHead(500); return res.end('DB error: '+e.message); }~~
~~  }~~
~~  res.end('hello from node\n');~~

~~});~~

~~server.listen(port,'127.0.0.1');~~
```

Flask Endpoints & Chatbot

~~Your Flask app should expose:~~

~~• GET /pyapp → health text~~

~~• POST /pyapp/ask → JSON {question: "..."} returns {answer, used_context}~~

~~Apache adds a friendly alias: POST /bot → forwards to /pyapp/ask~~

Minimal wsgi.py skeleton with DB + OpenAI proxy call

```
~~from flask import Flask, request, jsonify~~

~~from chatgpt_client import ask_chatgpt~~

~~from remote_api import lookup_hcp~~

~~from db import find_physicians_by_last~~

~~##########################################################################
######~~

~~# There's a known issue with Gunicorn overriding Python's 'requests'
library #~~

~~##########################################################################
######~~


~~app = Flask(__name__)~~


~~@app.get("/")~~

~~def hello():~~

~~    return "Hello World! -- Flask is running. POST /ask with
{'question':'...'}"~~


~~@app.post("/ask")~~

~~def ask():~~
```

```
    ~~data = request.get_json(silent=True) or {}~~
    ~~q = data.get("question")~~
    ~~if not q:~~
        ~~return jsonify(error='Send JSON {"question":"..."}'), 400~~

    ~~# Optional: naive last-name extraction (e.g., 'Find Dr. Smith')~~
    ~~import re~~
    ~~last = None~~
    ~~m = re.search(r"(?:dr\.?\s+)?([A-Z][a-zA-Z]+)$", q.strip())~~
    ~~if m: last = m.group(1)~~

    ~~print(">>> last:", last, flush=True)~~

    ~~# Remote API~~
    ~~remote = lookup_hcp(last)~~
    ~~remote_snippet = str(remote)[:300]~~
    ~~print(">>> remote_snippet:", remote_snippet, flush=True)~~

    ~~# DB lookup~~
    ~~db_snippet = ""~~
    ~~rows = find_physicians_by_last(last, limit=5) if last else []~~
    ~~if rows:~~
        ~~lines = [f"{r['first_name']} {r['last_name']} (NPI {r['npi']}) –~~
~~{r['specialty']} – {r['city']}, {r['state']}" for r in rows]~~
        ~~db_snippet = "Possible matches:\\n" + "\\n".join(lines)~~
    ~~print(">>> db_snippet:", db_snippet, flush=True)~~

    ~~# ChatGPT~~
    ~~messages = [~~
~~        {"role":"system","content":"You are a helpful SunshineSpend bot. Be concise. Admit uncertainty."},~~

~~        {"role":"user","content": f"Q: {q}\\n\\nContext (API): {remote_snippet}\\n\\nContext (DB): {db_snippet}"}~~

~~    ]~~

~~    answer = ask_chatgpt(messages)~~

~~    return jsonify(answer=answer, used_context=bool(db_snippet or remote_snippet))~~
```

Testing

```
~~# Basic Auth protected — replace PASS with your team password~~

~~curl -s -u teamXX:PASS https://teamX.sunshinespend.com/pyapp~~


~~# Chatbot call~~

~~curl -s -u teamXX:PASS -X POST https://teamX.sunshinespend.com/bot \~~

~~  -H 'Content-Type: application/json' -d '{"question":"hello"}'~~


~~# Node route~~

~~curl -s -u teamXX:PASS https://teamX.sunshinespend.com/nodeapp/dbcheck~~
```

Where to Keep Secrets

• PHP: place secrets in a file outside web/ (e.g., /srv/groups/teamXX/.env.php), chmod 600, and include it from PHP.

• Flask: use environment variables set by the admin (DB_* and OPENAI_PROXY_URL) or read your own .env safely.

• Node: read from process.env and/or a local config file not committed to Git.

Troubleshooting

• 401 Unauthorized → wrong Basic Auth. Ask the admin to reset your team password if needed.

• 403 Forbidden → permission issue; ensure files are under your team folder and readable by Apache (web/).

• 503 Service Unavailable → backend not listening or code error.

```
~~# Logs~~

~~# Apache per-team:~~

~~tail -n 80 /srv/groups/teamXX/logs/apache-error.log~~


~~# Flask:~~

~~journalctl -u gunicorn-teamXX --no-pager | tail -n 100~~


~~# Node:~~

~~journalctl -u node-teamXX --no-pager | tail -n 100~~
```