boiVisualiser Setup Guide

Hardware Inventory Visualisation Tool

Version 1.0

May 22, 2025

Ahmed Al-Alousi

https://github.com/ahmedalalousi/boiVisualiser

Contents

1 Introduction **3**

2 Quick Setup (Automated) **3**

2.1 Setup Script Options 3

2.2 Platform-Specific Setup Procedures 4

2.2.1 macOS Configuration 4

2.2.2 Linux Configuration 4

2.2.3 Universal Setup Steps 4

3 Manual Setup **4**

3.1 Prerequisites 4

3.1.1 pyenv Installation 5

3.1.2 Python Version Installation 5

3.2 Installation Steps 5

3.2.1 Repository Cloning 5

3.2.2 Virtual Environment Creation 5

3.2.3 Dependency Installation 5

4 Usage Instructions **6**

4.1 Environment Activation 6

4.2 PlantUML to JSON Conversion 6

4.3 Visualisation Generation 6

4.4 Command-Line Options 6

5 Troubleshooting **6**

5.1 pyenv Not Found After Installation 7

5.2 Permission Errors on macOS 7

5.3 Virtual Environment Activation Issues 7

5.4 Dependency Installation Failures 7

6 Project Structure **7**

6.1 Core Components 8

6.2 Data Flow 8

7 Conclusion **8**

1 Introduction

The boiVisualiser is a comprehensive hardware inventory visualisation tool designed to parse PlantUML hardware diagrams and convert them into interactive HTML visualisations. This guide provides detailed instructions for setting up the development environment and using the automated setup script.

The tool consists of several Python scripts that work together to process hardware inventory data from PlantUML files or CSV sources, converting them into JSON format and generating interactive web-based visualisations using D3.js.

2 Quick Setup (Automated)

The easiest method to begin using boiVisualiser is through the automated setup script. This approach handles all dependencies, environment configuration, and installation procedures automatically.

To initiate the automated setup process, execute the following commands in your terminal:

```
# Download and run the setup script
curl -O
   https://raw.githubusercontent.com/ahmedalalousi/boiVisualiser/main/setup.py
python3 setup.py
```

The automated setup script performs the following operations sequentially:

1. Verifies pyenv installation (installs if required)
2. Configures Python 3.13.3 via pyenv
3. Creates a dedicated virtual environment
4. Clones the repository from GitHub
5. Installs all necessary dependencies

2.1 Setup Script Options

The setup script accepts several command-line arguments to customise the installation process:

```
python3 setup.py [options]

Options:
  --repo-dir DIRECTORY    Directory to clone the repository
                         (default: ./boiVisualiser)
  --venv-name NAME       Name for virtual environment
                         (default: boiVisualiser-env)
  --skip-clone           Skip cloning if repository already exists
     locally
```

2.2 Platform-Specific Setup Procedures

The setup script adapts its behaviour based on the detected operating system, ensuring optimal configuration for each platform.

2.2.1 macOS Configuration

For macOS users, the setup script performs the following additional steps:

- Verifies Homebrew installation and installs if not present
- Installs pyenv and pyenv-virtualenv via Homebrew package manager
- Automatically configures shell environment for pyenv integration
- Handles Apple Silicon (M1/M2) specific PATH configurations

2.2.2 Linux Configuration

For Linux distributions, the setup process includes:

- Installation of pyenv using the official installer script
- Automatic shell configuration for bash and zsh environments
- Dependency verification and installation prompts

2.2.3 Universal Setup Steps

Regardless of the operating system, the following operations are performed:

- Installation of Python 3.13.3 via pyenv
- Creation of an isolated virtual environment
- Repository cloning from the official GitHub source
- Installation of all project dependencies from requirements.txt

3 Manual Setup

For users who prefer manual installation or require customised setup procedures, the following steps provide a comprehensive manual installation guide.

3.1 Prerequisites

Before beginning the manual setup process, ensure the following prerequisites are installed and properly configured:

3.1.1 pyenv Installation

pyenv serves as the Python version manager and is essential for maintaining consistent Python environments across different systems.

For macOS systems using Homebrew:

```
brew install pyenv pyenv-virtualenv
```

For Linux systems:

```
curl https://pyenv.run | bash
```

3.1.2 Python Version Installation

Install the specific Python version required by the project:

```
pyenv install 3.13.3
```

3.2 Installation Steps

Follow these sequential steps to complete the manual installation:

3.2.1 Repository Cloning

Clone the project repository to your local development environment:

```
git clone https://github.com/ahmedalalousi/boiVisualiser.git
cd boiVisualiser
```

3.2.2 Virtual Environment Creation

Create and activate a dedicated virtual environment for the project:

```
pyenv virtualenv 3.13.3 boiVisualiser-env
pyenv activate boiVisualiser-env
```

3.2.3 Dependency Installation

Install all required Python packages using pip:

```
pip install -r requirements.txt
```

4 Usage Instructions

After completing the setup process, the boiVisualiser tools are ready for use. Begin by activating the virtual environment and utilising the available command-line tools.

4.1 Environment Activation

Always activate the virtual environment before using the tools:

```
pyenv activate boiVisualiser-env
```

4.2 PlantUML to JSON Conversion

Convert PlantUML hardware diagrams to JSON format for processing:

```
python puml_to_json.py --input hardware.puml --output data.json
```

4.3 Visualisation Generation

Generate interactive HTML visualisations from PlantUML files:

```
python visualise_hardware.py --input hardware.puml --output
   visualization.html --open-browser
```

The `--open-browser` flag automatically opens the generated visualisation in your default web browser upon completion.

4.4 Command-Line Options

Both tools provide comprehensive command-line options for customising the processing workflow:

- `--input`: Specifies the input file or directory path
- `--output`: Defines the output file location
- `--input-type`: Explicitly sets the input format (puml or csv)
- `--temp-dir`: Specifies temporary directory for intermediate files
- `--open-browser`: Automatically opens output in web browser

5 Troubleshooting

This section addresses common issues encountered during setup and usage of the boiVisualiser tools.

5.1 pyenv Not Found After Installation

If pyenv commands are not recognised after installation, restart your terminal session or manually reload your shell configuration:

For zsh users:

```
source ~/.zshrc
```

For bash users:

```
source ~/.bashrc
```

5.2 Permission Errors on macOS

macOS users may encounter permission prompts during installation. Accept any requests to install Xcode command line tools when prompted, as these are required for compiling Python extensions.

5.3 Virtual Environment Activation Issues

If virtual environment activation fails, verify that pyenv is properly configured in your shell environment. Manually add the following configuration to your shell profile:

```
echo 'export PYENV_ROOT="$HOME/.pyenv"' >> ~/.zshrc
echo 'export PATH="$PYENV_ROOT/bin:$PATH"' >> ~/.zshrc
echo 'eval "$(pyenv init -)"' >> ~/.zshrc
echo 'eval "$(pyenv virtualenv-init -)"' >> ~/.zshrc
```

Replace `.zshrc` with `.bashrc` if using bash as your default shell.

5.4 Dependency Installation Failures

If package installation fails, ensure you are operating within the activated virtual environment and have the latest pip version:

```
pyenv activate boiVisualiser-env
pip install --upgrade pip
pip install -r requirements.txt
```

6 Project Structure

Understanding the project structure facilitates effective usage and potential contributions to the codebase.

6.1 Core Components

The boiVisualiser consists of several key components:

- **puml_to_json.py**: Parses PlantUML files and converts them to JSON format
- **visualise_hardware2.py**: Generates interactive HTML visualisations
- **csv2PlantUML.py**: Converts CSV inventory data to PlantUML format
- **hardware_inventory_visualisation.html**: Template for HTML output
- **setup.py**: Automated setup and configuration script
- **requirements.txt**: Python dependency specifications

6.2 Data Flow

The typical data processing workflow follows this sequence:

1. CSV files or PlantUML diagrams serve as input sources
2. Data is parsed and converted to standardised JSON format
3. JSON data is embedded into HTML template
4. Interactive visualisation is generated using D3.js
5. Output is displayed in web browser for analysis

7 Conclusion

The boiVisualiser provides a comprehensive solution for hardware inventory visualisation, combining the power of PlantUML diagramming with interactive web-based presentations. The automated setup script ensures rapid deployment across different platforms, whilst the manual installation options provide flexibility for customised environments.

For additional support, feature requests, or bug reports, please visit the project repository at `https://github.com/ahmedalalousi/boiVisualiser` or submit issues through the GitHub interface.