@serve.zone/isocreator

a tool to create isos for managed edge devices

- readme.md for @serve.zone/isocreator
- changelog.md for @serve.zone/isocreator

readme.md for @serve.zone/isocreator

> Ubuntu ISO customization tool for PC and Raspberry Pi with WiFi and cloud-init configuration

License: MIT

Overview

isocreator is a command-line tool that creates customized Ubuntu Server ISOs with pre-configured:

- □□ **WiFi credentials** (via cloud-init)
- □□ **User accounts** and SSH keys
- □□ **Pre-installed packages**
- □□ **Custom boot scripts**
- ⚙ **Full cloud-init configuration**

Perfect for:

- Raspberry Pi deployments
- Headless server installations
- Automated fleet provisioning
- Development environments

Features

- **Multi-Platform Support**: PC (x86_64) and Raspberry Pi (ARM64)
- **Multi-Version Support**: Ubuntu 22.04 LTS, 24.04 LTS, and future versions
- **Cloud-Init Integration**: Full cloud-init user-data and network-config support
- **Caching System**: Intelligent ISO caching with multi-version support
- **Flexible Configuration**: YAML files, CLI flags, or interactive mode

- **USB Bootable**: Creates ISOs that can be written directly to USB drives

Installation

via npm

```
npm install -g @serve.zone/isocreator
```

via Direct Script

```
curl -sSL https://code.foss.global/serve.zone/isocreator/raw/branch/main/install.sh | sudo bash
```

System Dependencies

isocreator requires the following tools to be installed:

Ubuntu/Debian:

```
sudo apt install xorriso syslinux-utils
```

macOS:

```
brew install xorriso syslinux
```

Quick Start

Interactive Mode

```
isocreator build
```

Using a Config File

```
# Generate a template
isocreator template create --output myconfig.yaml

# Edit the config file
nano myconfig.yaml

# Build the ISO
isocreator build --config myconfig.yaml
```

Using CLI Flags

```
isocreator build \
  --ubuntu-version 24.04 \
  --arch amd64 \
  --wifi-ssid "MyWiFi" \
  --wifi-password "secret123" \
  --ssh-key ~/.ssh/id_rsa.pub \
  --hostname "myserver" \
  --output ./custom-ubuntu.iso
```

Configuration

Example Config File

```
version: "1.0"

# Base ISO settings
iso:
  ubuntu_version: "24.04"
  architecture: "amd64"  # or arm64 for Raspberry Pi

# Output settings
output:
  filename: "ubuntu-custom.iso"
  path: "./output"
```

```
# WiFi configuration (via cloud-init)
network:
  wifi:
    ssid: "MyWiFi"
    password: "secret123"

# Cloud-init configuration
cloud_init:
  hostname: "myserver"

  # User accounts
  users:
    - name: "admin"
      ssh_authorized_keys:
        - "ssh-rsa AAAAB3NzaC1yc2E..."
      sudo: "ALL=(ALL) NOPASSWD:ALL"
      shell: "/bin/bash"

  # Packages to install on first boot
  packages:
    - docker.io
    - git
    - htop

  # Commands to run on first boot
  runcmd:
    - systemctl enable docker
    - systemctl start docker
    - echo "Setup complete!"

# Custom boot scripts
boot_scripts:
  - name: "setup-docker"
    path: "./scripts/setup-docker.sh"
```

Commands

Build

Build a customized ISO:

```
isocreator build [OPTIONS]
```

Options:

- `--config <file>` - Use a YAML config file
- `--ubuntu-version <version>` - Ubuntu version (22.04, 24.04, etc.)
- `--arch <arch>` - Architecture (amd64 or arm64)
- `--wifi-ssid <ssid>` - WiFi SSID
- `--wifi-password <password>` - WiFi password
- `--ssh-key <path>` - Path to SSH public key
- `--hostname <name>` - System hostname
- `--output <path>` - Output ISO path

Cache Management

List cached ISOs:

```
isocreator cache list
```

Download an ISO to cache:

```
isocreator cache download 24.04 --arch amd64
```

Clean old cached ISOs:

```
isocreator cache clean
isocreator cache clean --older-than 6m
```

Templates

Generate a config template:

```
isocreator template create
isocreator template create --output myconfig.yaml
```

Validation

Validate a config file:

```
isocreator validate myconfig.yaml
```

Use Cases

Raspberry Pi Home Server

```
iso:
  ubuntu_version: "24.04"
  architecture: "arm64"

network:
  wifi:
    ssid: "HomeNetwork"
    password: "homepass123"

cloud_init:
  hostname: "pi-server"
  users:
    - name: "pi"
      ssh_authorized_keys:
        - "ssh-rsa AAAAB3..."
      sudo: "ALL=(ALL) NOPASSWD:ALL"

  packages:
    - docker.io
    - docker-compose

  runcmd:
    - systemctl enable docker
```

Headless Development Server

```
iso:
  ubuntu_version: "24.04"
  architecture: "amd64"

cloud_init:
  hostname: "devbox"

  users:
    - name: "developer"
      ssh_authorized_keys:
        - "ssh-rsa AAAAB3..."
      sudo: "ALL=(ALL) NOPASSWD:ALL"

  packages:
    - build-essential
    - git
    - docker.io
    - nodejs
    - npm
```

Architecture

isocreator is built with:

- **Deno** - Modern TypeScript runtime
- **Cloud-init** - Industry-standard cloud instance initialization
- **xorriso** - ISO 9660 filesystem manipulation
- **Self-contained binaries** - Zero runtime dependencies

Development

Prerequisites

- Deno 1.40+
- xorriso
- syslinux-utils (for isohybrid)

Setup

```
git clone https://code.foss.global/serve.zone/isocreator.git
cd isocreator
deno task cache
```

Development Mode

```
deno task dev --help
```

Testing

```
deno task test
```

Build Binaries

```
deno task compile
```

Contributing

Contributions are welcome! Please feel free to submit issues and pull requests.

License

MIT License - see license file for details

Support

- **Issues**: https://code.foss.global/serve.zone/isocreator/issues
- **Documentation**: https://code.foss.global/serve.zone/isocreator

Made with ❤ by Lossless GmbH

changelog.md for @serve.zone/isocreator

2025-10-24 - 1.2.3 - fix(deps)

Update Deno dependencies and configuration, commit lockfile, and adjust plugins exports

- Add deno.lock with pinned std library versions for reproducible Deno builds
- Remove deno.lock from .gitignore so the lockfile is committed
- Remove unused @std/fmt export from ts/plugins.ts and from deno.json imports
- Adjust deno.json compiler options (remove allowJs)
- Add local .claude/settings.local.json (local permissions/config)

2025-10-24 - 1.2.2 - fix(logging)

Refactor logging and plugin imports: remove push.rocks dependencies and provide a simple console-based logger with a compatibility shim

- Replaced smartlog-based logger with a simple console-based log utility in ts/logging.ts; added a small logger shim (logger.log) for compatibility.
- Removed push.rocks exports from ts/plugins.ts and consolidated external imports to std modules (@std/*); updated deno.json to remove push.rocks packages and keep @std/cli.
- Updated code to use the new console logger and simplified plugin surface; this reduces external dependencies and simplifies runtime footprint.
- Added .claude/settings.local.json (local settings file) — development/local config only, no runtime change.

2025-10-24 - 1.2.1 - fix(deno)

Set Deno nodeModulesDir to 'auto' and add local .claude settings

- Add nodeModulesDir: "auto" to deno.json to enable automatic node_modules handling for compatibility with npm-scoped imports.

- Add .claude/settings.local.json with local tooling permissions for development environments (non-functional, local config only).
- This is a tooling/config change only — no runtime code paths were modified.

2025-10-24 - 1.2.0 - feat(ci)

Improve release workflow: compile multi-platform binaries, generate checksums, create Gitea release and upload assets, and add local settings

- Rename workflow job to build-and-release and set up Deno v2
- Verify deno.json version matches the Git tag before building
- Compile pre-built binaries for Linux (x64, arm64), macOS (x64, arm64) and Windows (x64)
- Generate SHA256 checksums (SHA256SUMS.txt) for all compiled binaries
- Extract or generate release notes and create a Gitea release via API
- Upload compiled binaries and checksums as release assets
- Delete existing release if present and clean up old releases (keep last 3)
- Add local .claude/settings.local.json for developer environment permissions

2025-10-24 - 1.1.0 - feat(core)

Initial project scaffold and implementation: Deno CLI, ISO tooling, cloud-init generation, packaging and installer scripts

- Add Deno-based CLI (commands: build, cache, template, validate) and entry point (mod.ts)
- Implement ISO management classes: iso-downloader, iso-cache, iso-extractor, iso-packer, iso-builder
- Add cloud-init generator and configuration manager with YAML template and validation
- Add packaging/distribution tooling: npm wrapper, postinstall installer, binary wrapper, compile script, install/uninstall scripts
- Add CI release workflow, changelog, README, license, and configuration files (deno.json, package.json, templates)

All notable changes to isocreator will be documented in this file.

The format is based on Keep a Changelog, and this project adheres to Semantic Versioning.

[Unreleased]

Added

- Initial project structure
- Deno-based CLI framework
- Binary compilation for multiple platforms
- npm distribution with postinstall binary download
- Direct installation script

[1.0.0] - TBD

Added

- Ubuntu ISO customization for PC (x86_64) and Raspberry Pi (ARM64)
- Cloud-init configuration support
- WiFi pre-configuration via cloud-init network-config
- User account and SSH key injection
- Package installation via cloud-init
- Custom boot script support
- ISO caching system with multi-version support
- Interactive mode for guided setup
- Config file support (YAML)
- CLI flag support for quick customization
- Template generation for configuration files