🍓 RuntipiOS - Guide Complet d'Installation

Vue d'Ensemble du Projet

RuntipiOS est un système d'exploitation Raspberry Pi OS Lite personnalisé qui installe automatiquement Runtipi avec une configuration WiFi simplifiée via smartphone, utilisant Balena WiFi-Connect comme portail captif.

Caractéristiques Principales

- ✅ Installation automatique de Runtipi au premier démarrage
- 📱 Configuration WiFi sans écran via portail captif
- 🌐 Page de statut web avec logo Runtipi
- ⚙️ Configuration centralisée via `config.yml`
- 🔄 Build automatisé via GitHub Actions
- 🐳 Docker et Docker Compose pré-installés

Structure Complète des Fichiers

Voici la structure complète à créer dans votre repository GitHub :

```
runtipios/
├── .github/
│   └── workflows/
│       └── build-release.yml     # Workflow GitHub Actions
├── scripts/
│   ├── build-image.sh            # Script principal de build
│   ├── customize-os.sh           # Customisation du système
│   ├── install-wifi-connect.sh   # Installation WiFi-Connect
│   ├── install-runtipi.sh        # Installation Runtipi
│   └── setup-services.sh         # Configuration services systemd
├── config.yml                    # Configuration principale (ÉDITABLE)
├── Dockerfile                    # Image Docker pour le builder
├── README.md                     # Documentation
├── LICENSE                       # Licence MIT
└── .gitignore                    # Fichiers à ignorer
```

📄 Fichier 1: `config.yml` (Configuration Principale)

Ce fichier contient **toutes les variables configurables** du projet. Vous pouvez le modifier avant de lancer le build.

Variables importantes:

- `raspios.version`: Version de Raspberry Pi OS (format YYYY-MM-DD)
- `raspios.url`: URL de téléchargement de l'image de base
- `runtipi.version`: Version de Runtipi à installer (ex: v3.8.0)
- `wifi_connect.version`: Version de WiFi-Connect (ex: 4.4.7)
- `wifi_connect.ssid`: Nom du réseau WiFi de configuration
- `system.hostname`: Nom d'hôte du système
- `system.default_user`: Nom d'utilisateur par défaut
- `system.default_password`: Mot de passe par défaut
- `build.image_size`: Taille de l'image en Go
- `build.compress`: Activer la compression (true/false)
- `build.compression_format`: Format (xz, gz, zip)

🐳 Fichier 2: `Dockerfile`

Crée l'image Docker du builder qui contiendra tous les outils nécessaires pour générer l'image Raspberry Pi.

Contient:

- Debian Bookworm comme base
- Outils de build (debootstrap, qemu, parted, kpartx)
- Outils de compression (xz, gzip, zip)
- Python 3 pour parser le YAML
- Structure de répertoires (/build/scripts, /build/output)

🔧 Fichier 3: `scripts/build-image.sh`

Script principal qui orchestre toute la création de l'image.

Étapes:

1. Parse le fichier config.yml
2. Télécharge l'image Raspberry Pi OS de base
3. Extrait et agrandit l'image
4. Monte l'image avec des loop devices
5. Configure le système (chroot)
6. Installe WiFi-Connect et configure Runtipi
7. Démonte et compresse l'image finale

Nécessite: Mode privilégié (`--privileged`) pour manipuler les loop devices

🔧 Fichier 4: `scripts/customize-os.sh`

Script exécuté dans le chroot pour personnaliser le système.

Actions:

- Configure hostname, timezone, locale
- Installe Docker et Docker Compose
- Crée l'utilisateur par défaut
- Active SSH et Avahi (mDNS)
- Installe les packages additionnels
- Crée la page de statut web avec logo Runtipi
- Configure le MOTD (Message du jour)

🔧 Fichier 5: `scripts/install-wifi-connect.sh`

Installe et configure Balena WiFi-Connect.

Actions:

- Télécharge le binaire WiFi-Connect pour ARM64
- Crée le script de vérification de connectivité
- Configure le service systemd
- Détecte automatiquement Ethernet vs WiFi

Comportement:

- Si Ethernet connecté → Pas de portail captif
- Si WiFi configuré → Pas de portail captif
- Sinon → Lance le portail captif "RuntipiOS-Setup"

🔧 Fichier 6: `scripts/install-runtipi.sh`

Script d'installation de Runtipi, exécuté au premier démarrage.

Actions:

- Vérifie la connexion Internet (30 tentatives)
- Clone le repository Runtipi
- Checkout de la version spécifiée
- Lance l'installation officielle
- Crée un fichier JSON avec les infos d'installation
- Désactive le service après installation réussie

Statuts:

- `downloading`: Téléchargement en cours
- `installing`: Installation en cours
- `configuring`: Configuration en cours
- `completed`: Installation réussie
- `failed`: Erreur

⚙ Fichier 7: `scripts/setup-services.sh`

Configure les services systemd nécessaires.

Services créés:

`runtipi-installer.service`

- Type: oneshot
- Démarre après: network-online.target, wifi-connect.service
- Condition: Fichier /etc/runtipi-configured n'existe pas
- Action: Lance install-runtipi.sh
- Timeout: 30 minutes

`wifi-connect.service`

- Type: simple
- Démarre après: NetworkManager.service
- Action: Lance wifi-connect-check.sh
- Restart: on-failure

Fichier 8: `.github/workflows/build-release.yml`

Workflow GitHub Actions pour automatiser le build et la release.

3 Jobs:

1. `build-image`

- Extrait la version du tag ou génère une version dev
- Valide config.yml
- Nettoie l'espace disque
- Configure QEMU pour ARM
- Build l'image Docker builder
- Push vers GitHub Container Registry

2. `build-rpi-image`

- Pull l'image builder
- Lance le build de l'image Raspberry Pi
- Vérifie que l'image .img est créée
- Génère les checksums SHA256
- Upload comme artifact GitHub

3. `create-release`

- Télécharge l'artifact
- Vérifie les checksums
- Génère les release notes automatiquement
- Crée une release GitHub avec l'image

Déclenchement:

- Push d'un tag `v*` (ex: v1.0.0)
- Push sur la branche main
- Déclenchement manuel (workflow_dispatch)

Utilisation du Système

Étape 1: Setup GitHub

1. Créez un nouveau repository GitHub
2. Copiez tous les fichiers dans le repository
3. Modifiez `config.yml` selon vos besoins
4. Commitez et poussez

Étape 2: Lancer le Build

Option A: Via Tag

```
git tag v1.0.0
git push origin v1.0.0
```

Option B: Manuel

- GitHub → Actions → Build and Release RuntipiOS → Run workflow

Étape 3: Récupérer l'Image

1. Attendez 30-45 minutes (durée du build)
2. Releases → Téléchargez `RuntipiOS-*.img.xz`
3. Vérifiez le checksum avec `SHA256SUMS`

Étape 4: Flasher l'Image

1. Utilisez Raspberry Pi Imager ou Etcher
2. Sélectionnez l'image téléchargée
3. Sélectionnez la carte microSD (min 8 Go)
4. Flashez

Étape 5: Premier Démarrage

1. Insérez la carte SD dans le Raspberry Pi
2. Branchez l'alimentation
3. Attendez 2-3 minutes

Si Ethernet branché: Système directement accessible

Si WiFi uniquement:

1. Cherchez le réseau "RuntipiOS-Setup" sur votre smartphone
2. Connectez-vous
3. Le portail captif s'ouvre automatiquement
4. Sélectionnez votre réseau WiFi
5. Entrez le mot de passe
6. Validez

Étape 6: Installation de Runtipi

L'installation de Runtipi démarre automatiquement (10-15 minutes).

Suivre la progression:

- Page de statut: `http://runtipios.local:8080`
- Via SSH: `sudo tail -f /var/log/runtipi-install.log`

Étape 7: Accéder à Runtipi

Une fois l'installation terminée:

- URL mDNS: `http://runtipios.local`
- URL IP: `http://&lt;IP-du-Pi&gt;`

Identifiants par défaut:

- User: `runtipi`
- Password: `runtipi`

⚠ **Changez immédiatement le mot de passe !**

Personnalisations Courantes

Changer le SSID du portail WiFi

Dans `config.yml`:

```
wifi_connect:
  ssid: "MonServeur-Config"
  password: "monpassword"
```

Ajouter un mot de passe au portail

Dans `config.yml`:

```
wifi_connect:
  password: "secureme123"
```

Changer l'utilisateur et le mot de passe

Dans `config.yml`:

```
system:
  default_user: "admin"
  default_password: "VotreMotDePasse123!"
```

Utiliser une version spécifique de Runtipi

Dans `config.yml`:

```
runtipi:
  version: "v3.7.0"
```

Changer la taille de l'image

Dans `config.yml`:

```
build:
  image_size: 16  # 16 Go au lieu de 8
```

Ajouter des packages

Dans `config.yml`:

```
packages:
  install:
    - neofetch
    - tmux
    - screen
```

Modifier la page de statut web

Éditez le HTML dans `scripts/customize-os.sh` (section "Page de statut web").

Pour ajouter le logo Runtipi:

```
<div>
    <img>
</div>
```

Dépannage

Le build GitHub Actions échoue

Erreur "No space left on device":

- Le workflow nettoie déjà l'espace disque
- Réduisez `build.image_size` dans config.yml

Erreur "Loop device not available":

- Le flag `--privileged` est nécessaire
- Déjà configuré dans le workflow

Le portail WiFi ne s'ouvre pas

1. Vérifiez la connexion au réseau "RuntipiOS-Setup"
2. Désactivez les données mobiles sur Android
3. Ouvrez manuellement: `http://192.168.4.1`

Runtipi ne s'installe pas

Vérifier les logs:

```
ssh runtipi@runtipios.local
sudo journalctl -u runtipi-installer.service -f
```

Relancer l'installation:

```
sudo systemctl start runtipi-installer.service
```

Impossible d'accéder via .local

Linux/Mac: Devrait fonctionner nativement

Windows: Installez Bonjour Print Services

Alternative: Utilisez l'adresse IP directement:

```
ssh runtipi@runtipios.local
ip addr show
```

Monitoring et Logs

Voir le statut des services

```
# WiFi-Connect
sudo systemctl status wifi-connect

# Runtipi installer
sudo systemctl status runtipi-installer

# Docker
sudo systemctl status docker

# Avahi (mDNS)
sudo systemctl status avahi-daemon
```

Logs en temps réel

```
# Installation Runtipi
sudo tail -f /var/log/runtipi-install.log

# Service WiFi-Connect
sudo journalctl -u wifi-connect -f

# Tous les services
sudo journalctl -f
```

Vérifier la connectivité

```
# Interfaces réseau
ip addr show

# Connexions WiFi
nmcli device status

# Test Internet
ping -c 3 google.com
```

Sécurité

Changer le mot de passe par défaut

```
ssh runtipi@runtipios.local
passwd
```

Désactiver l'authentification par mot de passe SSH

```
# 1. Copier votre clé publique
ssh-copy-id runtipi@runtipios.local

# 2. Désactiver le password auth
sudo nano /etc/ssh/sshd_config
# Modifier: PasswordAuthentication no

# 3. Redémarrer SSH
sudo systemctl restart ssh
```

Configurer le pare-feu

```
# Installer ufw
sudo apt install ufw

# Autoriser SSH
sudo ufw allow 22/tcp

# Autoriser Runtipi
sudo ufw allow 80/tcp
sudo ufw allow 443/tcp

# Activer
sudo ufw enable
```

Optimisations

Réduire la taille de l'image

1. Dans `config.yml`, réduire `packages.install`
2. Ajouter plus de packages dans `packages.remove`
3. Dans `customize-os.sh`, ajouter:

```
apt-get clean
rm -rf /var/cache/apt/archives/*.deb
rm -rf /var/log/*.log
```

Accélérer le build

1. Utiliser un runner GitHub auto-hébergé
2. Activer le cache Docker (déjà fait)
3. Pré-télécharger l'image Raspberry Pi OS

Performance du Raspberry Pi

1. Overclocker (avec précaution)
2. Utiliser une carte SD rapide (UHS-I U3 ou mieux)
3. Refroidissement passif/actif recommandé

Ressources

Documentation

- Runtipi Documentation
- Balena WiFi-Connect
- Raspberry Pi OS
- GitHub Actions

Outils

- Raspberry Pi Imager
- Etcher
- Docker Desktop

Communauté

- Runtipi Discord
- Raspberry Pi Forums

✅ Checklist de Déploiement

Avant de créer votre première release:

- [ ] Repository GitHub créé
- [ ] Tous les fichiers copiés
- [ ] `config.yml` modifié selon vos besoins
- [ ] Mot de passe par défaut changé dans config.yml
- [ ] SSID WiFi personnalisé (optionnel)
- [ ] Taille d'image adaptée à votre carte SD
- [ ] Tests locaux effectués (optionnel)
- [ ] Tag git créé et poussé
- [ ] Workflow GitHub Actions lancé
- [ ] Release créée avec l'image
- [ ] Checksums vérifiés
- [ ] Image flashée et testée sur Raspberry Pi

🎉 Félicitations !

Vous avez maintenant un système RuntipiOS complet et personnalisable !

Prochaines étapes:

1. Partagez votre configuration dans les issues GitHub
2. Contribuez aux améliorations
3. Aidez d'autres utilisateurs dans les discussions

Bon déploiement ! 🚀