🧪 Guia Completo de Testes - Orange Pi Provisioning

Este guia fornece instruções detalhadas para testar o sistema de provisionamento em diferentes cenários e ambientes.

📋 Índice

🛠️ Pré-requisitos para Testes

Ambiente de Desenvolvimento

```
# Verificar versões necessárias
docker --version          # >= 20.10
docker compose version    # >= 2.0
shellcheck --version      # >= 0.7
jq --version              # >= 1.6
```

Hardware Necessário

- **MicroSD Cards**: Pelo menos 2 cartões de 8GB+ (para testes paralelos)
- **Orange Pi**: Zero 3 (2GB) e/ou Zero 2W (1GB) para testes reais
- **Leitor de MicroSD**: USB 3.0 recomendado para velocidade
- **Rede WiFi**: Configurada e acessível

Configuração Inicial

```
# Clonar repositório
git clone <repository-url>
cd orange-pi-provisioning

# Construir ambiente de teste
docker compose build

# Verificar estrutura
docker compose run --rm provisioner ls -la
```

🤖 Testes Automatizados

1. Validação de Scripts Shell

```
# Executar shellcheck em todos os scripts
find scripts/ -name "*.sh" -exec shellcheck {} \;

# Teste específico por script
shellcheck scripts/provision-manager.sh
shellcheck scripts/deploy-ender3.sh
shellcheck scripts/deploy-laser.sh
```

2. Validação de Configurações JSON

```
# Validar todos os arquivos JSON
find configs/ -name "*.json" -exec jq empty {} \;

# Teste específico
jq empty configs/projects-config.json
jq empty configs/state-persistence.json
```

3. Testes de Build Docker

```
# Teste de build básico
docker compose build

# Teste de build com cache limpo
docker compose build --no-cache

# Verificar imagem criada
docker images | grep orange-pi-provisioning
```

4. Testes de Funcionalidade Básica

```
# Teste de execução do container
docker compose run --rm provisioner echo "Container funcionando"

# Teste de acesso aos scripts
docker compose run --rm provisioner ls -la scripts/

# Teste de permissões
docker compose run --rm provisioner bash -c "
  for script in scripts/*.sh; do
    if [ -x \"\$script\" ]; then
      echo \"✅ \$script é executável\"
    else
      echo \"❌ \$script não é executável\"
    fi
  done
"
```

🖱️ Testes Manuais

1. Teste de Interface Principal

```
# Executar interface interativa
docker compose run --rm provisioner scripts/provision-manager.sh

# Verificar menu principal
# - Opção 1: Coletar informações locais
# - Opção 2: Deploy Ender3
# - Opção 3: Deploy Laser
# - Opção 4: Validar deployment
# - Opção 5: Sair
```

2. Teste de Coleta de Informações

```
# Executar coleta manual
docker compose run --rm provisioner scripts/collect-local-info.sh

# Verificar arquivos gerados
docker compose run --rm provisioner ls -la state/
docker compose run --rm provisioner cat state/local-info.json
```

3. Teste de Scripts Individuais

```
# Teste do script de deploy Ender3 (modo dry-run)
docker compose run --rm provisioner bash -c "
  export DRY_RUN=true
  scripts/deploy-ender3.sh
"

# Teste do script de deploy Laser (modo dry-run)
docker compose run --rm provisioner bash -c "
  export DRY_RUN=true
  scripts/deploy-laser.sh
"
```

4. Teste de Validação

```
# Teste de validação sem hardware
docker compose run --rm provisioner bash -c "
  export MOCK_MODE=true
  scripts/validate-deployment.sh ender3
"
```

🔧 Testes em Ambiente Real

1. Preparação do Ambiente

```
# Verificar dispositivos USB
lsblk -d -o NAME,SIZE,TRAN | grep usb

# Inserir MicroSD e identificar dispositivo
# Exemplo: /dev/sdb
```

2. Teste Completo - Projeto Ender3

```
# Passo 1: Coletar informações locais
docker compose run --rm provisioner scripts/collect-local-info.sh

# Passo 2: Executar deploy completo
docker compose run --rm --privileged provisioner scripts/deploy-ender3.sh

# Passo 3: Inserir MicroSD no Orange Pi e aguardar boot (5-10 min)

# Passo 4: Validar deployment
docker compose run --rm provisioner scripts/validate-deployment.sh ender3
```

3. Teste Completo - Projeto Laser

```
# Executar deploy completo
docker compose run --rm --privileged provisioner scripts/deploy-laser.sh

# Inserir MicroSD no Orange Pi e aguardar boot

# Validar deployment
docker compose run --rm provisioner scripts/validate-deployment.sh laser
```

4. Teste de Recuperação de Estado

```
# Simular interrupção durante deploy
# Ctrl+C durante execução

# Verificar estado persistente
docker compose run --rm provisioner cat state/ender3-deployment.json

# Retomar deployment
docker compose run --rm provisioner scripts/deploy-ender3.sh
```

🔄 Testes de Regressão

1. Teste de Compatibilidade

```
# Testar com diferentes versões de imagem Armbian
# Editar configs/projects-config.json
# Alterar "armbian_image_url" para versões diferentes

# Testar deploy
docker compose run --rm --privileged provisioner scripts/deploy-ender3.sh
```

2. Teste de Configurações Diferentes

```
# Backup da configuração original
cp configs/projects-config.json configs/projects-config.json.bak

# Testar com IPs diferentes
jq '.projects.ender3.network.static_ip = "192.168.1.150"' configs/projects-con-
fig.json > temp.json
mv temp.json configs/projects-config.json

# Executar teste
docker compose run --rm --privileged provisioner scripts/deploy-ender3.sh

# Restaurar configuração
mv configs/projects-config.json.bak configs/projects-config.json
```

3. Teste de Múltiplos Deployments

```
# Deploy sequencial dos dois projetos
docker compose run --rm --privileged provisioner scripts/deploy-ender3.sh
# Trocar MicroSD
docker compose run --rm --privileged provisioner scripts/deploy-laser.sh

# Validar ambos
docker compose run --rm provisioner scripts/validate-deployment.sh ender3
docker compose run --rm provisioner scripts/validate-deployment.sh laser
```

⚡ Testes de Performance

1. Tempo de Build

```
# Medir tempo de build
time docker compose build

# Medir tempo de build sem cache
time docker compose build --no-cache
```

2. Tempo de Deploy

```
# Medir tempo de deploy completo
time docker compose run --rm --privileged provisioner scripts/deploy-ender3.sh
```

3. Uso de Recursos

```
# Monitorar uso durante deploy
docker stats

# Em outro terminal, executar deploy
docker compose run --rm --privileged provisioner scripts/deploy-ender3.sh
```

4. Teste de Carga

```
# Executar múltiplos containers simultaneamente
docker compose run --rm provisioner scripts/collect-local-info.sh &
docker compose run --rm provisioner scripts/validate-deployment.sh ender3 &
wait
```

🐛 Resolução de Problemas

1. Problemas Comuns

Container não inicia

```
# Verificar logs
docker compose logs

# Reconstruir imagem
docker compose build --no-cache
```

Scripts não executam

```
# Verificar permissões
docker compose run --rm provisioner ls -la scripts/

# Corrigir permissões se necessário
chmod +x scripts/*.sh
```

MicroSD não detectado

```
# Verificar dispositivos
lsblk

# Executar com privilégios
docker compose run --rm --privileged provisioner scripts/deploy-ender3.sh
```

2. Logs e Debugging

```
# Habilitar modo debug
export DEBUG=true
docker compose run --rm provisioner scripts/deploy-ender3.sh

# Verificar logs detalhados
docker compose run --rm provisioner ls -la logs/
docker compose run --rm provisioner tail -f logs/deploy-ender3-*.log
```

3. Limpeza de Ambiente

```
# Limpar containers
docker compose down
docker system prune -f

# Limpar volumes
docker volume prune -f

# Limpar imagens
docker image prune -f
```

📊 Relatórios de Teste

1. Gerar Relatório de Validação

```
# Executar validação completa
docker compose run --rm provisioner scripts/validate-deployment.sh ender3

# Verificar relatório gerado
docker compose run --rm provisioner ls -la reports/
docker compose run --rm provisioner cat reports/validation_report_*.md
```

2. Exportar Logs

```
# Copiar logs para host
docker compose run --rm provisioner tar -czf /tmp/logs.tar.gz logs/
docker cp $(docker compose run --rm provisioner echo $HOSTNAME):/tmp/logs.tar.gz ./
logs.tar.gz
```

3. Métricas de Sucesso

- **Build Success Rate**: 100% dos builds devem ser bem-sucedidos
- **Deploy Success Rate**: 95%+ dos deploys devem ser bem-sucedidos
- **Validation Success Rate**: 90%+ das validações devem passar
- **Time to Deploy**: < 15 minutos para deploy completo
- **Time to Validate**: < 5 minutos para validação completa

🎯 Checklist de Testes

Antes de Cada Release

- [ ] Todos os testes automatizados passam
- [ ] Teste manual completo executado
- [ ] Teste em ambiente real com hardware
- [ ] Documentação atualizada
- [ ] Logs limpos e informativos
- [ ] Performance dentro dos limites aceitáveis
- [ ] Segurança validada
- [ ] Compatibilidade testada

Testes Críticos

- [ ] Deploy Ender3 funciona completamente
- [ ] Deploy Laser funciona completamente
- [ ] Validação SSH funciona
- [ ] Validação de serviços funciona
- [ ] Recuperação de estado funciona
- [ ] Interface interativa funciona
- [ ] Coleta de informações locais funciona

Nota: Este guia deve ser atualizado conforme novas funcionalidades são adicionadas ao sistema.