# Futura Toolkit

Padronização e automação do ambiente de desenvolvimento

# ❗ O problema que queremos resolver

- “Funciona na minha máquina”
- Versões diferentes de ferramentas entre devs
- Setup demorado para novos desenvolvedores e após formatação da máquina
- Atualizações quebrando o ambiente local
- Dependência de instalações manuais no sistema operacional

👉 Alto custo de tempo, retrabalho e inconsistência entre ambientes

# 💡 A premissa do Toolkit

**Regra principal:**

Não instalar nada além do Docker
(com exceção dos projetos onde o desenvolvedor vai editar código)

- Infraestrutura, dependências e serviços rodam em containers
- O projeto em desenvolvimento pode rodar fora do container, se necessário
- O Toolkit não impõe como o dev executa o código da aplicação
- Ele garante que todo o resto do ambiente esteja padronizado

# 🧠 Conceito central

**Ambiente como código**

- Scripts Bash versionados
- Infraestrutura definida em Docker
- Mesma experiência em qualquer máquina
- Atualizações transparentes

Para quem executa os scripts, nada muda:
o ambiente simplesmente continua funcionando, mesmo após atualizações.

# 🚀 Início rápido

```
git clone git@bitbucket.org:futura_plataforma/toolkit.git
cd toolkit
./init.sh   # executado uma vez
./setup.sh
```

O que acontece nesse processo:

- Validação do ambiente
- Preparação da estrutura local
- Configuração inicial
- Clonagem de todos os repositórios necessários para trabalhar
- Cópia automática do `.env.example` para `.env` quando o arquivo não existe
- Criação automática de rede Docker externa `plataforma-dev`
  Todas as máquinas e containers do ambiente de desenvolvimento operam nessa mesma rede

# 📁 Organização do repositório

**Estrutura clara e modular**

- `config/` → configurações de ambiente (SSH, AWS)
- `utils/` → funções e utilitários reutilizáveis
- `db-utils/` → utilidades relacionadas a banco de dados
- `wrappers/` → scripts que encapsulam ou reutilizam scripts cujo original está em outro repositório
- `docker-compose.*.yml` → definição da infraestrutura

Cada pasta tem um propósito.

# 🔧 O que o Toolkit automatiza

**Tarefas comuns do dia a dia:**

- Configuração de SSH para Bitbucket
- Criação de chaves SSH
- Criação de arquivos AWS
- Clonagem e atualização de múltiplos repositórios
- Padronização de rebase antes do Pull Request
- Criação automática de rede Docker externa `plataforma-dev`
- Subida de infraestrutura com Docker

Menos passos manuais, menos erro humano.

# 🔑 SSH sem gargalos de rede

**Configuração automática de SSH para o Bitbucket**

- Uso da porta 443
- Resolve problemas de bloqueio e limitação de velocidade na porta 22
- Funciona melhor em redes corporativas
- Aplica permissões seguras automaticamente

```
./config/bitbucket_ssh_443.sh
```

# ☁️ AWS local simplificado

**Criação automática de arquivos AWS**

- Gera `~/.aws/config` e `~/.aws/credentials`
- Conteúdo padrão para desenvolvimento local
- Não sobrescreve arquivos existentes
- Seguro para rodar mais de uma vez

```
./config/create_aws_config_files.sh
```

# 🔄 Repositórios sempre alinhados

**Clonar ou atualizar tudo com um comando**

- Clona os repositórios caso não existam
- Atualiza repositórios já existentes
- Garante que cada projeto esteja na branch esperada
- Mantém o ambiente local alinhado com o padrão da equipe

```
./clone-ou-update-repos.sh
```

# 🧩 Rebase padronizado antes do PR

**Script `pre-pr.sh`**

- Executado dentro do projeto
- Impede execução em branches proibidas
- Força rebase interativo
- Gera log do rebase
- Ajuda a manter histórico limpo e organizado

Pull Requests mais fáceis de revisar e manter.

# 🧠 Dicas rápidas

| Ação | Script |
| --- | --- |
| Gerar chaves SSH | criar-ssh-key.sh |
| Configurar SSH Bitbucket | bitbucket_ssh_443.sh |
| Criar arquivos AWS | create_aws_config_files.sh |
| Clonar / atualizar repos | clone-ou-update-repos.sh |
| Padronizar rebase PR | pre-pr.sh |

# Estrutura local de diretórios

**Estrutura local gerada e esperada pelos scripts**

```
~/repos/
├── frontend/
├── base-microservice/
├── emissor-fiscal/
├── terraformfuturamensageria/
├── projetos-lambda/
├── teste-plataforma/
└── toolkit/
```

Os scripts assumem essa organização para funcionar corretamente.

# ✅ Benefícios finais

**O impacto na prática:**

- Setup muito mais rápido
- Ambiente previsível
- Menos dependência do sistema operacional
- Atualizações sem fricção
- Foco no desenvolvimento, não no ambiente

# ❓ Dúvidas