🐳 Docker Setup - Security Incident Framework

Este documento fornece instruções completas para executar o **Security Incident Classification Framework** usando Docker Compose, permitindo executar experimentos automatizados com todos os 27 modelos SLM configurados.

📋 Pré-requisitos

- **Docker Engine** >= 20.10
- **Docker Compose** >= 2.0
- **Pelo menos 16GB de RAM** (recomendado: 32GB)
- **50GB+ de espaço livre** em disco (para modelos)
- **Conexão com internet** (para download dos modelos)

🚀 Início Rápido

1. Clonar e Navegar para o Diretório

```
cd /path/to/security-incident-framework
```

2. Construir e Iniciar os Serviços

```
# Construir e iniciar todos os serviços
docker compose up -d

# Verificar status dos serviços
docker compose ps
```

PROF

3. Configurar Modelos Ollama (Primeira Execução)

```
# Inicializar todos os 27 modelos SLM automaticamente
docker compose run --rm model-setup

# Ou verificar modelos disponíveis
docker compose exec ollama ollama list
```

4. Executar Experimentos

```
# Modo dry-run (visualizar comandos sem executar)
docker compose run --rm framework ./docker-script.sh --dry-run
```

```
# Execução completa do experimento
docker compose run --rm framework ./docker-script.sh

# Executar experimento específico manualmente
docker compose run --rm framework python main.py data/ --columns target
--model ollama_mistral_7b --technique progressive_hint --output xlsx
```

📁 Estrutura de Volumes

O Docker Compose monta os seguintes diretórios:

```
Diretório Local → Container
├── ./data          → /app/data          (Dados de entrada)
├── ./results       → /app/results       (Resultados gerados)
├── ./logs          → /app/logs          (Logs de execução)
└── ./config        → /app/config        (Configurações)
```

🔧 Serviços Docker

1. **ollama** - Servidor de Modelos SLM

- **Imagem**: `ollama/ollama:latest`
- **Porta**: `11434:11434`
- **Volume**: `ollama_data` (persistente)
- **Recursos**: 4-8GB RAM reservados

2. **framework** - Aplicação Principal

- **Build**: Dockerfile local
- **Dependências**: ollama (aguarda health check)
- **Volumes**: dados, resultados, logs, config

PROF

3. **model-setup** - Inicializador de Modelos

- **Build**: Dockerfile local
- **Função**: Download automático dos 27 modelos SLM
- **Execução**: sob demanda (`docker compose run model-setup`)

🔍 Comandos Úteis

Gerenciamento de Serviços

```
# Iniciar todos os serviços
docker compose up -d

# Parar todos os serviços
```

```
docker compose down

# Visualizar logs
docker compose logs -f framework
docker compose logs -f ollama

# Remover tudo (incluindo volumes)
docker compose down -v
```

Execução de Experimentos

```
# Dry-run completo (ver todos os 108 comandos)
docker compose run --rm framework ./docker-script.sh --dry-run

# Execução real de todos os experimentos
docker compose run --rm framework ./docker-script.sh

# Executar apenas um modelo/técnica específicos
docker compose run --rm framework python main.py data/ \
    --columns target \
    --model ollama_mistral_7b \
    --technique progressive_hint \
    --output xlsx \
    --config config/docker_config.json
```

Gerenciamento de Modelos

```
# Listar modelos instalados
docker compose exec ollama ollama list

# Baixar modelo específico
docker compose exec ollama ollama pull mistral:7b

# Remover modelo
docker compose exec ollama ollama rm mistral:7b

# Verificar status do Ollama
curl http://localhost:11434/api/tags
```

PROF

Debug e Manutenção

```
# Acessar shell do container da aplicação
docker compose run --rm framework bash

# Acessar shell do container Ollama
```

```
docker compose exec ollama bash

# Verificar recursos utilizados
docker stats

# Limpar containers parados
docker compose down --remove-orphans
```

📊 Execução de Experimentos

Experimento Completo Automatizado

O script `docker-script.sh` executa **108 classificações**:

- **27 modelos SLM** × **4 técnicas de prompt** = 108 execuções
- **Técnicas**: progressive_hint, progressive_rectification, self_hint, hypothesis_testing
- **Tempo estimado**: 4-8 horas (dependendo do hardware)

Estrutura de Resultados

```
results/
├── classification_results_YYYYMMDD_HHMMSS.xlsx
├── progressive_hint/
│   ├── ollama_mistral_7b_results.xlsx
│   ├── ollama_falcon3_10b_results.xlsx
│   └── ...
├── progressive_rectification/
├── self_hint/
└── hypothesis_testing/
```

Logs Detalhados

PROF

```
logs/
├── framework_YYYYMMDD.log          (Log principal)
├── ollama_interactions.log         (Interações com modelos)
├── classification_errors.log       (Erros de classificação)
└── performance_metrics.log         (Métricas de performance)
```

⚡ Configurações de Performance

Recursos Recomendados

```
# Para modificar no docker compose.yml
services:
  ollama:
```

```
    deploy:
      resources:
        reservations:
          memory: 8G      # Mínimo recomendado
        limits:
          memory: 16G     # Para modelos grandes
```

Otimizações

1. **SSD Storage**: Use SSD para melhor performance de I/O
2. **GPU Support**: Para usar GPU com Ollama:

```
services:
  ollama:
    deploy:
      resources:
        reservations:
          devices:
            - driver: nvidia
              count: 1
              capabilities: [gpu]
```

🔧 Solução de Problemas

Problema: Ollama não responde

```
# Verificar logs do Ollama
docker compose logs ollama

# Reiniciar serviço Ollama
docker compose restart ollama

# Verificar conectividade
curl http://localhost:11434/api/tags
```

PROF

Problema: Modelos não encontrados

```
# Executar setup de modelos novamente
docker compose run --rm model-setup

# Verificar modelos disponíveis
docker compose exec ollama ollama list
```

Problema: Erro de memória

```
# Aumentar recursos no docker compose.yml
# Ou executar menos modelos simultaneamente
# Verificar uso de recursos
docker stats
```

Problema: Erro de permissões

```
# Ajustar permissões dos diretórios
sudo chown -R $USER:$USER ./data ./results ./logs

# Ou executar como root (não recomendado)
docker compose run --rm --user root framework ./docker-script.sh
```

📈 Monitoramento

Acompanhar Progresso

```
# Logs em tempo real
docker compose logs -f framework

# Status dos containers
watch docker compose ps

# Uso de recursos
watch docker stats
```

Métricas de Execução

PROF

O framework gera automaticamente:

- **Accuracy, Precision, Recall, F1-Score**
- **Confusion Matrix**
- **Tempo de execução por modelo**
- **Uso de recursos**

🚪 Finalização

```
# Parar todos os serviços
docker compose down

# Remover volumes (CUIDADO: perde modelos baixados)
docker compose down -v
```

```
# Limpeza completa
docker system prune -a
```

💡 Dicas Importantes

1. **Primeira execução**: Reserve tempo para download dos modelos (1-2 horas)
2. **Backup**: Faça backup da pasta `results/` regularmente
3. **Recursos**: Monitore RAM e armazenamento durante execução
4. **Logs**: Use `tail -f logs/framework_*.log` para acompanhar progresso
5. **Interrupção**: Use `Ctrl+C` para parar execução de forma segura

🎯 **Experimento completo**: 108 classificações automatizadas com todos os modelos SLM!