Ferramentas de sustentação ao ambiente SIGUFRN

Repositório: https://git.ufla.br/sig-ufrn/ambiente-docker-sustentacao

Este repositório reúne serviços auxiliares para sustentação, observabilidade, proxy reverso, disponibilidade e apoio operacional dos ambientes SIGUFRN.

Os componentes principais são:

- pgbouncer: pool de conexões para PostgreSQL.
- haproxy: proxy reverso HTTPS com regras de segurança e balanceamento.
- grafana: stack de monitoramento com Grafana, VictoriaMetrics, Loki, Alloy e Nginx.
- gatus: página de status e healthcheck de endpoints.

Pré-requisitos

- Docker instalado.
- Docker Compose instalado.
- Acesso aos serviços monitorados ou encaminhados.
- mkcert, quando for necessário gerar certificados locais para *.localhost.
- apache2-utils, quando for necessário criar .htpasswd para autenticação básica no Nginx da stack de monitoramento.

Instalação do apache2-utils:

```
sudo apt install apache2-utils
```

Estrutura Geral

```
.
├── gatus/
│   ├── config.yml
```

```
│   ├── data/
│   └── docker-compose.yml
├── grafana/
│   ├── docker-compose.yml
│   ├── docker-compose-master.yml
│   ├── docker-compose-node.yml
│   ├── config.alloy
│   ├── config-node.alloy
│   ├── loki-config.yaml
│   ├── nginx.conf
│   ├── provisioning/
│   └── certs/
├── haproxy/
│   ├── docker-compose.yml
│   ├── haproxy.cfg
│   ├── certs/
│   └── security/
└── pgbouncer/
    ├── compose.yaml
    ├── pgbouncer.ini
    ├── userlist.txt
    └── testes/
```

Portas e Acessos

Serviço	Porta	Acesso
PgBouncer	6432	PostgreSQL via pool de conexões.
HAProxy HTTP	80	Redirecionamento para HTTPS.
HAProxy HTTPS	443	Entrada HTTPS para aplicações SIG
HAProxy stats	8404	http://127.0.0.1:8404/stats
Gatus	8081	http://127.0.0.1:8081
Grafana local	3000	http://127.0.0.1:3000 no compose completo
Grafana master	443	https://grafana.localhost via Nginx
Loki master	443	https://loki.localhost via Nginx

VictoriaMetrics master	443	https://metrics.localhost via Nginx

PgBouncer

Diretório: pgbouncer/

O PgBouncer atua como pool de conexões entre aplicações e PostgreSQL. O Compose usa a imagem cleanstart/pgbouncer:1.24.1 e expõe a porta 6432.

Arquivos principais

- compose.yaml: define o container pg_bouncer, porta 6432 e volumes de configuração.
- pgbouncer.ini: define bancos, host PostgreSQL, modo de pool, limites e usuários administrativos.
- userlist.txt: define usuários e senhas aceitos pelo PgBouncer.
- testes/: contém teste Python para simular usuários acessando o banco.

Configuração

Em pgbouncer.ini, os bancos são definidos na seção [databases]:

```
ufla_administrativo_20251015 = host=172.17.0.1 port=5432 dbname=ufla_administrativo_20251015
ufla_sigaa_20251209 = host=172.17.0.1 port=5432 dbname=ufla_sigaa_20251209
ufla_sistemas_comum_20250622 = host=172.17.0.1 port=5432 dbname=ufla_sistemas_comum_20250622
```

A configuração atual usa:

- pool_mode = transaction
- listen_addr = 0.0.0.0
- listen_port = 6432
- auth_type = plain
- max_client_conn = 100
- default_pool_size = 10
- min_pool_size = 5
- reserve_pool_size = 5

Antes de usar em ambiente compartilhado, revise userlist.txt, admin_users, stats_users e senhas.

Execução

```
docker compose up -d
```

Execute o comando dentro de pgbouncer/.

Uso pela aplicação

Com a configuração atual, a aplicação deve apontar para a porta 6432 em vez de 5432.

No postgres-ds.xml, altere a porta da conexão PostgreSQL:

```
5432 -> 6432
```

Testes

Diretório: pgbouncer/testes/

```
python3 -m venv venv
source venv/bin/activate
python3 -m pip install -r ./requirements.txt
cp .env.sample .env
python3 teste.py
```

Configure o .env antes de executar o teste.

HAProxy

Diretório: haproxy/

O HAProxy atua como proxy reverso HTTPS para aplicações SIG e inclui regras de segurança para limitar métodos, normalizar URLs, bloquear padrões suspeitos e proteger contra excesso de conexões/requisições.

Arquivos principais

- docker-compose.yml: define o container haproxy, volumes, portas e host.docker.internal.
- haproxy.cfg: configuração principal do proxy, TLS, stats, ACLs, redirecionamentos e backend.
- certs/: certificados TLS usados pelo HAProxy.
- security/blacklist_paths.lst: caminhos bloqueados.
- security/forbidden_extensions.lst: extensões bloqueadas.
- security/suspicious_patterns.reg: padrões suspeitos bloqueados.

Certificado local

Para desenvolvimento, use mkcert:

```
mkcert "*.localhost"
cat _wildcard.localhost.pem _wildcard.localhost-key.pem > certs/_wildcard.localhost-combined.pem
```

O Compose espera o arquivo:

```
haproxy/certs/_wildcard.localhost-combined.pem
```

DNS local

Adicione ao /etc/hosts:

```
127.0.0.1       sipac.localhost sigrh.localhost sigadmin.localhost sigaa.localhost
```

Execução

```
docker compose up -d
```

Execute o comando dentro de haproxy/.

Acessos

- SIGAA: https://sigaa.localhost
- SIPAC: https://sipac.localhost
- SIGRH: https://sigrh.localhost
- SIGAdmin: https://sigadmin.localhost
- HAProxy stats: http://127.0.0.1:8404/stats

O usuário/senha atual do stats é user:password. Altere antes de usar fora de desenvolvimento.

Backend

O backend atual encaminha para:

```
host.docker.internal:8080
```

Isso permite encaminhar chamadas para o JBoss em execução no host ou em outro container acessível pelo gateway do Docker.

Gatus

Diretório: gatus/

O Gatus monitora endpoints HTTP/HTTPS e exibe uma página de status. O Compose usa a imagem twinproduction/gatus, expõe 8081 no host e persiste dados em SQLite.

Arquivos principais

- docker-compose.yml: define o container gatus, porta 8081, volumes e host.docker.internal.
- config.yml: define storage e endpoints monitorados.
- data/: armazena o banco SQLite em /data/data.db dentro do container.

Endpoints monitorados

O arquivo config.yml contém grupos de produção e homologação. Cada endpoint valida condição HTTP 200 a cada 30s.

Exemplos:

- appserver3: http://177.105.6.3:8080/public/jsp/portal.jsf
- appserver4: http://177.105.6.4:8080/public/jsp/portal.jsf
- apps-via-haproxy: https://sipac.ufla.br/public/jsp/portal.jsf
- sigteste2: http://177.105.6.20:8080/public/jsp/portal.jsf
- sigteste3: http://177.105.6.24:8080/public/jsp/portal.jsf

Execução

```
docker compose up -d
```

Execute o comando dentro de gatus/.

Depois acesse:

```
http://127.0.0.1:8081
```

Grafana e Monitoramento

Diretório: grafana/

A stack de monitoramento usa:

- Grafana para visualização.
- VictoriaMetrics para armazenamento de métricas.
- Loki para armazenamento de logs.
- Alloy para coleta de métricas e logs.

- Nginx no modo master, para expor Grafana, Loki e VictoriaMetrics com HTTPS e autenticação básica.

Modos de execução

Há três arquivos Compose principais:

- docker-compose.yml: stack completa local com Grafana, Loki, VictoriaMetrics e Alloy no mesmo host.
- docker-compose-master.yml: master de monitoramento com Nginx, Grafana, Loki e VictoriaMetrics.
- docker-compose-node.yml: nó coletor com Alloy enviando métricas/logs para o master.

Stack completa local

Use quando todos os componentes rodam no mesmo host.

```
docker compose up -d
```

Acesso ao Grafana:

```
http://127.0.0.1:3000
```

Credenciais atuais:

```
admin / admin
```

Altere a senha antes de usar em ambiente compartilhado.

Master de monitoramento

O modo master sobe:

- nginx
- grafana
- loki
- victoria-metrics

Execução:

```
docker compose -f docker-compose-master.yml --project-name grafana-master up -d
```

O Nginx expõe:

- https://grafana.localhost
- https://loki.localhost

- https://metrics.localhost

Node de monitoramento

O modo node sobe apenas o Alloy para coletar métricas/logs do host e enviar para o master.

Execução:

```
docker compose -f docker-compose-node.yml --project-name grafana-node up -d
```

O docker-compose-node.yml usa variáveis como:

- POSTGRES_EXPORTER_DATA_SOURCE_URI
- JMX_EXPORTER_INSTANCE
- JMX_EXPORTER_URL
- JMX_EXPORTER_USER
- JMX_EXPORTER_PASS
- METRICS_ENDPOINT
- METRICS_USER
- METRICS_PASS
- LOKI_ENDPOINT
- LOKI_USER
- LOKI_PASS

Revise esses valores antes de subir o node.

Certificados do Grafana master

Para desenvolvimento, gere certificados com mkcert dentro de grafana/certs/:

```
cd certs/
mkcert *.localhost
```

Adicione ao /etc/hosts:

```
127.0.0.1       grafana.localhost
127.0.0.1       loki.localhost
127.0.0.1       metrics.localhost
```

Se o node precisar confiar na CA local, execute:

```
docker compose -f docker-compose-node.yml exec alloy update-ca-certificates
docker compose -f docker-compose-node.yml restart alloy
```

Autenticação básica do Nginx

O master usa .htpasswd para proteger Loki e VictoriaMetrics.

Crie o arquivo em grafana/.htpasswd:

```
htpasswd -c .htpasswd dgti
```

Depois configure o mesmo usuário e senha nos nodes, em docker-compose-node.yml, nas variáveis:

- METRICS_USER
- METRICS_PASS
- LOKI_USER
- LOKI_PASS

Métricas coletadas pelo Alloy

O Alloy coleta:

- métricas do host via prometheus.exporter.unix;
- métricas de containers via prometheus.exporter.cadvisor;
- métricas do PostgreSQL via prometheus.exporter.postgres;
- métricas da aplicação Java via JMX Exporter;
- logs dos containers Docker via loki.source.docker.

No modo local, os dados são enviados para victoria-metrics:8428 e loki:3100 pela rede Docker monitoring.

No modo node, os dados são enviados para endpoints remotos definidos por ambiente:

```
https://metrics.localhost/prometheus/api/v1/write
https://loki.localhost/loki/api/v1/push
```

Usuário PostgreSQL para métricas

Para PostgreSQL 10 ou superior:

```
CREATE USER postgres_exporter;
ALTER USER postgres_exporter WITH PASSWORD 'password';
GRANT CONNECT ON DATABASE postgres TO postgres_exporter;
GRANT pg_monitor to postgres_exporter;
```

Para PostgreSQL anterior à versão 10, consulte grafana/README.md, pois é necessário criar schema, funções e views auxiliares.

Datasources e dashboards

Os datasources são provisionados em:

```
grafana/provisioning/datasources/datasource.yaml
```

Datasources configurados:

- VictoriaMetrics, como Prometheus em http://victoria-metrics:8428.
- Loki, em http://loki:3100.

Os dashboards são provisionados em:

```
grafana/provisioning/dashboards/files/
```

Arquivos existentes:

- jmx.json
- 12486_rev2.json
- 9628_rev8.json
- 21743_rev3.json

Operações Comuns

Ver containers

```
docker compose ps
```

Execute dentro do diretório do serviço correspondente.

Ver logs

```
docker logs -f <container>
```

Exemplos:

```
docker logs -f pg_bouncer
docker logs -f haproxy
docker logs -f gatus
docker logs -f grafana
docker logs -f alloy
docker logs -f loki
```

```
docker logs -f victoria-metrics
```

Reiniciar serviço

```
docker compose restart
```

Ou reinicie um container específico:

```
docker restart <container>
```

Atualizar imagens

```
docker compose pull
docker compose up -d
```

No Grafana, o justfile registra comandos de pull para os projetos master e node:

```
docker compose -f docker-compose-node.yml --project-name grafana-node pull
docker compose -f docker-compose-master.yml --project-name grafana-master pull
```

Segurança e Arquivos Sensíveis

Revise credenciais antes de usar em ambientes compartilhados ou produção.

Pontos sensíveis:

- pgbouncer/userlist.txt contém usuários/senhas do PgBouncer.
- pgbouncer/pgbouncer.ini usa auth_type = plain.
- haproxy/haproxy.cfg contém stats auth user:password.
- grafana/docker-compose.yml usa GF_SECURITY_ADMIN_PASSWORD=admin.
- grafana/docker-compose-node.yml contém senhas de Metrics/Loki/JMX.
- grafana/.htpasswd é ignorado pelo Git e deve ser criado localmente.

- grafana/certs/ é ignorado pelo Git e deve ser preparado localmente.

O .gitignore do diretório ignora:

- pgbouncer/testes/venv/
- pgbouncer/testes/.env
- pgbouncer/testes/app.log
- grafana/certs/
- grafana/.htpasswd

Solução de Problemas

PgBouncer não conecta ao PostgreSQL

Confira host, porta e banco em pgbouncer.ini. Confirme também se o usuário em userlist.txt corresponde ao usuário usado pela aplicação.

Aplicação continua conectando direto no PostgreSQL

Confirme se o postgres-ds.xml foi alterado para usar a porta 6432.

HAProxy não sobe por erro de certificado

Verifique se o arquivo haproxy/certs/_wildcard.localhost-combined.pem existe e se o volume no docker-compose.yml aponta para o caminho correto.

Domínios *.localhost não resolvem

Revise o /etc/hosts e confirme se os nomes usados no navegador batem com os server_name ou ACLs configurados.

Grafana master retorna erro 401 para Loki ou Metrics

Confira se .htpasswd foi criado e se usuário/senha nos nodes são iguais aos configurados no Nginx.

Alloy não envia métricas ou logs

Confira as variáveis METRICS_ENDPOINT, LOKI_ENDPOINT, credenciais e conectividade com o master. Se estiver usando certificado local, atualize os certificados dentro do container Alloy.

Gatus mostra endpoint indisponível

Teste a URL configurada em gatus/config.yml a partir do host e revise a condição esperada, atualmente [STATUS] == 200.

Revisão #2
Criado 2026-08-02 12:37:45 UTC por OBEDE JESSE CARVALHO
Atualizado: 2026-08-03 09:51:53 UTC por OBEDE JESSE CARVALHO