Curso DevOps

Aula 13 - Arquitetura Cloud Native

Prof. Esp. Guilherme Jorge Aragão da Cruz

guilherme.cruz@alumni.usp.br

linkedin.com/in/guijac

Roteiro

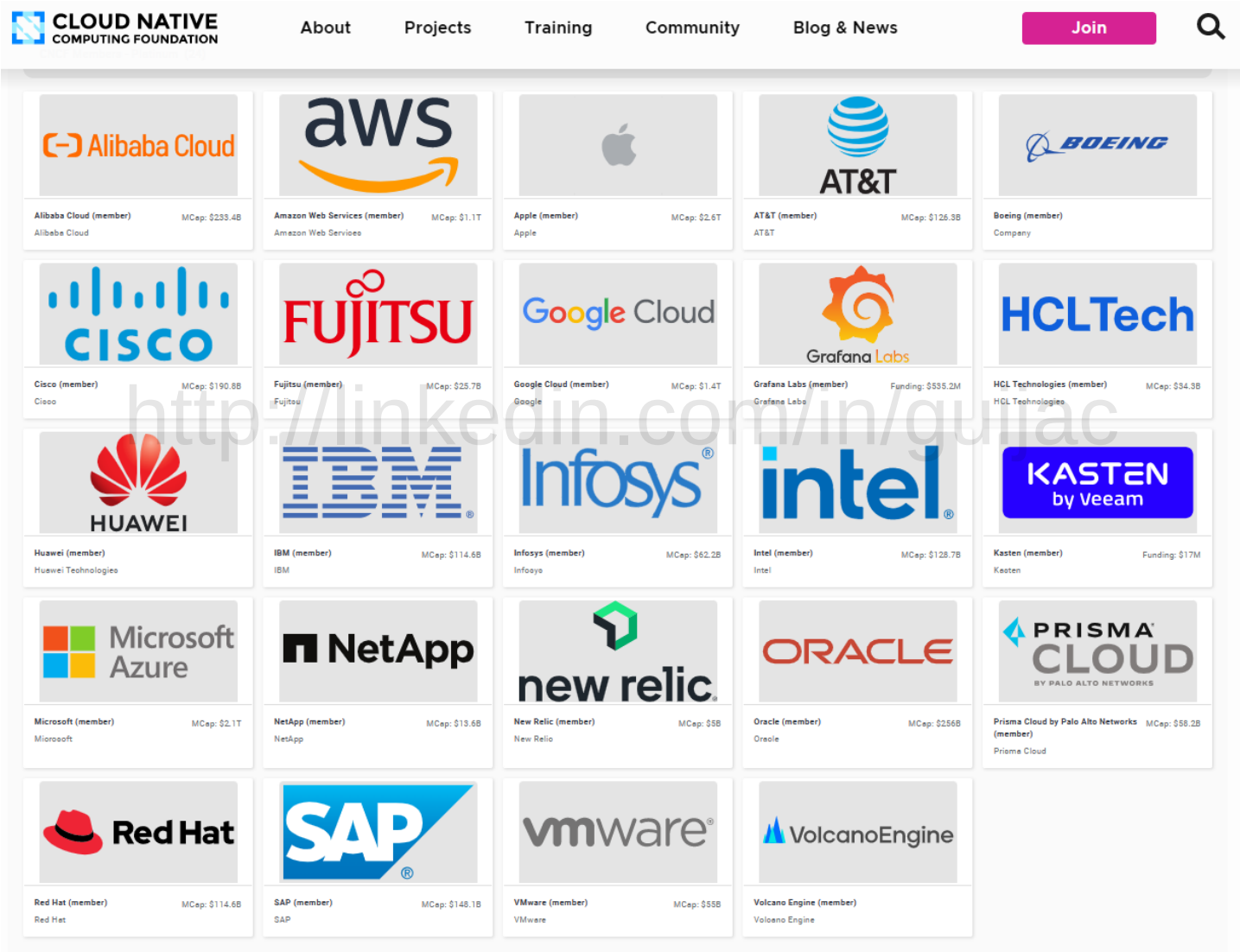
- Definição;
- A Cloud Native Computing Foundation;
- Visão Geral;
- Alguns Números;
- Microsserviços:
  - Definição;
  - Principais Componentes;
  - Principais Características.
- Metodologia 12 Fatores (The Twelve-Factor App):
  - Definição;
  - Base de Código;
  - Dependências;
  - Construa, Lance, Execute;
  - Processos;
  - Logs.
- Referências Bibliográficas.

Cloud Native: Definição

> *Tecnologias nativas ao cloud empoderam empresas a criarem e rodarem aplicações escaláveis em ambientes modernos e dinâmicos, como **nuvens** públicas, privadas e híbridas. **Contêineres**, **malhas de serviço**, **microsserviços**, **infraestruturas imutáveis** e **APIs** exemplificam bem esta estratégia.*
>
> *Tais técnicas permitem criar sistemas de **baixo acoplamento**, **resilientes**, **gerenciáveis** e **observáveis**. Combinadas com **automações** robustas, elas permitem alterações de alto impacto de forma frequente e previsível, com o mínimo de esforço.*
>
> *A Cloud Native Computing Foundation procura impulsionar a adoção desse paradigma fomentando e sustentando um ecossistema de projetos de **código aberto** e não atrelados a nenhum fornecedor. Nós democratizamos padrões do estado-da-arte para tornar essas inovações acessíveis a todos.*

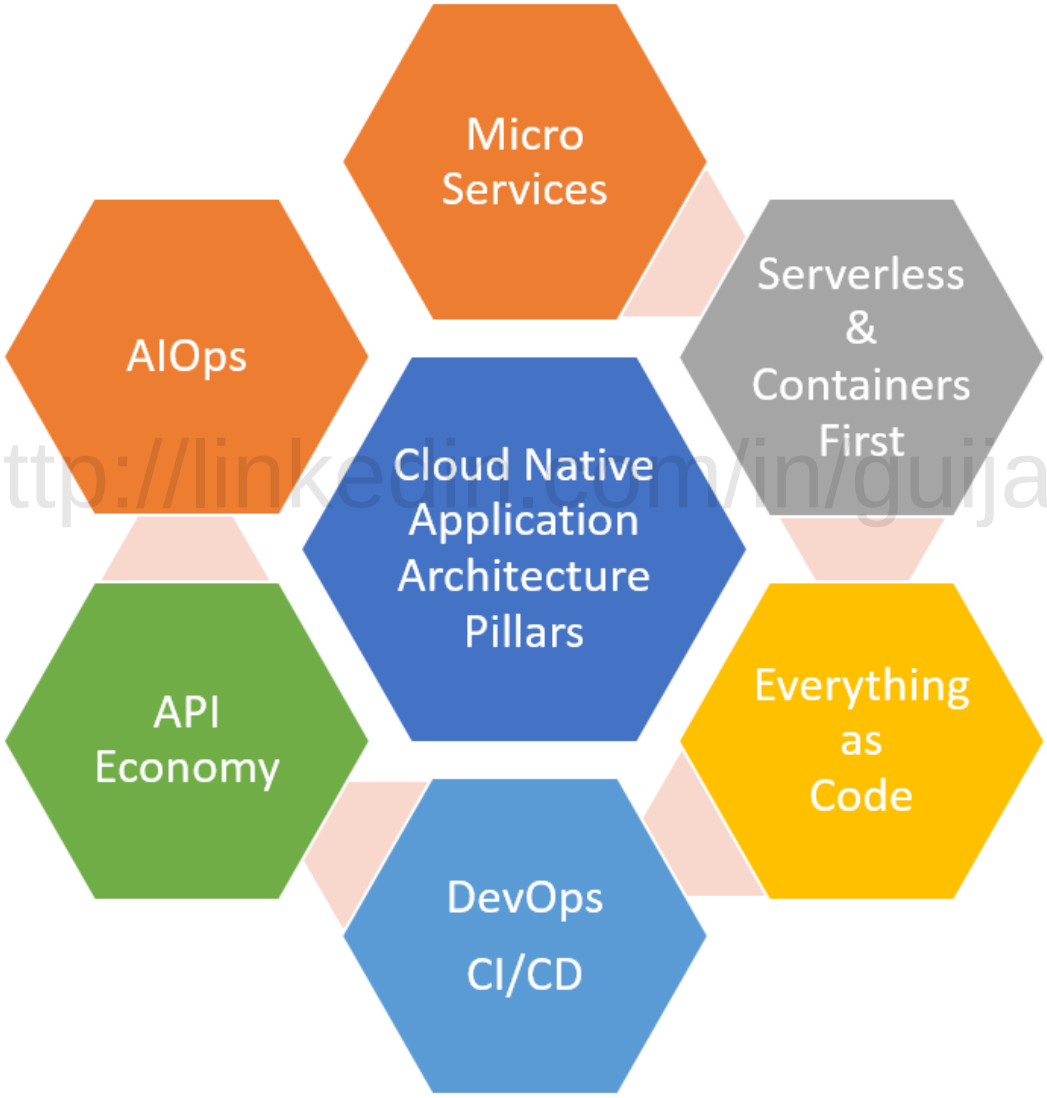
CNCF – Cloud Native Computing Foundation (2022)

Cloud Native Computing Foundation



[CNCF – Cloud Native Computing Foundation]

Cloud Native: Visão Geral



Fonte: Tech Blog » Cloud Native Application Architecture Pillars (baghel.com)

Cloud Native: Alguns Números

Cloud Native: Alguns Números

Cloud Native: Alguns Números

- Mais de 400 serviços. Implanta mais de 50 vezes por dia[1];

Cloud Native: Alguns Números

- Mais de 400 serviços. Implanta mais de 50 vezes por dia[1];

Cloud Native: Alguns Números

- Mais de 400 serviços. Implanta mais de 50 vezes por dia[1];

- Mais de 600 serviços. Implanta mais de 100 vezes por dia[2];

Cloud Native: Alguns Números

- Mais de 400 serviços. Implanta mais de 50 vezes por dia[1];

NETFLIX

- Mais de 600 serviços. Implanta mais de 100 vezes por dia[2];

Cloud Native: Alguns Números

- Mais de 400 serviços. Implanta mais de 50 vezes por dia[1];

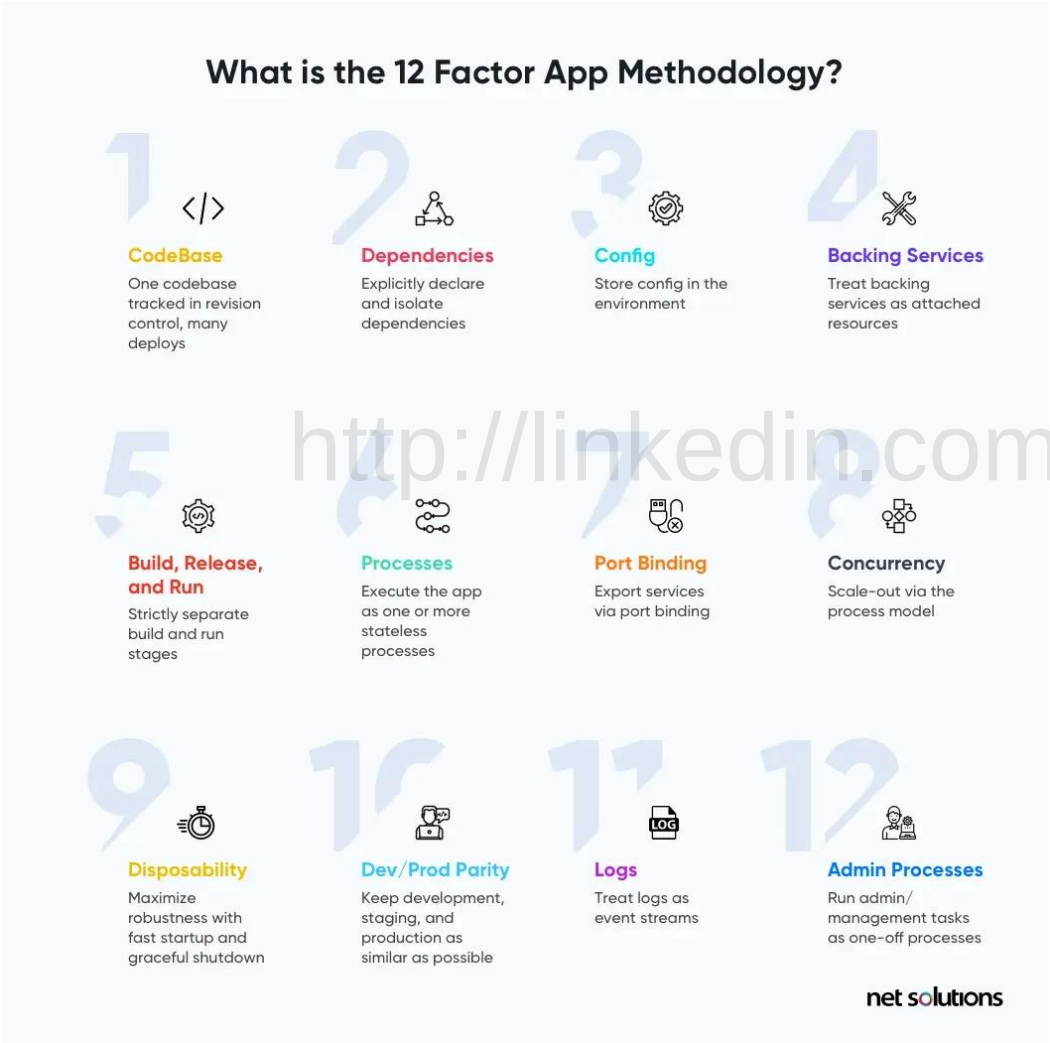
NETFLIX

- Mais de 600 serviços. Implanta mais de 100 vezes por dia[2];

- Mais de 3.000 serviços. Implanta mais de 1.000 vezes por dia[3].

[1] <https://blog.nubank.com.br/engenharia-de-software-como-funciona-no-nubank/>
[2] <https://www.infoq.com/news/2013/06/netflix/>
[3] <https://www.cs.columbia.edu/~ruigu/papers/socc18-final100.pdf>

The Twelve-Factor App



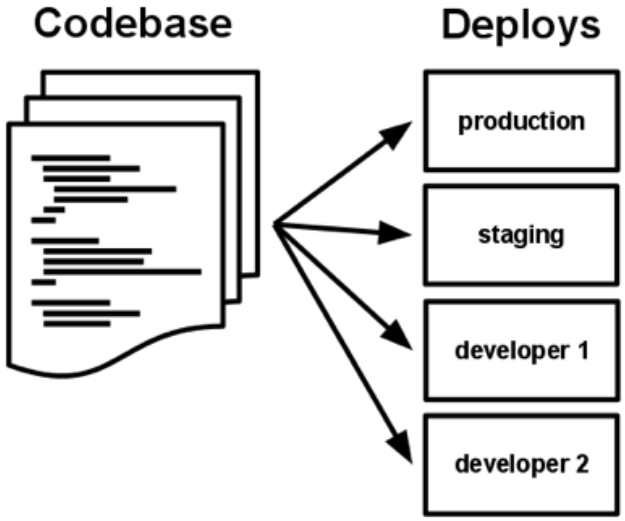
> “Conjunto de princípios publicado por Adam Wiggins (Heroku) e descreve uma maneira de fazer software que, quando seguida, permite que as organizações criem códigos que possam ser lançados de forma confiável, dimensionados rapidamente e mantidos de maneira consistente e previsível.”

RED HAT (2021)

Fonte: [The 12-Factor App Methodology Explained | Net Solutions]

The Twelve-Factor App: 1. Base de Código

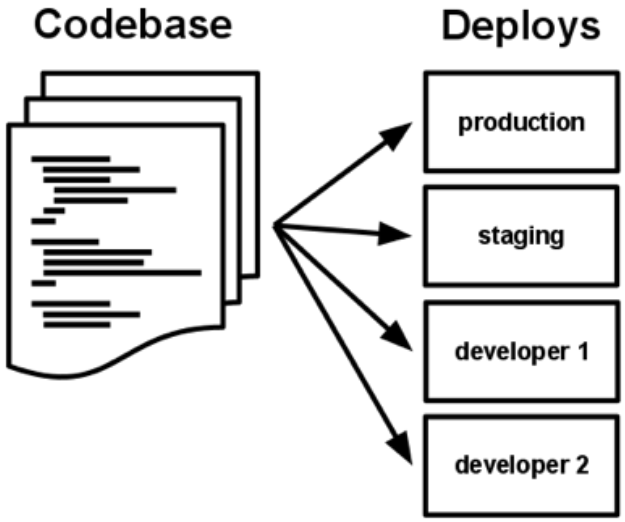
- **Uma base de código com rastreamento utilizando controle de revisão, muitos *deploys*:**
  - Uma aplicação 12 fatores é sempre rastreada em um **sistema de controle de versão**;
  - Existe apenas uma base de código por aplicação, mas existirão vários *deploys* da mesma.



Fonte: [The Twelve-Factor App (traduzido) (12factor.net)]

The Twelve-Factor App: 1. Base de Código

- **Uma base de código com rastreamento utilizando controle de revisão, muitos *deploys*:**
  - Uma aplicação 12 fatores é sempre rastreada em um **sistema de controle de versão**;
  - Existe apenas uma base de código por aplicação, mas existirão vários *deploys* da mesma.



Fonte: The Twelve-Factor App (traduzido) (12factor.net)

Fonte: Elaboração própria (2023)

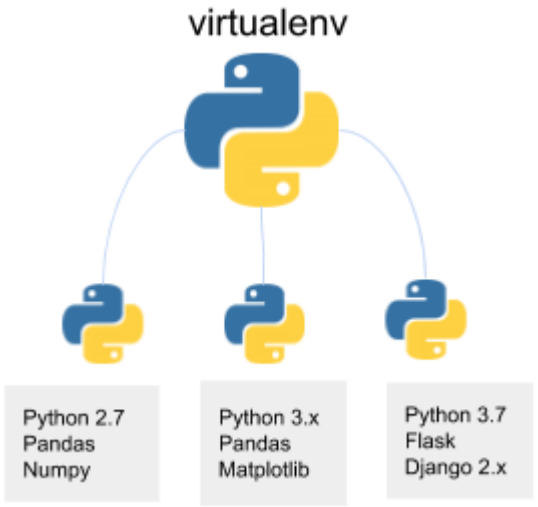
The Twelve-Factor App: 2. Dependências

- **Declare e isole explicitamente as dependências:**
  - Bibliotecas instaladas por meio de um sistema de pacotes podem ser instaladas em todo o sistema (“*site packages*”) ou com escopo dentro do diretório da aplicação (“*vendoring*” ou “*building*”);
  - Uma aplicação doze-fatores nunca confia na existência implícita de pacotes em todo o sistema. Ela **declara suas dependências por meio de um manifesto**;
  - Um dos benefícios da declaração de dependência explícita é a simplificação da configuração da aplicação para novos desenvolvedores.

The Twelve-Factor App: 2. Dependências

- **Declare e isole explicitamente as dependências:**
  - Bibliotecas instaladas por meio de um sistema de pacotes podem ser instaladas em todo o sistema ("*site packages*") ou com escopo dentro do diretório da aplicação ("*vendoring*" ou "*building*");
  - Uma aplicação doze-fatores nunca confia na existência implícita de pacotes em todo o sistema. Ela **declara suas dependências por meio de um manifesto**;
  - Um dos benefícios da declaração de dependência explícita é a simplificação da configuração da aplicação para novos desenvolvedores.

Fonte: Python: Docker image



Fonte: Python: why should we create a VE?

The Twelve-Factor App: 2. Dependências

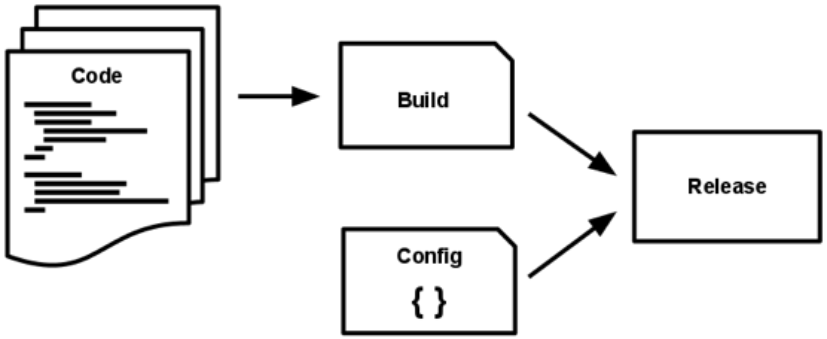
- **Virtualenv:**
  - Utilizado para tratar situações de diferentes versões de uma determinada biblioteca, para diferentes aplicações, como a appX utilizando mysqlclient:1.0 e appY utilizando mysqlclient:2.0;
  - Em contêineres é recomendado para situações de "*multi-stage build*", nas quais um comando "pip install" pode ser executado mais de uma vez.


```
# Opcional para contêineres com uma única aplicação ou sem "multi-stage build":
# Empacota todas as dependências do projeto precisa e armazena em um diretório(VIRTUAL_ENV),
# desta forma, nenhum pacote é instalado diretamente no sistema operacional/contêiner.
ENV VIRTUAL_ENV=/opt/venv
RUN python3 -m venv $VIRTUAL_ENV
ENV PATH="$VIRTUAL_ENV/bin:$PATH"
```

Dockerfile com uso do Virtualenv

The Twelve-Factor App: 5. Construa, lance, execute

- **Separe estritamente os estágios de construção e execução:**
  - O app doze-fatores usa **separação** estrita entre os **estágios** de construção (*build*), lançamento (*release*) e execução (*run/deploy*);
  - Por exemplo, é impossível alterar código em tempo de execução, já que não há meios de se propagar tais mudanças de volta ao estágio de construção;
  - Cada lançamento (*release*) deve sempre ter um identificador de lançamento único, tal qual o *timestamp* do lançamento.

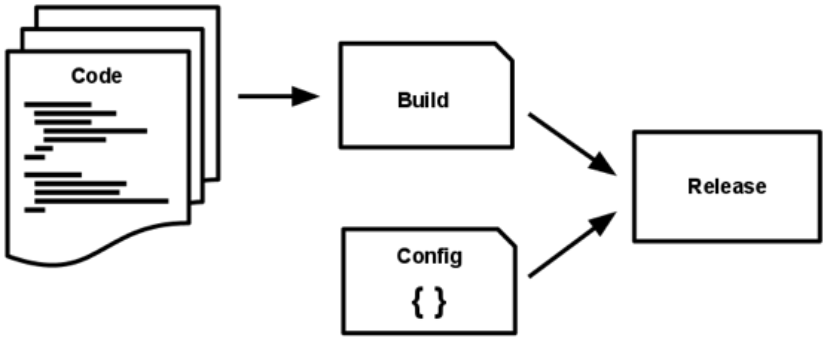


Fonte: The Twelve-Factor App (traduzido) (12factor.net)

Fonte: Do zero ao primeiro Job com Gitlab CI

The Twelve-Factor App: 5. Construa, lance, execute

- **Separe estritamente os estágios de construção e execução:**
  - O app doze-fatores usa **separação** estrita entre os **estágios** de construção (*build*), lançamento (*release*) e execução (*run/deploy*);
  - Por exemplo, é impossível alterar código em tempo de execução, já que não há meios de se propagar tais mudanças de volta ao estágio de construção;
  - Cada lançamento (*release*) deve sempre ter um identificador de lançamento único, tal qual o *timestamp* do lançamento.



Fonte: The Twelve-Factor App (traduzido) (12factor.net)

Fonte: Do zero ao primeiro Job com Gitlab CI

The Twelve-Factor App: 6. Processos

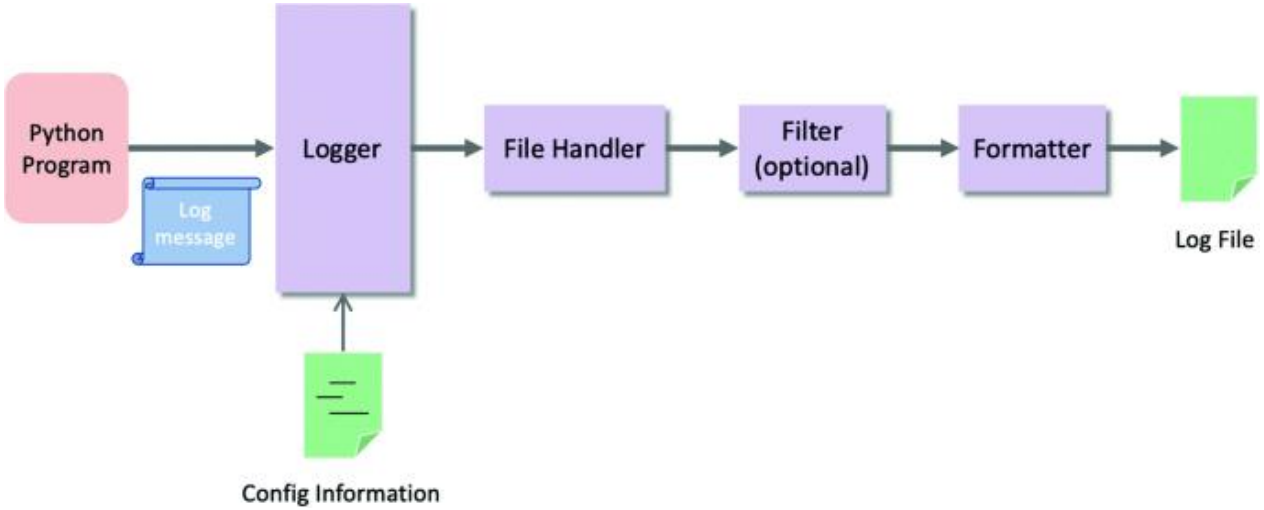
- **Execute a aplicação como um ou mais processos que não armazenam estado:**
  - Aplicações doze-fatores são *stateless* (não armazenam estado);
  - Quaisquer dados que necessitam de persistência devem ser armazenados em um serviço de apoio (fator 4) *stateful* (que armazena o seu estado), tipicamente uma **base de dados**;
  - Sessões persistentes são uma violação do doze-fatores e nunca devem ser utilizadas ou invocadas. Dados do estado da sessão são bons candidatos para uso em um **recurso que ofereça tempo de expiração**.

The Twelve-Factor App: 6. Processos

- **Execute a aplicação como um ou mais processos que não armazenam estado:**
  - Aplicações doze-fatores são *stateless* (não armazenam estado);
  - Quaisquer dados que necessitam de persistência devem ser armazenados em um serviço de apoio (fator 4) *stateful* (que armazena o seu estado), tipicamente uma **base de dados**;
  - Sessões persistentes são uma violação do doze-fatores e nunca devem ser utilizadas ou invocadas. Dados do estado da sessão são bons candidatos para uso em um **recurso que ofereça tempo de expiração**.

The Twelve-Factor App: 11. Logs

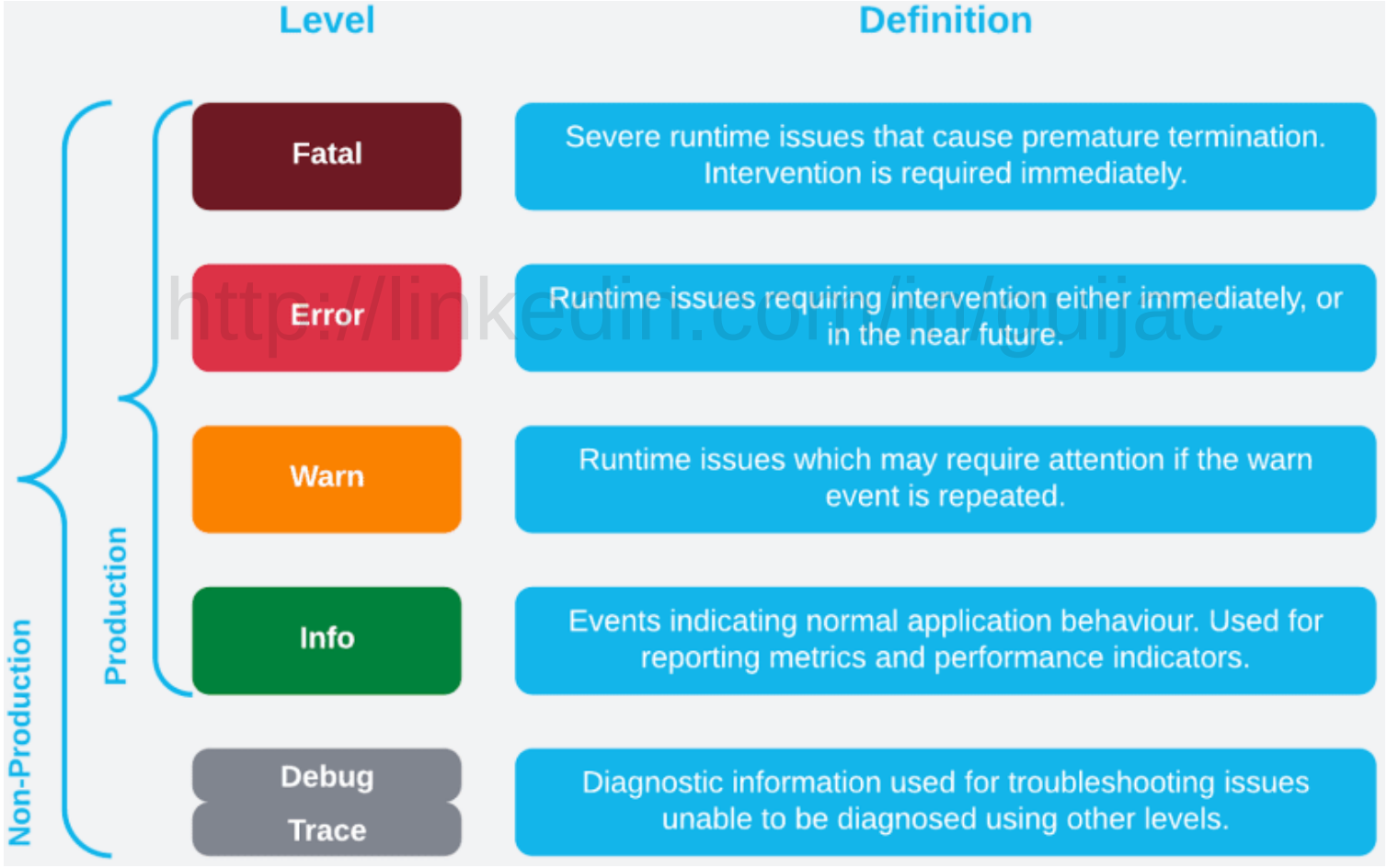
- **Trate logs como fluxos de eventos:**
  - Logs são o fluxo de eventos agregados e ordenados por tempo coletados dos fluxos de saída de todos os processos em execução e serviços de apoio;
  - Uma aplicação doze-fatores nunca se preocupa com o roteamento ou armazenagem do seu fluxo de saída. Ela não deve tentar escrever ou gerir arquivos de logs;
  - Um fluxo de evento pode ser modelado e enviado para um sistema indexador, que permite observar o comportamento de uma aplicação.



Fonte: Logging in Python | SpringerLink

The Twelve-Factor App: 11. Logs

- **Boas práticas com logs:**



Fonte: Logs - Why, good practices, and recommendations - DEV Community

The Twelve-Factor App: 11. Logs

- **Usando logs com Python:**

```python
import logging
import json

# Configuração básica do logger
logger = logging.getLogger()
logger.setLevel(logging.INFO)

# Handler para escrever os logs em stdout
handler = logging.StreamHandler()
handler.setFormatter(logging.Formatter(
    '{"time": "%(asctime)s", "level": "%(levelname)s", "message": %(message)s}'
))
logger.addHandler(handler)

# Logando um evento
logger.info(json.dumps({
    "event": “my-event",
    “var1": var1,
    “var2": var2
}))
```

Código Python utilizando log de eventos através da lib logging

The Twelve-Factor App: 11. Logs

- **Usando logs com Python:**
  - O log é gerado em formato JSON e o atributo “message” carrega os atributos especificados dentro da construção do log, sendo neste cenário:
    - **event**: identificador único do evento da aplicação;
    - **url**: pode ser recuperada do atributo request.url;
    - **user_agent**: pode ser recuperada do atributo request.headers.get("User-Agent");
    - **nome**: variável que foi recebida através do método “GET”.


```
{
    "time": "2023-05-09 15:39:49,449",
    "level": "INFO",
    "message": {
      "event": "hello-success",
      "url": "http://127.0.0.1:5000/hello?name=guijac",
      "user_agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/112.0.0.0 Safari/537.36 Edg/112.0.1722.39",
      "nome": "guijac"
    }
  }
```

Saída de um evento de log de uma aplicação Python utilizando a lib logging

Referências Bibliográficas

12FACTOR. The Twelve-Factor App. Disponível em https://12factor.net/pt_br/. Acesso em 19 jan 2025;

CNCF. **Cloud Native Definition v1.0**. Disponível em https://github.com/cncf/toc/blob/main/DEFINITION.md#portugu%C3%AAs-brasileiro. Acesso em 19 jan 2025;

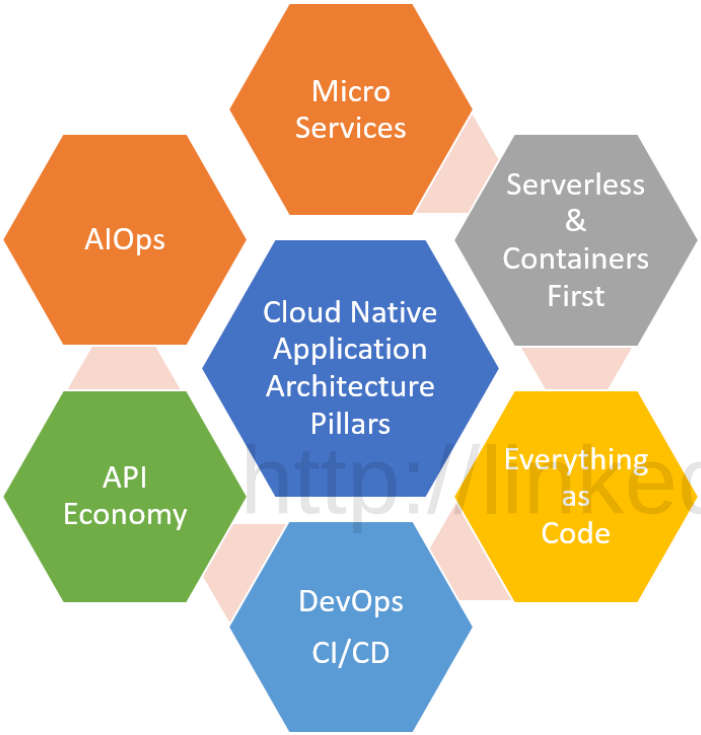
ERL, Thomas. **SOA Princípios de design de serviços**. Pearson Prentice Hall, São Paulo, 2014.

FOWLER, Martin; LEWIS, James. **Microservices a definition of this new architectural term**. Medium, 2014. Disponível em http://martinfowler.com/articles/microservices.html. Acesso em 19 jan 2025;

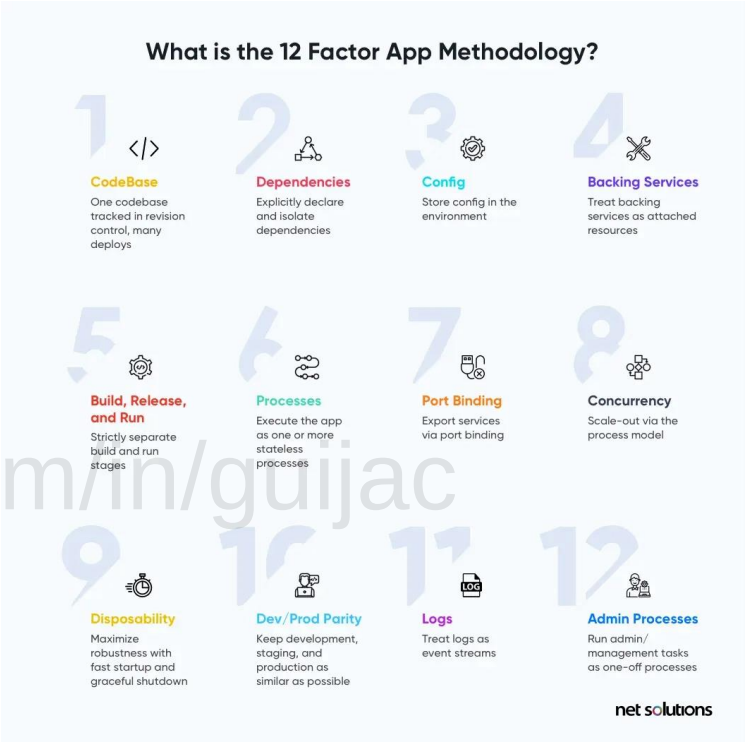
GUIJAC**. Microsserviços: Uma Visão Geral — Parte I**. Disponível em https://guijac.medium.com/microsservi%C3%A7os-uma-vis%C3%A3o-geral-parte-i-88c0c9c547ee. Acesso em 19 jan 2025;

JUNG, Matthias et al. **Microservices on AWS**. Amazon Web Services, Inc., New York, NY, USA, Tech. Rep, 2016.

Por hoje (de teoria!) é só!



Fonte: Tech Blog » Cloud Native Application Architecture Pillars



Fonte: The 12-Factor App Methodology Explained

Prof. Esp. Guilherme Jorge Aragão da Cruz

guilherme.cruz@alumni.usp.br

linkedin.com/in/guijac

Licença

- Este conteúdo está licenciado sob a Licença Creative Commons Atribuição-NãoComercial-CompartilhaIgual 4.0 Internacional (CC BY-NC-SA 4.0).
- Todos os direitos autorais sobre este conteúdo pertencem ao autor, e este material não pode ser usado comercialmente sem autorização expressa.
- Para ver o texto completo da licença, acesse o https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode.

Prof. Esp. Guilherme Jorge Aragão da Cruz

guilherme.cruz@alumni.usp.br

linkedin.com/in/guijac