Microsserviços em produção

7COMm
Serviços e Soluções em TI

Agenda

- Entregado serviços seguros
- Externalizando a configuração
- *Deploy* de microsserviços

Entregando serviços seguros

7COMm
Serviços e Soluções em TI

Segurança de uma aplicação

1. Autenticação
2. Autorização
3. Auditoria
4. Comunicação segura interprocessos

Autenticação

- Verifica a identidade da aplicação ou do usuário
- Feita normalmente através de *user_id/password* ou *API key* e *secrets*

Autorização

- Verifica se o usuário tem permissão para executar a operação especificada
- Normalmente são utilizadas uma combinação de papeis e listas de controle de acessos (ACLs)
- Cada usuário pode ter um ou mais papeis que adicionam permissão para um determinado recurso

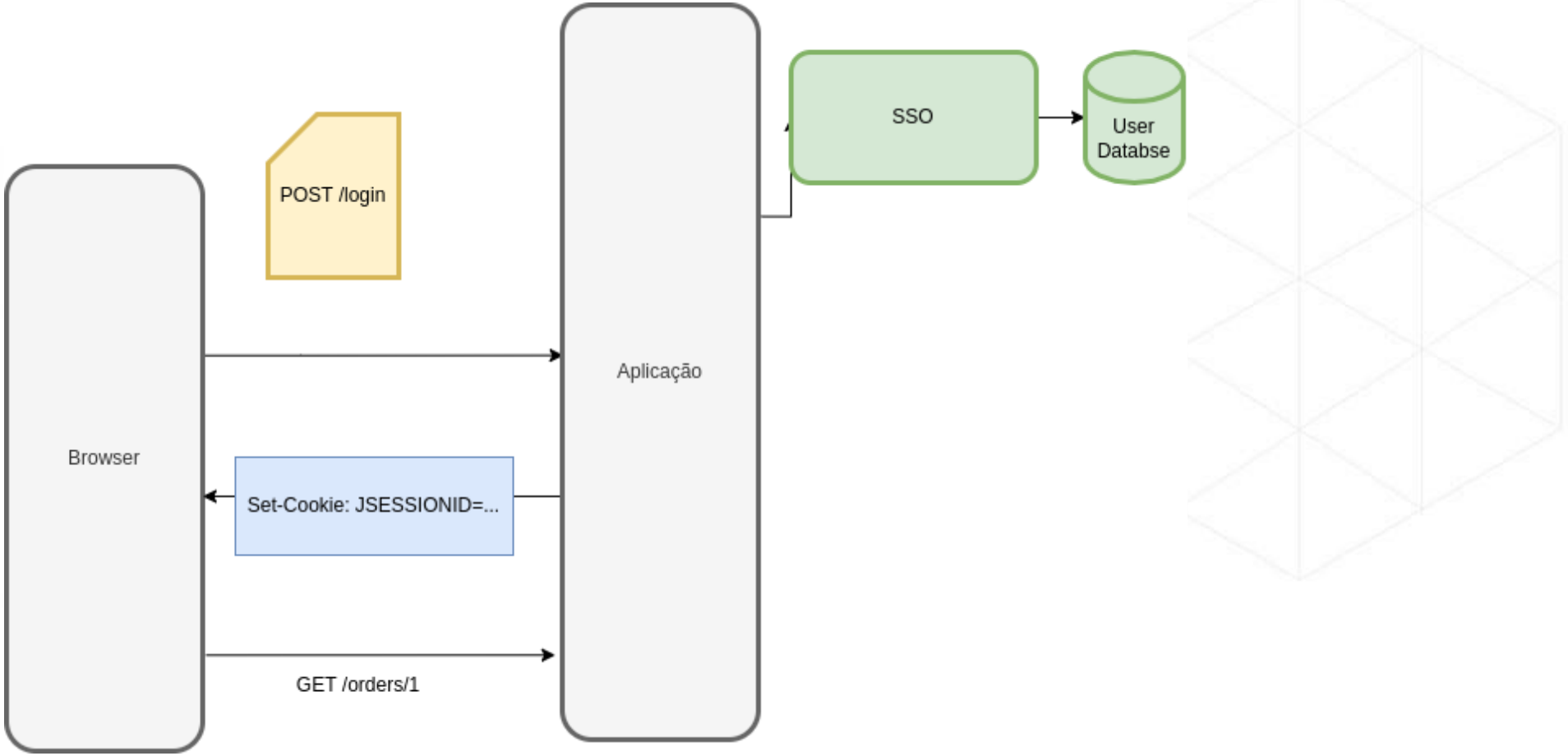
Auditoria

- A capacidade de monitorar as operações que um usuário realiza
- Ajuda a detectar pontos de segurança
- Ajuda no suporte ao usuário
- Reforça *compliance*

Comunicação segura

- Toda a comunicação entre serviços deve ser segura
- TLS
- Autenticação

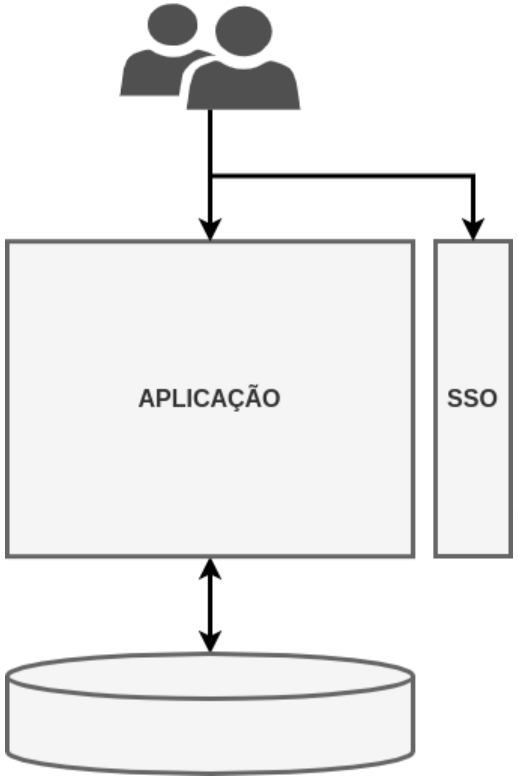
Segurança em monolito



Segurança em uma arquitetura de microsserviços

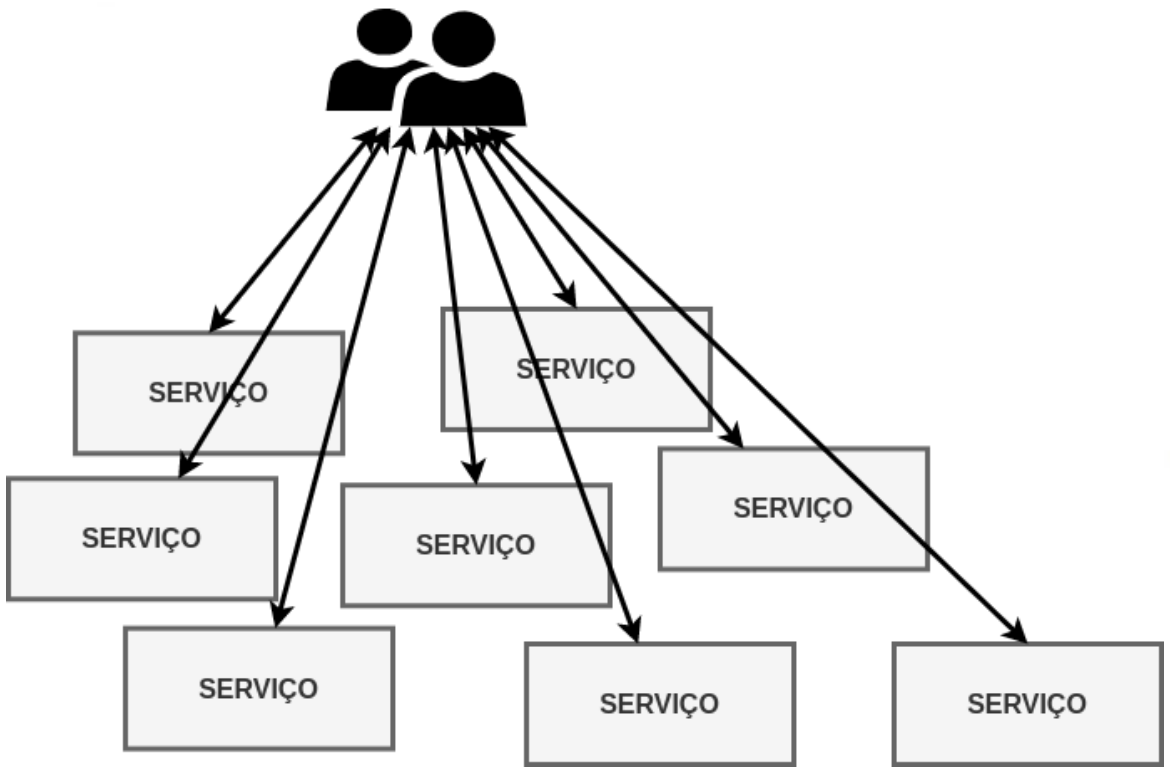
- Não devem seguir exemplos de uma arquitetura monolita
- Não utilizam sessão
- Servidores não compartilham sessão
- Sessão centralizada quebra o baixo acoplamento

Autenticação no monolito



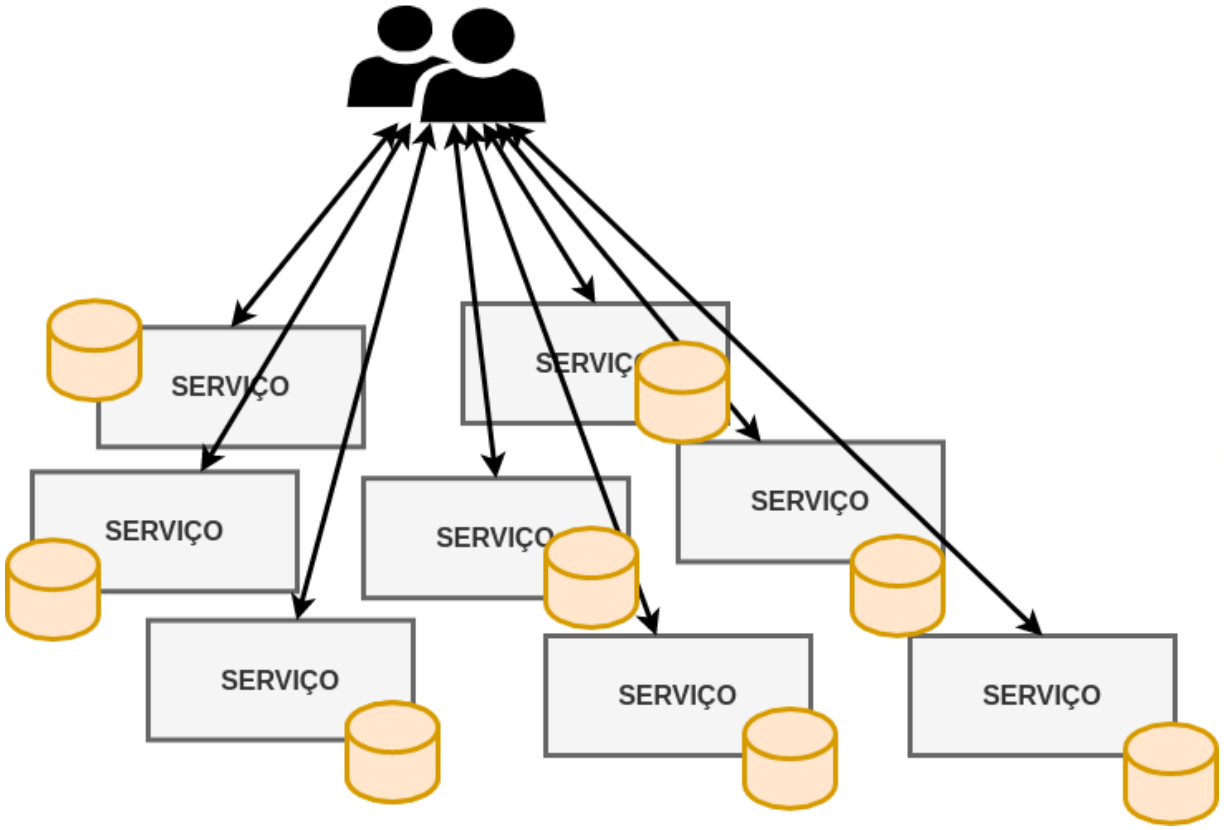
- Base de dados única
- Sessão de usuário centralizada

Autenticação em microsserviços



- Bancos descentralizados?

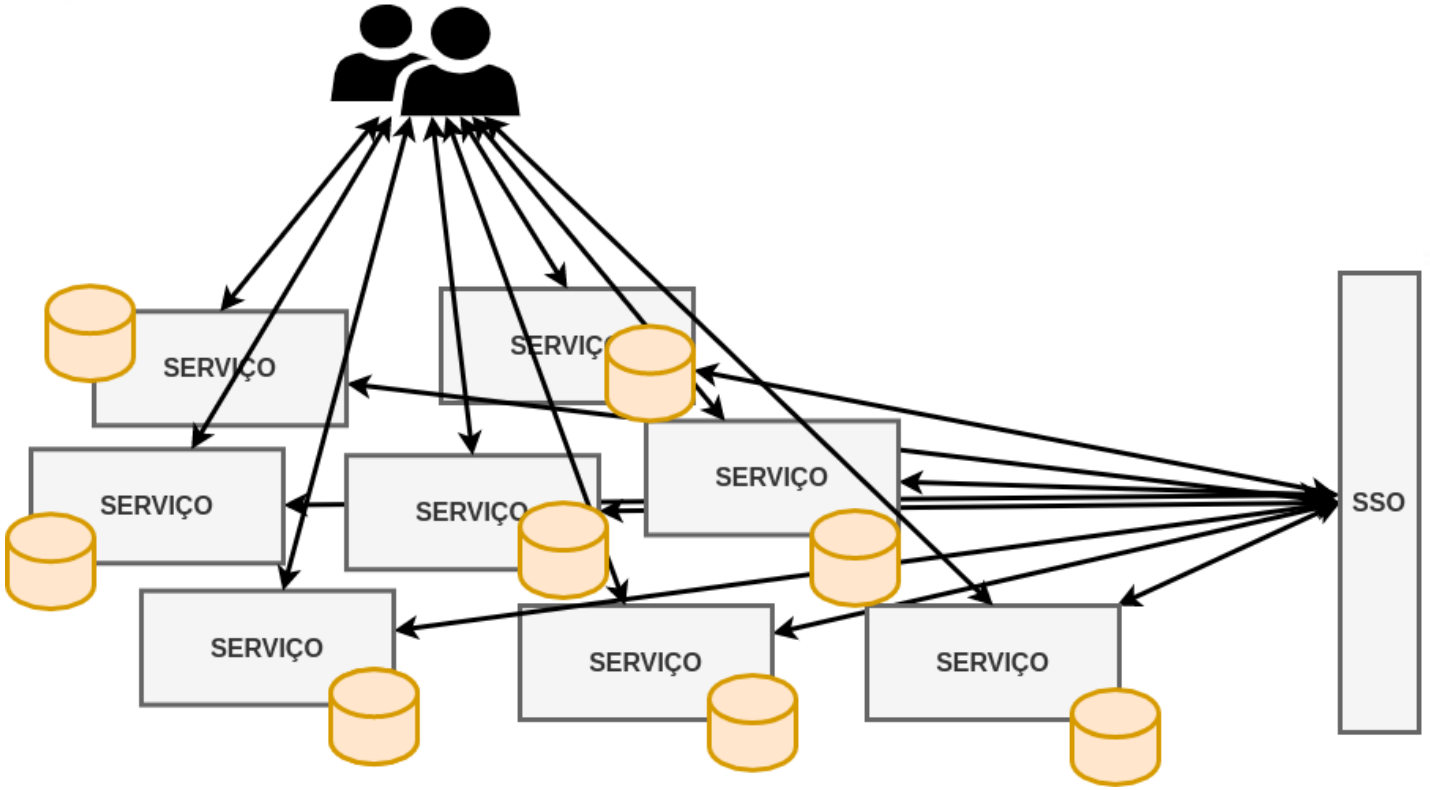
Autenticação em microsserviços



- Single Responsability
- Dificuldade de manutenção

- SSO?

Autenticação em microsserviços



- Alto acoplamento

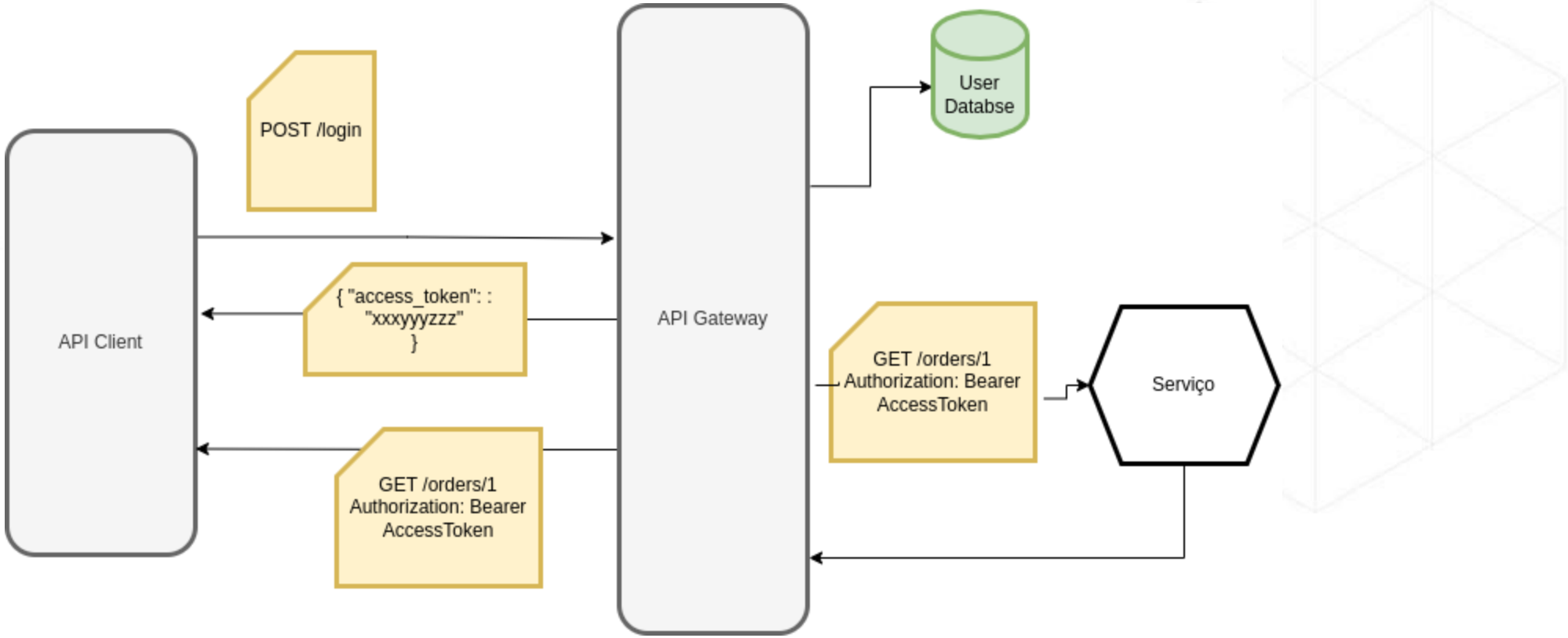
Autenticação no API Gateway

- Autenticação em cada serviço pode não ser ideal, pois permite requisições não autenticadas na rede interna
- Aumenta a complexidade da aplicação como um todo
- Uma melhor alternativa é implementar autenticação no próprio API Gateway

Autenticação no API Gateway

- Evita-se complexidade no ecossistema de autenticação
- Somente um lugar para lidar com a segurança da aplicação
- Somente ele deve se preocupar com a heterogeneidade das autenticações

Autenticação no API Gateway



Autorização

- Não necessariamente deve ser implementada no API Gateway
- Aumenta o acoplamento entre o API gateway e os serviços
- Requerem profundo conhecimento do domínio dos serviços
- Como comunicar a identidade do usuário aos outros serviços?
- http://microservices.io/patterns/security/access-token.html

Tokens de Autorização

- Uma vez autenticado, pode-se gerar tokens de autorização
- Estes podem ser validados nos próprios serviços
- Cada token pode conter os papéis de cada usuário

Passando identidade e papeis em um JWT

- Token com informações sobre o usuário
- Maneira segura de compartilhar tokens, identidade de usuários e papéis
- Pode possuir uma data de expiração
- Assinado com uma chave, o que assegura que terceiros não possam modificar este token
- Uma desvantagem é que este token é irrevogável
- Jwt.io

Header

```
base64enc({
    "alg":  "HS256",
    "typ":  "JWT"
})
```

Payload

```
base64enc({
      "iss": "toptal.com",
      "exp": 1426420800,
      "company": "Toptal",
      "awesome": true
})
```

Signature

```
HMACSHA256(
  base64enc(header)
  + '.' +,
  base64enc(payload)
  , secretKey)
```

toptal

JWT - Header

- Consiste normalmente de duas informações
  - Tipo de token
  - Algoritmo de assinatura

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

JWT - Payload

- Contém as *claims*
- Dados do usuário

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "admin": true
}
```

JWT - Assinatura

- Aplica-se o algoritmo do header + secret conhecido somente pelo serviço nos dois primeiros campos do token

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secret)
```

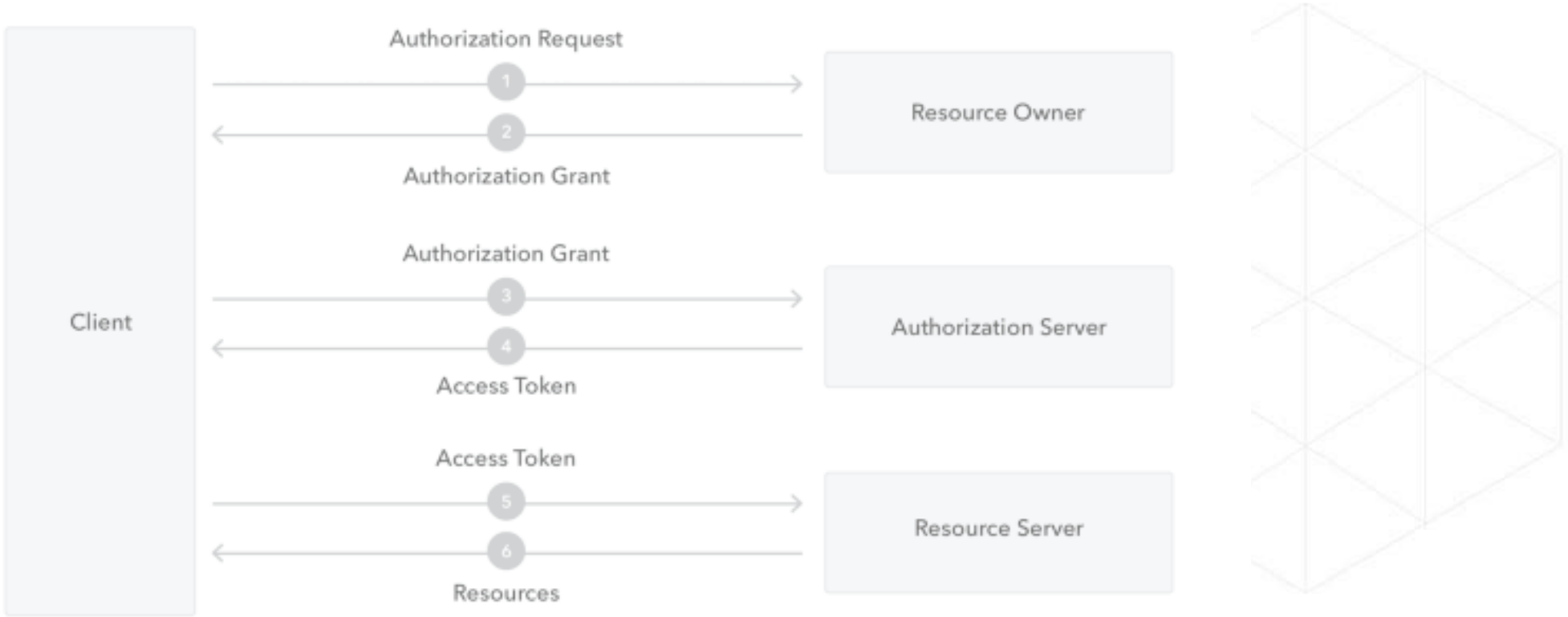
JWT

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4
gRG9lIiwiaXNTb2NpYWwiOnRydWV9.
4pcPyMD09olPSyXnrXCjTwXyr4BsezdI1AVTmud2fU4
```

OAuth 2

- Protocolo por procuração
- Permite acesso ***limitado*** aos dados de usuários
- Autorização baseada em *Roles*

OAuth 2


Client
Authorization Request
1
2
Authorization Grant
Resource Owner
Authorization Grant
3
4
Access Token
Authorization Server
Access Token
5
6
Resources
Resource Server

Externalizando Configurações

7COMm
Serviços e Soluções em TI

Configurações

- Informações que mudam de ambiente em ambiente
- Devem ser externalizadas
- Armazenadas em lugar seguro
- http://microservices.io/patterns/externalized-configuration.html

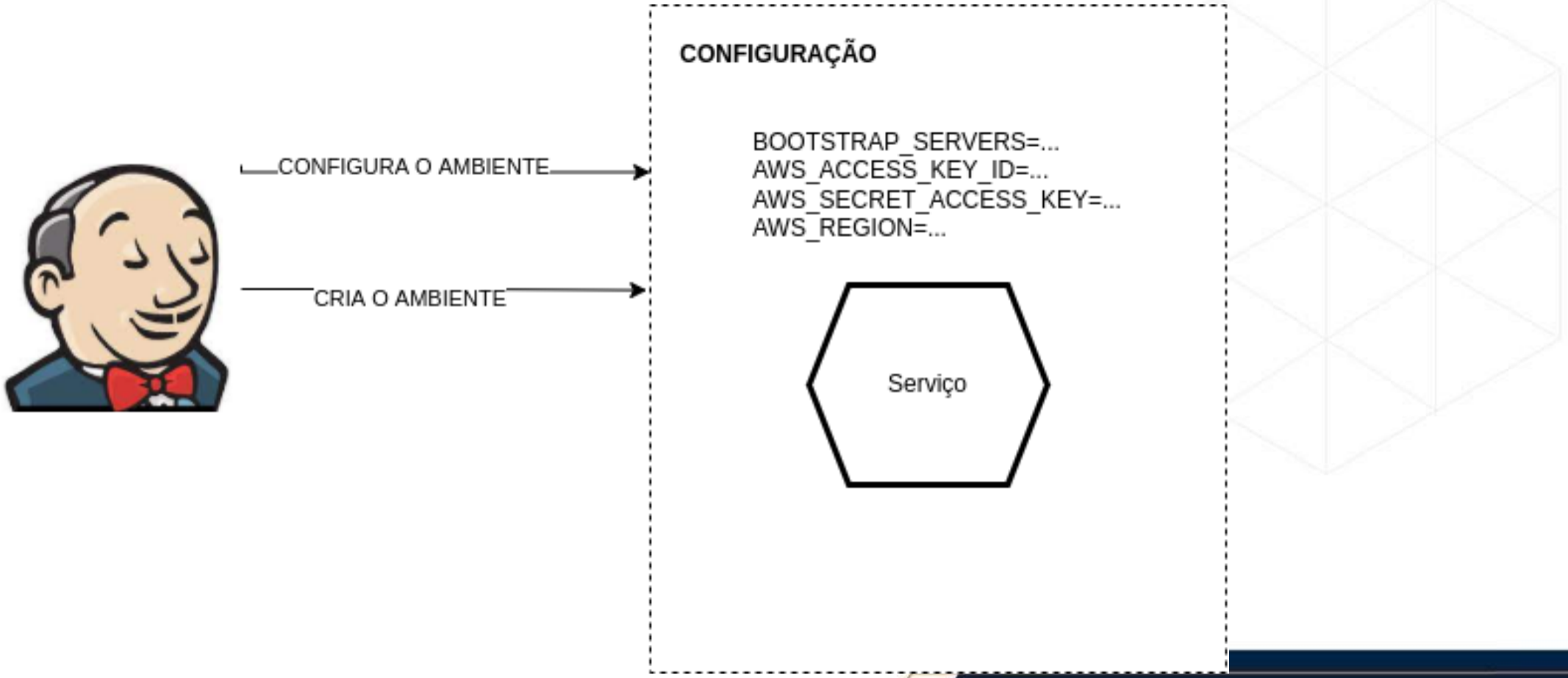
Configurações

- Push Model
- Pull Model

Push Model

- O responsável pelo deploy passa a configuração para a instancia do serviço
  - Variáveis de ambiente
  - Arquivo de configuração
  - Argumentos de linha de comando
- Deve ser especificado a maneira que essas configurações são repassadas ao *deploy*

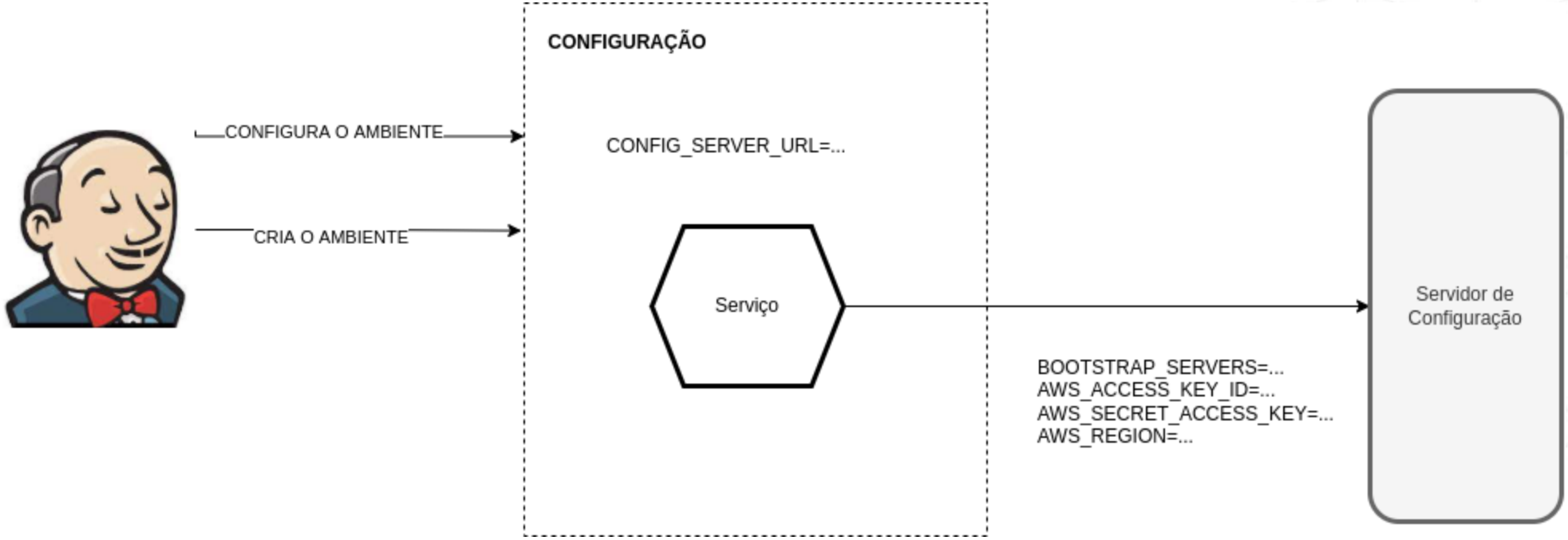
Push Model


CONFIGURAÇÃO
BOOTSTRAP_SERVERS=...
AWS_ACCESS_KEY_ID=...
AWS_SECRET_ACCESS_KEY=...
AWS_REGION=...
CONFIGURA O AMBIENTE
CRIA O AMBIENTE
Serviço

Pull Model

- A instância do serviço lê as configurações de um servidor
- Fornece configuração centralizada
- Pode ser armazenada em locais como *Git*, *NoSQL* ou em servidores de configuração

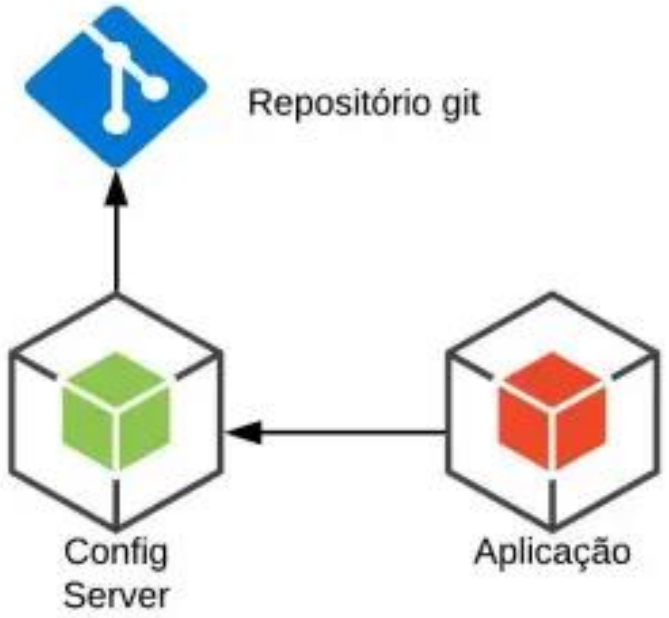
Pull Model


CONFIGURAÇÃO
CONFIGURA O AMBIENTE
CONFIG_SERVER_URL=...
CRIA O AMBIENTE
Serviço
Servidor de Configuração
BOOTSTRAP_SERVERS=...
AWS_ACCESS_KEY_ID=...
AWS_SECRET_ACCESS_KEY=...
AWS_REGION=...

Spring Cloud Config

- O *client* recupera configurações do servidor e injeta no *ApplicationContext*
- Fornece configuração centralizada
- Descriptação de dados sensíveis
- Reconfiguração dinâmica
- https://cloud.spring.io/spring-cloud-config/reference/html/

Spring Cloud Config



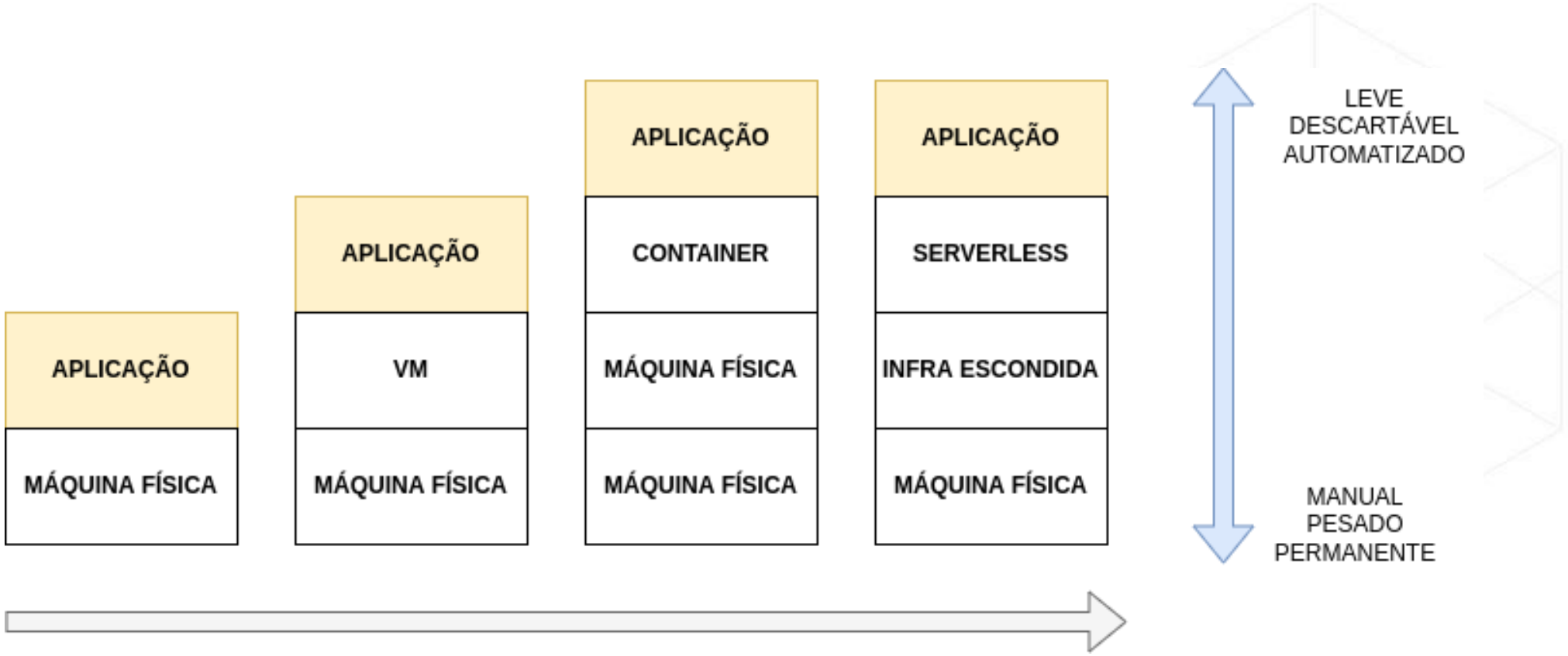
Entregando Microsserviços

7COMm
Serviços e Soluções em TI

Deploy

- Anos 90
  - Pesados servidores de aplicação
  - Normalmente o entregável era feito junto com instruções para o time de operações
- Meados dos anos 2000
  - Pesados servidores começam a dar espaço a conteineres leves, como *Tomcat* e *Jetty*
  - Máquinas virtuais começaram a substituir máquinas físicas
- Hoje
  - Próprio time é responsável pelo *deploy*
  - Muitas vezes o pipeline de *deploy* é 100% automatizado
  - Conteineres são utilizados sobre as máquinas virtuais dentro de um ambiente cloud
  - *Serverless*

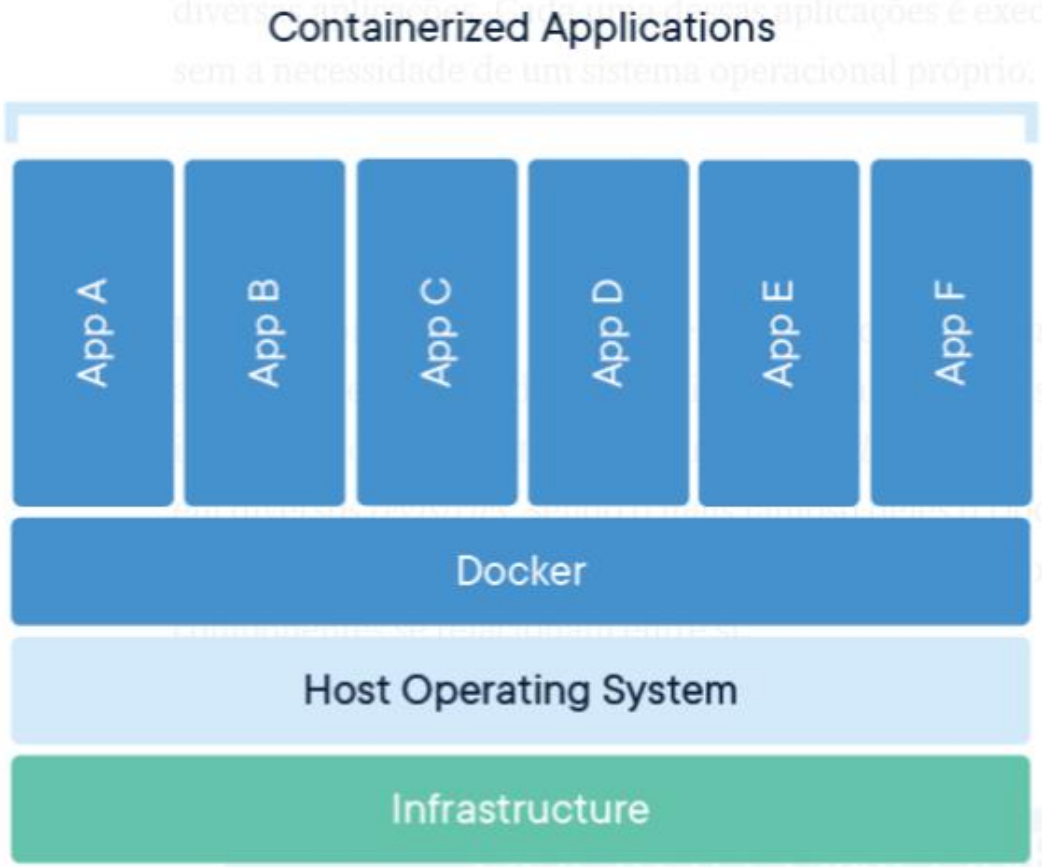
Deploy


APLICAÇÃO
MÁQUINA FÍSICA
APLICAÇÃO
VM
MÁQUINA FÍSICA
APLICAÇÃO
CONTAINER
MÁQUINA FÍSICA
APLICAÇÃO
SERVERLESS
INFRA ESCONDIDA
MÁQUINA FÍSICA
LEVE
DESCARTÁVEL
AUTOMATIZADO
MANUAL
PESADO
PERMANENTE

Deploy através de containers

- http://microservices.io/patterns/deployment/service-per-container.html
- Pode-se especificar os recursos, como memória e cpu
- Encapsula a *stack* de tecnologia
- Instâncias dos serviços são isoladas

Deploy


Containerized Applications
App A
App B
App C
App D
App E
App F
Docker
Host Operating System
Infrastructure

Deploy

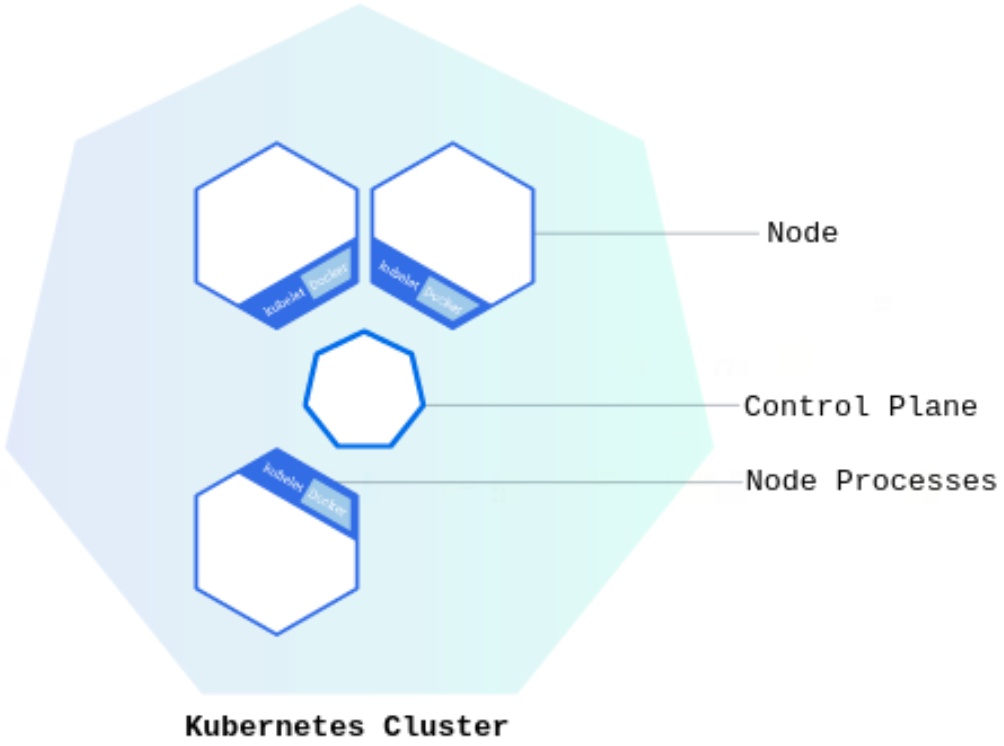
Kubernetes

- Orquestrador de contêineres *docker*
- Provê uma maneira de manter o número desejado de instancias de cada serviço
- Trata um conjunto de máquinas com *docker* como um *pool* de recursos
- https://kubernetes.io/

Kubernetes

- Gerenciamento de Recursos
  - Trata um cluster de máquinas como um pool de CPU, memória e armazenamento
- Scheduling
  - Escolhe a máquina que vai rodar nosso container
  - Trabalha com o conceito de afinidade e anti-afinidade
- Gerenciamento de serviços
  - Assegura que o número desejado de instancias saudáveis estão rodando ao mesmo tempo
  - Faz o balanceamento de carga
  - Faz o devido *rollback* caso seja necessário

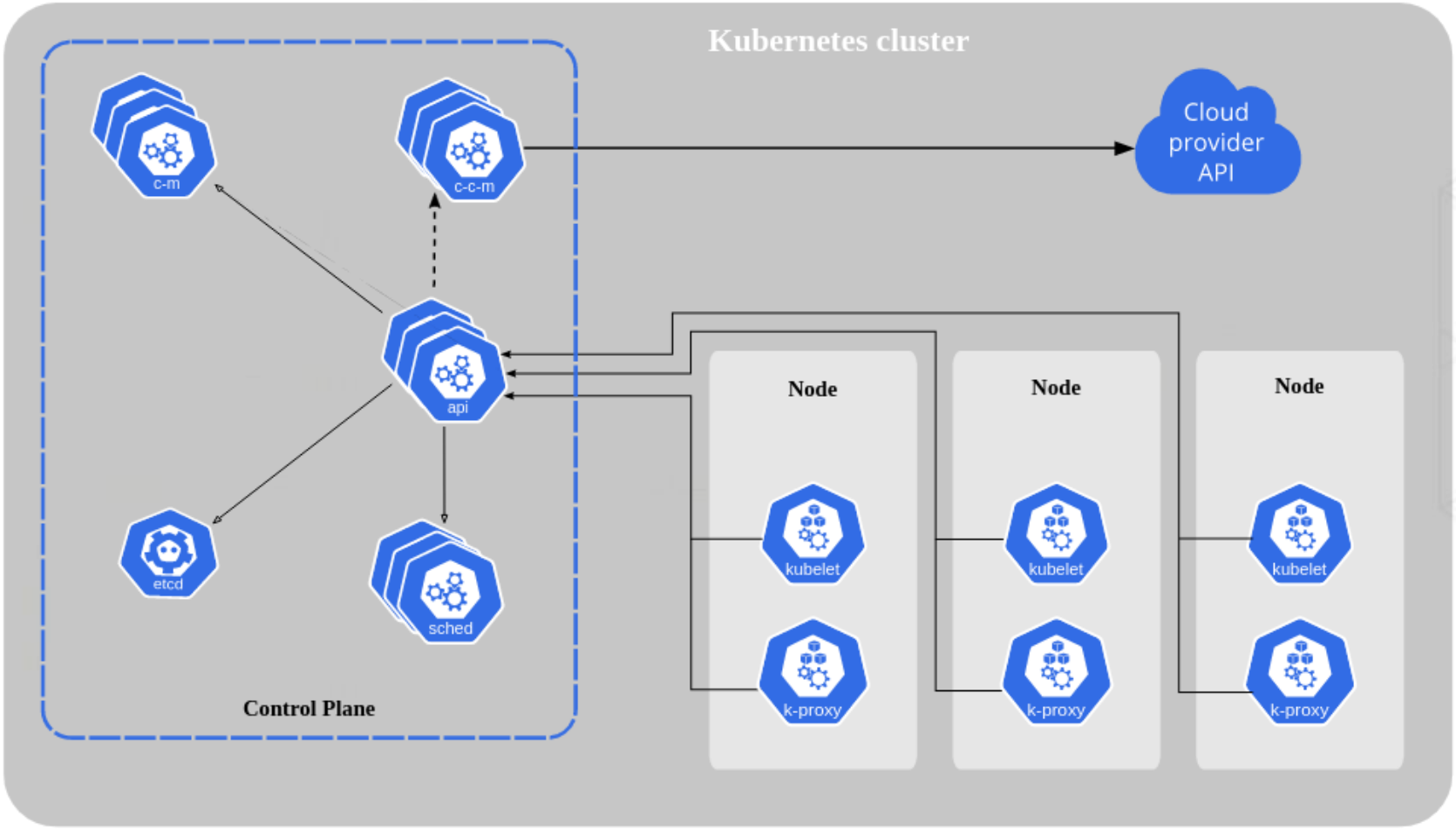
Kubernetes


kubelet
Docker
kubelet
Docker
Node
Control Plane
kubelet
Docker
Node Processes
Kubernetes Cluster

Kubernetes - Arquitetura

- Roda em um cluster de máquinas
- Cada máquina é um *master* ou um *worker*
- Um master é responsável por diversos componentes
  - Api Server
  - Etcd
  - Scheduler
  - Controller manager
- Um *node* roda outros componentes
  - Kubelet
  - Kube-proxy
  - Pods

Kubernetes - Arquitetura



Pod

- A unidade básica de *deploy*
- Consiste de um ou mais contêineres que compartilham *IPs* e volumes

Deployment

- Especificação declarativa de um *Pod*
- *Controller* que assegura que o determinado número de instâncias estão rodando ao mesmo tempo
- Suporta versionamentos e *rollbacks*

Deployment

```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: hello
spec:
  template:
    # This is the pod template
    spec:
      containers:
      - name: hello
        image: busybox
        command: ['sh', '-c', 'echo "Hello, Kubernetes!" && sleep 3600']
      restartPolicy: OnFailure
    # The pod template ends here
```

Service

- Provê uma localização estática/estável
- Uma maneira de *service discovery* provido pela infraestrutura
- Normalmente os DNS e endereços IP são fornecidos somente dentro do *kubernetes*

ConfigMap

- Coleção de *chave-valor* que definem as configurações externas para uma ou mais serviços
- Um *pod* pode referenciar um *ConfigMap* para definir suas variáveis de ambiente
- Pode utilizar, também, para criar arquivos de configuração no container
- É possível guardar informações sensíveis em um tipo de *ConfigMap* chamado *Secret*

Kubernetes – Instalando e utilizando

- Como kubernetes são baseados em clusteres, sempre houve uma certa limitação em rodar local
- No entanto, hoje em dia temos algumas facilidades
  - Minikube
  - KinD

Kubernetes – Minikube

- Kubernetes local
- Ambiente para aprendizado e desenvolvimento
- Roda clusters com somente um nó

Kubernetes – KinD

- Roda o kubernetes local através de conteineres docker
- Leve e simples de instalar
- https://kind.sigs.k8s.io/

OBRIGADO!

Centro
Rua Formosa, 367 - 29º andar Centro, São Paulo - SP, 01049-000

Alphaville
Avenida Ipanema, 165 - Conj. 113/114 Alphaville, São Paulo - SP,06472-002

+55 (11) 3358-7700
contact@7comm.com.br

7COMm
Serviços e Soluções em TI