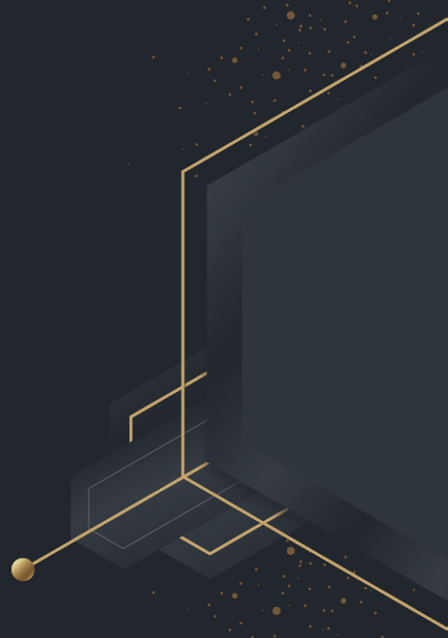
API Gateway com Kong



Daniel Abreu Dantas - contatodanieldantasdev@gmail.com - IP: 172.226.128.45

Uma breve apresentação

Minha carreira tem sido dedicada a ajudar as empresas a tomarem as melhores decisões estratégicas em arquitetura e desenvolvimento de software.

Sou defensor do código limpo, acredito que sempre pode haver uma solução que pode facilitar o dia-a-dia dos usuários.

patrickreinan.com

Objetivos

- Entender o que é um API Gateway
- Demonstrar a utilização de um API Gateway usando o Kong
    - Habilitar autenticação e autorização e outros recursos de segurança
    - Gerar dados e analisar questões de observabilidade
    - Controlar o tráfego da aplicação

Parece pouco, mas é muita coisa...

O que é um API Gateway

Um API Gateway é uma camada de serviço responsável por intermediar o tráfego entre as aplicações externas e o backend, que permite atuar em cima de questões como:

- Controle de Tráfego
- Segurança
- Observabilidade

Funcionalidades do API Gateway

Segurança

- **Autenticação** - OAuth, Basic, API Key
- **Autorização** - controle de acesso por grupos
- **Restrição por IP** - somente IPs autorizados podem acessar
- **Detecção por bot** - identifica padrões comuns de tentativas de ataque e bloqueia
- **Certificados na borda** - o certificado digital pode ficar no Gateway sem a necessidade de ser implementado nas aplicações.

Observabilidade

- **Logging -** Informações relevantes sobre os acessos a aplicação
- **Metrics -** Informações quantitativas sore a saúde do API Gateway e aplicações
    - **Latency -** tempo de resposta da aplicação
    - **Saturation -** quantidade de recurso está sendo utilizado
    - **Errors -** quantidade de erros da aplicação
    - **Traffic -** quantidade de tráfego
- **Tracing -** Inicio do rastreio das requisições que passam pelo proxy.

Controle de Tráfego

- **Throttling -** Limita o número de requisições que pode ser feito por um cliente
- **Routing -** Reescrita da rota para o backend
- **Caching -** Cache de requisições no API Gateway para entregas mais rápidas e reduzir chamadas ao backend

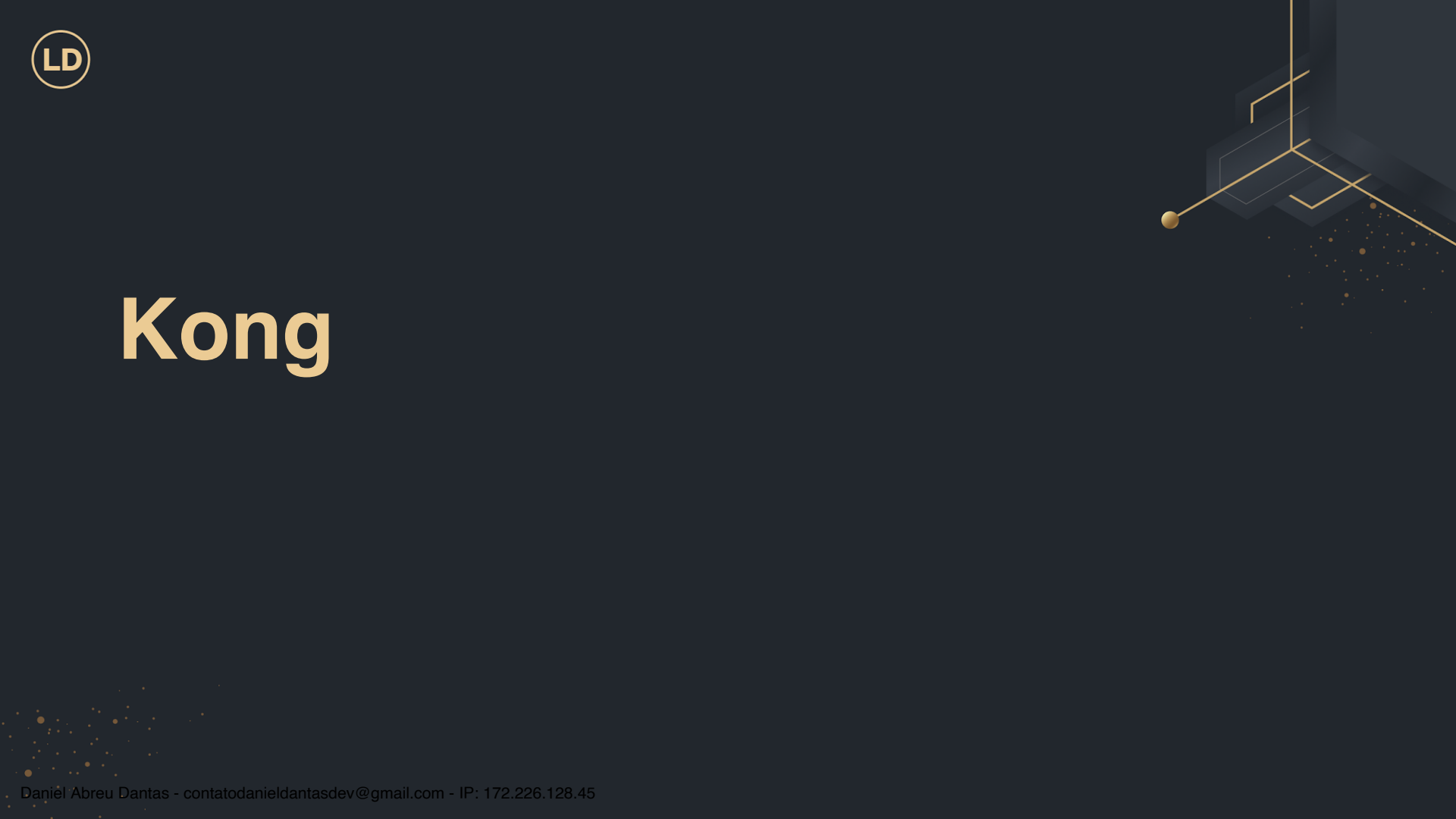
Por que usar um API Gateway?

Benefícios

- Centralizar as requisições as aplicações
- Controlar quem acessa o que
- Limitar o volume de acessos
- Cache de requisições
- Prover dados de observabilidade.
- Desenvolvedores podem focar na regra de negócio

Desafios

- Risco de ser um Single Point of Failure
- Adiciona uma camada ao tráfego de requisições



Kong

O que é o Kong

O Kong é um API Gateway de uma empresa que leva o mesmo nome.

É uma ferramenta escrita em Lua que utilizar o NGINX e é extensível através de plugins.

Características

- Latência abaixo de milissegundos
- +50 mil requisições por segundo
- Agnóstico
- Extensível
- Escalável
- Configuração por interface de usuário, API ou declarativa permitindo automações

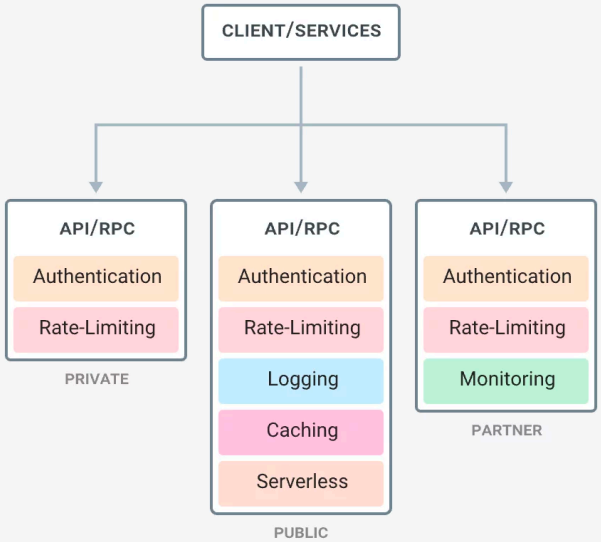
Conceitos básicos

- **Consumer**: Quem vai utilizar os serviços expostos
- **Route**: Como requisitar o serviço
- **Service**: Referência ao serviço a ser requisitado
- **Upstream**: Caminho e Load Balancer do destino do serviço
- **Plugin**: Aplica ações durante a requisição.

Se você sentir falta de um plugin, você pode criar um!

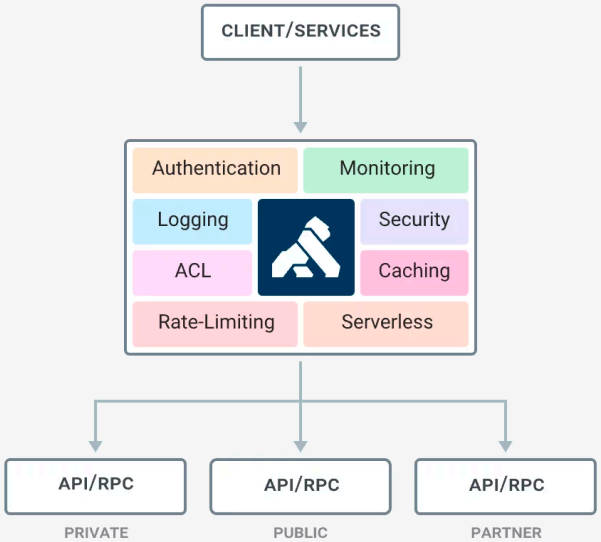
E como seria sem e com o Kong?

The Redundant Old Way



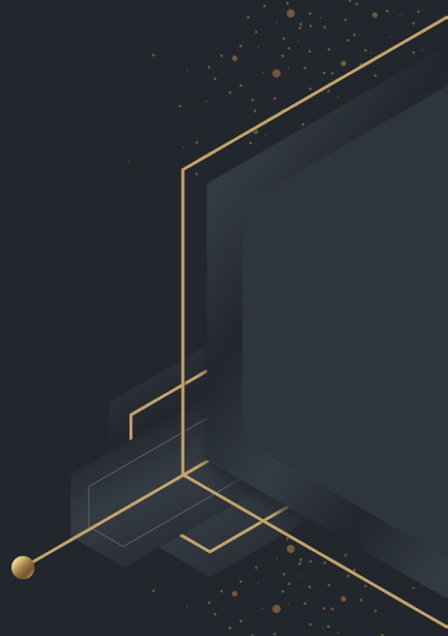
- ✗ Common functionality is duplicated across multiple services
- ✗ Systems tend to be monolithic and hard to maintain
- ✗ Difficult to expand without impacting other services
- ✗ Productivity is inefficient because of system constraints

The Kong Way

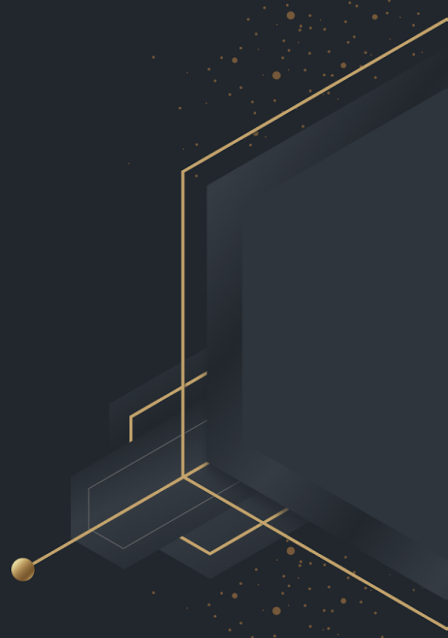


- ✓ Kong orchestrates common functionality
- ✓ Build efficient distributed architectures ready to scale
- ✓ Expand functionality from one place with a simple command
- ✓ Focus on your product and let Kong do the REST

DEMO



Autenticação com Keycloak



Daniel Abreu Dantas - contatodanieldantasdev@gmail.com - IP: 172.226.128.45

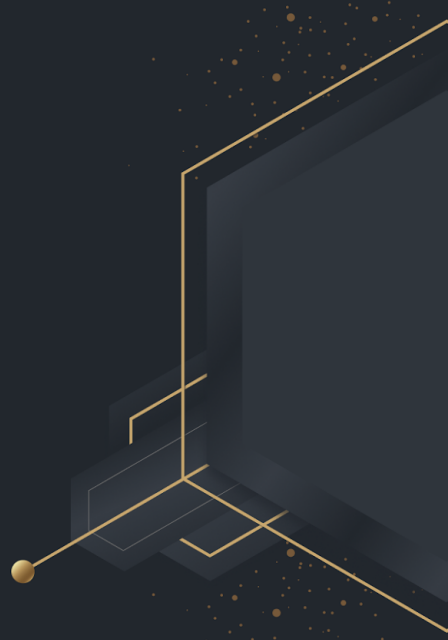
O que é o Keycloak?

Keycloak é uma ferramenta de gestão de identidade e acessos que elimina a necessidade de lidar com armazenamento de senhas, e autenticação de usuários e aplicações.

Conceitos básicos

- **Realm**: área dentro para gerenciar objetos
- **Client**: serviços que usam o Keycloak para gerar autenticação para usuários ou aplicações
- **Users**
- **Roles**: identificam um tipo ou categoria de usuário
- **Groups**: organização de usuários por tipos específicos

Um pouco sobre tokens…



Tokens JWT

O processo de autenticação com o Keycloak é feito através de tokens JWT

Os tokens podem ser gerados via:

- API em nome da aplicação
- API em nome do usuário
- Fluxo OpenID

Tokens JWT

Um token JWT é formado por:

- **Header**: Informações de metadados
- **Payload**: Dados (claims) trafegados dentro do token
- **Signature**: Assinatura para validar integridade do token

Exemplo de JWT Token

eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJ0NnJjTkNYQTVtdWpRMWh1LXRueS1KV0pXc0JXd1IwS29wYzVaUGpCdmNrIn0.eyJleHAiOjE2NTUzMzkxNjUsImlhdCI6MTY1NTMzOTEwNSwianRpIjoiYjRkNjZkZmItNmE0Yi00MDZmLTg0OTAtMjY4NjUwZGZkZGFlIiwiaXNzIjoiaHR0cDovL2xvY2FsaG9zdDo4MDgwL3JlYWxtcy9tYXN0ZXIiLCJzdWIiOiJkZTk1ZjQ3ZS1mN2JhLTQxMmEtOGVjZS0zMzFjYjRmMWY1MTgiLCJ0eXAiOiJCZWFyZXIiLCJhenAiOiJhZG1pbi1jbGkiLCJhY3IiOiIxIiwic2NvcGUiOiJvcGVuaWQgZW1haWwgcHJvZmlsZSIsImNsaWVudEIkIjoiYWRtaW4tY2xpIiwiZW1haWxfdmVyaWZpZWQiOmZhbHNlLCJjbGllbnRIb3N0IjoiMTcyLjI3LjAuMSIsInByZWZlcnJlZF91c2VybmFtZSI6InNlcnZpY2UtYWNjb3VudC1hZG1pbi1jbGkiLCJjbGllbnRBZGRyZXNzIjoiMTcyLjI3LjAuMSJ9.VHhf8VoQWDnFvqocP2mpTVkIQ-jTQlhhhuaBt3d5cWJRc6y67yFnTe9TCKwDEPoxlpKvTqUNO4L4kQ1zJvEVtM1cc8OrL7jhvgwVEQtY2mXggcIYhZJ1yigwdTB-fSTEoqiN-rs304n81fXGWFaSILkLM5JC1oLYviXYsOzV41Bt7H8UoQCxchRJ0X0JzXiZoMEGhMMqebG4WDc0OLcMHcKGYOZwP4LxkZ2rCceH0JWypFbmFO7oyfQs37pYt_m5RWTUduLZ8Gu0djfWJzkzUTFahtRacdyIXsL0ADwDfcgAFN-mWk_auojQwCS0f6JMDKHddqnWBtZ4Y0gupWi3bQ

Tokens JWT

Quando uma aplicação recebe um token ela deve validar a assinatura e outros parâmetros do token para efetuar a autenticação.

As informações de autorização devem ser lidas das claims.

Tokens JWT

As claims comuns de validação do token são:

aud: Identifica quem deve receber o token
sub: identifica o usuário
iss: identifica quem emitiu o token

DEMO