Partiklo

MVP

Arquitetura inicial

De acordo com a visão do Partiklo:

“ZERO configuração! O Partiklo pode ser configurado “on-the-fly”, não necessitando de “properties” e nem de “xml” ou “yaml” para entrar no ar. Você pode configurá-lo via API REST ou Web e salvar toda a configuração para uso posterior.”

Traduzindo isto para uma arquitetura, deve ser possível subir um servidor Partiklo e esquecer! Se for necessário, outros servidores Pariklo podem ser ativados, se encontrarem e trabalharem em conjunto.

Tal arquitetura se faz compatível com mecanismos de controle de microsserviços como o Kubernetes ou o Docker Compose.

Outro aspecto da visão do Partiklo, que afeta sua arquitetura, é:

“O Partiklo se integra com vários orquestradores de serviços, como o Kubernetes, da Google. Seus servidores distribuídos, sem exigência de “master node” podem manter a comunicação de sua empresa funcionando, mesmo em caso de falhas.”

O ponto chave é a inexistência de um “master node”. Em outras palavras, cada nó do Partiklo é um “master” independente. Isto tem um impacto grande na distribuição dos dados entre os vários nós.

Para atender a este princípio, cada instância do Partiklo é totalmente independente, mantendo sua própria “datastore”. Cada instância deve ser capaz de encontrar as outras e comunicá-las sobre alterações. Assim, com o tempo, cada instância tem uma cópia completa dos dados.

O MVP deverá ser extremamente “enxuto” e funcional, evitando YAGNIs (You Ain't Gonna Need It), então, as soluções mais simples possíveis que permitam atender ao documento de visão serão adotadas.

Plataforma

- Linguagem de programação: python 3.6;
- Persistência: pacote “dataset” (https://dataset.readthedocs.io/en/latest/), com gravação no disco local de cada servidor;
- Protocolo: HTTPS com certificado auto-assinado;
- Arquitetura de comunicação: REST;

- Mecanismo de sincronização: Multicast;

Casos de uso arquiteturalmente significativos:

1 – Boot de servidor

1. Verificar datastore. Vazia? Assumir primeira instalação;
2. Ler configuração;
3. Descobrir outros “peers” (Multicast), atualizar tabela de replicação;
4. Disponível.

2 – Registro de novo usuário

1. Receber dados do usuário (nome, e-mail, senha, chave pública);
2. Registrar na datastore local e avisar aos “peers”;

3 – Envio de mensagem

1. Validar assinatura;
2. Validar destinatário; Não encontrado? Verificar com “peers”;
3. Salvar mensagem. Enviar para “peers”;

4 – Recebimento de mensagem

1. Validar usuário;
2. Obter mensagens (próprias e grupo). Mais de 15 minutos? Perguntar a “peers” por novas mensagens;
3. Marcar recebimento. Avisar “peers” do recebimento.

5 – Novo “peer”

1. Receber aviso de novo “peer”;
2. Responder com seu endereço IP.

6 – Consistência eventual

1. Enviar todas as atualizações depois de um determinado “timestamp”.

O MVP permitirá a criação de grupos e estes serão tratados da mesma forma que os outros casos acima.

Para enviar uma mensagem, o usuário precisa assiná-la com sua chave pública. Inicialmente, as mensagens não serão encriptadas.

Consistência eventual

Deverá ser criado um componente de consistência eventual, marcando o “timestamp” de todos os “peers” e atualizando os dados. Este componente deverá ser embutido no próprio servidor Partiklo e executado em intervalos regulares.