DevOps

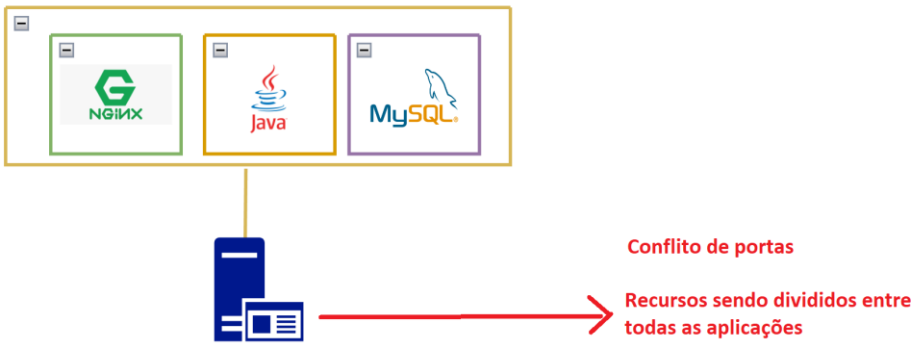
Uma **definição rápida** sobre **DevOps** é que **ela** é uma cultura que **visa unificar práticas de desenvolvimento com práticas de operação**, em um português claro, o **dev** tem que saber como fazer um deploy e monitoramento decente da sua aplicação *(práticas de infra).*

Assim como o mundo de **desenvolvimento** tem diversas práticas *(SOLID, TDD, TESTES etc...)*, o mundo de **operações** também tem *(CI/CD, PROVISIONAMENTO, CONTEINERIZAÇÃO, etc..)* e todas elas merecem resenhas próprias.

O que é **docker** e como ele **se encaixa na cultura** de **DevOps**

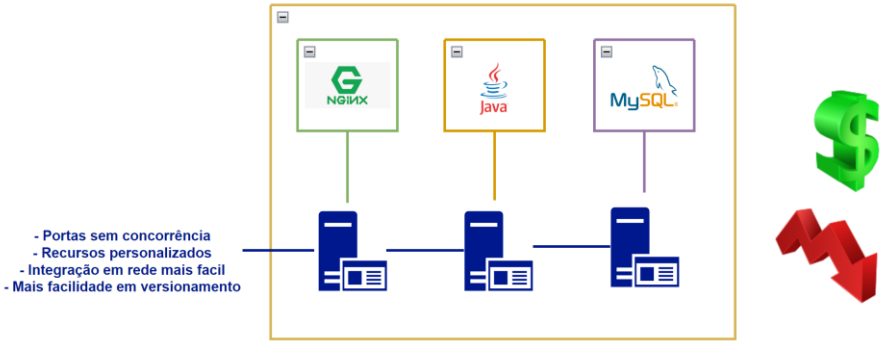
Pra entender a importância do **Docker**, primeiro é importante saber o cenário que ele se propõe a resolver. Imaginando então que eu tenha um **sistema** que pra funcionar precise de uma ***aplicação Java***, um ***banco de dados*** e um ***balanceador de carga.***

Eu poderia tentar fazer esse **sistema funcionar** em uma **única máquina**, mas eu teria diversos problemas, como por exemplo: *portas conflitantes, complexidade no controle de versões, dificuldade pra escalar, desperdício de recurso* e etc...:



Uma **possível solução** seria ter **uma máquina dedicada para cada aplicação do sistema**, isso faria com que cada **máquina fosse personalizada para atender cada tipo de aplicação**, além de ter portas únicas para facilitar a integração em uma rede.

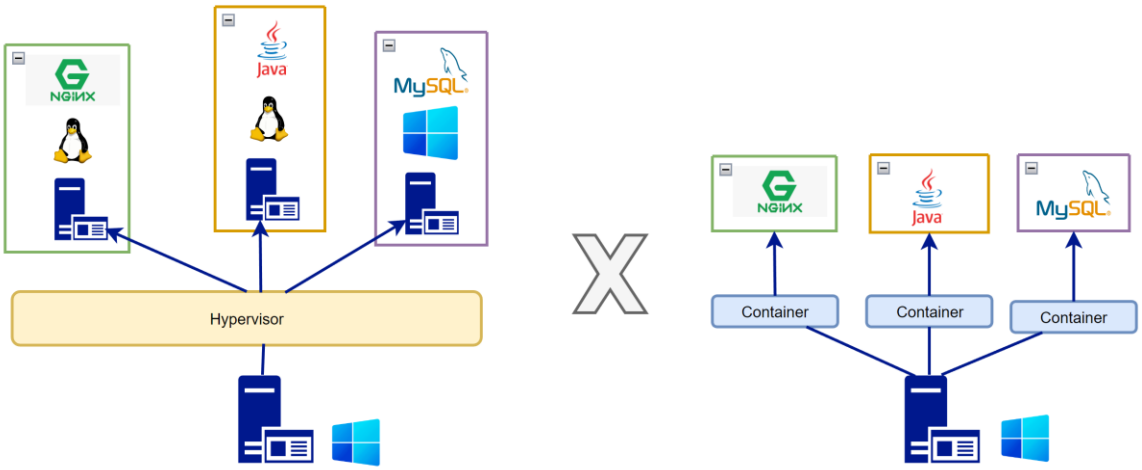
Só que não, se você precisar comprar uma máquina toda vez que tiver um componente no sistema **tu vai falir:**



Virtualização x Conteinerização

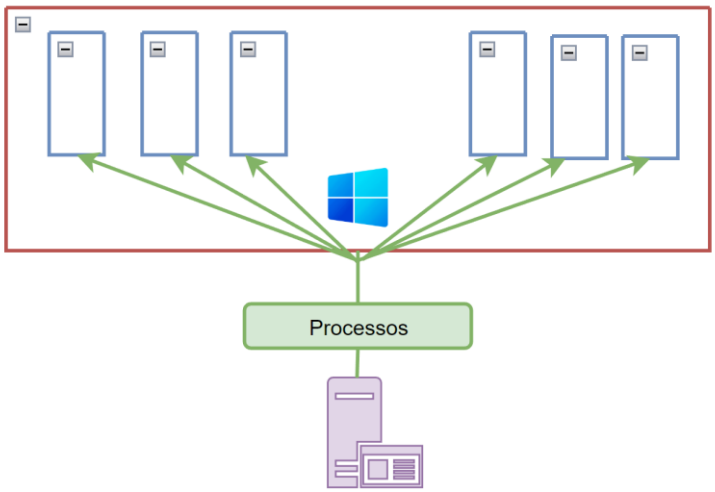
Então uma solução bem difundida e utilizada é a virtualização, que é simplesmente você emular outras máquinas em uma só. Então você pega uma **máquina host**, bota uma camada de ***hypervisor*** (emulador) e cria **outras máquinas isoladas**, com seus ***próprios SO, Dependências e aplicações.***

Só que uma dúvida ficou nesse tipo de **solução**: Eu realmente preciso de um *hypervisor*? Com um **SO e tudo??** Aí surgiu o **container**, que tem como proposta **eliminar essa camada de** *hypervisor* **e rodar o ambiente TAMBÉM de forma isolada** no **container** sem precisar emular uma nova máquina**:**

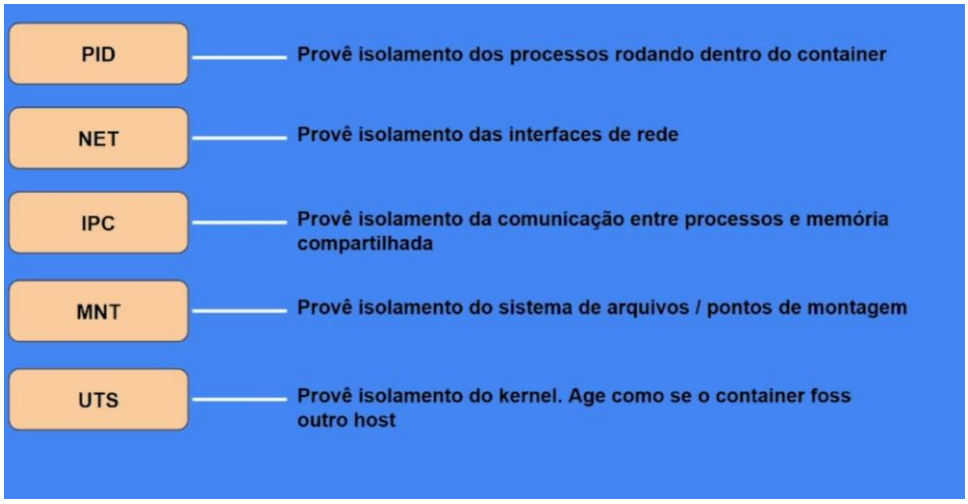


Como um container funciona?

Na verdade, porque um **container** é melhor *(teoricamente)* que uma **máquina virtual**? Primeiro que o **container roda como um processo**, ou seja, ele se comporta na sua **máquina host** como um processo normal *(igual tu rodar um outro programa)*, então eu não preciso de uma camada de **hypervisor** e nem de um SO a parte, isso já **diminui o custo de recursos** ao usar **container:**



Mas como um **container** pode ser isolado se ele está rodando como um processo do SO??? Para **garantir o isolamento dos containers** *(tanto deles mesmos quanto do SO host)* o docker possuí um conceito chamado **Namespaces:**



Dentre esses **Namespaces** o ultimo **"UTS"** é um dos mais importantes, já que essa camada de isolamento faz com que o **container** tenha um "pedaço" do kernel do linux *(um pedaço específico do SO)* que faz com que não precisemos instalar um **SO** completo no **container.**