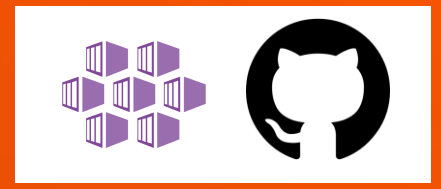
12 Factor Apps em AKS



Por que AKS é o ambiente ideal para 12 factor apps

avanade

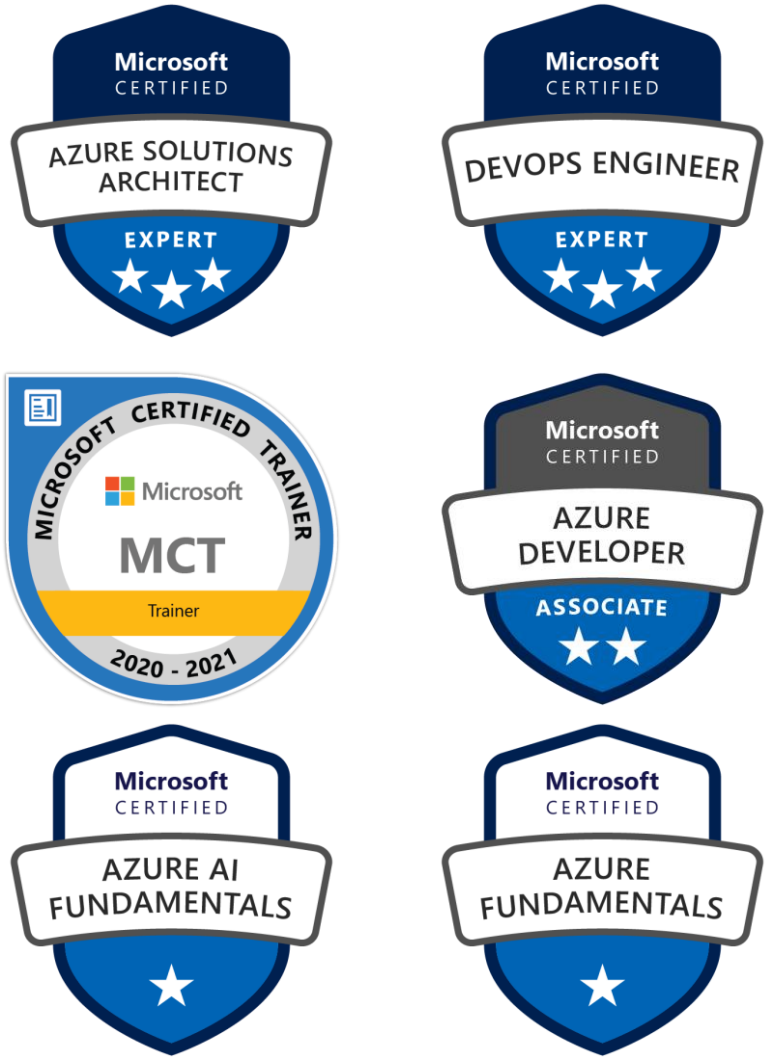
Sobre mim

Diretor – Enterprise Tech Arch Leader
https://www.linkedin.com/in/aracz/
andre.racz@avanade.com

- + de 20 anos de carreira
- + de 15 anos em Arquitetura
- 3 anos de Avanade

Projetos em grandes empresas dos segmentos: Bancos / Seguradoras / Setor Público / Industrias / Farmacêutico / Hospitais / Energia

Tecnologias: Java, .Net, NodeJS, Kubernetes, Cloud, Devops, IA



avanade

O que são 12 factor apps?

- Criadas pelo time da Heroku em 2011: https://12factor.net
- Independente de linguagem e plataforma
- Princípios gerais para aplicações que:
  - Possuem configuração declarativa
  - Possuem contrato claro com o ambiente operacional
  - Adequadas para deployment em nuvem
  - Resilientes
  - Fáceis de escalar

avanade

Azure Kubernetes Service (AKS)

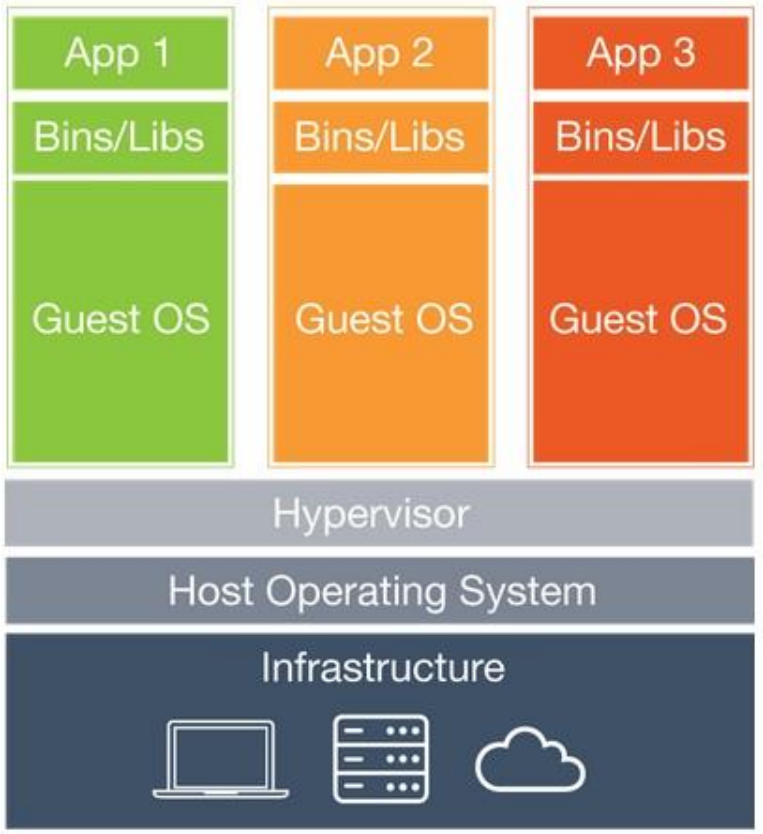
Kubernetes made easy

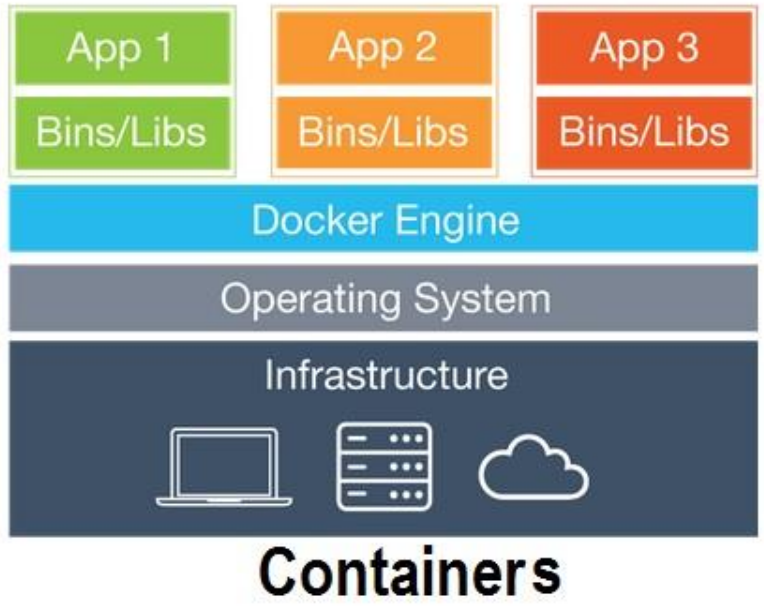
Kubernetes é uma plataforma open-source para automatizar o deployment, escalação e operações de containers de aplicação entre um cluster de hosts. Você é responsável pela Arquitetura, construção e gestão.

AKS é a plataforma de Kubernetes gerenciada da Microsoft; A Azure é responsável pela gestão, escala e operação dos nós de controle e os usuários escalam os nós de trabalho e deployam os workloads.

avanade

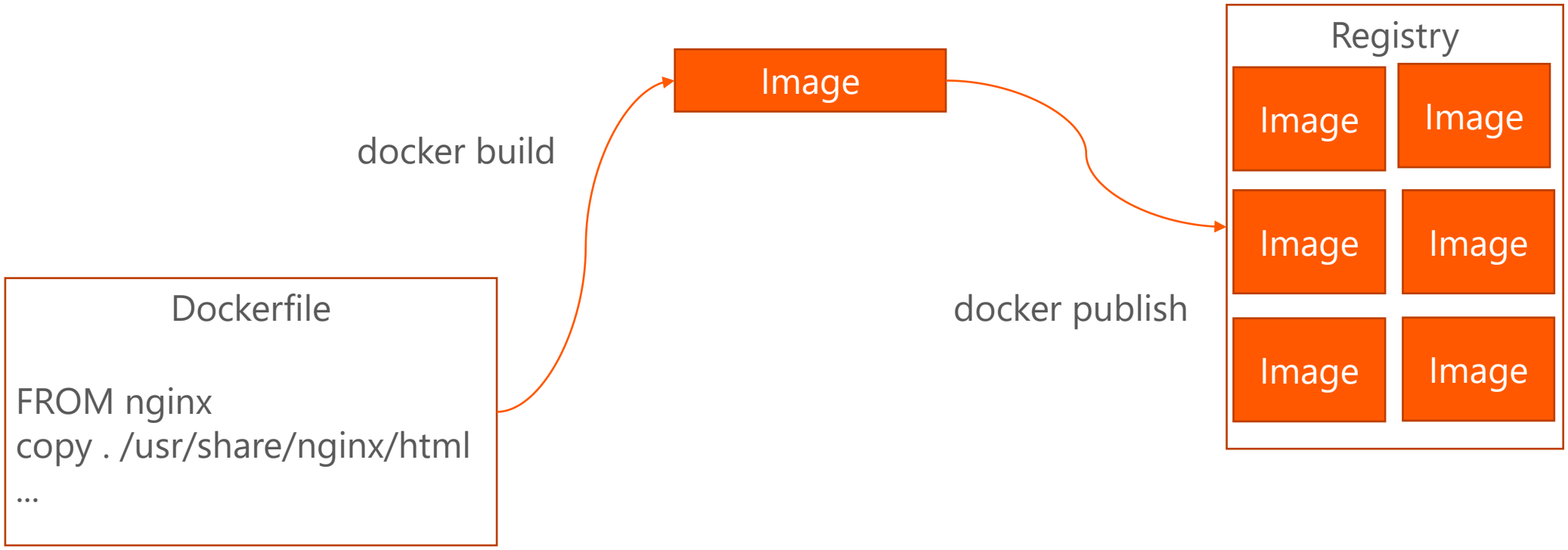
VMs x Containers



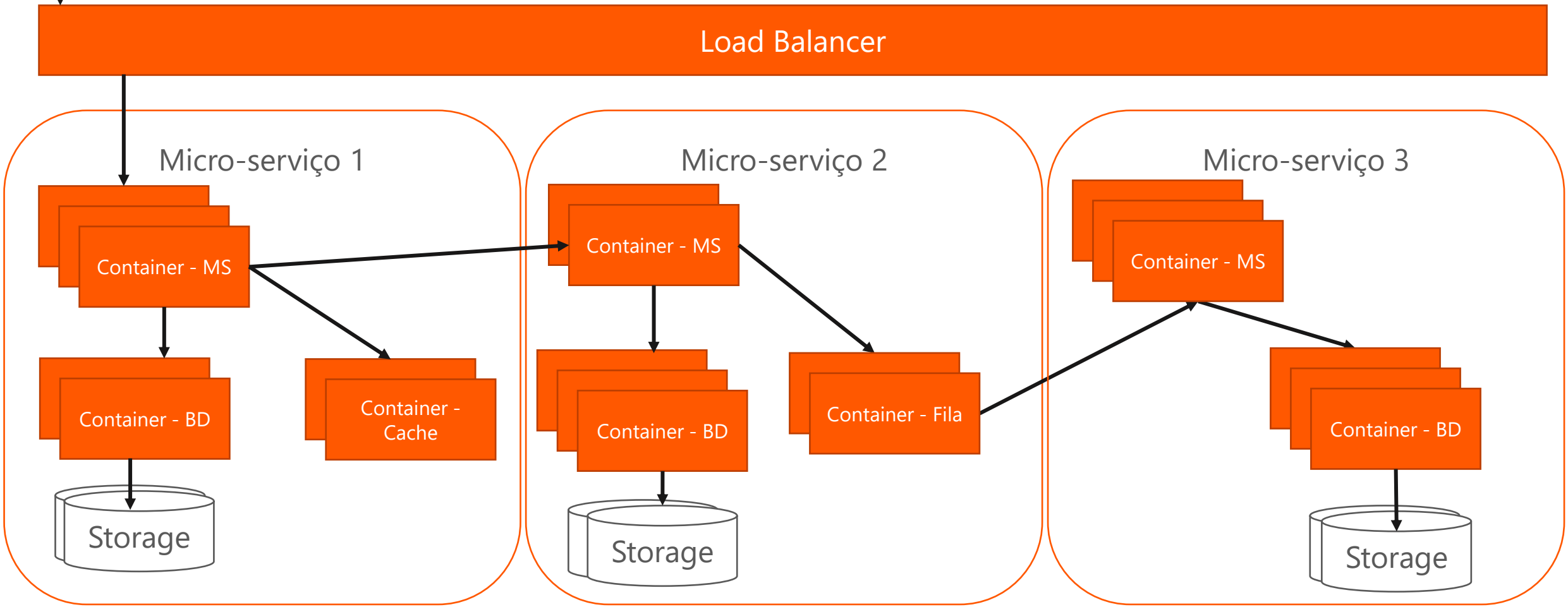


avanade

Docker - Conceitos



Por que Orquestração de Containers?



avanade

Por que Orquestração de Containers?

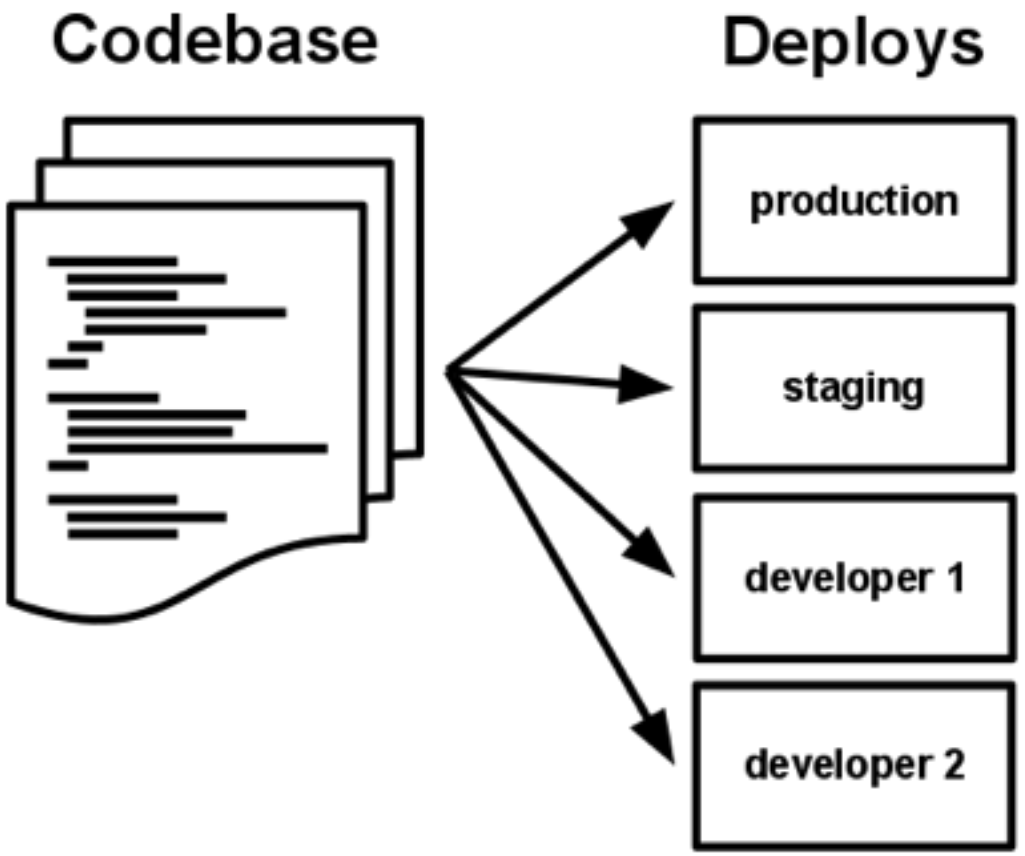
Para que seus containers dentro de sua infra estrutura não vire essa bagunça, se faz necessário o uso de orquestradores de containers. Os orquestradores otimizam a organização e o gerenciamento de suas aplicações distribuídas em containers. O orquestrador mais famoso é o Kubernetes

avanade

12 factor

I - Codebase

- Um repositório de Código, vários deployments
- Se tem múltiplas bases de código, não é uma aplicação, é um sistema distribuído

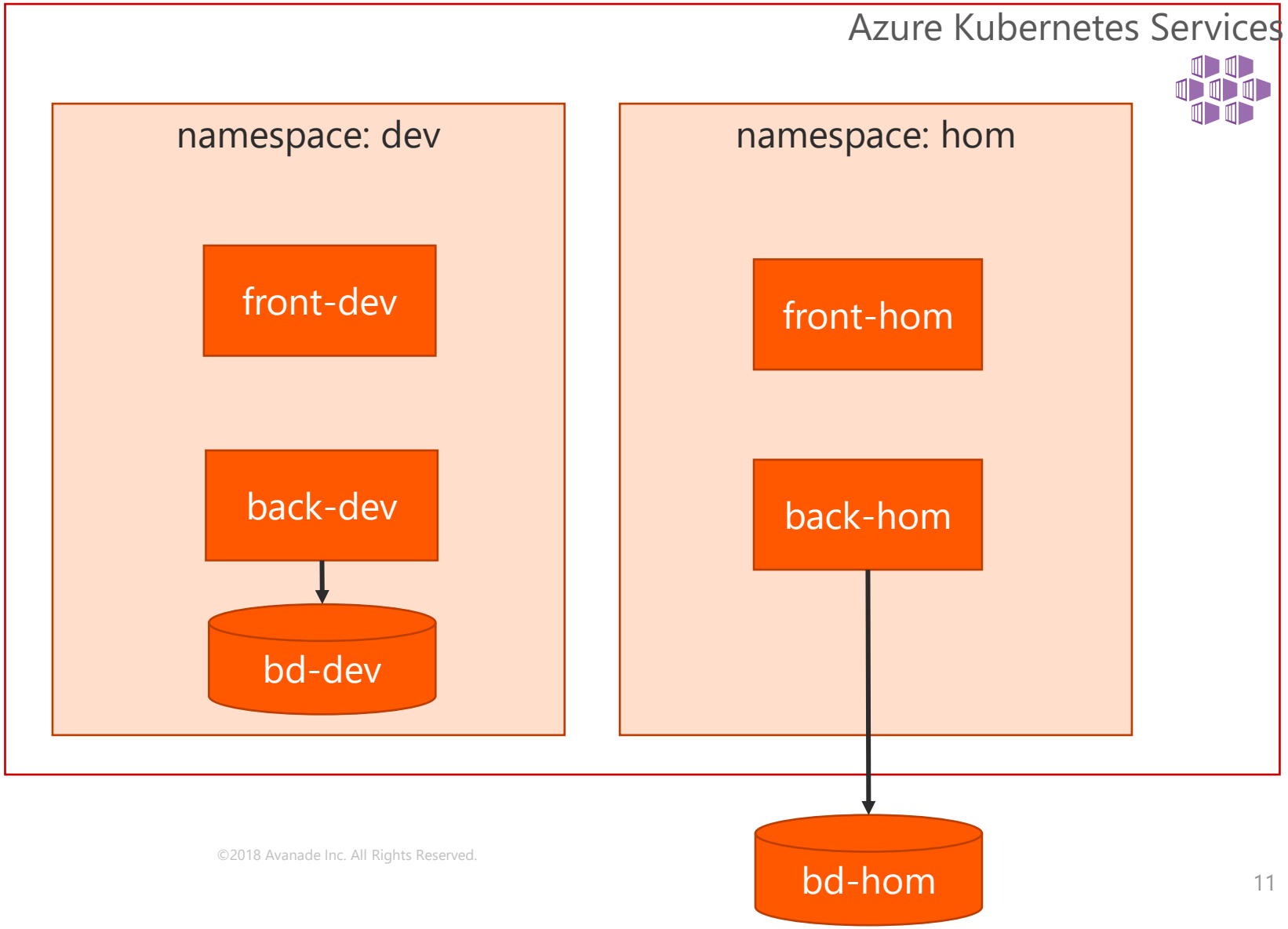


avanade

I – Codebase

REPO 1 – Front-end

REPO 2 – Back-end



avanade

II - Dependencies

- Declare e isole todas as dependências
- A maioria das linguagens suporta package managers:
  - Nuget, NPM, Maven, Rubygems
- Não conte com a existência de ferramentas de sistema
- Docker é ótimo para isso!

avanade

Docker e dependencias

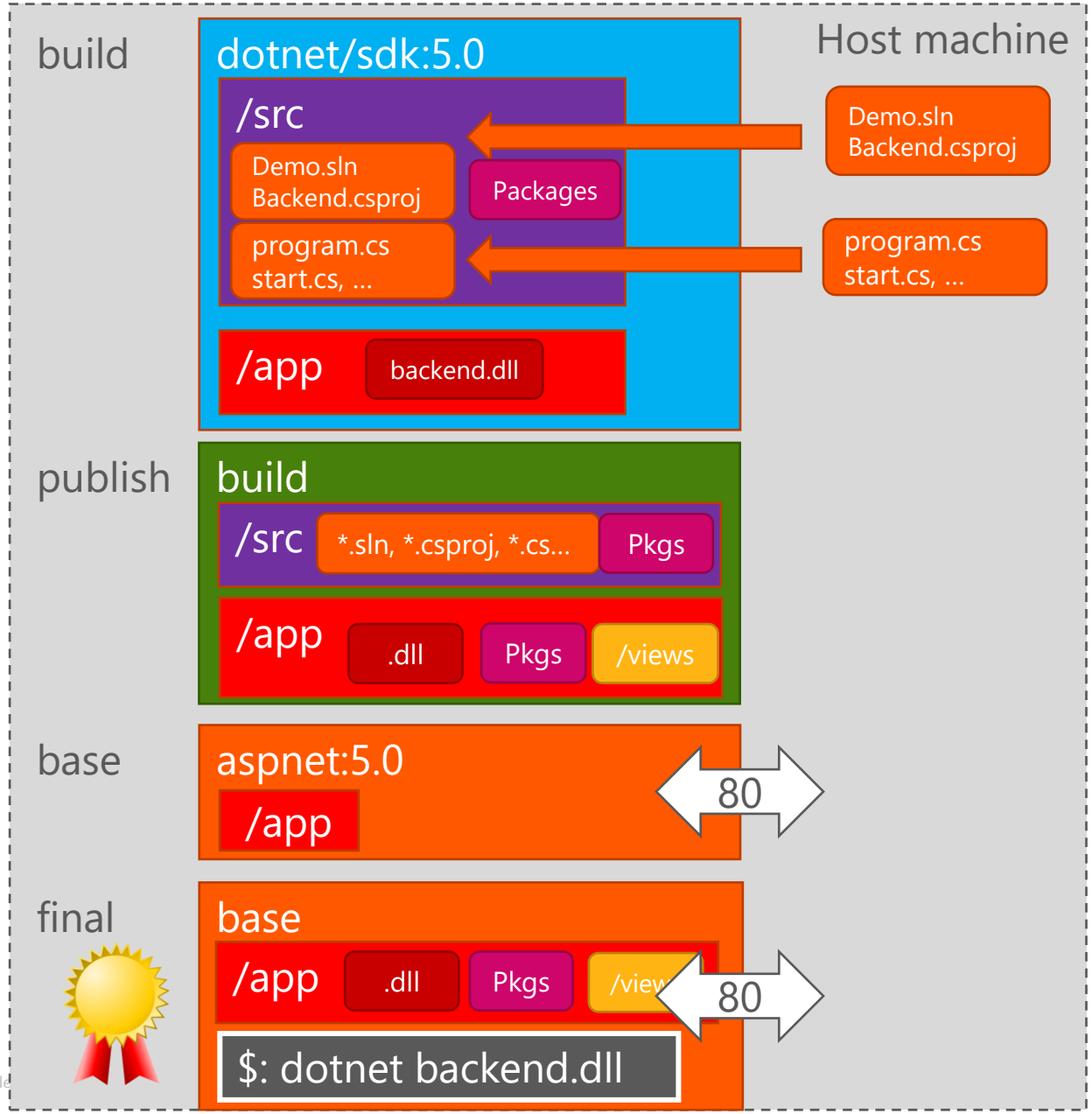
```
FROM mcr.microsoft.com/dotnet/sdk:5.0 AS
build
WORKDIR /src
COPY *.sln ./
COPY backend/backend.csproj backend/
RUN dotnet restore

COPY . .
WORKDIR /src/backend
RUN dotnet build -c Release -o /app

FROM build AS publish
RUN dotnet publish -c Release -o /app

FROM mcr.microsoft.com/dotnet/aspnet:5.0 AS
base
WORKDIR /app
EXPOSE 80

FROM base AS final
WORKDIR /app
COPY --from=publish /app .
ENTRYPOINT ["dotnet", "backend.dll"]
```



avanade

III - Config

- Configuração é tudo que muda entre um ambiente e outro
- Não use constantes para armazenar ela no código
- Configuração deve ser provida para a aplicação como variáveis de ambientes
- Não agrupe elas em arquivos por ambiente (produção, teste, desenvolvimento)

avanade

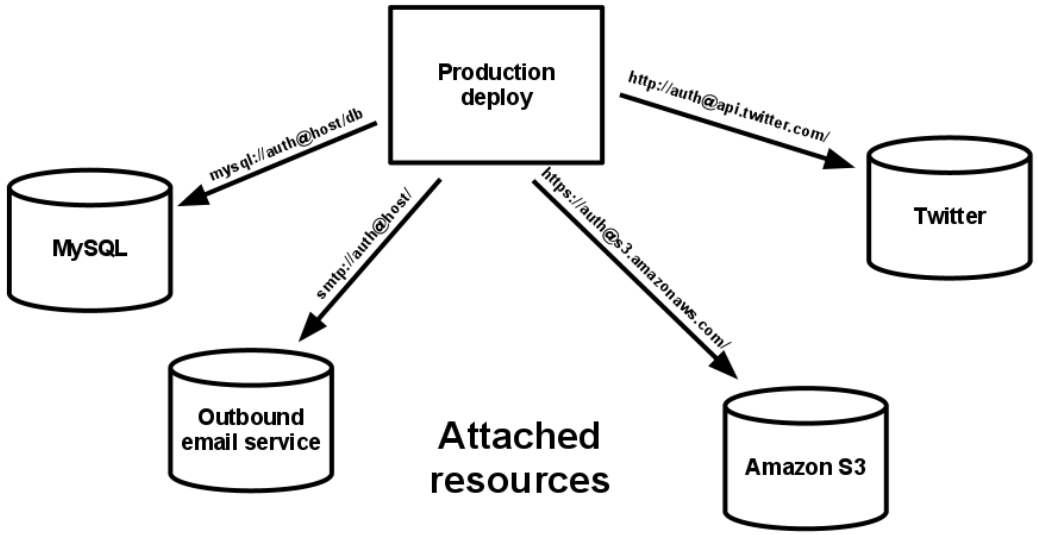
III – Config em Kubernetes

```
containers:
  - name: container1
    image: imagem:version
    envFrom:
    - configMapRef:
        name: myConfigMap
    - secretRef:
        name: mySecret
    env:
    - name: ENV_VAR1
      value: valor
    - name: ENV_VAR2
      value: valor2
```

avanade

IV – Backing Services

- Backing Service é qualquer serviço externo consumido pela aplicação:
  - Banco de dados, cache, filas, outros serviços
- Não faça distinção entre serviços internos e externos.
- Um serviço específico deve ser fácil de substituir

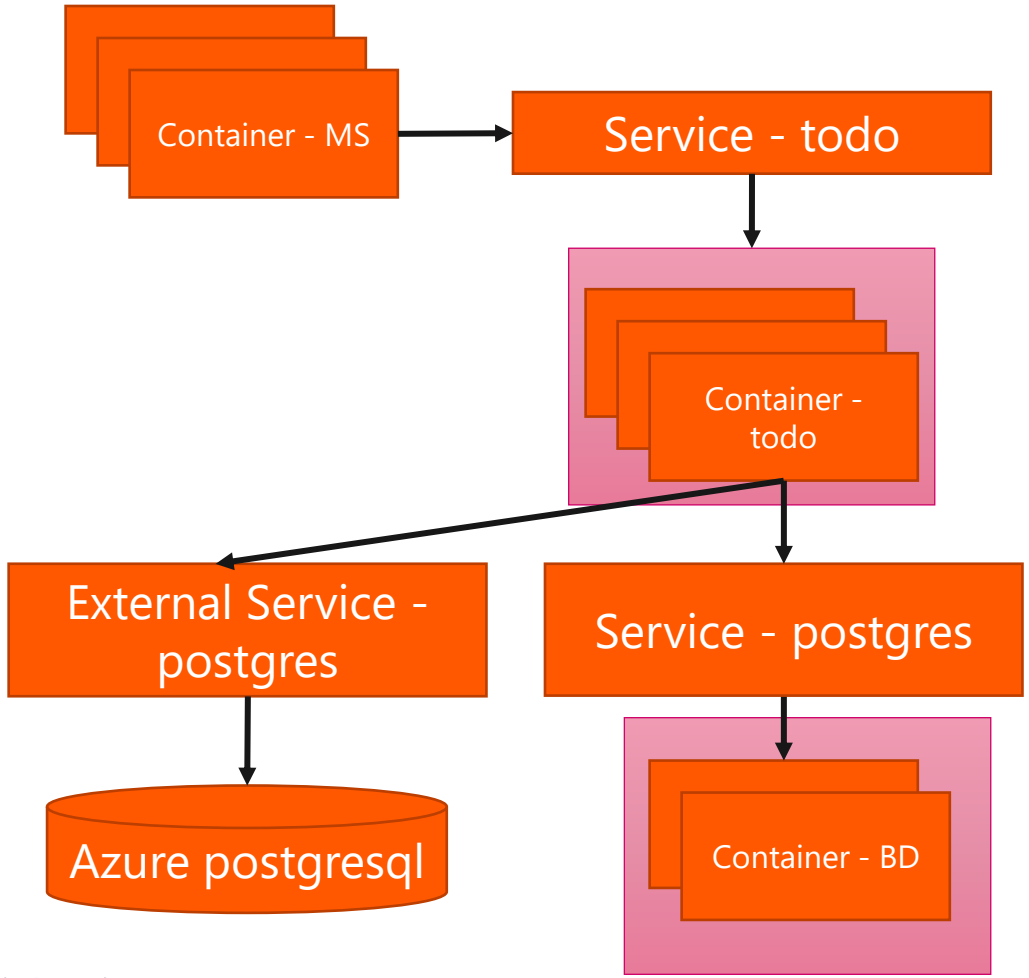


avanade

IV – Backing Services

- Kubernetes representa os endpoints para conectar diferentes workloads, com o tipo de objeto Service.

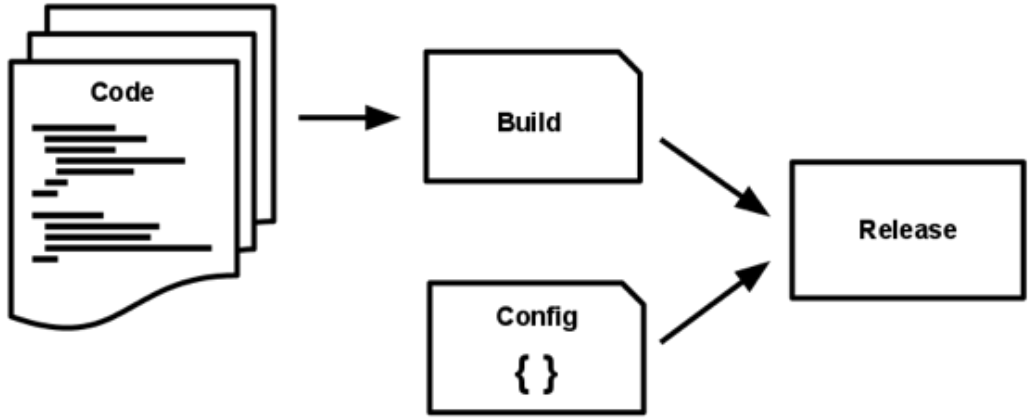
- Os seguintes tipos de Service estão disponíveis:
  - ClusterIp – Interno ao cluster
  - LoadBalancer – Exposto para fora do cluster
  - NodePort – expõe o serviço em uma porta fixa no nó
  - ExternalName – expõe um serviço pelo nome DNS externo



avanade

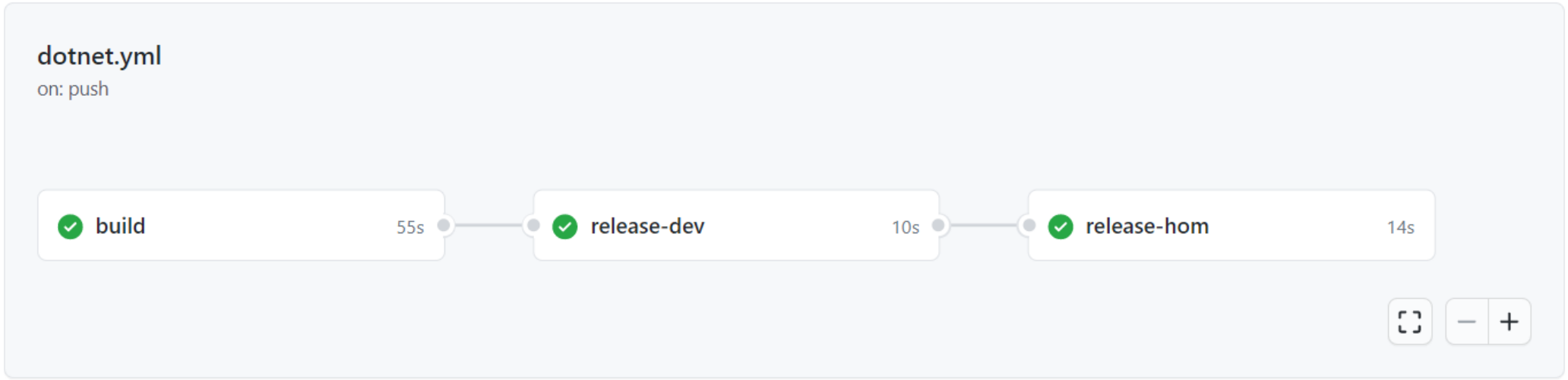
V – Build, release and run Services

- 3 passos no ciclo de vida da aplicação:
  - Build: transormar o Código e executável
  - Release: aplicar a configuração para um build e ambiente
  - Run: iniciar os processo
- Cada build e release tem que ser unicamente identificado
- Rodar dever ser o mais simples possível



avanade

V – Build, release and run Services



docker build

helm upgrade

helm upgrade

avanade

Helm

- Ferramenta de template para Kubernetes
- Gera “releases” compostos de múltiplos objetos
- Permite atualização do release completo
- Permite utilização de lógicas, variáveis
- Permite customização das variáveis por ambiente

avanade

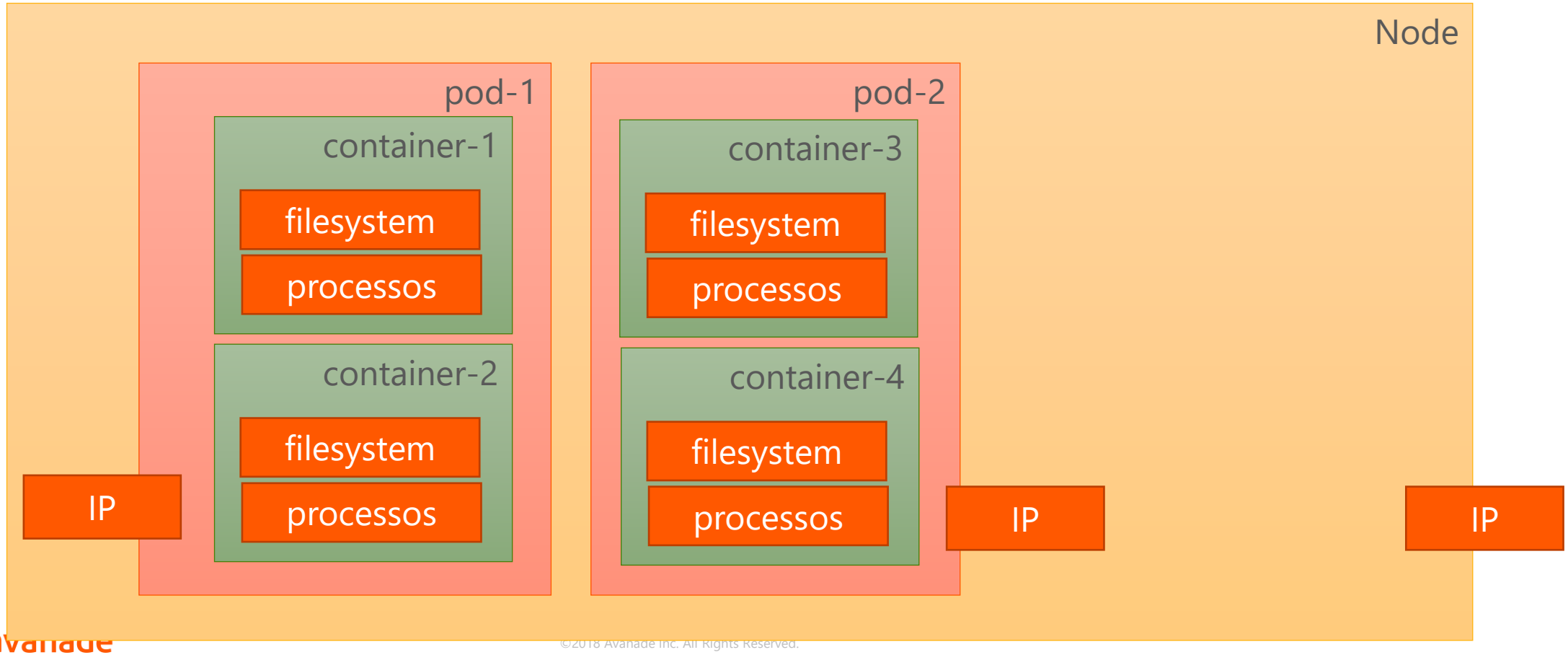
VI - Processes

- A aplicação roda como um ou mais processos.
- Processos não compartilham nada entre si (não tem estado)
- Recursos locais podem ser usados em uma única transação.

avanade

VI - Processes

Modelo de processos do kubernetes



VII – Port binding

- Exponha sua aplicação como uma porta
- Não dependa de um servidor de aplicações ou web externos
- Use servidores embarcados se necessário
- Qualquer protocol pode ser utilizado

avanade

VII – Port binding

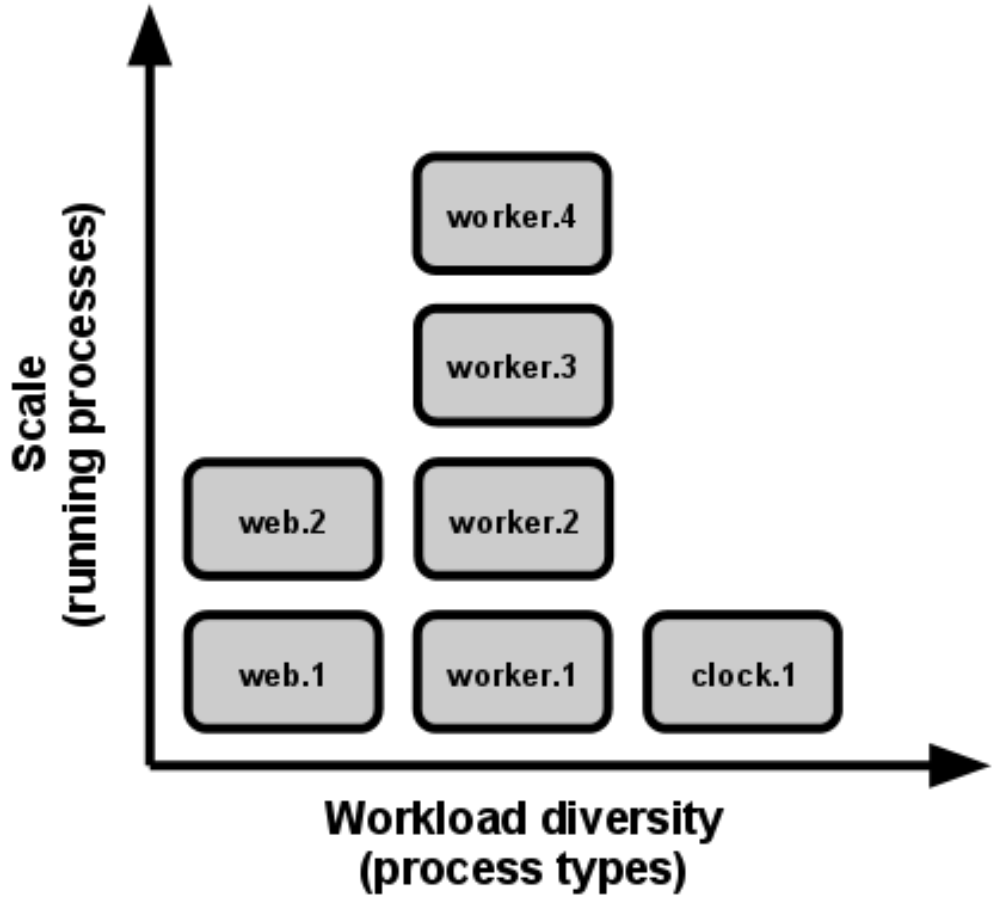
- Docker: EXPOSE 80
- Kubernetes Pod:

```
        ports:
          - name: http
            containerPort: 80
            protocol: TCP
```

avanade

VIII – Concurrency

- Escale pelo modelo de processos (mais processos para aumentar a carga)
- Processos individuais podem ter mais de uma thread
- Processos não devem depender de outros processos
- Confie no ambiente de operação para manter os processos rodando



avanade

VIII – Concurrency

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nomeDeployment
spec:
  replicas: 3
```

```yaml
apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
  name: nomeHpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: nomeDeployment
  minReplicas: 1
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 70
```

avanade

IX - Disposability

- Processos são descartáveis
- Inicie Rápido:
  - Faz com que escalar seja fácil
  - Novas versões entram rápido no ar
- Graceful shutdown:
  - Quando receber o sinal SIGTERM
  - Pare de processar novas requisições
  - Termine o trabalho atual

avanade

IX - Disposability

- Natural no Docker
- Para Kubernetes:
  - Parar de responder nas probes
  - Terminar de processar as requisições atuais

avanade

Kubernetes Probes

- Meio pelo qual o Kubernetes monitora os serviços:
- 3 probes disponíveis:
  - startupProbe: Chamada enquanto o serviço está subindo para saber se já finalizou a inicialização. Após a resposta positiva, para de ser chamada
  - readinessProbe: Chamada para saber se o serviço pode receber novas requisições
  - livenessProbe: Chamada para saber se o serviço está “vivo”

avanade

X – Dev / Prod parity

- Todos os ambientes devem ser similares
- Use os mesmos tipos de “backing services” para todos os ambientes
- Garanta que todas as versões entre ambientes estão equalizadas
- As mesmas pessoas devem desenvolver e rodar os serviços
- Você deve se esforçar para minimizar o tempo entre os deploys de produção

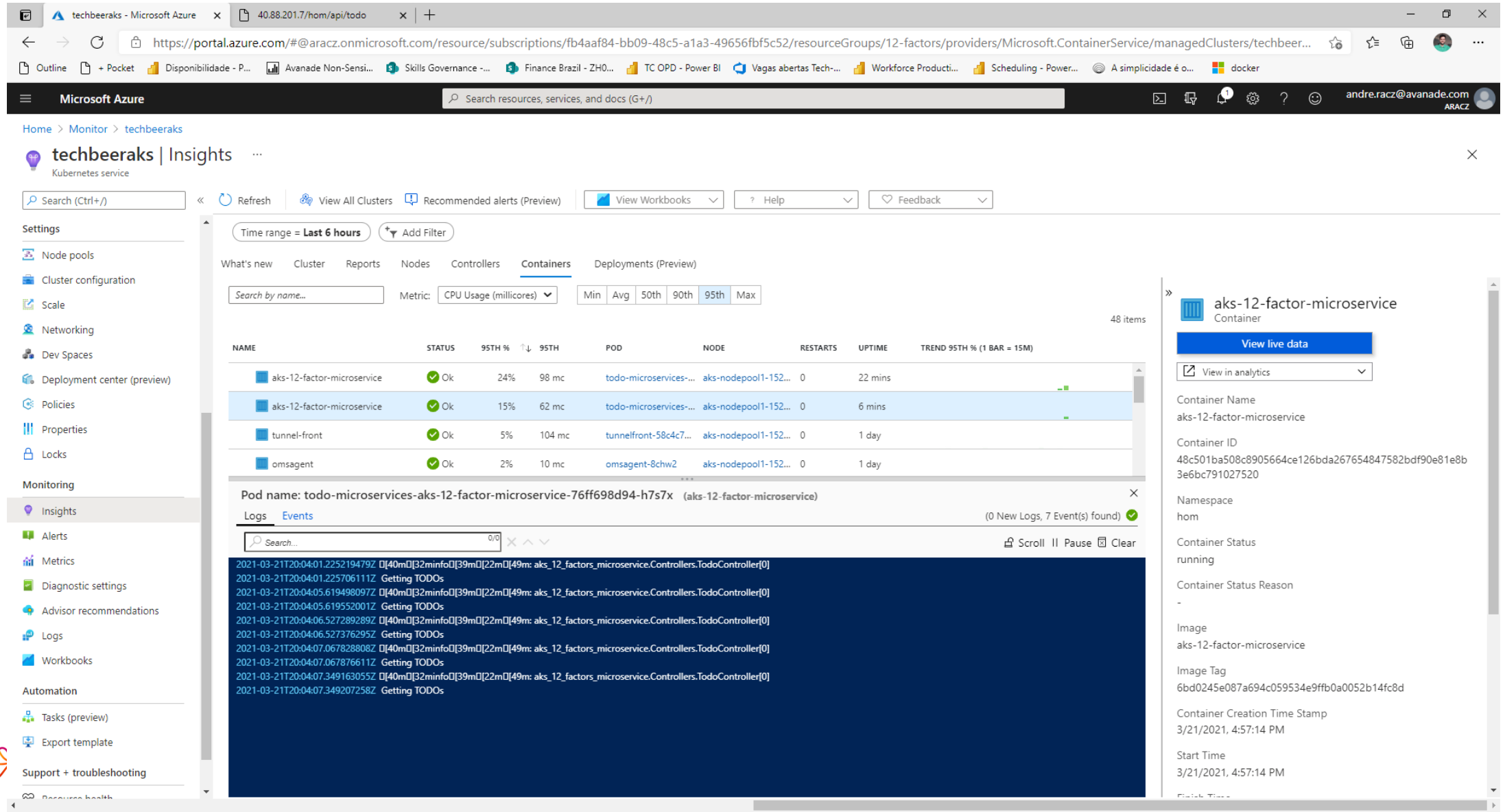
avanade

XI – Logs

- Trate os logs como streams de eventos
- 1 evento por linha (exceto stacktraces)
- Padronizar o formato
- Logar para STDOUT
- Aplicação não deve ser preocupar com armazenamento dos logs
- O ambiente deve cuidar dos logs

avanade

XI – Logs – AKS + Azure Container Insights



XII – Admin processes

- Rode ferramentas administrativas como processos esporádicos
- Devem rodar no mesmo ambiente que a aplicação
- Devem ser entregues junto com a aplicação
- Devem ter controle de versão

avanade

XII – Admin processes em AKS

- É possível conectar direto em um container da aplicação para rodar commandos:

  kubectl exec -it -n namespace nomecontainer -- sh

- É possível rodar jobs e cronjobs utilizando a mesma imagem do container (https://andrewlock.net/deploying-asp-net-core-applications-to-kubernetes-part-8-running-database-migrations-using-jobs-and-init-containers/)
- Linguagens de script facilitam isso

avanade

Obrigado

Repositórios:

- Principal: https://github.com/andreracz/12-factors-aks-main
- Microserviço: https://github.com/andreracz/12-factors-aks-microservice
- Front: https://github.com/andreracz/12-factors-aks-front

avanade

avanade

avanade

Alternate Title Slide

Go to https://aka.avanade.com/CoverPhotos for additional cover photos