INSTITUTO FEDERAL
Paraíba

Campus Cajazeiras

PROGRAMAÇÃO P/ WEB 2

3

11. IMPLANTANDO MICROSSERVIÇOS

PROF. DIEGO PESSOA

DIEGO.PESSOA@IFPB.EDU.BR

@DIEGOEP

CST em Análise e Desenvolvimento de Sistemas

ROTEIRO

- Introdução aos quatro principais padrões de implantação, como funcionam, seus benefícios e desvantagens:
  - Implantação de pacote de linguagem específica
  - Implantação de serviço como VM
  - Implantação de um serviço como container
  - Implantação serverless
- Implantando serviços com Kubernetes
- Usando um service mesh para separar implantação da release
- Implantando serviços com AWS Lambda
- Como escolher um padrão de implantação

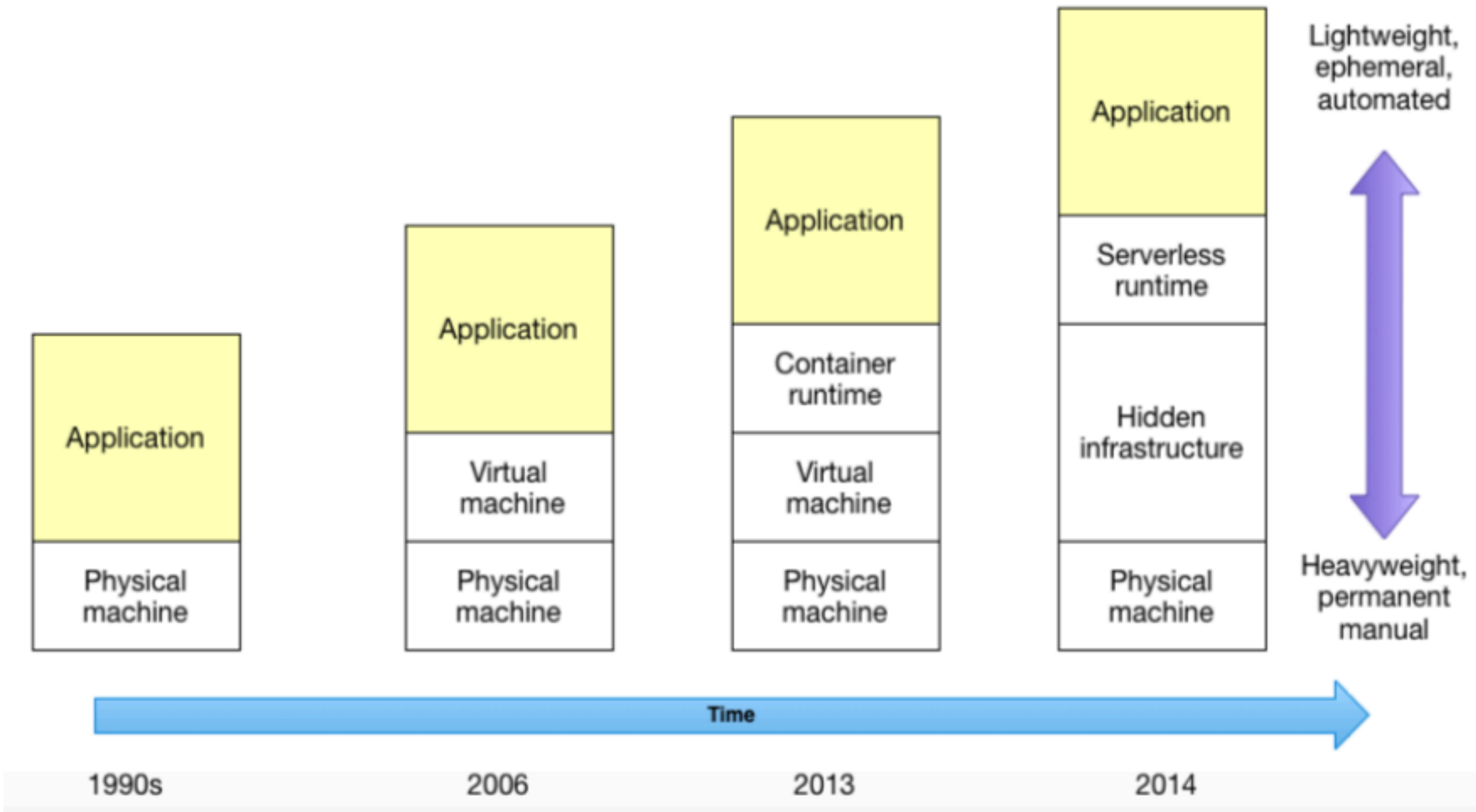
O PROCESSO DE IMPLANTAÇÃO

- Implantação (*deployment*) é uma combinação de dois conceitos relacionados: processo e arquitetura

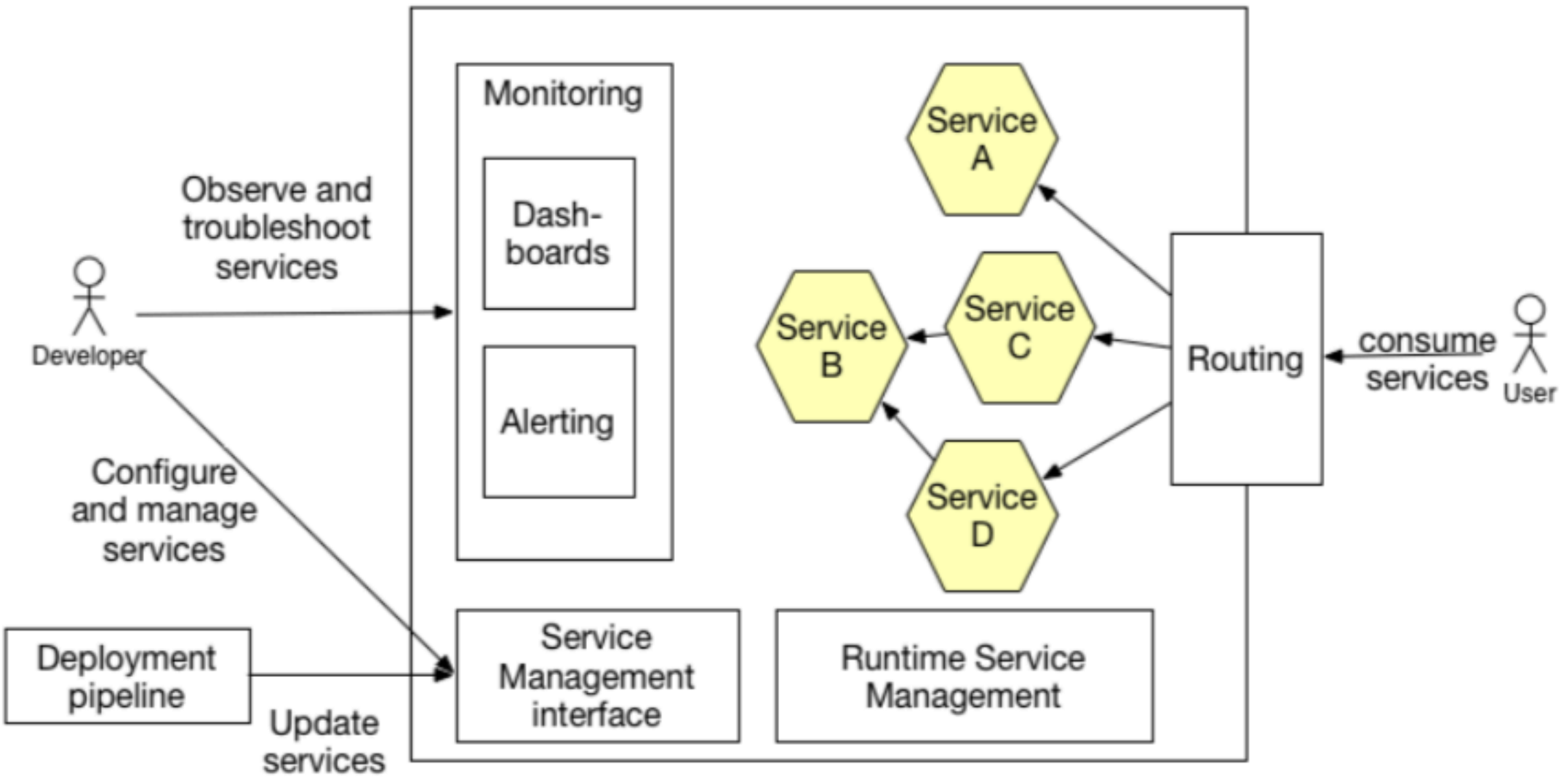
- O processo de implantação são as etapas que precisam ser realizadas por pessoas (dev e ops) para pôr o software em produção

- A arquitetura de implantação define a estrutura do ambiente em que o software irá rodar

FORMAS DE IMPLANTAÇÃO AO LONGO DO TEMPO



VISÃO SIMPLIFICADA DO AMBIENTE DE PRODUÇÃO

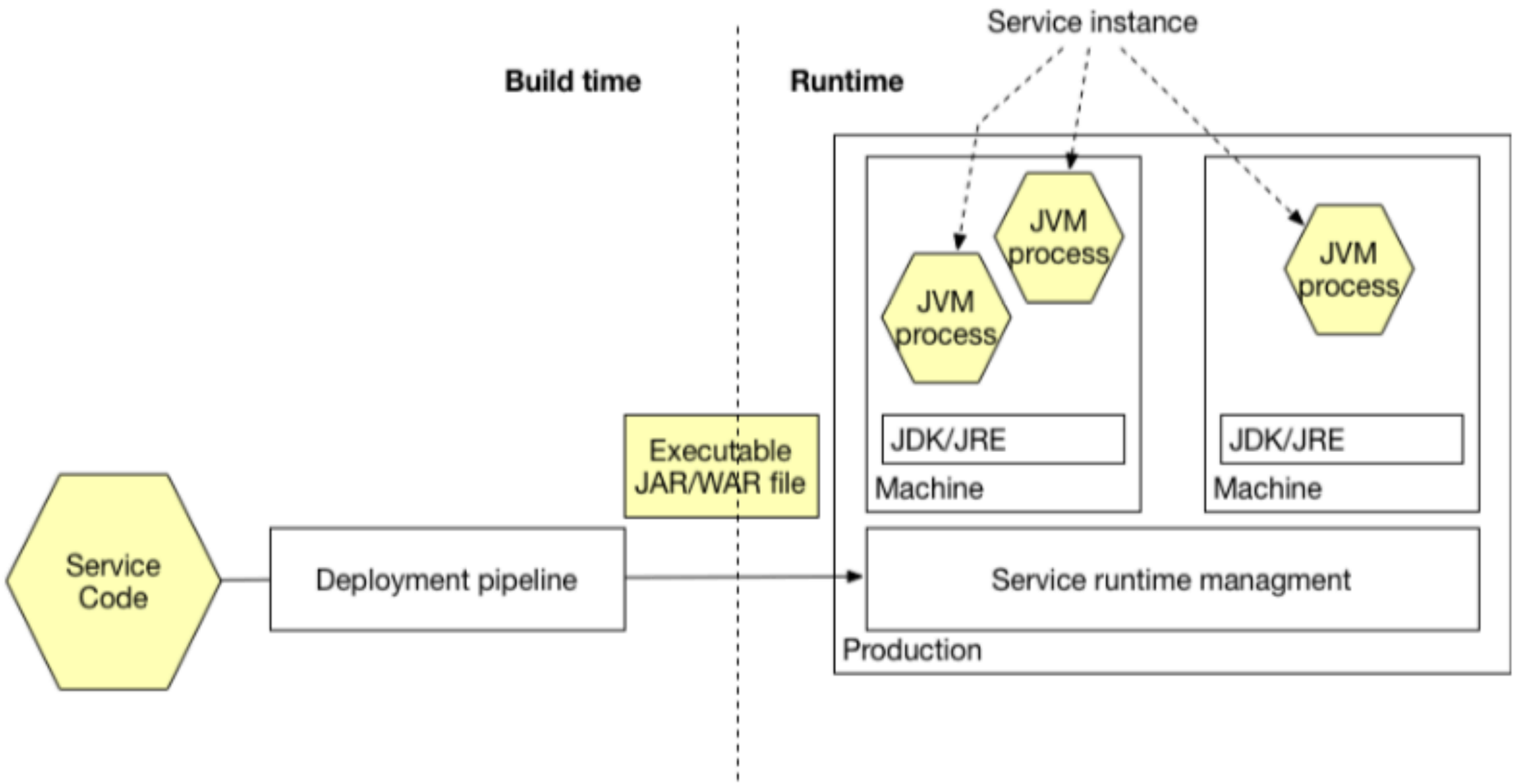


OPÇÕES DE IMPLANTAÇÃO

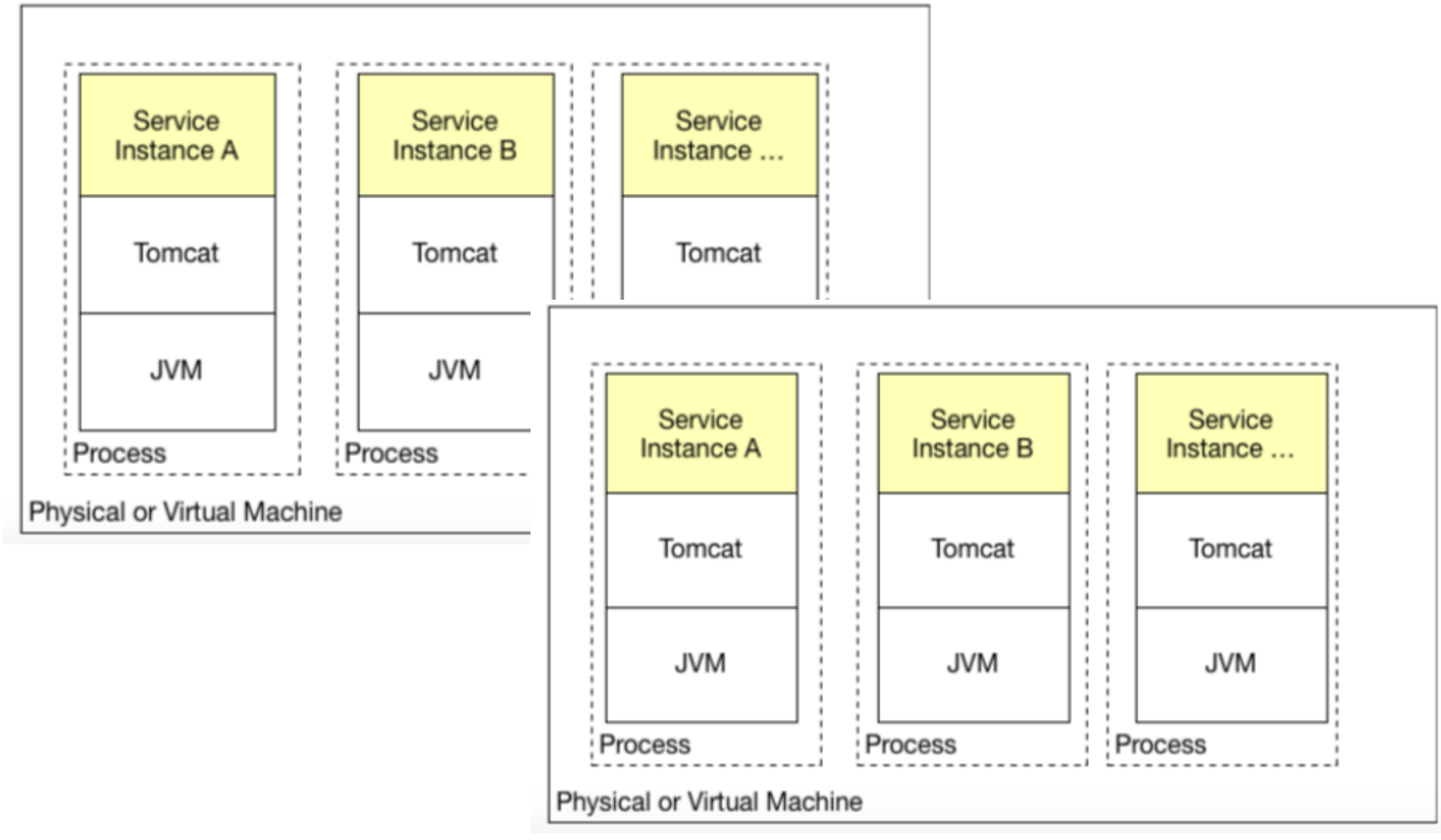
- Implantação de serviços como pacotes de linguagem específica (ex.: arquivos JAR, WAR)
- Implantação como imagem de máquina virtual, encapsulando o stack de tecnologias do serviço
- Implantação como containers, que são mais leves do que máquinas virtuais (ex.: Docker + Kubernetes)
- Implantação de serviços usando serverless deployment, que é ainda mais moderno do que os containers (ex.: AWS Lambda)

1. IMPLANTANDO SERVIÇOS USANDO PACOTES DE LINGUAGEM ESPECÍFICA

IMPLANTAÇÃO DE PACOTES DE LINGUAGEM ESPECÍFICA



IMPLANTANDO MÚLTIPLAS INSTÂNCIAS DE SERVIÇOS NA MESMA MÁQUINA



IMPLANTAÇÃO DE PACOTES DE LINGUAGEM ESPECÍFICA – BENEFÍCIOS

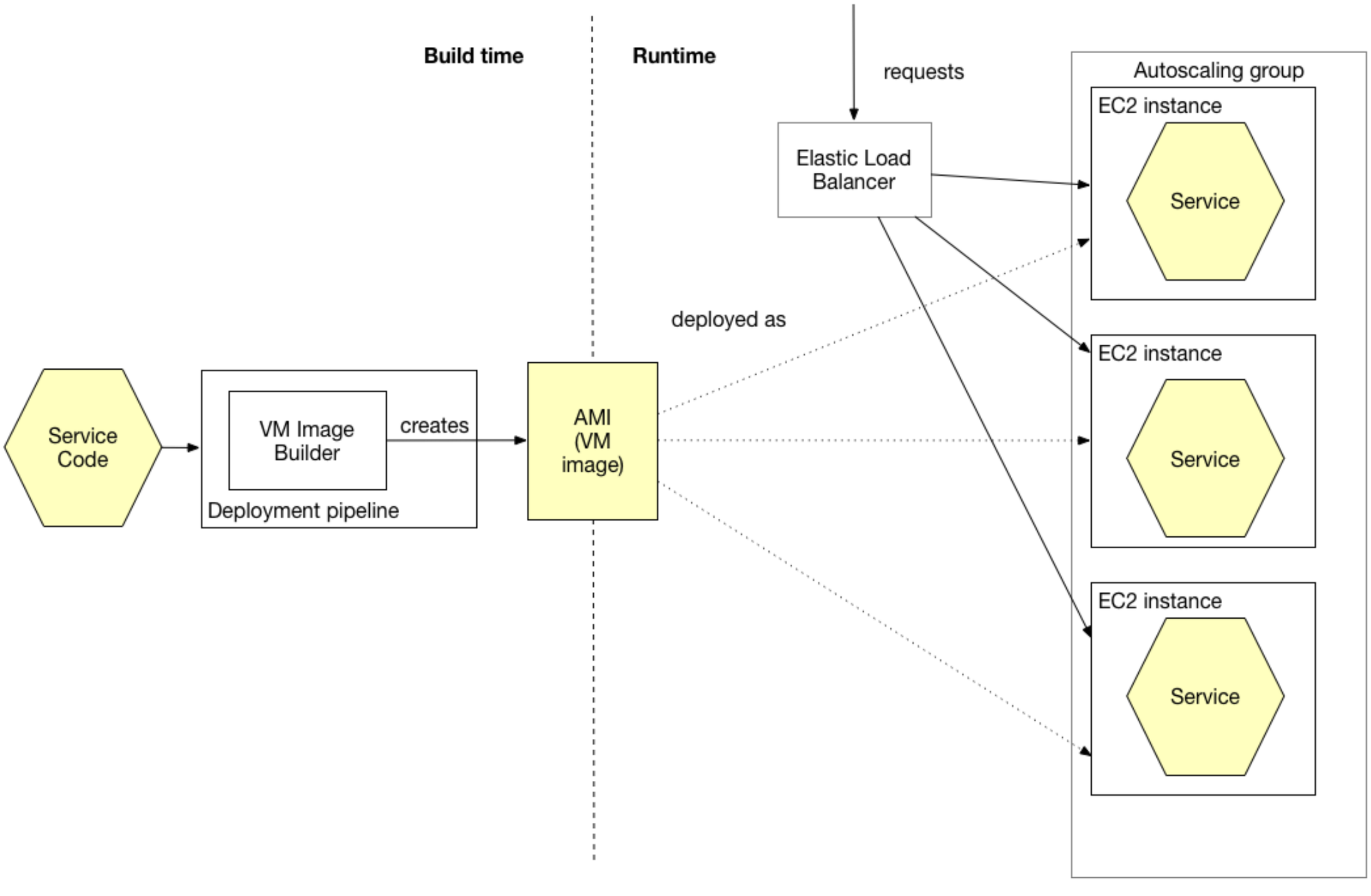
- **Implantação rápida**
  - Basta copiar o artefato e iniciar
- **Utilização eficiente de recursos, especialmente quando múltiplas instâncias rodam na mesma máquina**
  - Instâncias compartilham a mesma máquina, o mesmo sistema operacional e até o mesmo processo (ex.: Tomcat)

IMPLANTAÇÃO DE PACOTES DE LINGUAGEM ESPECÍFICA – DESVANTAGENS

- **Falta de encapsulamento do stack de tecnologias:**
  - Versão do tomcat, jdk, banco e pacotes internos do SO.
- **Falta de controle sobre os recursos consumidos por uma instância de serviço**
  - Uma instância pode consumir todo o CPU ou memória da máquina e “derrubar" os outros serviços que compartilham o mesmo recurso.
- **Falta de isolamento quando executadas múltiplas instâncias de serviços numa mesma máquina**
  - Uma instância de serviço com problemas impactará nas outras instâncias que estão rodando.
- **Dificuldade em determinar automaticamente onde colocar as instâncias dos serviços**
  - Qual SO utilizar para as aplicações? Quanto de memória precisa ter? Quanto de CPU?

2. IMPLANTANDO SERVIÇOS EM MÁQUINAS VIRTUAIS

IMPLANTANDO SERVIÇOS EM MÁQUINAS VIRTUAIS



BENEFÍCIOS DE IMPLANTAR SERVIÇOS COMO VMS

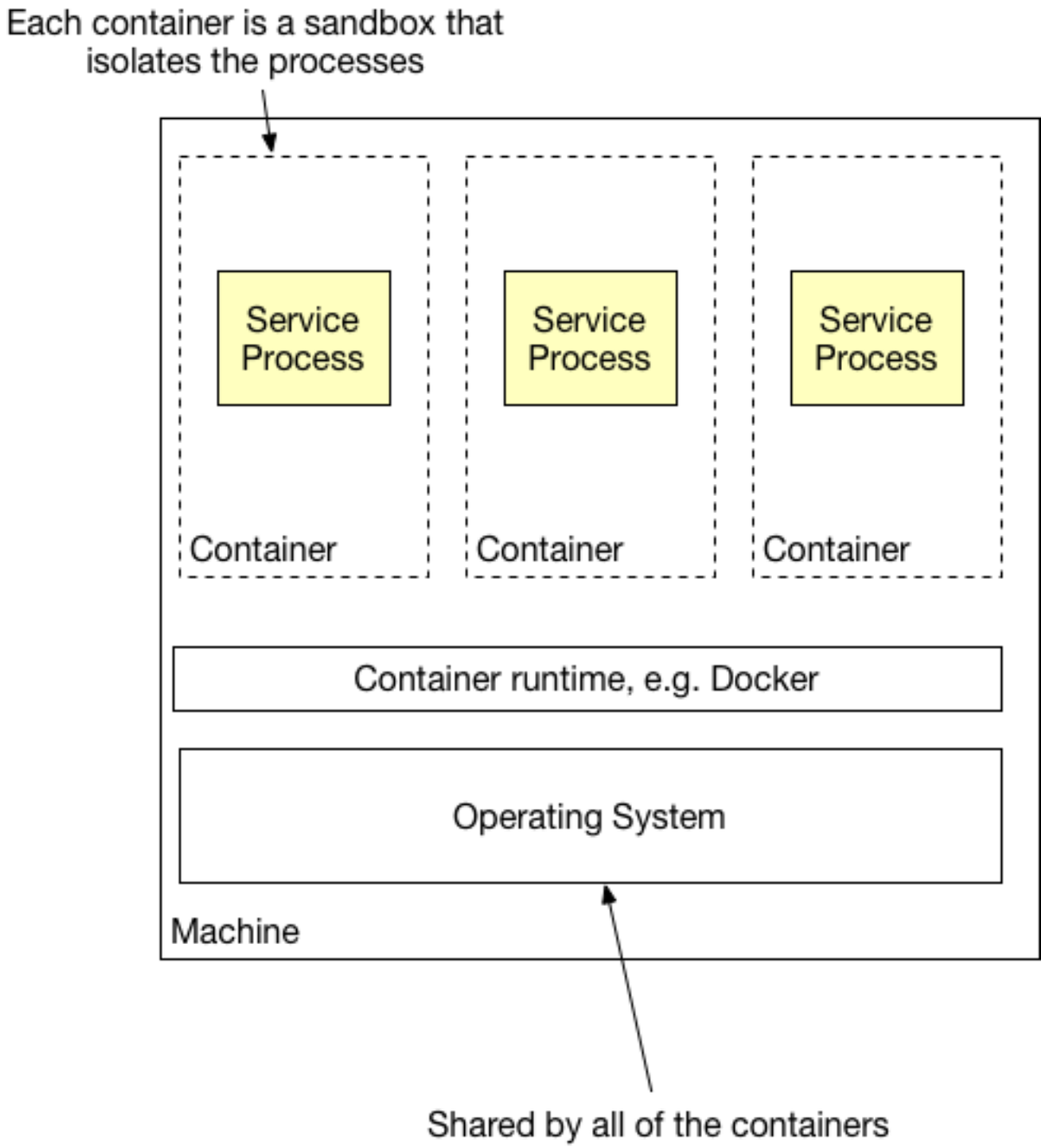
- A máquina virtual encapsula o stack de tecnologias
- Instâncias de serviços isoladas
- Alavanca infraestrutura em nuvens maduros

DESVANTAGENS DE IMPLANTAR SERVIÇOS COMO VMS

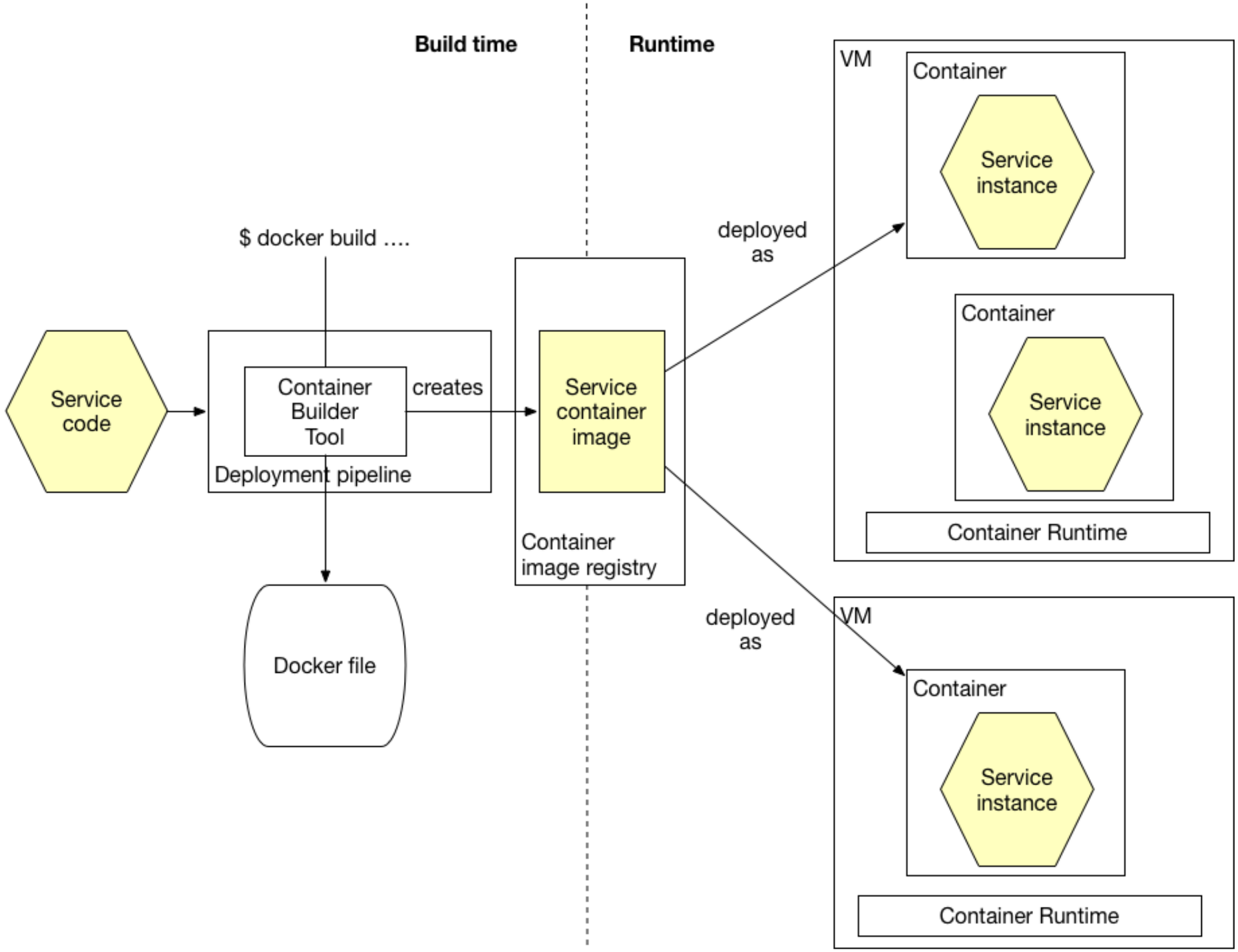
- Utilização de recursos menos eficiente
- Implantação relativamente lenta
- Sobrecarga na administração do sistema

3. IMPLANTANDO SERVIÇOS COMO CONTAÍNERS

CONCEITO DE CONTAINERS



IMPLANTAÇÃO DE SERVIÇOS COMO CONTAINERS



BENEFÍCIOS DE IMPLANTAR SERVIÇOS COMO CONTAINERS

- Encapsulamento do stack de tecnologias
- Instâncias de serviços são isoladas
- Controle dos recursos das instâncias dos serviços

DESVANTAGENS DE IMPLANTAR SERVIÇOS COMO CONTAINERS

- Desenvolvedor responsável pelo processo de administrar as imagens dos containers (definição do runtime)

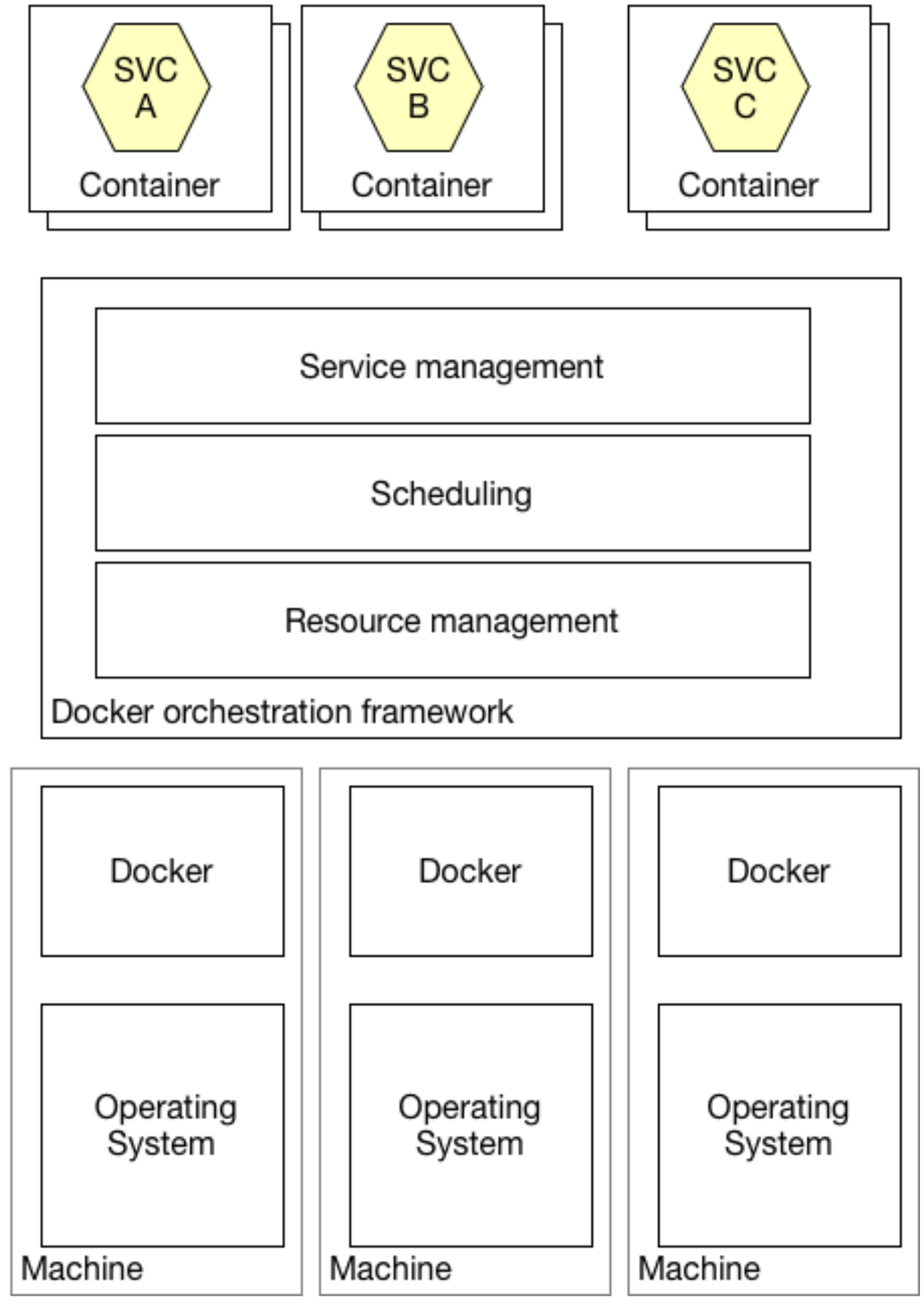
3. IMPLANTANDO UMA APLICAÇÃO COM KUBERNETES

KUBERNETES

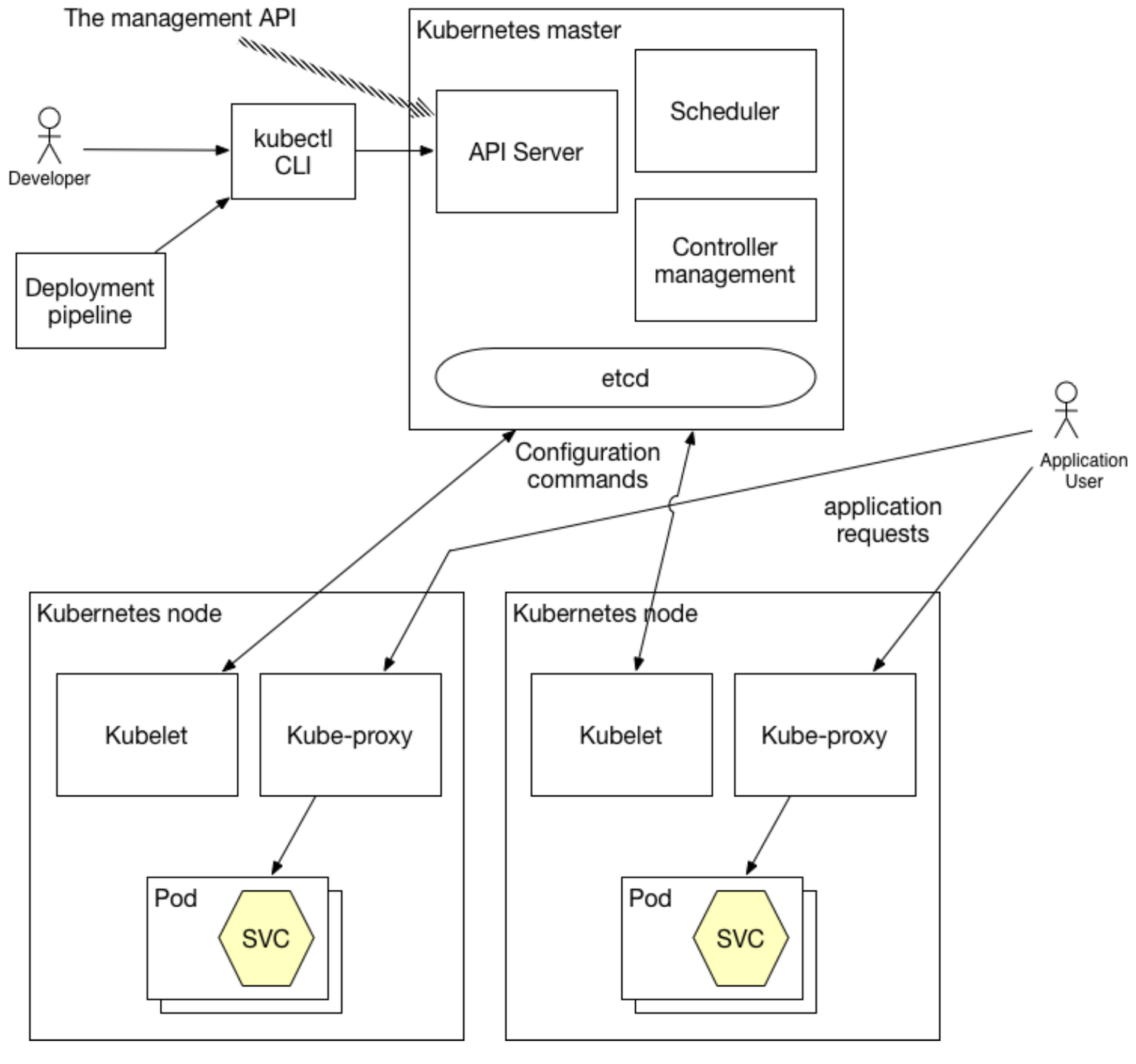
- O uso de Docker Compose é útil para desenvolvimento e testes. Porém, para rodar serviços confiáveis em produção, é preciso usar um runtime de containers mais sofisticado, como o Kubernetes.
- Kubernetes é um framework de orquestração de containers Docker. Trata-se de uma camada de software acima do Docker que mapeia um conjunto de máquinas em um pool de recursos para rodar serviços.
- Isto possibilita manter um desejado número de instâncias de cada serviço rodando mesmo quando algumas instâncias do serviço ou máquinas parem de funcionar.
- A agilidade dos containers combinada com a sofisticação de Kubernetes são uma forma convincente de implantar serviços.

VISÃO GERAL DE KUBERNETES

- Kubernetes é um framework de orquestração de Docker.
- Ele trata um conjunto dem áquinas rodando Docker como um pool de recursos.
- Você simplesmente diz ao Kubernetes para rodar N instâncias do seu serviço e ele cuida do resto



ARQUITETURA DO KUBERNETES



ARQUITETURA DO KUBERNETES

- API server - API REST para implantar e gerenciar serviços, usado pela interface de comandos kubectl
- tcd - Um banco NoSQL que guarda dados do cluster
- scheduler - seleciona um nó para rodar um container
- controller manager - garante que o estado atual do cluster seja o desejado.
- kublet - cria e gerencia pods rodando no nó
- kube-proxy - gerencia o roteamento, incluindo load-balancing entre pods
- pods - os serviços da aplicação

KUBERNETES – CONCEITOS-CHAVES

- *Pod* - unidade básica de implantação no Kubernetes. Consiste de um ou mais containers que compartilham um endereço IP e um volume de armazenamento.
- *Deployment* - uma especificação declarativa de um pod. É um controlador que garante que o número de instâncias desejado no pod estarão rodando todo o tempo. Suporta versionamento e rollbacks.
- *Service* - provê clientes de serviços com uma localização em rede através de um IP ou DNS.
- ConfigMap - uma coleção de pares chave/valor que definem a configuraçõa externa para um ou mais serviços. Pode definir variáveis de ambiente dos containers.

IMPLANTANDO USANDO KUBERNETES

```
apiVersion: extensions/v1beta1
kind: Deployment                                                    ❶
metadata:
  name: ftgo-restaurant-service                                     ❷
spec:
  replicas: 2                                                       ❸
  template:
    metadata:
      labels:
        app: ftgo-restaurant-service                                ❹
    spec:                                                           ❺
      containers:
      - name: ftgo-restaurant-service
        image: ftgo-restaurant-service:1.0.0.RELEASE
        ports:
        - containerPort: 8080                                       ❻
        env:                                                        ❼
          - name: SPRING_DATASOURCE_USERNAME
            valueFrom:
              secretKeyRef:
                name: ftgo-db-secret                                ❽
                key: username
          - name: SPRING_DATASOURCE_PASSWORD
            valueFrom:
              secretKeyRef:
                name: ftgo-db-secret
                key: password
          - name: SPRING_DATASOURCE_URL
            value: jdbc:mysql://mysql/eventuate
        ...
        livenessProbe:                                              ❾
          httpGet:
            path: /health
            port: 8080
          initialDelaySeconds: 30
          periodSeconds: 20
        readinessProbe:
          httpGet:
            path: /health
            port: 8080
          initialDelaySeconds: 30
          periodSeconds: 20
```

❶ Especifica que este objeto é do tipo deployment
❷ Nome do deployment
❸ Número de réplicas pod
❹ Dá a cada pod um rótulo chamado “app” cujo valor é “ftgo-restaurant-service”
❺ A especificação do pod, que define apenas um container
❻ A porta do cointaner
❼ As variáveis de ambiente do container, que são as configurações externas do serviço. Elas são lidas pelo Spring Boot, que as deixa disponível como propriedades no contexto da aplicação.
❽ Valores sensíveis que são recuperados do Kubernetes S

ROTEIRO PARA IMPLANTAÇÃO

- Instalação e configuração do kubectl
- kubectl apply -f arquivo.yml
- Criar segredo para login/senha:
- echo -n mysqluser > ./deployment/kubernetes/dbuser.txt echo -n mysqlpw > ./deployment/kubernetes/dbpassword.txt kubectl create secret generic ftgo-db-secret \
- --from-file=username=./deployment/kubernetes/dbuser.txt \ --from-file=password=./deployment/kubernetes/dbpassword.txt