Projeto 2

Kubernetes: Apresentação Teórica e Prática

Matheus Luis Oliveira da Silva

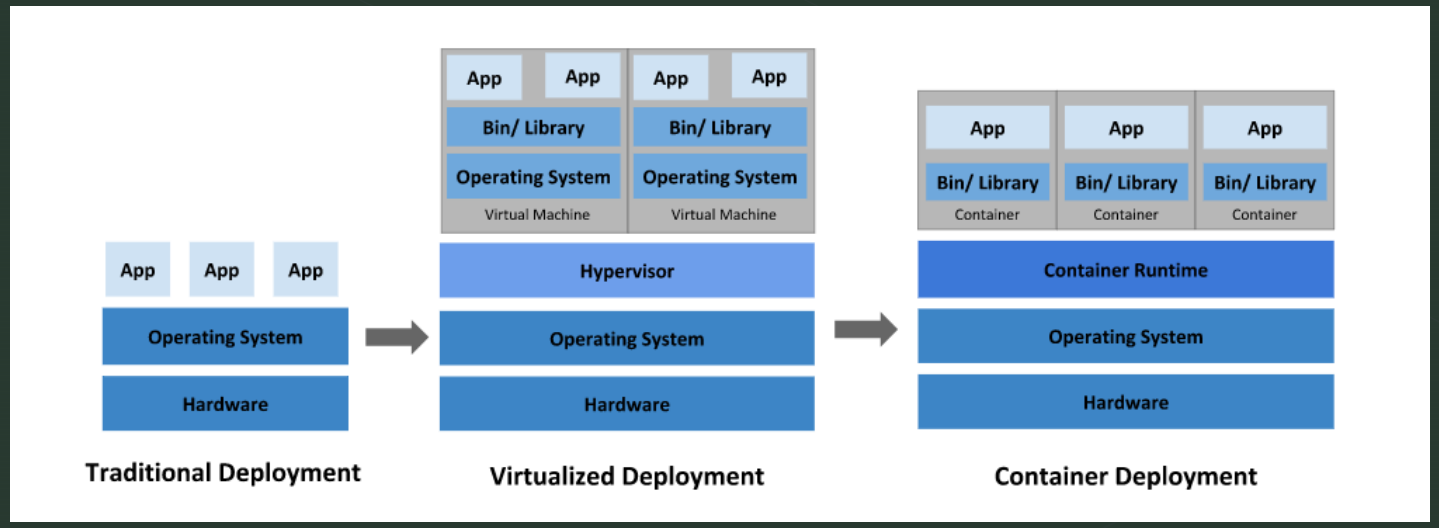
Opus Software

O que é Kubernetes?

- Desenvolvido pela Google;
- Nasce como um orquestrador de containers OpenSource;
- Provê alta disponibilidade e baixo downtime se bem aplicado;
- Promove escalabilidade e performance;
- Mecanismos de backup de alta eficiência.

História:

- A evolução do deployment de aplicações: tradicional, virtualização e containerização;
- Concretização da quebra do monolíto em microsserviços;
- Surge como um gerenciador, uma solução para o problema de estabilização e controle de containers, vai além de um simples serviço de orquestração.



Como utilizar ?

- É necessário que se tenha mais de uma máquina disponível, virtual ou física para a criação de um cluster.
- O cluster pode ser virtualizado, utilizando, para isso, alguma ferramenta gerenciadora de clusters: Minikube, KinD, etc.
- É preciso um runtime de containers para que o cluster possa criar seus pods e componentes: Docker, containerd, CRI-O, etc.
- Uma ferramenta para se comunicar com a api do kubernetes e gerenciar o cluster: Kubectl, por exemplo.

Kubernetes Vs Docker:

- São duas ferramentas distintas para dois propósitos distintos;
- Docker ofereceu soluções para o problema ddas limitações que a virtualização de máquinas tinham na época.
- Kubernetes, por sua vez, oferece solução para as limitações do Docker (ou qualquer outra ferramenta de containereização) trazendo ordem, automatização e desempenho para sistemas complexos.
- Nenhuma das tecnologias substituiu a outra, apenas se complementaram.

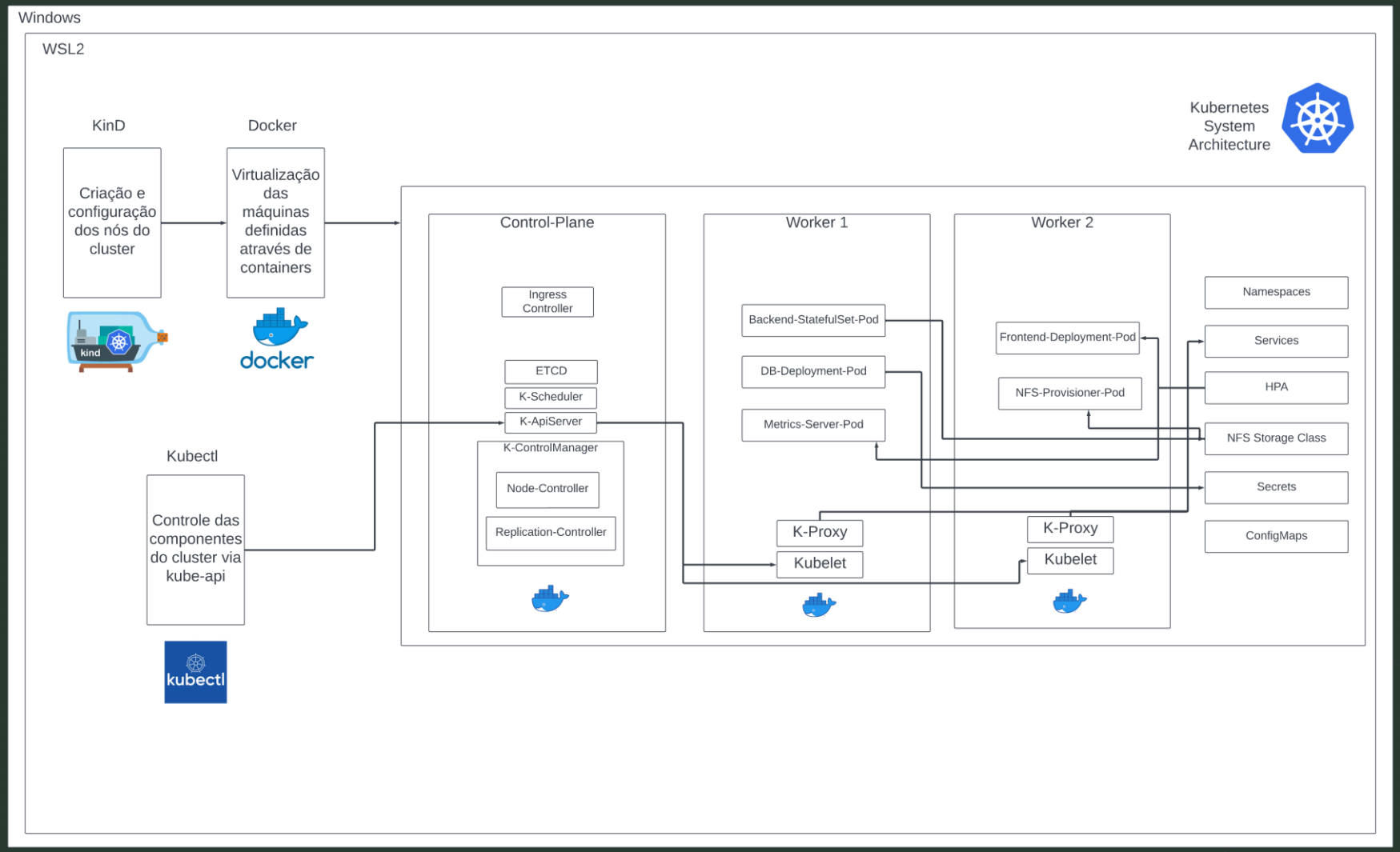
Kubernetes vs Docker:

- **Descoberta de serviço e balanceamento de carga**
- **Orquestração de armazenamento**
- **Lançamentos e reversões automatizadas**
- **Gerenciamento de nós e recursos**
- **Autocorreção**
- **Gerenciamento de configuração e de segredos**
  - https://kubernetes.io/pt-br/docs/concepts/overview/what-is-kubernetes/

Pods vs Containers:

- Várias definições possíveis, todas corretas:
- “Um pod é uma abstração superior de um ou mais containeres”
- “Um pod é um invólucro para containeres, no qual o container é empactoado para rodar num cluster”
- “Um pod é um slot no qual um container é alocado para ter seu armazenamento, rede e recursos compartilhados com outros containeres em um cluster”
- Basicamente, o pod pode ser enxergado como a evolução necessária do container para que esse pudesse ser usado de maneira prática num sistema clusterizado.

Arquitetura de um cluster kubernetes:



Componentes:

- Na prática:
  - Aplicação prática abordando:
    - Namespaces
    - Configmaps e Secrets
    - Criptografia do banco etcd
    - ReplicaSet, Deployments e Statefulsets
    - Nfs-client, SC, PVC e StatefulSet
    - Metrics-server e HPA
    - services e ingress
- Na teoria:
  - Abordagem teórica contemplando:
    - Daemonsets
    - Job e Cronjobs
    - RBAC (roles, service accounts, users)
    - CRD

DaemonSet

- Aplicações web → Deployments
- Banco de dados, backends → StatefulSets
- Jobs → CronJobs
- Nodes + pod assignment → DaemonSets
- Com eles é possível declarar um pod para seja atribuída uma cópia para cada um de um certo número de nós.
- Usado geralmente para atribuir a cada node do cluster serviços como os de armazenamento, monitoração de pods ou gerenciamento de logs, por exemplo.
- Quando um node é adicionado, o pod declarado no daemonSet é atribuído ao nó, e o mesmo é deletado caso haja deleção do node.

```yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: fluentd-elasticsearch
  namespace: kube-system
  labels:
    k8s-app: fluentd-logging
spec:
  selector:
    matchLabels:
      name: fluentd-elasticsearch
  template:
    metadata:
      labels:
        name: fluentd-elasticsearch
    spec:
      tolerations:
      # these tolerations are to have the daemonset runnable on control plane nodes
      # remove them if your control plane nodes should not run pods
      - key: node-role.kubernetes.io/control-plane
        operator: Exists
        effect: NoSchedule
      - key: node-role.kubernetes.io/master
        operator: Exists
        effect: NoSchedule
      containers:
      - name: fluentd-elasticsearch
        image: quay.io/fluentd_elasticsearch/fluentd:v2.5.2
        resources:
          limits:
            memory: 200Mi
          requests:
            cpu: 100m
            memory: 200Mi
        volumeMounts:
        - name: varlog
          mountPath: /var/log
      terminationGracePeriodSeconds: 30
      volumes:
      - name: varlog
        hostPath:
          path: /var/log
```

DaemonSet

Job e CronJob

- Jobs são utilizados quando se tem rotinas de começo meio e fim.
- Criam pods para executar a tarefa, indefinidamente até que um limite seja atingido ou a tarefa concluída.
- É possível definir:
  - Falhas toleradas, sucessos desejados, limitação de recursos, etc.
- CronJobs são utilizados para agendar, cronologicamente, Jobs para serem executados dentro de um cluster.
  - Rotina de relatórios.

- Jobs podem ser executados concomitantemente
- Vale ressaltar a importância do troubleshooting ao se lidar com jobs
- CronJobs são ainda mais sensíveis a erros.

```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: pi
spec:
  template:
    spec:
      containers:
      - name: pi
        image: perl:5.34.0
        command: ["perl",  "-Mbignum=bpi", "-wle", "print bpi(2000)"]
      restartPolicy: Never
  backoffLimit: 4
```

Job

CRD

- Um resource é um endpoint na API do kubernetes que guarda uma coleção de APIObjects
- API/v1 por exemplo é um endpoint group
- um Deployment é um recurso, um api object
- Custom Resources Definition, então, é autoexplicativo, ou seja, o usuario do cluster kubernetes pode definir seus próprios recursos com seus próprios objetos e, inclusive propriedades.

RBAC

- CA certificates:
  - São utilizados para que o kubernetes reconheça um usuário com suas devidas permissões.
  - Geralmente ficam na pasta /etc/kuberenetes/pki no control-plane
- CubeConfig file:
  - Caso seja criado um novo certificado, o componente cubeconfig é usado para anexar esse certificado e adicioná-lo ao cluster.
  - O cubeconfig padrão está localizado na pasta /home/matheusoliveira/.kube.
  - O administrador do cluster pode usar esse arquivo para criar um novo e disponibilizá-lo ao colega desenvolverdor, por exemplo.
  - Mas e as limitações desse colega? → Roles

RBAC

- Roles:
  - Roles são usados para garantir a um usuário restrições e permissões de acesso a um determinado componente de um cluster.
  - Também são definidas as permissões de ações que o usuário poderá tomar dentro de cada componente.
  - Geralmente são usados a nivel de namespaces, por exemplo, uma permissão de leitura para um grupo específico a um namespace específico.
  - Para permissões a nível de cluster, existem os clusterroles e clusterbindings.
- ServiceAccount:
  - Similar a autorização de um usuário, serviceAccounts são usadas para garantir permissões de acesso a pods e/ou aplicações.

Ambientes produtivos:

- Para um ambiente produtivo, vários pontos devem ser levados em consideração, dentre eles:
- Setup com alta disponibilidade:
  - Réplicas do control-plane;
  - Disponibilidade de worker nodes;
  - Gerenciamento de tráfego para o api-server.

Ambientes Produtivos:

- Scaling:
  - Utilizar os recursos estudados para garantir a escala do cluster
  - Manter todos os arquivos de declaração organizados e modularizados
  - Estudar e trabalhar os recursos disponíveis de hardware ou de serviço de cloud
- Segurança:
  - Criptografia de secrets e arquivos → evita acesso indevido a arquivos com dados do sistema
  - RBAC para limitar permissões de acesso → evita acesso a informações sensíveis expostas no ambiente
  - Segurança e criptografia dos armazenamentos utilizados → evita acesso de invasões feitas ao banco de dados do cluster ou da aplicação
  - Controle de tráfego de rede → evita ataques ou escutas de rede que possam se aproveitar de dados transmitidos sem criptografia

- FIM