Comprovantes - Prática Minikube e Kubernetes

Parte A - Preparar o ambiente

1. Verificar nó do cluster

COMANDO:

```
kubectl get nodes
```

SAÍDA:

```
NAME       STATUS   ROLES           AGE   VERSION
minikube   Ready    control-plane   12s   v1.34.0
```

2. Verificar pods do Ingress NGINX

COMANDO:

```
kubectl -n ingress-nginx get pods
```

SAÍDA:

```
NAME                                        READY   STATUS      RESTARTS   AGE
ingress-nginx-admission-create-9vb9s        0/1     Completed   0          25s
ingress-nginx-admission-patch-hk9zk         0/1     Completed   0          25s
ingress-nginx-controller-9cc49f96f-jwvxx    0/1     Running     0          25s
```

Parte B - Dockerizar a API e testar localmente

1. Teste do endpoint /healthz

COMANDO:

```
curl http://localhost:8000/healthz
```

SAÍDA:

```
{"status":"ok"}
```

2. Teste do endpoint /quotes/random

COMANDO:

```
curl http://localhost:8000/quotes/random
```

SAÍDA:

```
{"text":"Ship small, ship often."}
```

Parte C - Publicar no Minikube

1. Verificar recursos criados (Deployment, Service, Ingress)

COMANDO:

```
kubectl get deploy,svc,ing
```

SAÍDA:

```
NAME                          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/quotes-api    2/2     2            2           8s

NAME                 TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1        <none>        443/TCP   83s
service/quotes-api   ClusterIP   10.102.177.102   <none>        80/TCP    5s

NAME                                        CLASS   HOSTS          ADDRESS   PORTS   AGE
ingress.networking.k8s.io/quotes-api-ingress   nginx   quotes.local             80      3s
```

2. Verificar pods em estado Running

COMANDO:

```
kubectl get pods
```

SAÍDA:

```
NAME                          READY   STATUS    RESTARTS   AGE
quotes-api-5f4f6f9548-p6lr8   1/1     Running   0          10s
quotes-api-5f4f6f9548-qmnpf   1/1     Running   0          10s
```

3. Logs da aplicação

COMANDO:

```
kubectl logs -l app=quotes-api
```

SAÍDA:

```
INFO:     Started server process [1]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO:     10.244.0.1:43432 - "GET /healthz HTTP/1.1" 200 OK
INFO:     10.244.0.1:43438 - "GET /healthz HTTP/1.1" 200 OK
INFO:     10.244.0.1:60166 - "GET /healthz HTTP/1.1" 200 OK
INFO:     10.244.0.1:60176 - "GET /healthz HTTP/1.1" 200 OK
INFO:     10.244.0.1:60178 - "GET /healthz HTTP/1.1" 200 OK
INFO:     Started server process [1]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO:     10.244.0.1:42194 - "GET /healthz HTTP/1.1" 200 OK
INFO:     10.244.0.1:42206 - "GET /healthz HTTP/1.1" 200 OK
INFO:     10.244.0.1:56664 - "GET /healthz HTTP/1.1" 200 OK
INFO:     10.244.0.1:56674 - "GET /healthz HTTP/1.1" 200 OK
INFO:     10.244.0.1:56684 - "GET /healthz HTTP/1.1" 200 OK
```

4. Teste via Ingress - /healthz

COMANDO:

```
curl http://quotes.local/healthz
```

SAÍDA:

```
{"status":"ok"}
```

5. Teste via Ingress - /quotes/random

COMANDO:

```
curl http://quotes.local/quotes/random
```

SAÍDA:

```
{"text":"Ship small, ship often."}
```

Parte D - Observabilidade, Escala e Rollout

1. Logs da aplicação

COMANDO:

```
kubectl logs -l app=quotes-api
```

SAÍDA:

```
INFO:     Started server process [1]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO:     10.244.0.1:43432 - "GET /healthz HTTP/1.1" 200 OK
[... linhas similares de logs de healthz ...]
INFO:     10.244.0.1:56684 - "GET /healthz HTTP/1.1" 200 OK
```

2. Descrição detalhada de um pod

COMANDO:

```
kubectl describe pod -l app=quotes-api
```

SAÍDA:

```
Name:             quotes-api-5f4f6f9548-p6lr8
Namespace:        default
Priority:         0
Service Account:  default
Node:             minikube/192.168.49.2
Start Time:       Mon, 17 Nov 2025 20:26:57 -0300
Labels:           app=quotes-api
                  pod-template-hash=5f4f6f9548
Annotations:      <none>
Status:           Running
IP:               10.244.0.7
IPs:
  IP:           10.244.0.7
Controlled By:  ReplicaSet/quotes-api-5f4f6f9548
Containers:
  quotes-api:
    Container ID:   docker://d82663aab53bbedee1e86c547e4468061288d337779711a628ef89cbdd42adf2
    Image:          quotes-api:1.0.0
    Image ID:       docker://sha256:0dd1aa632941f3f7a726ede1df932006b98cdecf8539dfdb3a6de4f93728d7d4
    Port:           8000/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Mon, 17 Nov 2025 20:26:59 -0300
    Ready:          True
    Restart Count:  0
    Liveness:       http-get http://:8000/healthz delay=10s timeout=1s period=10s #success=1 #failure=3
    Readiness:      http-get http://:8000/healthz delay=3s timeout=1s period=5s #success=1 #failure=3
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-z74mn (ro)
Conditions:
  Type                        Status
  PodReadyToStartContainers   True
  Initialized                 True
  Ready                       True
  ContainersReady             True
  PodScheduled                True
```

3. Eventos do cluster

COMANDO:

```
kubectl get events --sort-by=.metadata.creationTimestamp | tail -n 20
```

SAÍDA:

```
LAST SEEN   TYPE     REASON                   OBJECT                            MESSAGE
2m20s       Normal   NodeAllocatableEnforced  node/minikube                     Updated Node Allocatable limit across pods
2m20s       Normal   NodeHasSufficientMemory  node/minikube                     Node minikube status is now: NodeHasSufficientMemo
2m20s       Normal   NodeHasNoDiskPressure    node/minikube                     Node minikube status is now: NodeHasNoDiskPressure
2m20s       Normal   NodeHasSufficientPID      node/minikube                     Node minikube status is now: NodeHasSufficientPID
2m20s       Normal   Starting                 node/minikube                     Starting kubelet.
2m16s       Normal   RegisteredNode           node/minikube                     Node minikube event: Registered Node minikube in C
2m13s       Normal   Starting                 node/minikube
66s         Normal   SuccessfulCreate         replicaset/quotes-api-5f4f6f9548  Created pod: quotes-api-5f4f6f9548-qmnpf
66s         Normal   Scheduled                pod/quotes-api-5f4f6f9548-p6lr8   Successfully assigned default/quotes-api-5f4f6f954
66s         Normal   ScalingReplicaSet        deployment/quotes-api             Scaled up replica set quotes-api-5f4f6f9548 from 0
66s         Normal   SuccessfulCreate         replicaset/quotes-api-5f4f6f9548  Created pod: quotes-api-5f4f6f9548-p6lr8
66s         Normal   Scheduled                pod/quotes-api-5f4f6f9548-qmnpf   Successfully assigned default/quotes-api-5f4f6f954
64s         Normal   Pulled                   pod/quotes-api-5f4f6f9548-p6lr8   Container image "quotes-api:1.0.0" already present
64s         Normal   Created                  pod/quotes-api-5f4f6f9548-qmnpf   Created container: quotes-api
64s         Normal   Started                  pod/quotes-api-5f4f6f9548-qmnpf   Started container quotes-api
64s         Normal   Pulled                   pod/quotes-api-5f4f6f9548-qmnpf   Container image "quotes-api:1.0.0" already present
64s         Normal   Started                  pod/quotes-api-5f4f6f9548-p6lr8   Started container quotes-api
64s         Normal   Created                  pod/quotes-api-5f4f6f9548-p6lr8   Created container: quotes-api
44s         Normal   Sync                     ingress/quotes-api-ingress        Scheduled for sync
```

4. Escalar aplicação para 4 réplicas

COMANDO:

```
kubectl scale deploy/quotes-api --replicas=4
```

SAÍDA:

```
deployment.apps/quotes-api scaled
```

COMANDO:

```
kubectl get pods
```

SAÍDA (4 réplicas Running):

```
NAME                          READY   STATUS    RESTARTS   AGE
quotes-api-5f4f6f9548-p6lr8   1/1     Running   0          99s
quotes-api-5f4f6f9548-qmnpf   1/1     Running   0          99s
quotes-api-5f4f6f9548-tw2jb   1/1     Running   0          8s
quotes-api-5f4f6f9548-x49gv   1/1     Running   0          8s
```

5. Rollout de nova versão (1.0.1)

COMANDO:

```
kubectl set image deploy/quotes-api quotes-api=quotes-api:1.0.1
```

SAÍDA:

```
deployment.apps/quotes-api image updated
```

COMANDO:

```
kubectl rollout status deploy/quotes-api
```

SAÍDA:

```
Waiting for deployment "quotes-api" rollout to finish: 2 out of 4 new replicas have been updated...
Waiting for deployment "quotes-api" rollout to finish: 2 out of 4 new replicas have been updated...
[... linhas similares de progresso do rollout ...]
Waiting for deployment "quotes-api" rollout to finish: 1 old replicas are pending termination...
deployment "quotes-api" successfully rolled out
```

COMANDO:

```
kubectl rollout history deploy/quotes-api
```

SAÍDA:

```
deployment.apps/quotes-api
REVISION  CHANGE-CAUSE
1         <none>
2         <none>
```

6. Testar nova versão via Ingress

COMANDO:

```
curl http://quotes.local/quotes/random
```

SAÍDA:

```
{"text":"New quote in version 1.0.1!"}
```

7. Rollout undo (voltar para versão anterior)

COMANDO:

```
kubectl rollout undo deploy/quotes-api
```

SAÍDA:

```
deployment.apps/quotes-api rolled back
```

COMANDO:

```
kubectl rollout status deploy/quotes-api
```

SAÍDA:

```
Waiting for deployment "quotes-api" rollout to finish: 2 out of 4 new replicas have been updated...
[... linhas similares de progresso do rollout ...]
Waiting for deployment "quotes-api" rollout to finish: 3 of 4 updated replicas are available...
deployment "quotes-api" successfully rolled out
```

Nota sobre opção utilizada

Opção utilizada para carregar imagem: Opção 2 (Build local + load) - mais rápida e permite testar localmente antes de carregar no Minikube.

Perguntas de Reflexão

1. Explique, com suas palavras, o fluxo Dev → Docker → Minikube/Kubernetes que você implementou.

O fluxo começou com o desenvolvimento da API em FastAPI e criação do `Dockerfile` para empacotar a aplicação. Fiz o build local da imagem (`docker build -t quotes-api:1.0.0`) e testei em container Docker antes de publicar no cluster. Em seguida, carreguei a imagem no Minikube usando `minikube image load` (opção 2: build local + load).

No Kubernetes, criei três recursos: Deployment para gerenciar réplicas dos pods, Service ClusterIP para expor a aplicação internamente, e Ingress para acesso externo via `quotes.local`. O Ingress NGINX roteia o tráfego para o Service, que distribui para os pods do Deployment.

2. Qual é o papel de cada objeto criado (Deployment, Service, Ingress) na arquitetura?

O **Deployment** gerencia o ciclo de vida dos pods: define quantas réplicas rodar, mantém os pods saudáveis através de health checks (readiness e liveness probes), e permite rollouts controlados de novas versões.

O **Service** fornece um ponto de acesso estável aos pods. Como os pods têm IPs efêmeros, o Service oferece um IP virtual (ClusterIP) e nome DNS interno, além de fazer load balancing entre os pods.

O **Ingress** é a camada de entrada externa, funcionando como reverse proxy que roteia requisições HTTP/HTTPS baseado em hostname e path. No nosso caso, recebe requisições para `quotes.local` e encaminha para o Service, que distribui para os pods.

3. O que muda, conceitualmente, entre rodar docker run local e expor a app via Ingress em um cluster Kubernetes?

Com `docker run` local, temos um container isolado com IP e porta acessíveis diretamente. É simples, mas limitado: falhas requerem reinício manual, múltiplas instâncias precisam ser gerenciadas separadamente, e não há balanceamento de carga automático.

Com Kubernetes e Ingress, ganhamos orquestração automatizada: alta disponibilidade com múltiplas réplicas, auto-recuperação quando pods falham, escalabilidade horizontal simples (`kubectl scale`), e roteamento inteligente. O Ingress permite expor múltiplos serviços através de um único ponto de entrada com roteamento baseado em hostname/path. Além disso, Kubernetes oferece health checks automáticos, rollouts graduais sem downtime, e rollback rápido, tudo gerenciado declarativamente via YAML.

4. Descreva um problema real que você enfrentou na prática e como diagnosticou e resolveu.

Ao testar a aplicação via Ingress, recebi erro de resolução DNS ao acessar `http://quotes.local/healthz`. O problema era a falta da entrada `quotes.local` no arquivo `/etc/hosts`, que requer privilégios de administrador.

Para diagnosticar, verifiquei o Ingress (`kubectl get ingress`) e os pods do NGINX (`kubectl -n ingress-nginx get pods`), confirmando que estavam corretos. O problema era realmente DNS local.

Como solução alternativa, usei `kubectl port-forward svc/quotes-api 8080:80` para mapear o Service localmente, permitindo testar via `http://localhost:8080` e validar que a aplicação funcionava corretamente no cluster.

5. Se essa API fosse parte de um sistema maior, que outros recursos de Kubernetes você considera importantes (ConfigMap, Secret, HPA, etc.) e por quê?

ConfigMaps para externalizar configurações que variam entre ambientes, permitindo alterações sem reconstruir imagens. **Secrets** para armazenar dados sensíveis (chaves, tokens, credenciais) de forma criptografada com políticas de acesso restritivas.

HPA (Horizontal Pod Autoscaler) para escalar automaticamente baseado em métricas (CPU, memória, requisições), lidando com picos de tráfego e otimizando custos. **ResourceQuotas e LimitRanges** para gerenciar recursos do cluster e garantir distribuição justa.

NetworkPolicies para segurança em rede, controlando comunicação entre pods e criando micro-segmentação. **PersistentVolumes** para dados persistentes (logs, cache, banco de dados), garantindo que não sejam perdidos quando pods são recriados.

Esses recursos trabalhariam em conjunto para criar uma arquitetura escalável, segura e resiliente.