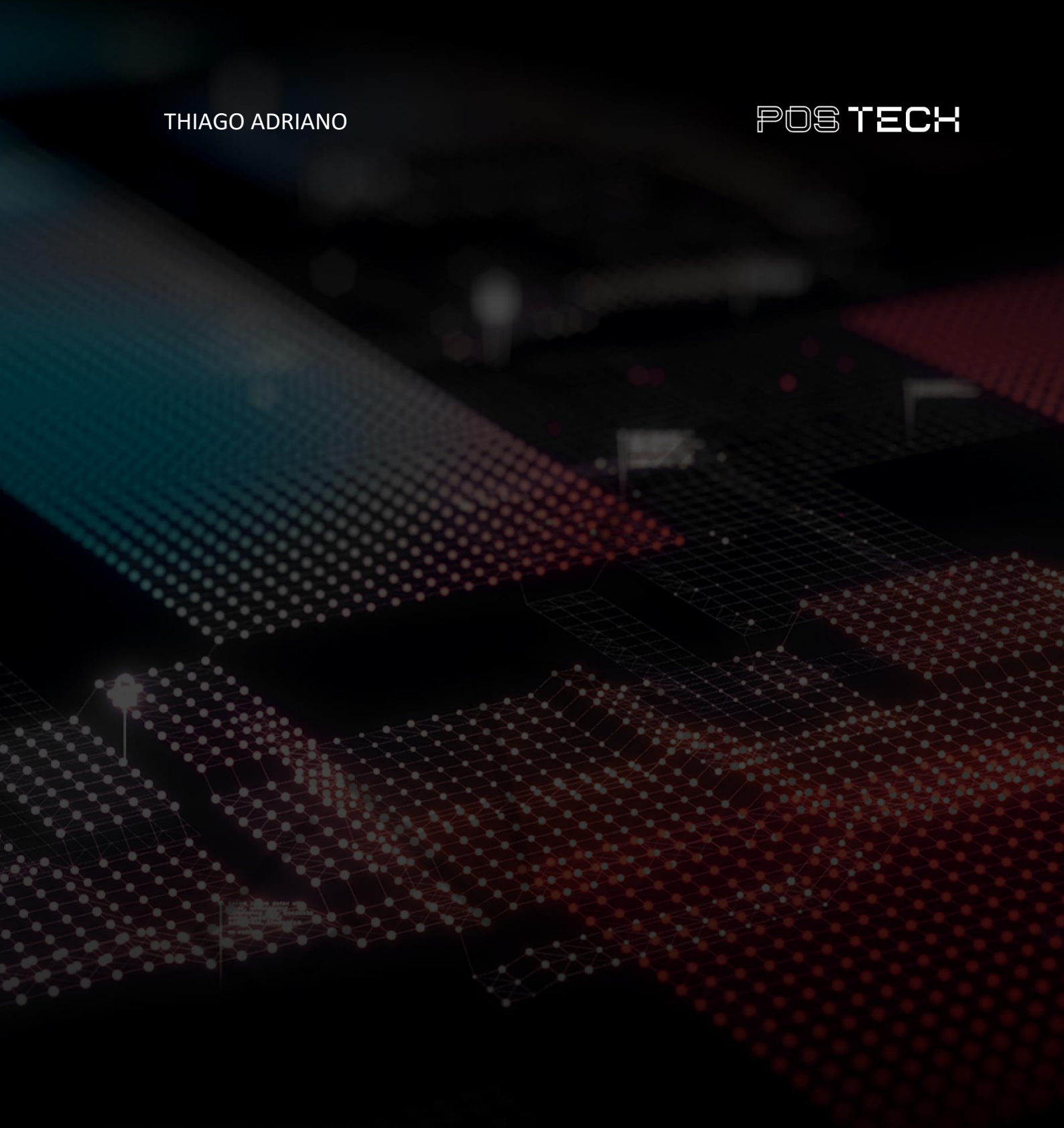
THIAGO ADRIANO



SOFTWARE ARCHITECTURE

KUBERNETES

AULA 07

SUMÁRIO

O QUE VEM POR AÍ?

Nesta aula, você aprenderá o que são as probes no Kubernetes e como elas podem ser usadas para garantir a disponibilidade e a confiabilidade dos aplicativos em containers.

HANDS ON

Nesta aula prática, os docentes demonstrarão como trabalhar com as probes do Kubernetes para garantir a disponibilidade e a confiabilidade dos aplicativos em containers. Eles mostrarão como configurar as diferentes probes, como a Liveness, a Readiness e a Startup, e como testar o comportamento do Kubernetes em diferentes cenários.

SAIBA MAIS

PROBES

As probes do Kubernetes são recursos importantes para garantir a confiabilidade dos aplicativos em um cluster Kubernetes. Elas são usadas para verificar a saúde de um aplicativo em execução em um Pod, garantindo que ele esteja respondendo corretamente às solicitações.

Elas são definidas como parte do manifesto do Pod Kubernetes, que é um arquivo YAML que descreve como o aplicativo deve ser implantado e executado. As configurações de probes incluem o tipo de probe, o caminho de solicitação e os intervalos de verificação.

Usar probes no Kubernetes é importante porque garante a confiabilidade do aplicativo em um cluster. As probes ajudam a garantir que o aplicativo esteja sempre em um estado saudável e pronto para receber solicitações, o que é especialmente importante em ambientes de produção. Sem as probes, os aplicativos podem falhar silenciosamente, sem que ninguém perceba, o que pode levar a interrupções do serviço e afetar negativamente a experiência do usuário.

Existem três tipos de probes no Kubernetes: Liveness, Readiness e Startup. A seguir aprenderemos cada um destes tipos.

Liveness é um tipo de probe que verifica se um aplicativo está em execução e respondendo às solicitações corretamente. A probe de Liveness ajuda a garantir a disponibilidade do aplicativo em um cluster Kubernetes, verificando regularmente se ele está em um estado saudável. Se o aplicativo não estiver respondendo ou estiver em um estado inoperante, a probe de Liveness reiniciará o Pod automaticamente.

A probe de Liveness é definida como parte do manifesto do Pod Kubernetes, que é um arquivo YAML que descreve como o aplicativo deve ser implantado e executado. As configurações da probe de Liveness incluem o tipo de probe (Liveness), o caminho de solicitação e os intervalos de verificação.

Por padrão, a probe de Liveness usa um endpoint HTTP para verificar a disponibilidade do aplicativo.

O fluxo seria o seguinte: o kubelet envia uma solicitação HTTP GET para o endpoint especificado e espera uma resposta 200 OK. Se a resposta for 200 OK, o aplicativo é considerado saudável e a verificação é bem-sucedida. Caso contrário, o Kubernetes assume que o aplicativo não está em execução corretamente e reinicia o pod.

No entanto, a probe de Liveness não se limita apenas ao uso de endpoints HTTP. Outros tipos de probes, como TCP e Exec, também podem ser usadas para verificar a disponibilidade do aplicativo. Por exemplo, uma probe de Liveness do tipo TCP pode ser usada para verificar se um aplicativo está em execução e ouvindo em uma porta específica.

Já a probe de Readiness é um recurso do Kubernetes que verifica se um aplicativo está pronto para receber solicitações; esta probe ajuda a garantir a confiabilidade do aplicativo em um cluster Kubernetes, garantindo que ele esteja pronto para lidar com solicitações antes que elas sejam enviadas.

As configurações da probe de Readiness incluem o tipo de probe (Readiness), o caminho de solicitação e os intervalos de verificação.

Ao contrário da probe de Liveness, que verifica se o aplicativo está em execução, a probe de Readiness verifica se o aplicativo está pronto para receber solicitações. Isso significa que, mesmo que o aplicativo esteja em execução, ele ainda pode não estar pronto para lidar com solicitações de entrada. Por exemplo, o aplicativo pode precisar de algum tempo para carregar dados ou inicializar como uma conexão com um banco de dados antes de estar pronto para lidar com solicitações.

A probe de Readiness é especialmente importante em aplicativos que exigem algum tempo para iniciar ou carregar dados.

Por padrão, a probe de Readiness do Kubernetes usa um endpoint HTTP para verificar a disponibilidade do aplicativo, o Kubernetes envia uma solicitação HTTP GET para o endpoint especificado e espera uma resposta 200 OK. Se a resposta for 200 OK, o aplicativo é considerado pronto para receber solicitações e a verificação é bem-sucedida. Caso contrário, o Kubernetes assume que o aplicativo não está pronto para receber solicitações e o remove do balanceador de carga.

Já a probe de Startup verifica se um aplicativo está sendo iniciado corretamente. Ela é importante em aplicativos que exigem algum tempo para inicializar e configurar corretamente antes de estar pronto para lidar com solicitações.

A probe de Startup é definida como parte do manifesto do Pod Kubernetes, que é um arquivo YAML que descreve como o aplicativo deve ser implantado e executado.

Ao contrário da probe de Liveness e da probe de Readiness, que verificam se o aplicativo está em execução e pronto para lidar com solicitações, respectivamente, a probe de Startup verifica se o aplicativo está sendo iniciado corretamente. Isso significa que ela pode ser usada para verificar se o aplicativo está sendo inicializado e configurado corretamente antes de estar pronto para lidar com solicitações.

Por padrão, a probe de Startup do Kubernetes usa um endpoint HTTP para verificar a disponibilidade do aplicativo. O Kubernetes envia uma solicitação HTTP GET para o endpoint especificado e espera uma resposta 200 OK. Se a resposta for 200 OK, o aplicativo é considerado iniciado e configurado corretamente e a verificação é bem-sucedida. Caso contrário, o Kubernetes assume que o aplicativo não está sendo iniciado e configurado corretamente e reinicia o Pod.

Assim como as outras probes, as configurações da probe de Startup podem ser ajustadas para atender às necessidades específicas do aplicativo e do ambiente.

Por exemplo, é possível configurar os intervalos de verificação da probe de Startup para verificar se o aplicativo está sendo iniciado e configurado corretamente com mais ou menos frequência, dependendo da carga e da importância do aplicativo.

Agora que já sabemos o que são as probes do Kubernetes, vejamos algumas das boas práticas que podemos utilizar quando trabalhamos com elas:

- É importante usar todas as três probes (Liveness, Readiness e Startup) para garantir a confiabilidade contínua dos aplicativos em um cluster.
- Configure os intervalos de verificação: ajuste os intervalos de verificação para atender às necessidades específicas do aplicativo e do ambiente. Por exemplo, é possível configurar os intervalos de verificação para verificar com mais ou menos frequência, dependendo da carga e da importância do aplicativo.
- Para evitar conflitos, use endpoints diferentes para cada tipo de probe.

- Use as configurações padrão do Kubernetes sempre que possível, pois elas foram projetadas para funcionar bem em uma variedade de cenários.
- As configurações avançadas, como os atrasos de inicialização e os intervalos de verificação personalizados, podem ser úteis em alguns casos, mas devem ser usadas com cuidado para evitar efeitos colaterais indesejados.
- Teste as probes regularmente para garantir que elas estejam funcionando corretamente e detectando problemas com o aplicativo.
- Use ferramentas de monitoramento, como Prometheus, para monitorar as probes e o desempenho do aplicativo em um cluster Kubernetes.
- Documente as configurações das probes em um manifesto do Pod ou em um arquivo de configuração, para facilitar a manutenção e o gerenciamento de probes em um cluster.
- Monitore as falhas de probe para detectar e solucionar problemas rapidamente e garantir a confiabilidade contínua do aplicativo.

Veja a figura 1 – “Exemplo de probes”, que é a imagem de um arquivo YAML que define um Pod com um container chamado “exemplo-container”, usando a imagem "exemplo:latest".

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: fiap-pod
spec:
  containers:
  - name: api-fiap
    image: api-fiap:latest #somente para exemplo
    ports:
    - containerPort: 80
    livenessProbe:
      httpGet:
        path: /health
        port: 80
      initialDelaySeconds: 30
      periodSeconds: 10
      timeoutSeconds: 5
    readinessProbe:
      httpGet:
        path: /ready
        port: 80
      initialDelaySeconds: 15
      periodSeconds: 5
      timeoutSeconds: 3
    startupProbe:
      httpGet:
        path: /startup
        port: 80
      initialDelaySeconds: 120
      periodSeconds: 30
      timeoutSeconds: 10
```

Figura 1 – Exemplo de probes
Fonte: Elaborado pelo autor (2023)

Analisando este arquivo, podemos observar que temos um container que expõe a porta 80 e usa os três probes para garantir que o aplicativo esteja em um estado saudável e pronto para lidar com solicitações.

A probe de Liveness é definida para verificar a disponibilidade do aplicativo a cada 10 segundos, com um tempo de espera máximo de 5 segundos. A probe de Readiness é definida para verificar se o aplicativo está pronto para receber solicitações a cada 5 segundos, com um tempo de espera máxima de 3 segundos. A probe de Startup é definida para verificar se o aplicativo está sendo iniciado corretamente com um atraso inicial de 120 segundos, verificando a cada 30 segundos e com um tempo de espera máximo de 10 segundos.

O QUE VOCÊ VIU NESTA AULA?

Nesta aula, você aprendeu a como trabalhar com as probes do Kubernetes para garantir a disponibilidade e a confiabilidade dos aplicativos em containers. Você aprendeu sobre os três tipos principais de probes: Liveness, Readiness e Startup, e como configurá-las para monitorar a saúde do seu aplicativo.

O que achou do conteúdo? Conte-nos no Discord! Estamos disponíveis na comunidade para tirar dúvidas, fazer networking, enviar avisos e muito mais.

REFERÊNCIAS

BASS, L. CLEMENTS, P. KAZMAN, R. **Software Architecture in Practice**. 3rd ed. [s.l.]: Addison-Wesley Professional, 2012.

PALAVRAS-CHAVE

Kubernetes. Probes. Startup.