

# Ephemeral Storage



Você está atento a ele???



BL<sup>2</sup>  
TI direto ao  
ponto!

# Persistent Storage

**Armazenamento persistente:** dados **podem** ficar armazenados por um longo tempo, independente de sua utilização.

Ex: banco de dados



# Ephemeral Storage

**Armazenamento efêmero:** temporário ou transitório, dados ficam armazenados somente pelo tempo mínimo no qual são necessários ou estão sendo utilizados.

- **Não há garantias sobre sua durabilidade.**



# Pods são efêmeros

Os *Pods* por definição são efêmeros, possuindo um ciclo de vida bem definido [1]. A grande vantagem dessa característica é permitir que eles possam ser criados e destruídos conforme necessidade, para atendimento das exigências do *deployment*, conforme vimos no vídeo 6.

# Armazenamento Efêmero no Kubernetes

- **Nós:** Podem usar dispositivos locais (discos) ou algumas vezes a própria memória RAM para gravar os dados efêmeros.
  - Se um nó apresentar falha, **seus dados efêmeros poderão ser perdidos**
  - Exemplos: *logs dos containers*, imagens e camadas de escrita dos *containers* em execução

# Armazenamento Efêmero no Kubernetes

- **Pods:** Além da camada de escrita dos *containers* em execução, podem ser montados volumes efêmeros nos *containers* [2]:
  - configMaps
  - secrets
  - downward API
  - emptyDir volumes



```
FROM tomcat
```

```
# Modo não interativo do terminal  
ARG DEBIAN_FRONTEND=noninteractive
```

```
# Instala dependências da imagem  
RUN apt update && apt install gettext-base -y
```

```
# Apaga webapps padrão do Tomcat  
RUN rm -rf /usr/local/tomcat/webapps/*
```



The diagram illustrates the architecture of a PostgreSQL cluster in a Kubernetes environment, showing the flow of logs from a pod to a central log storage and then to a registry.

**Cluster:** The main environment, represented by a large dashed green box. It contains:

- Pod A:** A dashed pink box representing a Kubernetes pod. It contains:
  - Container A1:** A dashed blue box representing a container. It contains a stack of four colored blocks (green, light blue, light blue, dark blue), representing the PostgreSQL instance.
- Log Storage:** A red cylinder icon with a red alarm clock, representing a central log storage or monitoring component.
- Log Viewer:** A black box containing a log snippet, representing a tool for viewing logs.

**Registry:** A dashed black box representing a log registry. It contains:

- NGINX Logo:** A green logo with the text "NGINX".
- Log Storage:** A stack of four colored blocks (light blue, light blue, light blue, dark blue), representing the log storage within the registry.

**Log Flow:** Red arrows indicate the flow of logs:

- From **Container A1** (specifically from the green block) to the **Log Storage** cylinder.
- From the **Log Storage** cylinder to the **Log Viewer** box.
- From the **Log Viewer** box to the **Registry** (specifically to the stack of blocks).

**Log Snippet:** The log viewer displays the following log entries:

```
2021-09-22 08:38:40 -03 [1454-1] [unknown]@[unknown] LOG: incomplete startup packet
2021-09-22 08:39:49 -03 [1401-2] LOG: database system was not properly shut down; automa
2021-09-22 08:39:49 -03 [1401-3] LOG: invalid record length at 0/1712498
2021-09-22 08:39:49 -03 [1401-4] LOG: redo is not required
2021-09-22 08:39:49 -03 [1401-5] LOG: MultiXact member wraparound protections are now en
2021-09-22 08:39:49 -03 [1319-1] LOG: database system is ready to accept connections
2021-09-22 08:39:49 -03 [1892-1] LOG: autovacuum launcher started
```

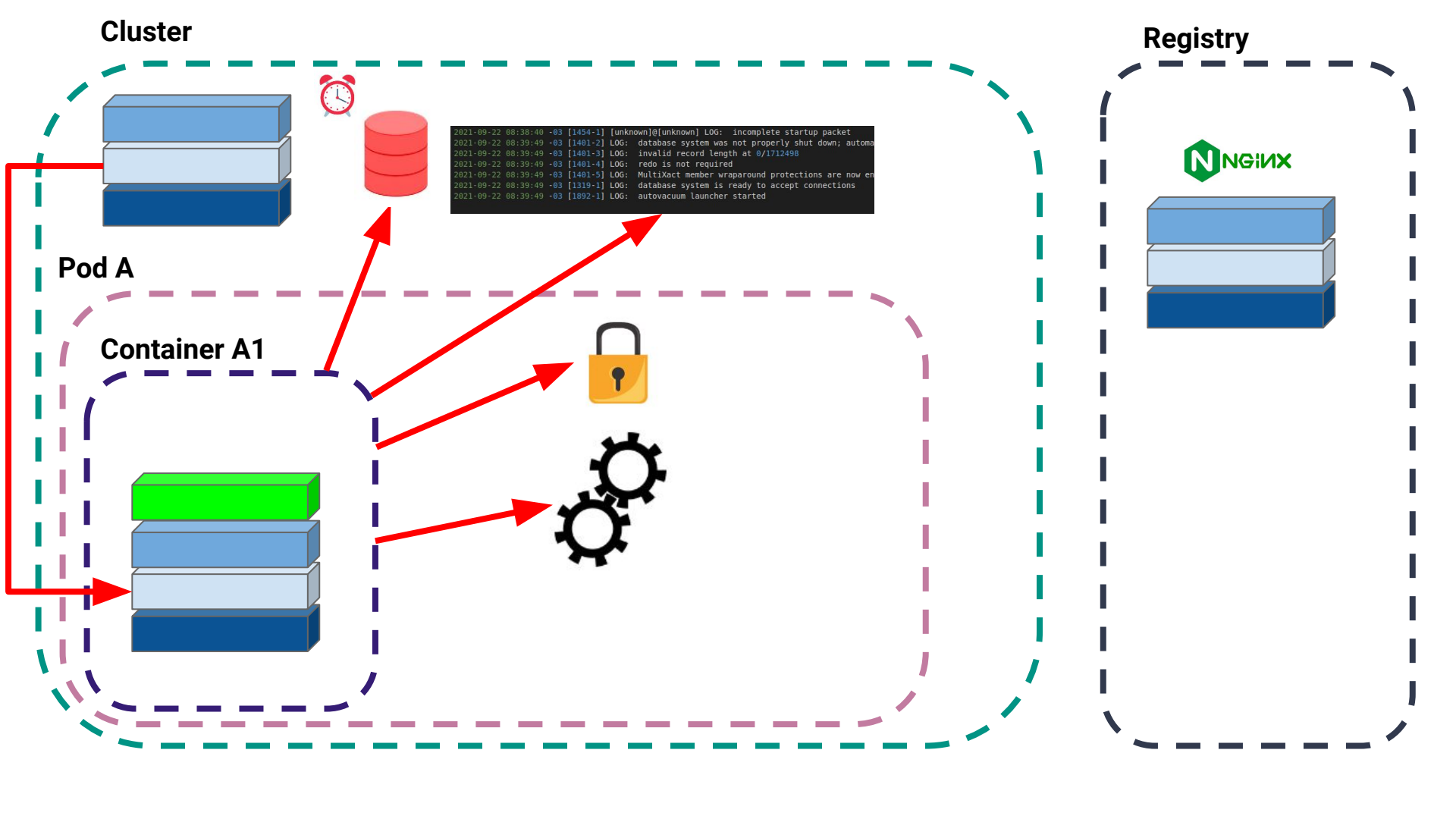
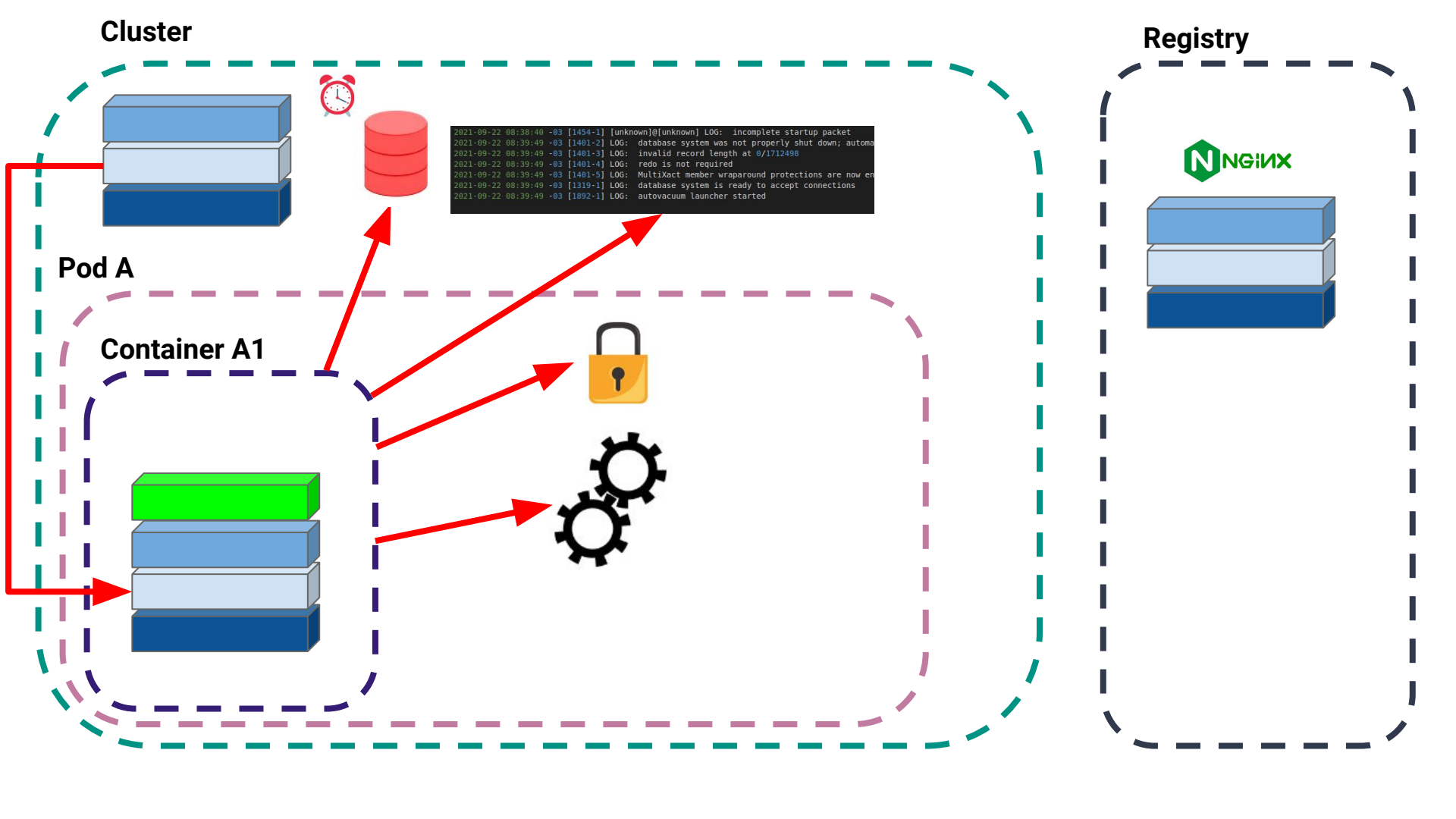
The diagram illustrates the architecture of a PostgreSQL cluster in a Kubernetes environment, showing the flow of logs from a pod to a central log collector and then to a registry.

**Cluster:** The main environment, represented by a large dashed green box. It contains:

- Pod A:** A dashed pink box containing **Container A1** (a dashed blue box). Inside Container A1 is a stack of four colored blocks (green, light blue, light blue, dark blue) representing the PostgreSQL instance.
- Log Collector:** A red alarm clock icon and a red cylinder icon representing a central log collector.
- Log Output:** A black box containing PostgreSQL log entries, such as: `2021-09-22 08:38:40 -03 [1454-1] [unknown]@[unknown] LOG: incomplete startup packet`, `2021-09-22 08:39:49 -03 [1401-2] LOG: database system was not properly shut down; automatic recovery in progress`, `2021-09-22 08:39:49 -03 [1401-3] LOG: invalid record length at 0/1712498`, `2021-09-22 08:39:49 -03 [1401-4] LOG: redo is not required`, `2021-09-22 08:39:49 -03 [1401-5] LOG: MultiXact member wraparound protections are now enabled`, `2021-09-22 08:39:49 -03 [1319-1] LOG: database system is ready to accept connections`, and `2021-09-22 08:39:49 -03 [1892-1] LOG: autovacuum launcher started`.

**Registry:** A dashed black box on the right containing the **NGINX** logo and a stack of four colored blocks (light blue, light blue, light blue, dark blue) representing the registry storage.

**Log Flow:** Red arrows indicate the flow of logs from the PostgreSQL instance in Container A1, through the log collector, to the central log output box, and finally to the registry.





# Perigos do Ephemeral Storage



# Dados não Duradouros

Como nós vimos, os *containers* podem gerar ou fazer uso de dados efêmeros, os quais são destruídos quando o *pod* é removido ou acidentalmente encerrado.

Por isso, suas aplicações devem funcionar de acordo com essa dinâmica.



# Espaço “Invisível”

Não podemos esquecer que a execução de um *container* implica em espaço em disco, que muitas vezes não levamos em consideração.

Por isso, devemos ficar atentos com:

- crescimento dos *logs*
- tamanhos das imagens
- dados temporários gerados



# Falhas

**Pod eviction:** O pod é finalizado para o Kubernetes recuperar espaço. Ocorre quando o espaço em disco utilizado é maior que um certo limiar (por padrão 85%) [3]

- 



# Falhas

- **\*\*kubelet: I0114 03:37:08.639450\*\* 4721**  
**image\_gc\_manager.go:300] [imageGCManager]: Disk usage**  
**on image filesystem is at 95% which is over the high**  
**threshold (85%).** Trying to free 3022784921 bytes down to  
the low threshold (80%).
- **\*\*kubelet: I0114 00:15:51.130499\*\* 4781**  
**eviction\_manager.go:344] eviction manager: must evict**  
**pod(s) to reclaim ephemeral-storage**

# Falhas

- The node was low on resource: ephemeral-storage. Container yyyy was using xxxxKi, which exceeds its request of 0.

# Falhas

**Problemas de schedule:** O pod não é inicializado em nós devido a indisponibilidade de espaço.

- | NAME     | READY | STATUS  | RESTARTS | AGE |
|----------|-------|---------|----------|-----|
| frontend | 0/2   | Pending | 0        | 10s |

# Falta de Limites

Por padrão, não há limites de *ephemeral storage* para os containers, o que pode facilitar a ocorrência dos problemas anteriores.

Felizmente o Kubernetes permite definir o mínimo de *ephemeral storage* que um *container* precisa para poder ser inicializado, e o máximo que ele pode utilizar.





# Pontos de atenção na utilização de Ephemeral Storage



- Dados não duradouros
- Espaço “invisível”
- Falhas
- Falta de Limites

# Referências

- [1] [Pod | Kubernetes Engine Documentation](#)
- [2] [Ephemeral Volumes](#)
- [3] [Kubernetes Pods Terminated - Exit Code 137](#)