Cenário 01

- 100 novas URLs por minuto;

- Ratio de leitura e escrita de 50:1

Tráfego estimado

- Estimado em 5.000 RPM

Armazenamento Estimado

DATETIME = 8 bytes
VARCHAR(n) = n * 2 bytes
INT = 4 bytes

- Cada row tem estimado 750 bytes;

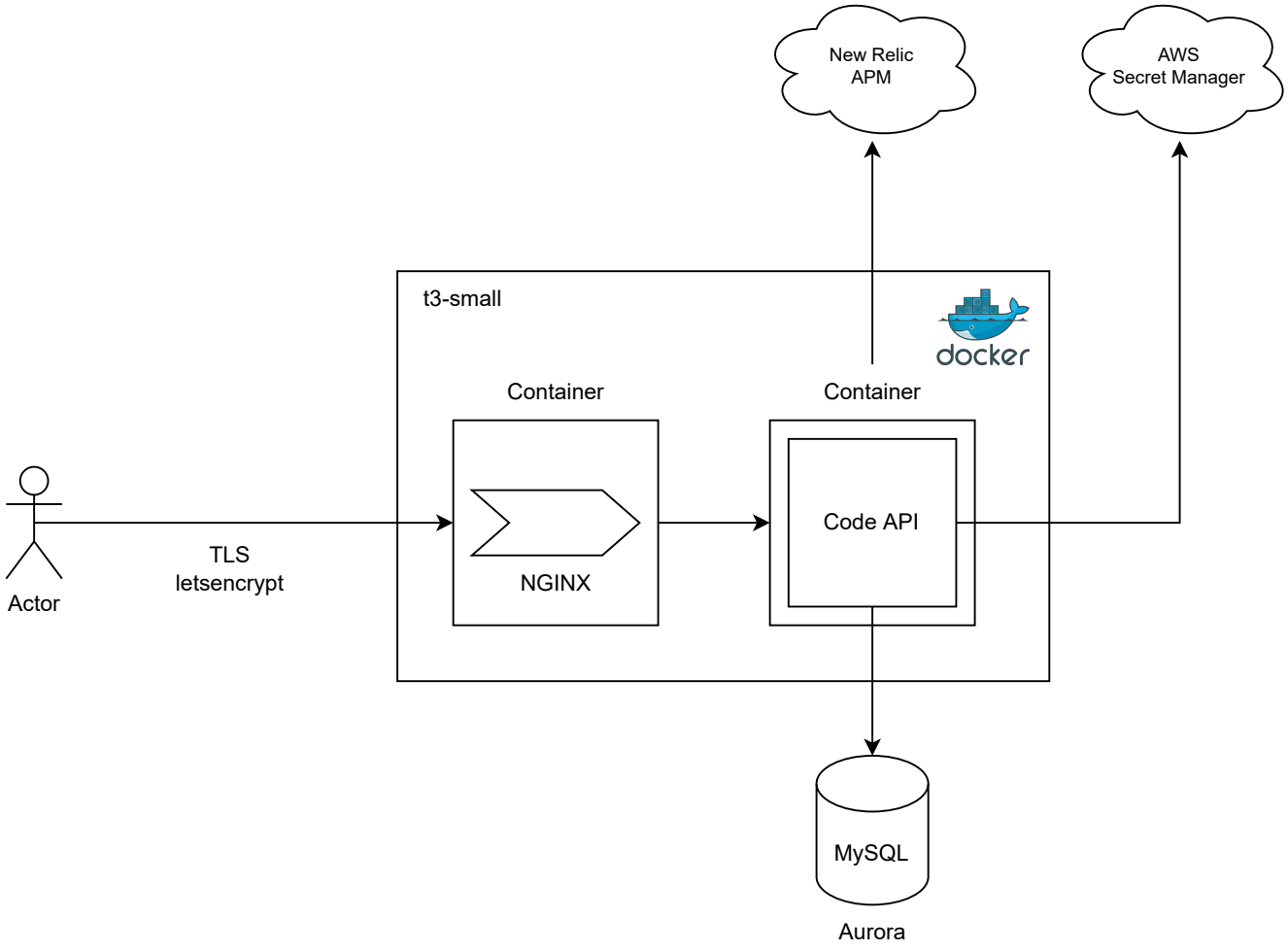
- 10 URLs por minuto X 60 minutos X
24 horas X 365 dias = 5.5 M rows por ano;

- Vamos precisar de 4.5 GB por ano;

Armazenamento

Table urls	
PK	hash: VARCHAR(6)
	target_url: VARCHAR
	created_at: DATETIME(3)
	expires_at: DATETIME(3)
FK	user_id: VARCHAR(36)

Table users	
PK	id: VARCHAR(36)
	name: VARCHAR(255)
unique	email: VARCHAR(255)
	password: VARCHAR(255)
	created_at: DATETIME(3)



Cenário 02

- 100 novas URLs por minuto;

- Ratio de leitura e escrita de 50:1

Tráfego estimado

- Estimado em 5.000 RPM

Armazenamento Estimado

DATETIME = 8 bytes
VARCHAR(n) = n * 2 bytes
INT = 4 bytes

- Cada row tem estimado 750 bytes;

- 10 URLs por minuto X 60 minutos X
24 horas X 365 dias = 5.5 M rows por ano;

- Vamos precisar de 4.5 GB por ano;

Armazenamento

Table urls	
PK	hash: VARCHAR(6)
	target_url: VARCHAR
	created_at: DATETIME(3)
	expires_at: DATETIME(3)
FK	user_id: VARCHAR(36)

Table users	
PK	id: VARCHAR(36)
	name: VARCHAR(255)
unique	email: VARCHAR(255)
	password: VARCHAR(255)
	created_at: DATETIME(3)

Cache

- Algoritmo para remover do cache os items não utilizadas;

- Escolhemos LRU dentre as LFU e FIFO

- Estratégia write-through de cache;

- Uso estimado: 100 RPM X 60 minutos X 24 horas = ~ 144K Requests

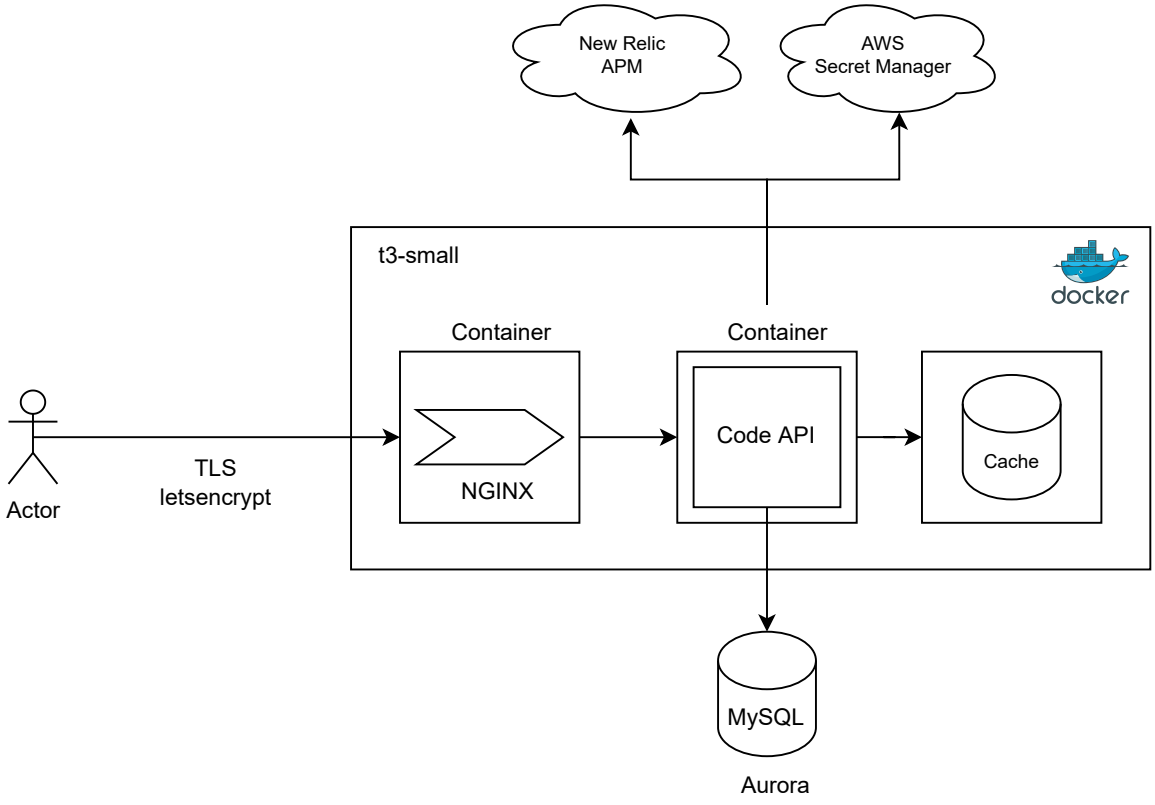
- Resultado: 0,8 * 144K * 750 bytes = 0.1GB

Gerador de hash

- Precisamos de um hash de 6 digítos de (A-Z, 0-9) = 65 BI de valores;

- Usar o algoritmo MD5 (para finalidade está ok)

- Salt randomico por geração



Cenário 03

- 100 novas URLs por minuto;

- Ratio de leitura e escrita de 500:1

Tráfego estimado

- Estimado em 5.000 RPM

Armazenamento Estimado

DATETIME = 8 bytes
VARCHAR(n) = n * 2 bytes
INT = 4 bytes

- Cada row tem estimado 750 bytes;

- 100 URLs por minuto X 60 minutos X 24 horas X 365 dias = 5.5 M rows por ano;

- Vamos precisar de 45 GB por ano;

Armazenamento

Table urls	
PK	hash: VARCHAR(6)
	target_url: VARCHAR
	created_at: DATETIME(3)
	expires_at: DATETIME(3)
FK	user_id: VARCHAR(36)

Table users	
PK	id: VARCHAR(36)
	name: VARCHAR(255)
unique	email: VARCHAR(255)
	password: VARCHAR(255)
	created_at: DATETIME(3)

Cache

- Algoritmo para remover do cache os items não utilizadas;

- Escolhemos LRU dentre as LFU e FIFO

- Estratégia write-through de cache;

- Uso estimado: 100 RPM X 60 minutos X 24 horas = ~ 144K Requests

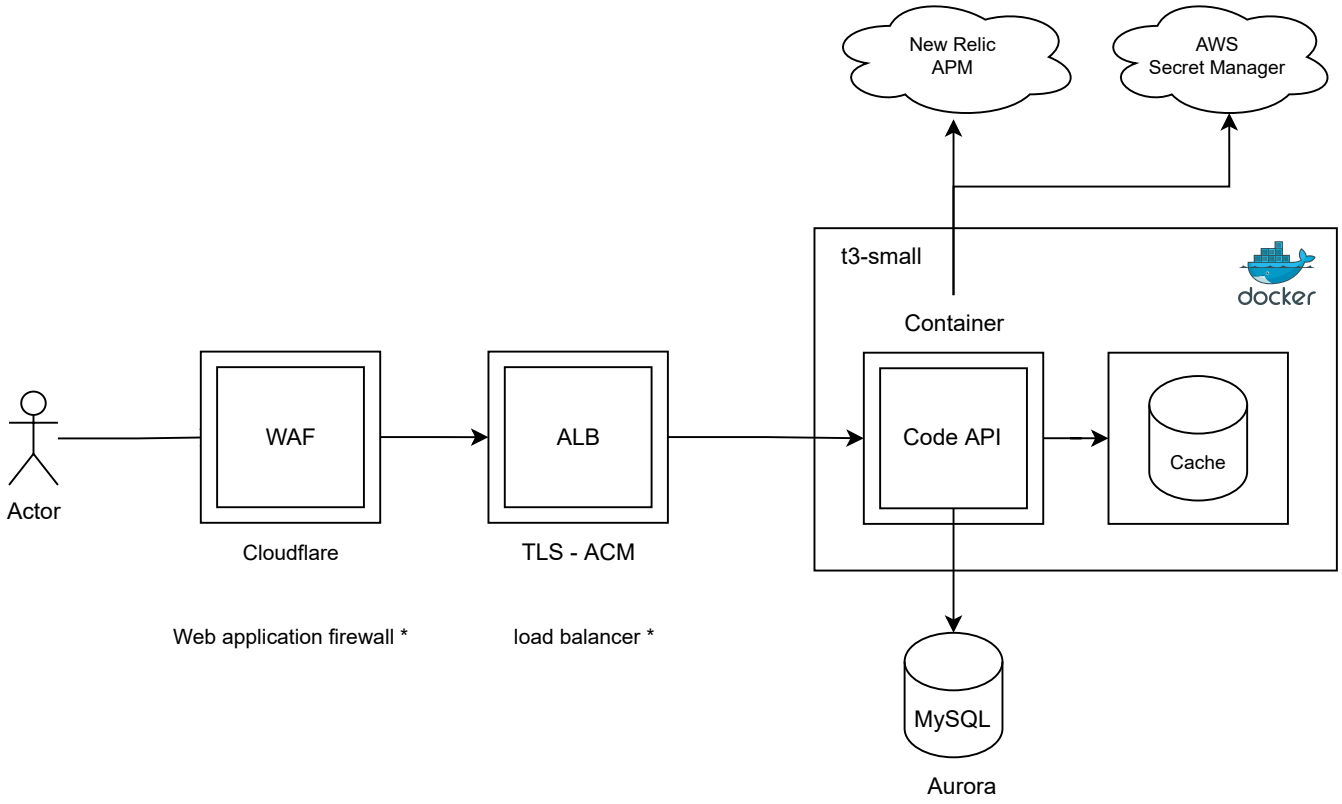
- Resultado: 0,8 * 144K * 750 bytes = 0.1GB

Gerador de hash

- Precisamos de um hash de 6 digítos de (A-Z, 0-9) = 65 BI de valores;

- Usar o algoritmo MD5 (para finalidade está ok)

- Salt randomico por geração



Cenário 04

- 1000 novas URLs por minuto;

- Ratio de leitura e escrita de 500:1

- Adicionar telemetria e métricas de negócio

Tráfego estimado

- Estimado em 500K RPM

-720 M reqs por dia

Armazenamento Estimado

DATETIME = 8 bytes
VARCHAR(n) = n * 2 bytes
INT = 4 bytes

- Cada row tem estimado 750 bytes;

- 1000 URLs por minuto X 60 minutos X 24 horas X 365 dias
= 525.600.000 rows por ano;

- Vamos precisar de 450 GB por ano;

Armazenamento

Table urls	
PK	hash: VARCHAR(6)
	target_url: VARCHAR
	created_at: DATETIME(3)
	expires_at: DATETIME(3)
FK	user_id: VARCHAR(36)

Table users	
PK	id: VARCHAR(36)
	name: VARCHAR(255)
unique	email: VARCHAR(255)
	password: VARCHAR(255)
	created_at: DATETIME(3)

Cache

- Algoritmo para remover do cache os items não utilizadas;

- Escolhemos LRU dentre as LFU e FIFO

- Estratégia write-through de cache;

- Uso estimado: 100 RPM X 60 minutos X 24 horas = ~ 144K Requests

- Resultado: 0,8 * 144K * 750 bytes = 0.1GB

Gerador de hash

- Precisamos de um hash de 6 digítos de (A-Z, 0-9) = 65 BI de valores;

- Usar o algoritmo MD5 (para finalidade está ok)

- Salt randomico por geração