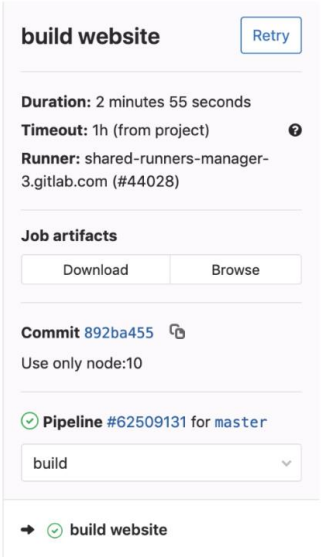
Uso de caché para optimizar la construcción

En esta lección, vamos a revisar cómo podemos utilizar las cachés con el fin de optimizar la velocidad de construcción.

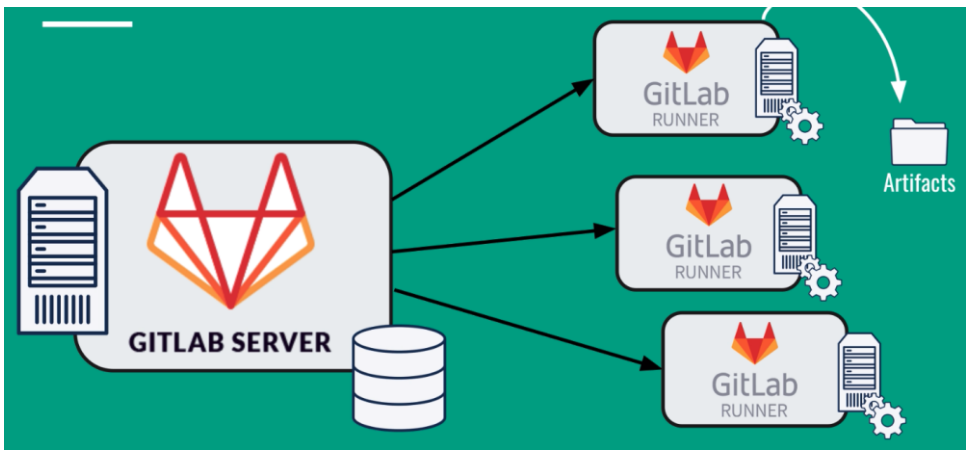
Probablemente hayas notado que algunos de los Jobs necesitan mucho tiempo para ejecutarse, especialmente los Jobs de construcción:



El Job “build website” necesita descargar algunas dependencias antes de poder ejecutarse, y esto lleva mucho tiempo.

Ahora, en comparación con otros servidores de ci más tradicionales como Jenkins, este tiempo extra que se tarda en descargar la imagen Docker, descargar las dependencias y todo eso, puede parecer una eternidad.

El por qué GitLab se comporta de esta manera se explica en la arquitectura de GitLab:



Previamente dijimos que cada Job se inicia en un entorno limpio, por lo que descargamos una imagen Docker limpia y sobre ella descargamos el código de nuestro repositorio Git, siempre desde cero, nunca reutilizando archivos previamente descargados.

Y si te estás preguntando, ¿qué podemos cachear?, la respuesta es: “todo lo que estamos descargando”.

Así que el candidato ideal para el almacenamiento en caché son esas dependencias externas del proyecto que no estamos almacenando y que necesitamos descargar todo el tiempo.

Comencemos a optimizar nuestro Job “build”, ya que es el que se encarga de descargar todas las dependencias en la carpeta node_modules.

Añadamos entonces una nueva instrucción llamada “cache”:

```
build website:
  stage: build
  cache:
    key: ${CI_COMMIT_REF_SLUG}
    paths:
      - node_modules/
  script:
    - echo $CI_COMMIT_SHORT_SHA
    - npm install
```

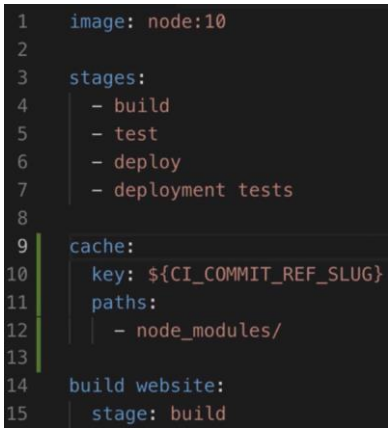
Cuando utilizamos la instrucción “cache”, tenemos que especificar una “key” y un “path”. Con el path, le decimos a GitLab qué guardar, y sólo necesitamos guardar el contenido de la carpeta node_modules, ya que es lo que se descarga constantemente y necesitaremos una y otra vez.

La segunda parte especifica el “key”, lo cual sirve para identificar esta caché y poder utilizarla después, podemos utilizar variables d eentorno o oncluso simples strings, como por ejemplo “my cache”:

```
build website:
  stage: build
  cache:
    key: mycache
    paths:
      - node_modules/
  script:
    - echo $CI_COMMIT_SHORT_SHA
    - npm install
```

En la práctica, es una buena idea tener una caché cuyo key se basa en el nombre del branch. En neustro caso, utilizaremos una variable de entorno que hace referencia al branch en el que estamos trabajando.

Hay una cosa más que debemos revisar, y esto es válido con un montón de cosas en gitLab. Podemos especificar la caché para un solo Job específico, como lo hicimos, agregamos la caché dentro del Job “build website”. Peor también podemos agregar la caché globalmente para todo el pipeline, de forma que podemos sacar el segmento de caché al exterior del Job:



```yaml
image: node:10

stages:
  - build
  - test
  - deploy
  - deployment tests

cache:
  key: ${CI_COMMIT_REF_SLUG}
  paths:
    - node_modules/

build website:
  stage: build
```