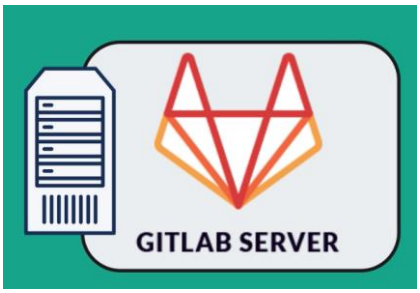
Introducción – GitLab Architecture

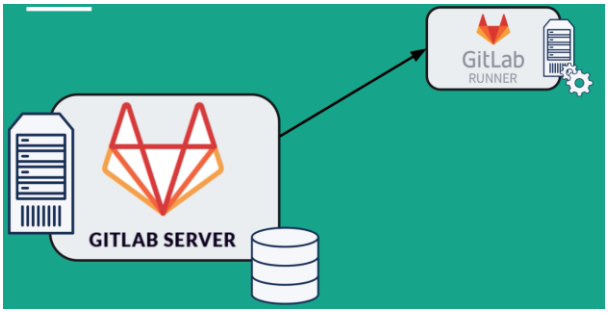
Ahora veamos cómo funciona exactamente GitLab desde el punto de vista de la arquitectura.

Lo primero que necesitamos es el GitLab Server:

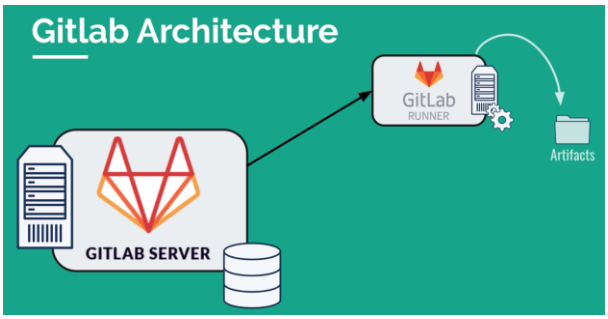


GitLab Server nos ofrece una interfaz, con la que podemos crear repositorios y gestionar todo lo relacionado con nuestro proyecto, y todo lo que hagamos se guardará en base de datos. Ese es el trabajo principal de GitLab Server.

Ahora, tan pronto como creamos un pipeline, ese pipeline también será gestionado por el GitLab Server, peor será delegado al GitLab Runner:



GitLab Server le dirá al GitLab Runner: “esto es lo que tienes que hacer a continuación, descarga la imagen, ejecuta los siguientes pasos, y finalmente, si hay algún artifact, asegúrate de guardarlo.”:

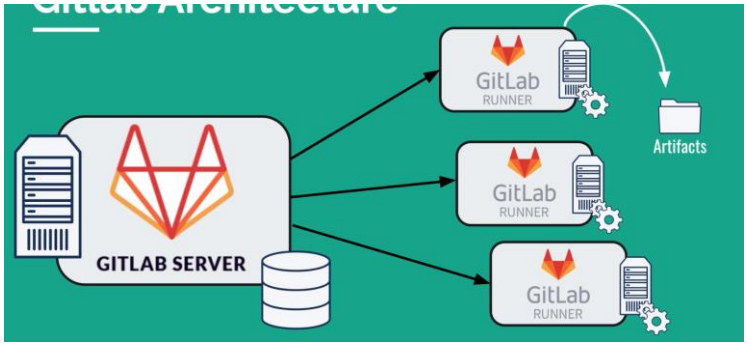


Y eso es todo.

El GitLab Server no se encarga de ejecutar los pasos, si no que gestiona todo el proceso. Se asegura de que el Runner está recogiendo ese trabajo, que la salida del Runner está en algún lugar almacenado, etc. Y esto permite una arquitectura muy escalable porque como mínimo

tendremos el GitLab Server y un GitLab Runner, porque sin un Runner no podríamos ejecutar ningún pipeline.

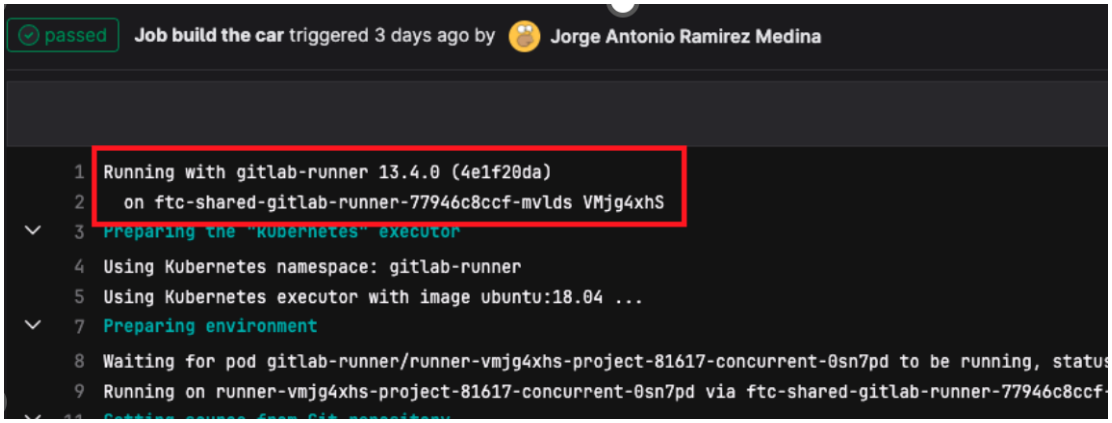
En caso de que sus necesidades sean mucho mayores, puede añadir tantos Runners como necesite:



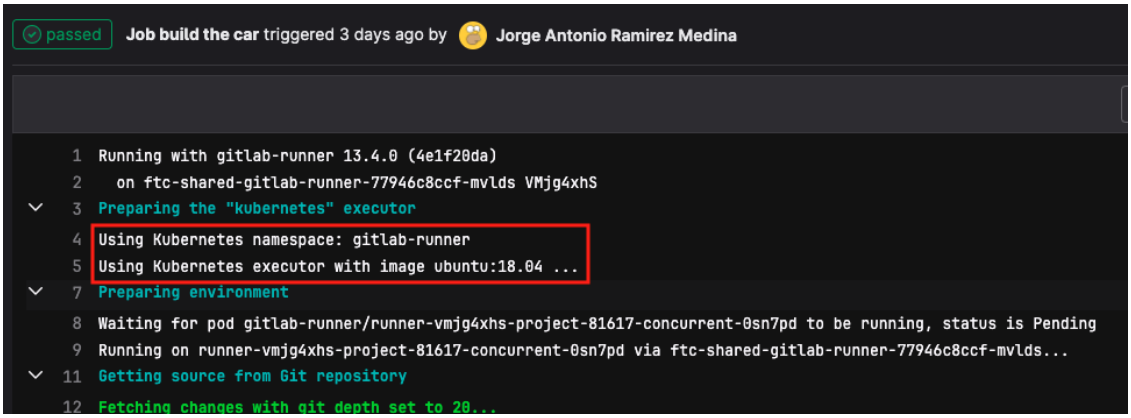
Como puede ver, es fácilmente escalable y puede ocurrir que algunos de sus pipelines tengn necesidades muy especiales o que los utilice sólo durante el día, pero no tanto durante la noche, o viceversa. Así que usted puede escalar los Runners fácilmente hacia arriba o hacia abajo.

En este caso, la arquitectura de GitLab es muy flexible y se ajusta perfectamente al funcionamiento de los Servidores de Integración Continua modernos.

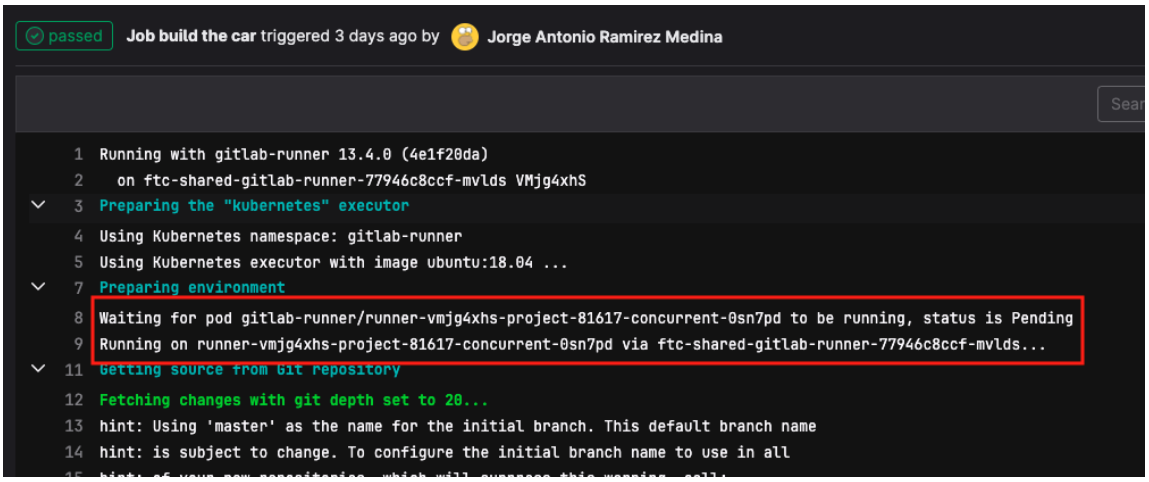
Así, si nos fijamos en el primer Job “build the car”, veremos que éste ha sido ejecutado por un GitLab Runner:



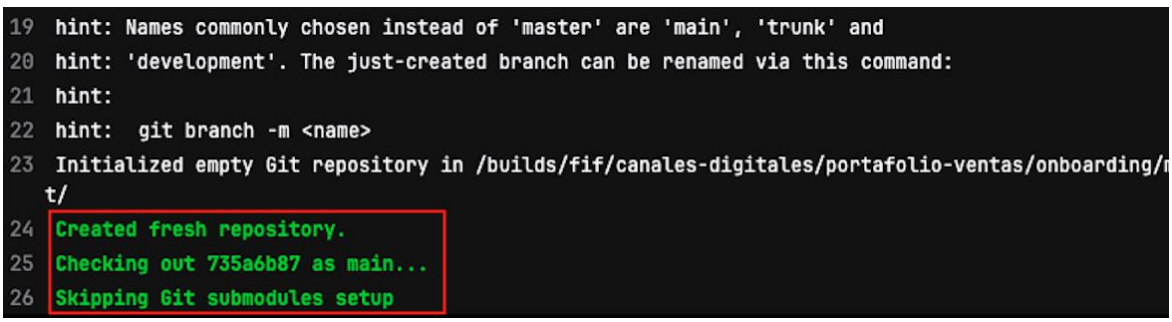
Y verá que, por default, GitLab Runner está usando una Imagen Docker Ubuntu v18.04:



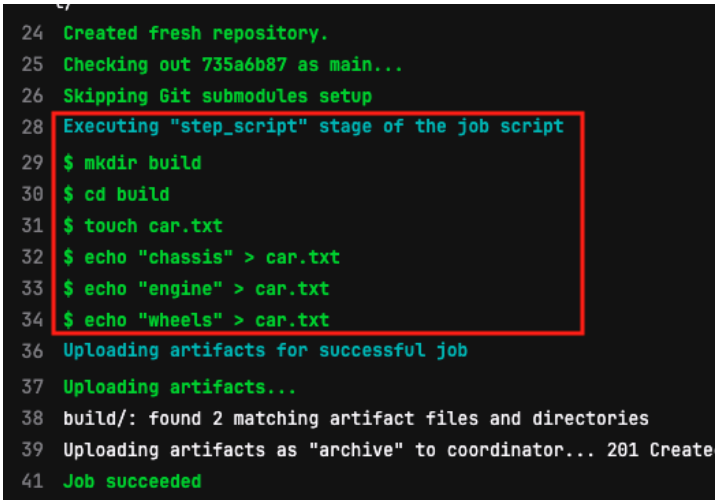
Y esta unidad está utilizando a su vez esta imagen y luego está iniciando el Runner:



Después, clona el proyecto del repositorio:



Después comienza a ejecutar los pasos que hemos puesto en el script:

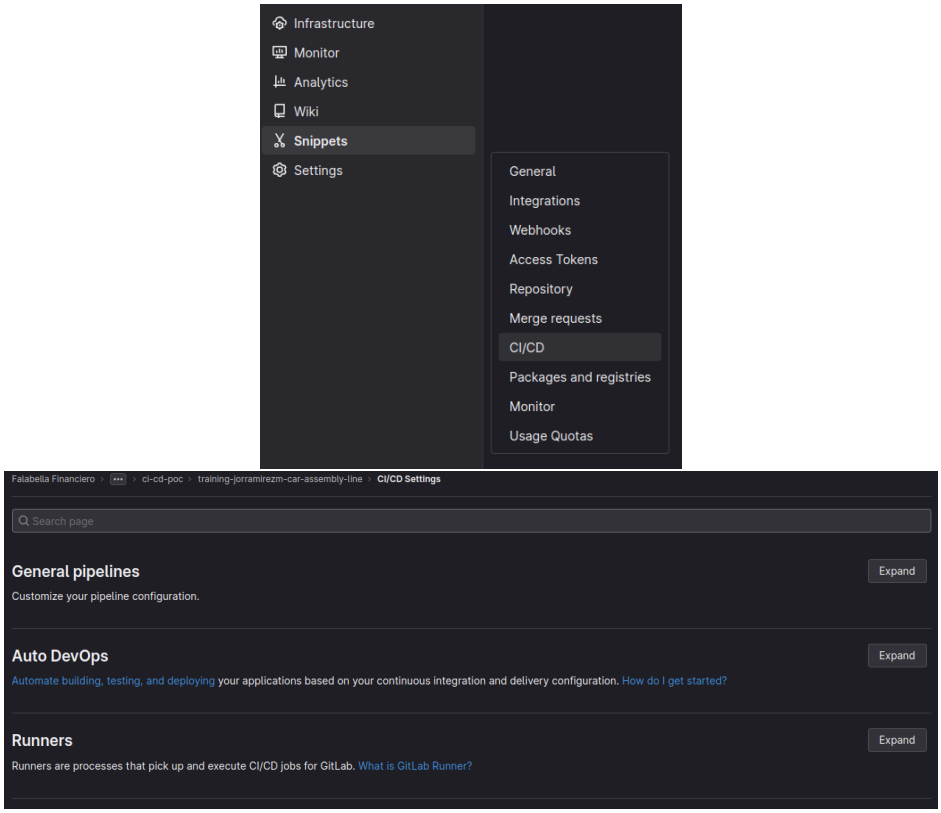


Y el último paso es, una vez generados los artifacts, los sube al "*Coordinator*", que en este caso el Coordinator es el propio GitLab server:

```
Uploading artifacts for successful job
Uploading artifacts...
build/: found 2 matching artifact files and directories
Uploading artifacts as "archive" to coordinator... 201 Created  id=19429894 responseStatus=201 Created token=64_ucZkH
```

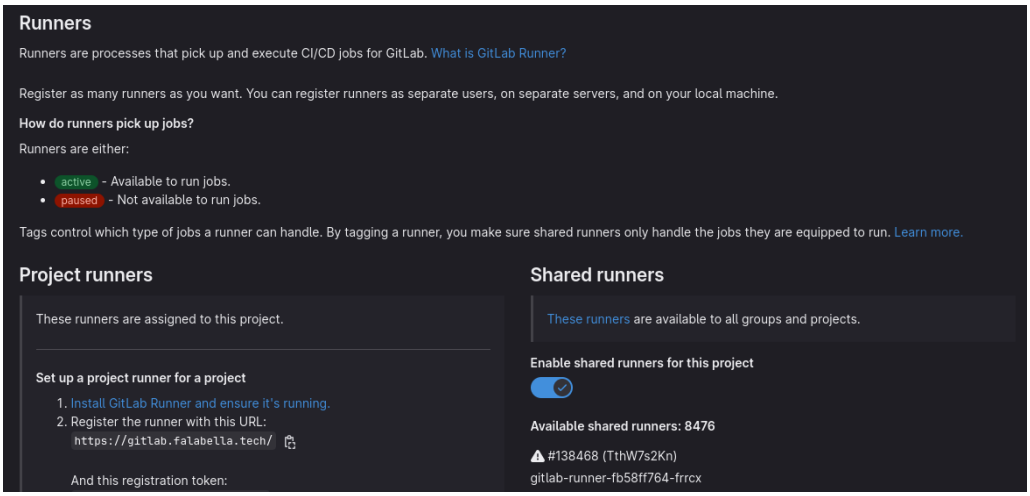
Después de que el Job finaliza de forma exitosa, el Runner detiene esta tarea. Así que tan pronto como esto ha terminado, la imagen se destruye y nadie más tendrá acceso a ella. Lo único que se está guardando en realidad son estos logs que puede inspeccionar para entender lo que ha pasado. Por supuesto, los artifacts del job se pueden descargar e inspeccionar.

Ahora que ya hablamos de Runners, veamos cómo están configurados actualmente. Si usted va a la opción Settings -> CI/CD, podrá encontrar la configuración de los Runners:



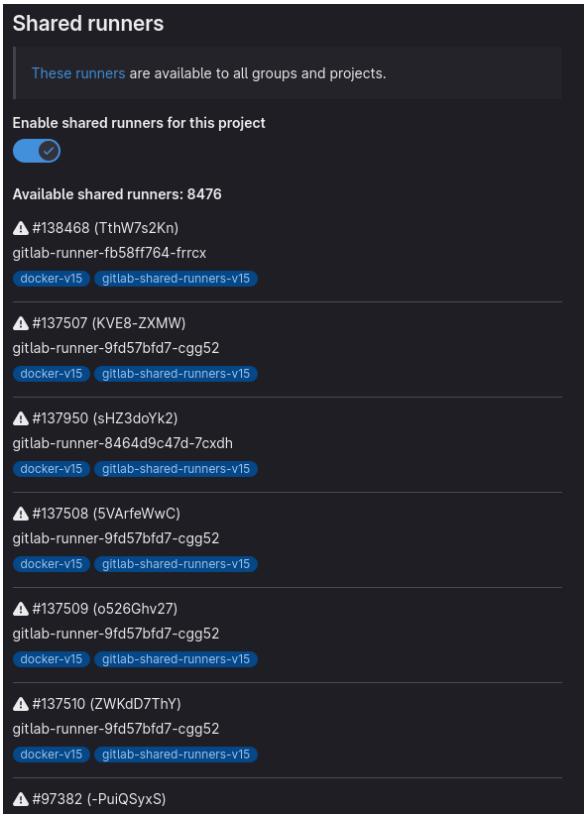
Demos click en Expand para ver el detalle de la configuración de los Runners de este proyecto.

Podemos ver que actualmente, nuestro proycto se está ejecutando con Runners compartidos (Shared), de los cuales tenemos 8,476:



Ahora, estos Shared Runners son proporcionados por GitLab y están disponibles para todos los usuarios de GitLab.

Podemos ver aquí estos Runners:



Pero, además de los Shared Runners, tenemos la posibilidad de crear nuestros propios Runners en caso de necesitar más potencia. Así que, por ejemplo, puede seguir usando los Shared Runners pero tener sus propios Runners. Los Runners personalizados le pueden servir cuando tiene Jobs muy intensivos en el consumo de CPU en los que los Shared Runners tal vez no tienen demasiado poder. Puede crear sus propios Runners con características muy específicas para trabajo muy intensivo y configurar qué proyectos específicos usarán ese Runner.

Así que GitLab es realmente flexible con lo que puede hacer con los Runners.

Esta fue una breve descripción de la arquitectura de GitLab y de lo que ocurre exactamente una vez que se ha iniciado un pipeline, cómo se ejecuta el código y cómo encajan y trabajan todas las piezas de GitLab.