Branches – Condicionar ejecución de Jobs a un branch

En esta lección vamos a echar un vistazo a cómo podemos utilizar los Merge Request y los Branches con el fin de mejorar nuestro flujo de trabajo general en Git cuando usamos GitLab CI.

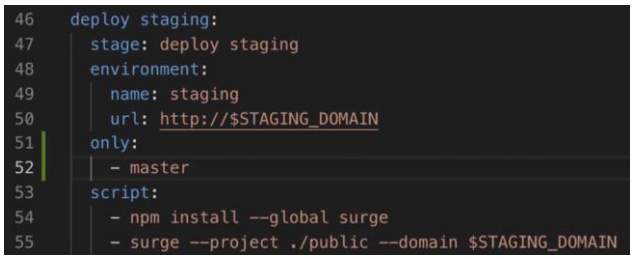
Trabajar con branches es una buena manera de evitar que un branch master no pueda desplegarse. Ya hemos visto anteriormente que cada vez que un desarrollador entra en el branch master y hace un cambio, un ejemplo simple fue la eliminación de un import, entonces todo el branch master ya no es desplegable. Y esto es, por supuesto, muy lamentable y realmente puede afectar a todo el trabajo.

La idea de trabajar con branches es que cada desarrollador o cada nuevo feature pueda ser desplegado a un branch específico, pueda ser probado allí, y cuando esté listo para el merge, pueda hacerce merge a master, y después pueda ser desplegado en producción. El uso de este flujo de trabajo realmente hace que sea mucho más fácil y garantiza que el branch master siempre sea desplegable. Esto es en realidad parte del CD, queremos ser capaces todo el timpo de desplegar el software, y si el branch master no está funcionando porque alguien ha cometido un error y todo deja de funcionar, entonces esto no es realmente ir en dirección del CI/CD.

Existen diferentes estrategias, también conocidas como branches models, para hacer frente a esta situación, GitFlow es un ejemplo de ellas, pero no entraremos en detalles.

Antes de que podamos utilizar branches, hay un par de cosas que tenemos que arreglar acerca de nuestro branch. Por default, si creamos un nuevo branch y hacemos push, nuestro pipeline se ejecutará inmediatamente para ese branch. Definitivamente no queremos desplegar nuestro branch de desarrollo en los Stages de producción ni staging, ni ejecutar las pruebas de producción en nuestro branch. Por suerte, podemos configurar eso, podemos configurar nuestros tres Jobs “deploy staging”, “deploy production”, “production tests” para que solo se ejecuten en el branch master.

Solo tenemos que especificar la instrucción “only” seguido del valor “master”:

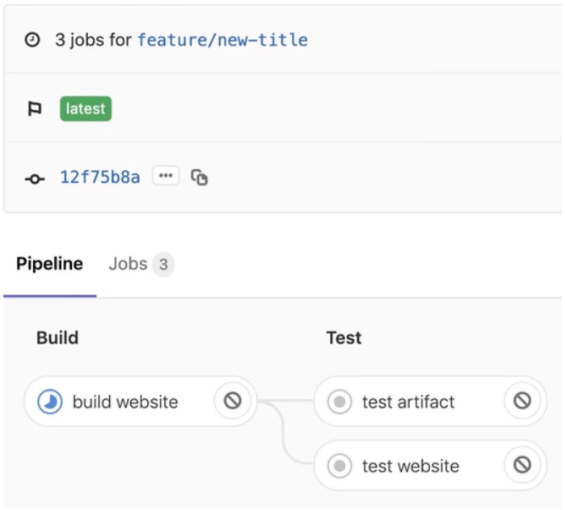


Hacemos lo mismo para el Job “deploy production” y “production test”:

```
deploy production:
  stage: deploy production
  environment:
    name: production
    url: $PRODUCTION_DOMAIN
  only:
    - master
  script:
    - npm install --global surge
    - surge --project ./public --domain $PRODUCTION_DOMAIN
```

```
production tests:
  image: alpine
  stage: production tests
  only:
    - master
  script:
    - apk add --no-cache curl
    - curl -s "https://$PRODUCTION_DOMAIN" | grep -q "Hi people"
    - curl -s "https://$PRODUCTION_DOMAIN" | grep -q "$CI_COMMIT_SHORT_SHA"
```

Antes de hacer commit de estos cambios debemos crear un branch, llamado por ejemplo "feature/new-title". Tan pronto como creamos este branch vemos que el pipeline se ejecuta:



Podemos ver que nuestro pipeline ahora más corto, solo ejecutamos el job build y luego test, y si todo se ejecuta correctamente, estaremos listos para hacer merge a master.