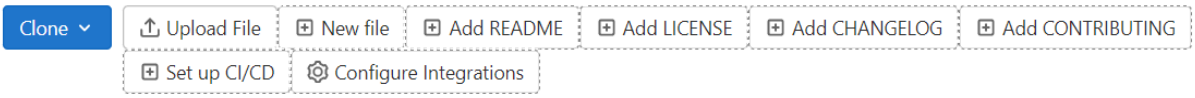
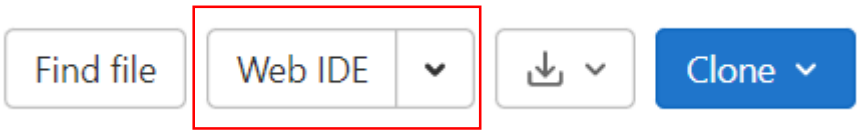
CI/CD TP

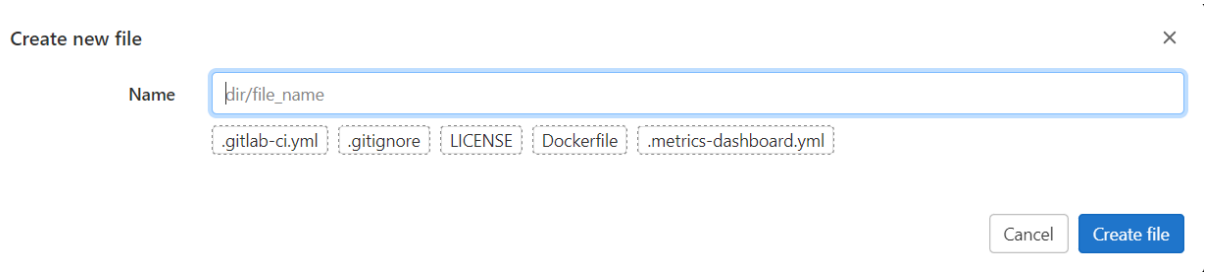
Nous allons voir ici la mise en place d'un pipeline CI/CD très simple pour comprendre le fonctionnement et la syntaxe du fichier .gitlab-ci.yml.

Dans votre dépôt Gitlab créez un nouveau projet vide. Via l'éditeur intégré à Gitlab créez un nouveau fichier soit en sélectionnant le web IDE, soit en cliquant sur le bouton "Set up CI/CD":





Sélectionnez comme nom de fichier .gitlab-ci.yml.



Lorsque vous créez un nouveau fichier gitlab-ci l'IDE vous propose un template pré-fait de pipeline. Nous n'allons pas l'utiliser mais vous pouvez voir déjà à quoi ressemble un fichier .gitlab-ci.yml. Notre pipeline va reprendre, très simplement, le processus de création d'un ordinateur.

Pour créer un ordinateur portable il vous faut un boitier, dans lequel vous mettez une carte mère (et bien d'autre composants évidemment) et ensuite vous ajoutez le clavier et l'écran.

Ce sont ces actions que nous allons indiquer dans notre première étape de que nous appelerons build. Commencez par rédiger un job build qui va indiquer de créer un dossier build et dedans un fichier ordinateur.txt dans lequel vous écrirez "Carte mère" puis "Clavier" puis "Ecran".

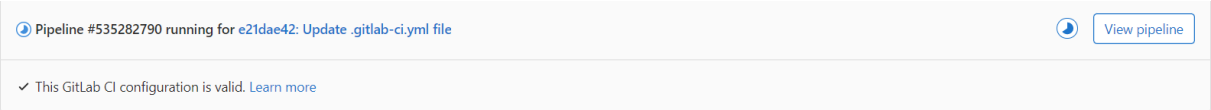
Auteur(s)	Relu, validé et visé par :		Date de création :	Date dernière MAJ :
Maxime PRZYBYLO	Jérôme CHRETIENNE : **Resp. Secteur NUMERIQUE OCCITANIE**			
	Florence CALMETTES : **Coordinatrice Filière SYST. & RESEAUX**		*Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.*	
	Sophie POULAKOS : **Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT**			

CI/CD TP

Notre job s'éffectuant dans un container il faut lui indiquer l'image à utiliser. Ne souhaitant faire que des commandes de base nous pouvons prendre une image classique et légère, en l'occurrence alpine. Vous indiquerez ensuite dans la partie script les étapes que vous voulez.

```
build laptop:
    image: alpine
    script:
        - echo "Création d'un ordinateur"
        - mkdir build
        - touch build/ordinateur.txt
        - echo "Carte mère" >> build/ordinateur.txt
        - echo "Clavier" >> build/ordinateur.txt
        - echo "Ecran" >> build/ordinateur.txt
```

Sauvegardez votre fichier en cliquant sur le bouton "commit changes" en bas de la page, attendez quelques secondes puis cliquez sur le bouton "view pipeline".



Si votre fichier n'est pas valide en terme de syntaxe vous pouvez cliquer sur le bouton Lint pour voir quel est le problème.

En cliquant sur "view pipeline" vous voyez les différents jobs que vous avez indiqué. Cliquez sur votre job "build laptop" afin de votre ce qu'il se passe dans votre container. Si tout se passe bien vous devriez voir que le runner démarre l'image alpine, affiche "Création d'un ordinateur" et crée bien le dossier, le fichier et son contenue.

Auteur(s)	Relu, validé et visé par :		Date de création :	Date dernière MAJ :
Maxime PRZYBYLO	Jérôme CHRETIENNE : **Resp. Secteur NUMERIQUE OCCITANIE**			
	Florence CALMETTES : **Coordinatrice Filière SYST. & RESEAUX**		*Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.*	
	Sophie POULAKOS : **Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT**			

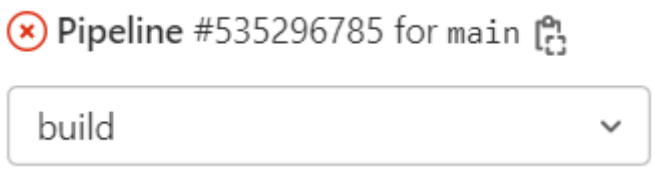
CI/CD TP

```
18  $ echo "Création d'un ordinateur"
19  Création d'un ordinateur
20  $ mkdir build
21  $ echo "Carte mère" >> build/ordinateur.txt
22  $ echo "Clavier" >> build/ordinateur.txt
23  $ echo "Ecran" >> build/ordinateur.txt
24  Uploading artifacts for successful job                                    00:02
25  Uploading artifacts...
26  build: found 2 matching files and directories
27  Uploading artifacts as "archive" to coordinator... 201 Created  id=2435470917 responseStatus=201 Created token=ihtri6hF
28  Cleaning up project directory and file based variables                    00:01
29  Job succeeded
```

Votre job s'est donc bien déroulé mais nous ne nous en sommes pas assurés. Il faut toujours tester son processus pour qu'il n'y ai pas de doute sur son bon déroulement. Nous allons donc rédiger un autre job pour tester la présence du fichier ainsi que son contenue. Pour tester la présence du fichier nous utiliserons la commande linux "test -f" et pour vérifier le contenue la commande "grep"

```
test laptop:
    image: alpine
    script:
        - test -f build/ordinateur.txt
        - grep "Carte mère" build/ordinateur.txt
        - grep "Clavier" build/ordinateur.txt
        - grep "Ecran" build/ordinateur.txt
```

Sauvegardez votre fichier en faisant un commit des changements et regardez votre pipeline. Vous devriez voir maintenant que vous disposez de deux jobs, "build laptop" et "test laptop". Si vous cliquez sur build laptop vous voyez le process décrit dans le premier job qui se conclut par "Job succeeded". Au bout d'un moment vous devriez voir sur la droite que votre pipeline a échoué.



Vous pouvez via ce menu déroulant sélectionner votre autre job pour voir ce qu'il s'est passé, à quel moment le processus s'est arrêté. On voit que le job s'est arrêté au moment du premier test de présence de fichier.

Auteur(s)	Relu, validé et visé par :		Date de création :	Date dernière MAJ :
Maxime PRZYBYLO	Jérôme CHRETIENNE : **Resp. Secteur NUMERIQUE OCCITANIE**			
	Florence CALMETTES : **Coordinatrice Filière SYST. & RESEAUX**		*Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.*	
	Sophie POULAKOS : **Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT**			

CI/CD TP

```
$ test -f build/ordinateur.txt
Cleaning up project directory and file based variables
ERROR: Job failed: exit code 1
```

Notre premier job s'est bien déroulé alors pourquoi ne pouvons-nous pas voir notre fichier dans notre job suivant ? Et bien cela est dù à cette phrase en bleu que vous voyez sur l'image. A la fin d'un job le runner va détruire le container qu'il a utilisé pour effectuer le job. Cela implique que tout ce qui a été créé dans ce container sera détruit également. Nous ne pouvons donc pas utiliser nos fichiers créés dans notre premier job pour notre deuxième. Nous devons pour ça passer par l'utilisation d'artefacts. Notre artefact va nous permettre non seulement de faire passer des données d'un job à un autre mais également de récupérer ces données sur notre ordinateur. Rajoutez donc ces lignes à la fin de votre job "build laptop".

```
artifacts:
    paths:
        - build
```

Nous indiquons ici que notre job aura des artefacts à sauvegarder, en l'occurrence le répertoire build. Faites un commit de vos changements et visualiser votre pipeline.

```
25 $ test -f build/ordinateur.txt
26 $ grep "Carte mère" build/ordinateur.txt
27 Carte mère
28 $ grep "Clavier" build/ordinateur.txt
29 Clavier
30 $ grep "Ecran" build/ordinateur.txt
31 Ecran
33 Cleaning up project directory and file based variables
35 Job succeeded
```

00:01

On voit ici que nos tests se sont déroulés correctement et que notre job à bien réussi. Nous pouvons voir dans notre job "build laptop" qu'il a bien créé les artefacts grâce à cette syntax :

```
28 Uploading artifacts for successful job
29 Uploading artifacts...
30 build: found 2 matching files and directories
31 Uploading artifacts as "archive" to coordinator... 201 Created  id=2435470917 responseStatus=201 Created token=ihtri6hF
```

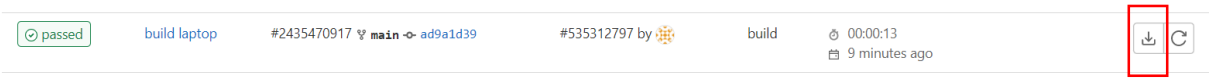
Auteur(s)	Relu, validé et visé par :		Date de création :	Date dernière MAJ :
Maxime PRZYBYLO	Jérôme CHRETIENNE : **Resp. Secteur NUMERIQUE OCCITANIE**			
	Florence CALMETTES : **Coordinatrice Filière SYST. & RESEAUX**		*Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.*	
	Sophie POULAKOS : **Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT**			

CI/CD TP

On peut voir également dans notre job "test laptop" qu'il télécharge bien les artefacts :

```
19  Downloading artifacts
20  Downloading artifacts for build laptop (2435470917)...
21  Downloading artifacts from coordinator... ok        id=2435470917 responseStatus=200 OK token=ihtri6hF
```

Vous pouvez visualiser vos jobs dans l'onglet "Jobs" de l'onglet "CI/CD" du menu latéral à gauche. Vous voyez sur cette vos différents jobs en cours et passés et leurs états. Sur cette même pas vous pouvez télécharger les artefacts des différents jobs. Téléchargez celui votre job et regardez ce qu'il contient.



Cela vous télécharge donc une archie "artifacts.rar" contant le dossier build et son fichier ordinateur.txt contenant bien "Carte mère", "Clavier" et "Ecran".

Nous avons donc maintenant un processus de CI complétement réalisé avec la création d'un produit puis le test de cette réalisation. Nous allons cependant modifier notre fichier pour qu'il soit plus organisé et optimisé.

Tout d'abord, même si gitlab est capable de comprendre qu'il doit d'abord faire notre premier job "build laptop" puis notre deuxième job "test laptop" nous allons lui définir un ordre d'étapes qui s'appelle des "stages". Nous aurons un stage "build" et un stage "test". Nous indiquerons ces stages au début du document:

```
stages:
    - build
    - test
```

Nous indiquerons ensuite dans nos jobs à quels stages ils appartiennent.

```
build laptop:
    image: alpine
    stage: build


test laptop:
    image: alpine
    stage: test
```

CI/CD TP

Faites un commit de vos changements et vérifiez que tout se passe toujours bien.
Nous allons maintenant rendre notre pipeline plus dynamique. Si nous décidons à un moment de changer le nom du fichier à créer nous devons le remplacer à 4 endroit. 4 mots à remplacer n'est pas énorme mais notre pipeline est très simple. Si nous avions un plus gros pipeline ça serait beaucoup plus compliqué. C'est pour ça que nous allons remplacer le nom du fichier par une variable que nous définirons. Nous pourrions définir cette variable dans notre job build comme ceci :

```
build laptop:
    image: alpine
    stage: build
    variables:
        FILENAME: "laptop.txt"
```

Etant donné que nous utiliserons cette variable dans notre autre job nous pourrions la définir une deuxième fois mais cela desservirai le principe des variables. Nous allons donc plutôt définir cette variable pour notre pipeline entier en la déclarant juste en dessous de de nos stages.

```
stages:
    - build
    - test

variables:
    FILENAME: laptop.txt
```

Et nous nous en servirons ensuite dans nos job en l'appelant précédé du symbole $.

```
build laptop:
    image: alpine
    stage: build
    script:
        - echo "Building a laptop"
        - mkdir build
        - echo "Motherboard" >> build/$FILENAME
        - echo "Keyboard" >> build/$FILENAME
    artifacts:
        paths:
            - build
```

Remplacez toutes les occurrence de ordinateur.txt par la variable FILENAME, effectuez votre commit et vous constaterez que le fichier créé, que vous pouvez télécharger, s'appel laptop.txt

Auteur(s)	Relu, validé et visé par :		Date de création :	Date dernière MAJ :
Maxime PRZYBYLO	Jérôme CHRETIENNE : **Resp. Secteur NUMERIQUE OCCITANIE**			
	Florence CALMETTES : **Coordinatrice Filière SYST. & RESEAUX**		*Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.*	
	Sophie POULAKOS : **Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT**			