

Orchestration de conteneurs et CI/CD

 ESGI Lyon - 2022/2023

- Présentation disponible à l'adresse: <https://dduportal.github.io/esgi-containers-ci-cd/2023>
- Version PDF de la présentation :  Cliquez ici
- This work is licensed under a Creative Commons Attribution 4.0 International License
- Code source de la présentation:  <https://github.com/dduportal/esgi-containers-ci-cd>







Comment utiliser cette présentation ?

- Pour naviguer, utilisez les flèches en bas à droite (ou celles de votre clavier)
 - Gauche/Droite: changer de chapitre
 - Haut/Bas: naviguer dans un chapitre
- Pour avoir une vue globale : utiliser la touche "o" (pour "**O**verview")
- Pour voir les notes de l'auteur : utilisez la touche "s" (pour "**S**peaker notes")

Bonjour !

La suite: vers le bas ↓

Damien DUPORTAL

- Staff Software Engineer chez CloudBees pour le projet Jenkins ☐☐☐
- Freelancer
- Me contacter :
 -  damien.duportal <chez> gmail.com
 -  dduportal
 -  Damien Duportal
 -  @DamienDuportal

Et vous ?



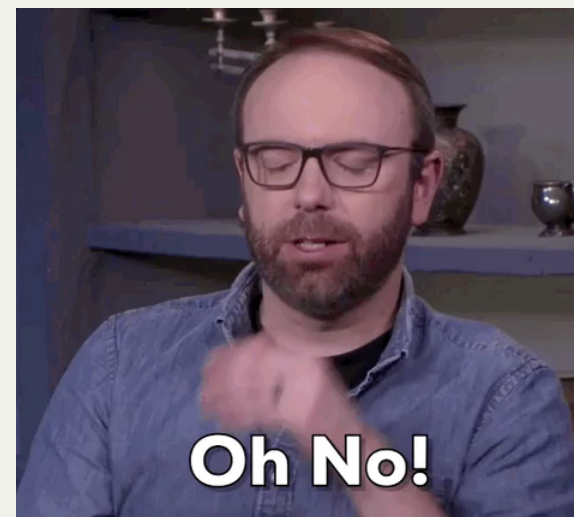
A propos du cours

- Alternance de théorie et de pratique pour être le plus interactif possible
- Compatible Distanciel / Présentiel
- Contenu entièrement libre et open-source
 - N'hésitez pas ouvrir des Pull Request si vous voyez des améliorations ou problèmes: [sur cette page](#) (🙄 wink wink)

Calendrier

- **Distanciel** 🛏 Mercredi 01 février 2023 - 14h → 17h15
- **Présentiel** 🎓 Mardi 28 février 2023 - 14h → 17h15
- **Présentiel** 🎓 Mercredi 01 mars 2023 - 08h → 17h16
- **Présentiel** 🎓 Jeudi 02 mars 2023 - 08h → 13h
- **Distanciel** 🛏 Vendredi 28/04 - 14h → 17h15

Evaluation



- Pourquoi ? s'assurer que vous avez acquis un minimum de concepts
- Quoi ? Une note sur 20, basée sur une liste de critères exhaustifs
- Comment ? Un projet GitHub (public) à me rendre (timing à déterminer ensemble)

Plan

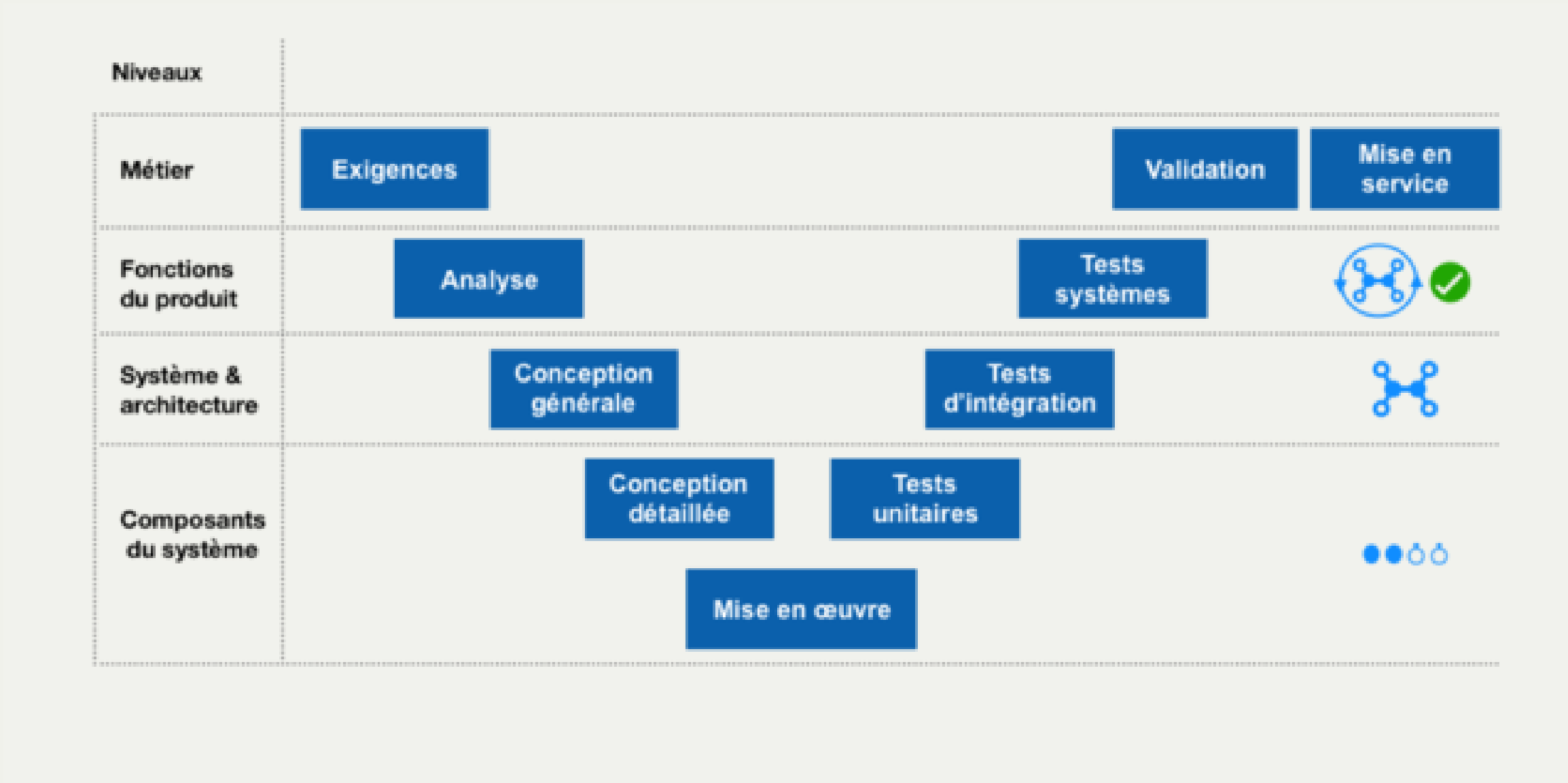


- Introduction
- Rappels de Ligne de commande et Docker
- Docker Avancé
- Orchestration de Containers avec Docker Swarm
- Intégration et Déploiement Continus (CI/CD)

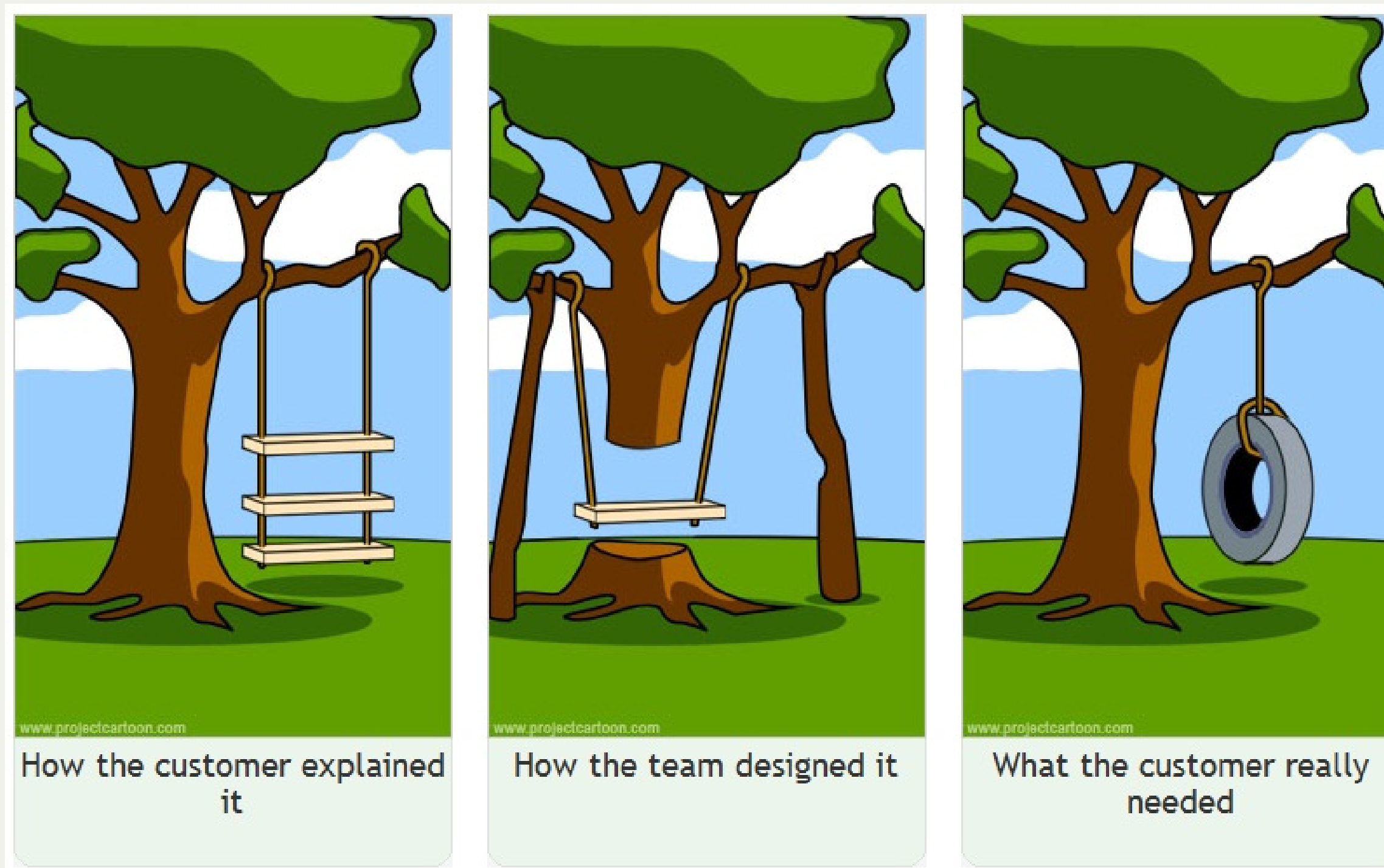
La suite: vers la droite ➡

Vous avez dit "DevOps" ?

Avant : le cycle en V

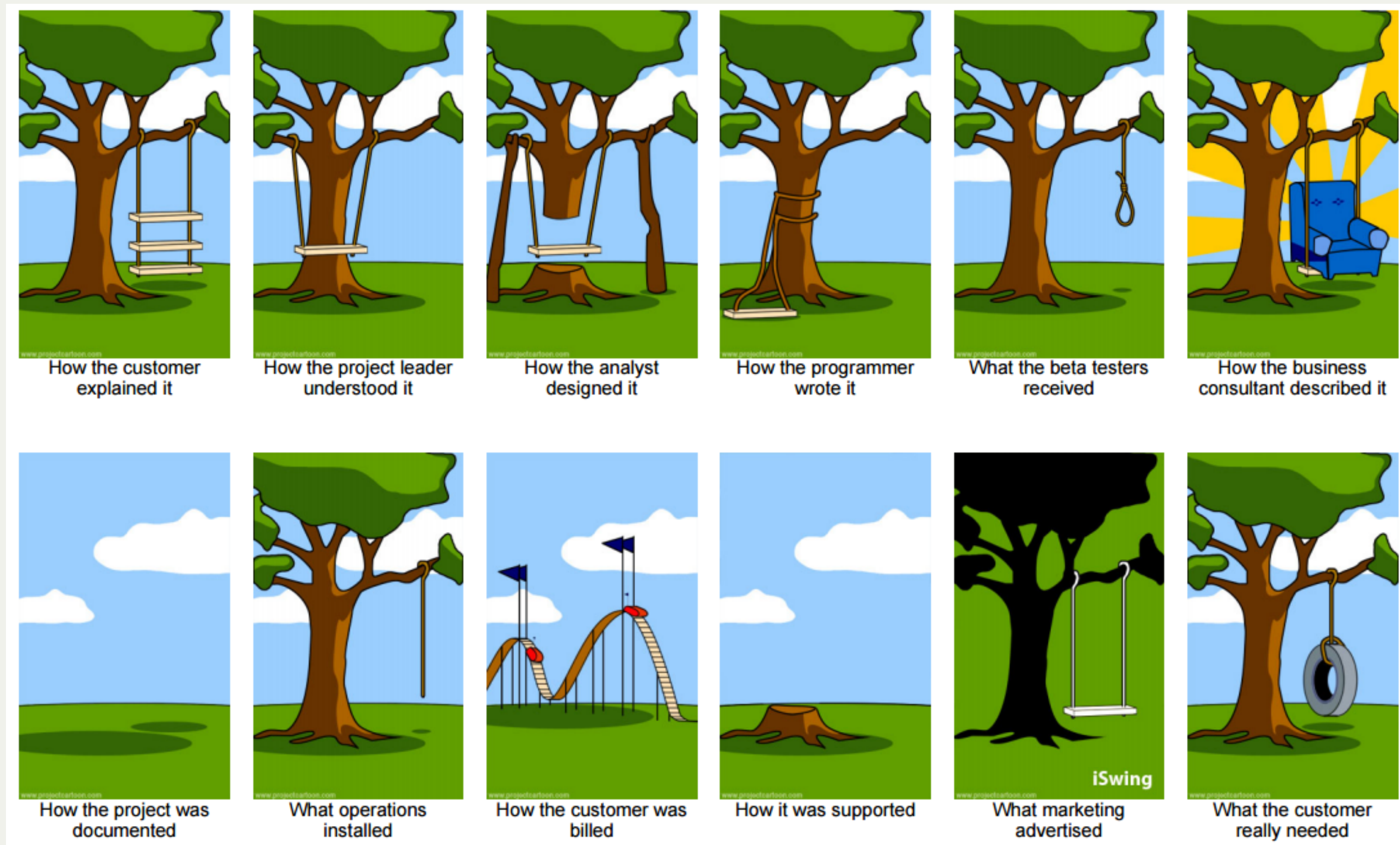


What could go right?



Copyright © <https://www.projectcartoon.com> under the Creative Commons Attribution 3.0 Unported License

Et dans la vraie vie c'est pire !



Copyright © <https://www.projectcartoon.com> under the Creative Commons Attribution 3.0 Unported License

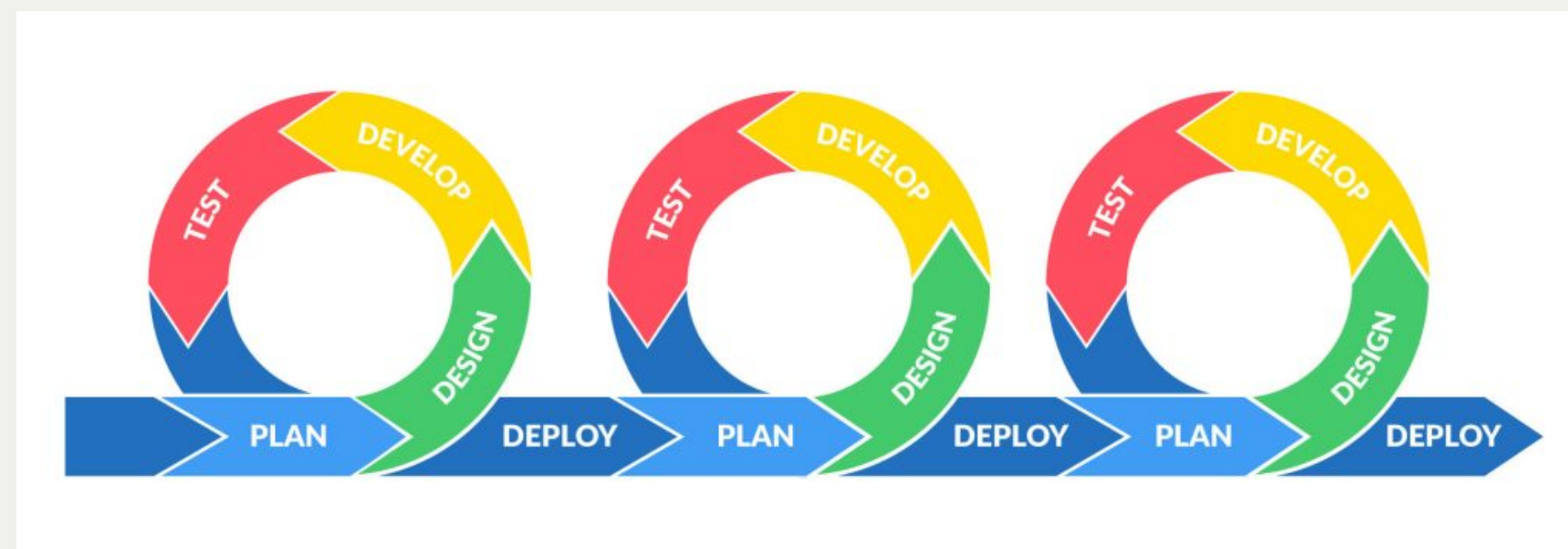
Problématique

Travail basé sur les hypothèses de départ + Temps écoulé entre la capture du besoin et la mise à disposition en production

- Le besoin a forcément évolué dans ce laps de temps
- Erreur de capture du besoin == coût de l'erreur décuplé avec le temps

Agile

- Travailler par petites itérations **complètes**
 - Commencer petit
 - Confronter le logiciel au plus tôt aux utilisateurs.
 - Refaire des hypothèses basées sur ce que l'on a appris, et recommencer !
- "Embrasser" le changement : Votre logiciel va changer en **continu**






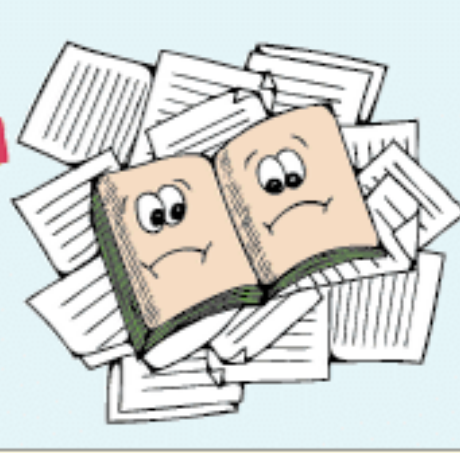
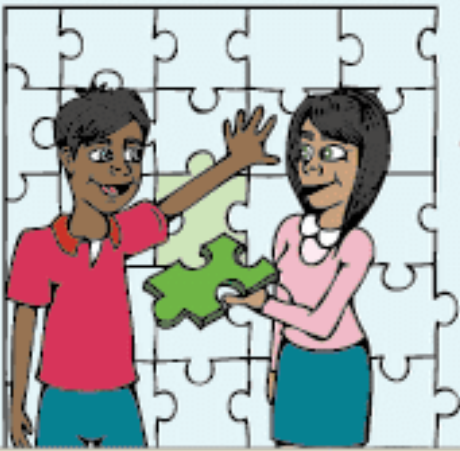



Source

Manifest Agile

- <https://agilemanifesto.org/iso/fr/manifesto.html>

Manifesto for Agile Software Development*

*"We are uncovering better ways of developing software by doing it and helping others do it.
Through this work we have come to value:*

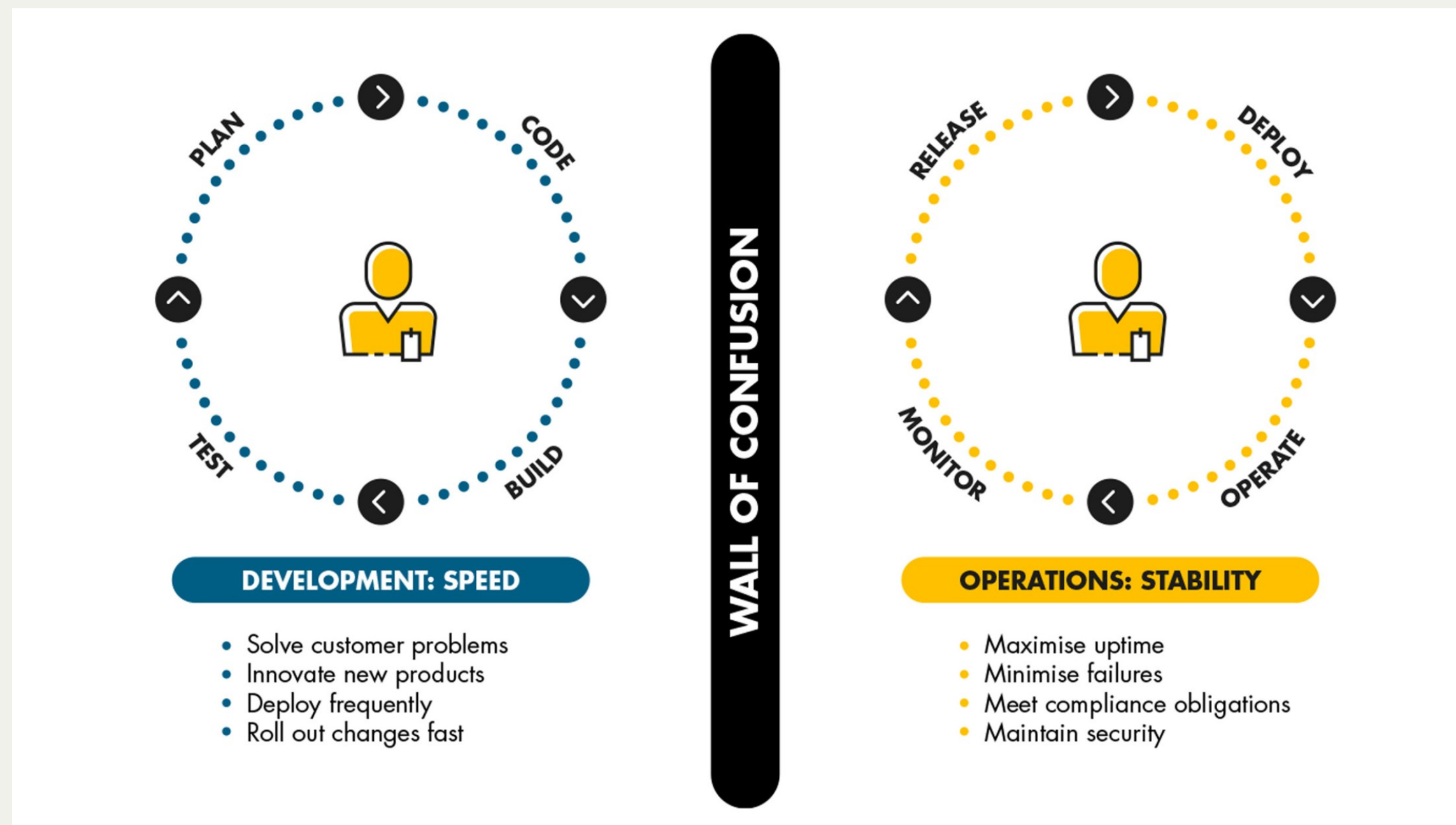
 Individuals & interactions	 Processes & tools	 Working software	 Comprehensive documentation
 Customer collaboration	 Contract negotiation	 Responding to change	 Following a plan

That is, while there is value in the items on the right, we value the items on the left more."

Knowledge TRAIN

Copyright © 2018 Knowledge Train Limited. *Quoted from www.agilemanifesto.org

Le mur de la confusion

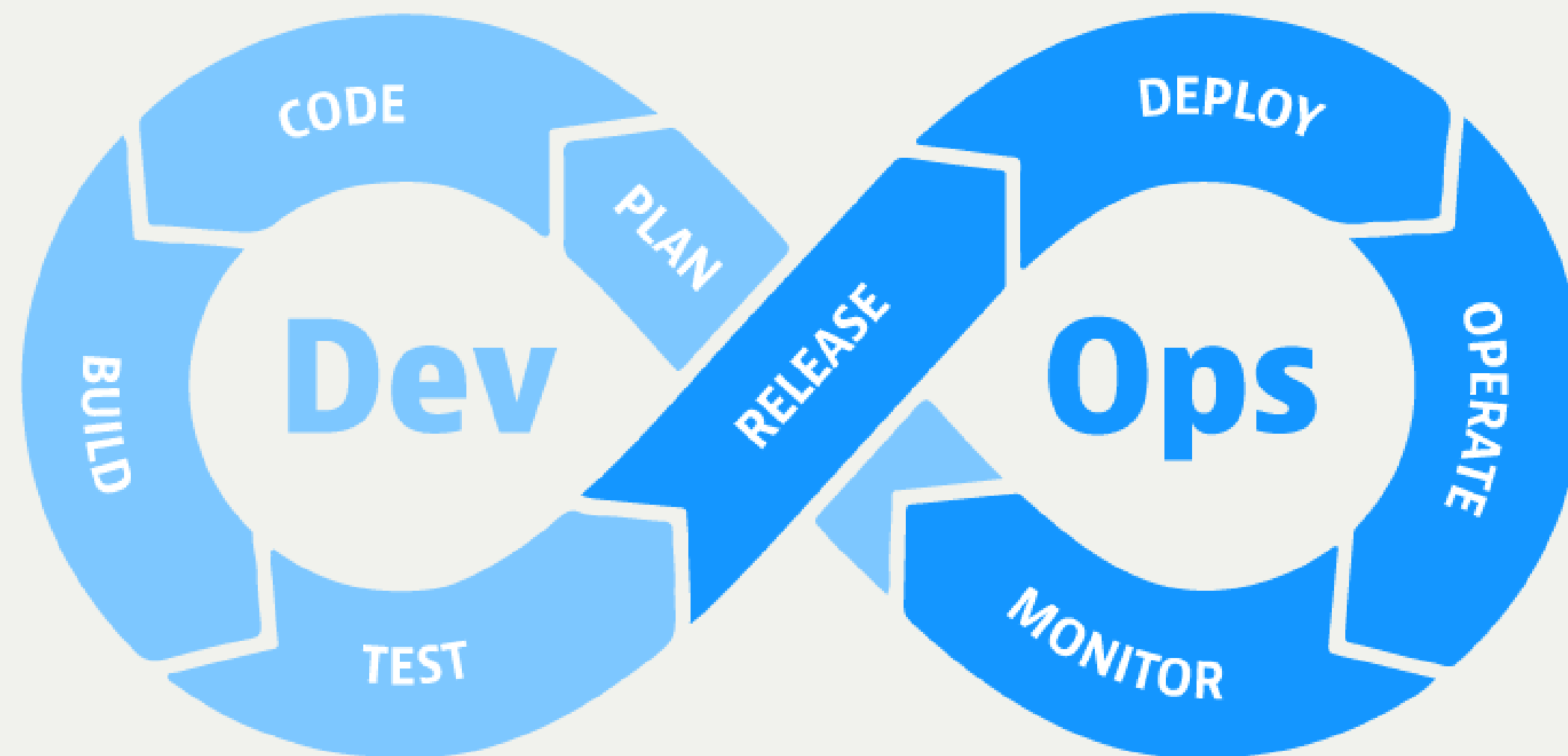


Source

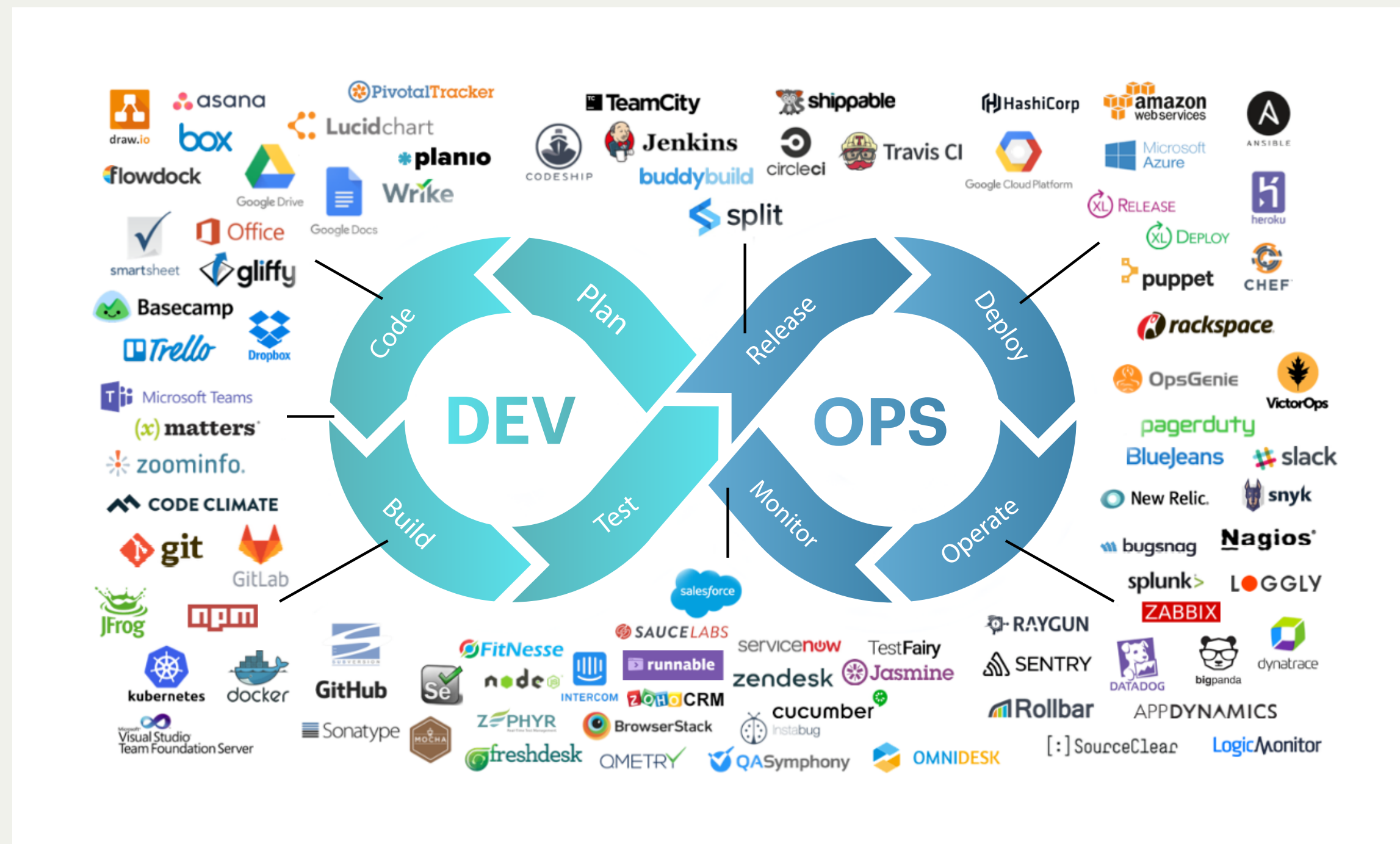
La clé : gérer le changement!

- Le changement ne doit pas être un événement, ça doit être la **norme**
- Faire en sorte que changer quelque chose soit:
 - Facile
 - Rapide
 - Stable
 - Sûr

Say Hello to DevOps



Heureusement, vous avez des outils à disposition !



Préparer votre environnement de travail

Outils Nécessaires


- Un navigateur web récent (et décent)
- Un compte sur  GitHub

GitPod

GitPod.io : Environnement de développement dans le ☁ "nuage"

- **But:** reproductible
- Puissance de calcul sur un serveur distant
- Éditeur de code VSCode dans le navigateur
- Gratuit pour 50h par mois (⚠)

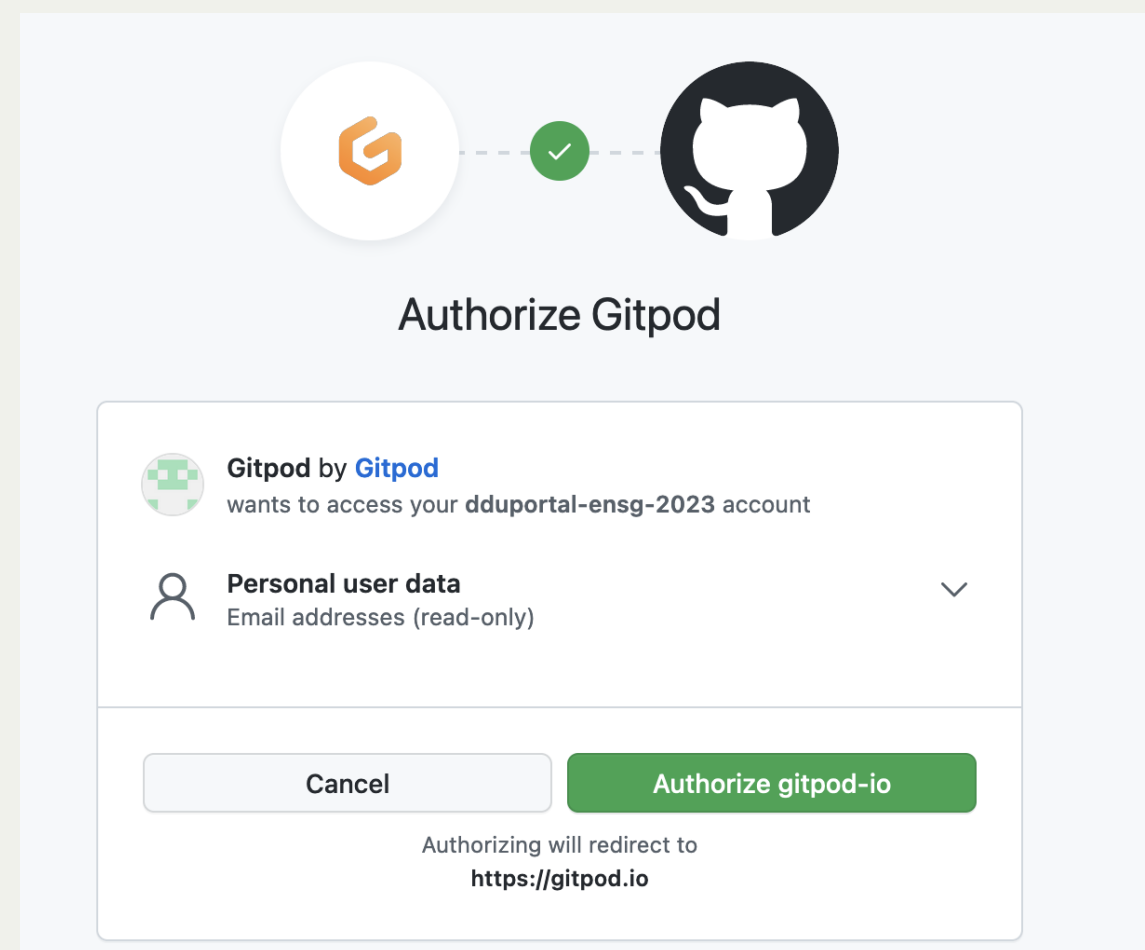
Démarrer avec GitPod

- Rendez vous sur <https://gitpod.io>
- Authentifiez vous en utilisant votre compte GitHub:
 - Bouton "Login" en haut à droite
 - Puis choisissez le lien " Continue with GitHub"

△ Pour les "autorisations", passez sur la slide suivante

Autorisations demandées par GitPod

Lors de votre première connexion, GitPod va vous demander l'accès (à accepter) à votre email public configuré dans GitHub :



⚠️ Passez à la slide suivante avant d'aller plus loin

Validation du Compte GitPod




GitPod vous demande votre numéro de téléphone mobile afin d'éviter les abus (service gratuit).
Saisissez un numéro de téléphone valide pour recevoir par SMS un code de déblocage :

User Validation Required

⚠ To use Gitpod you'll need to validate your account with your phone number. This is required to discourage and reduce abuse on Gitpod infrastructure.

Enter a mobile phone number you would like to use to verify your account. Having trouble? [Contact support](#)

Mobile Phone Number

 ▾

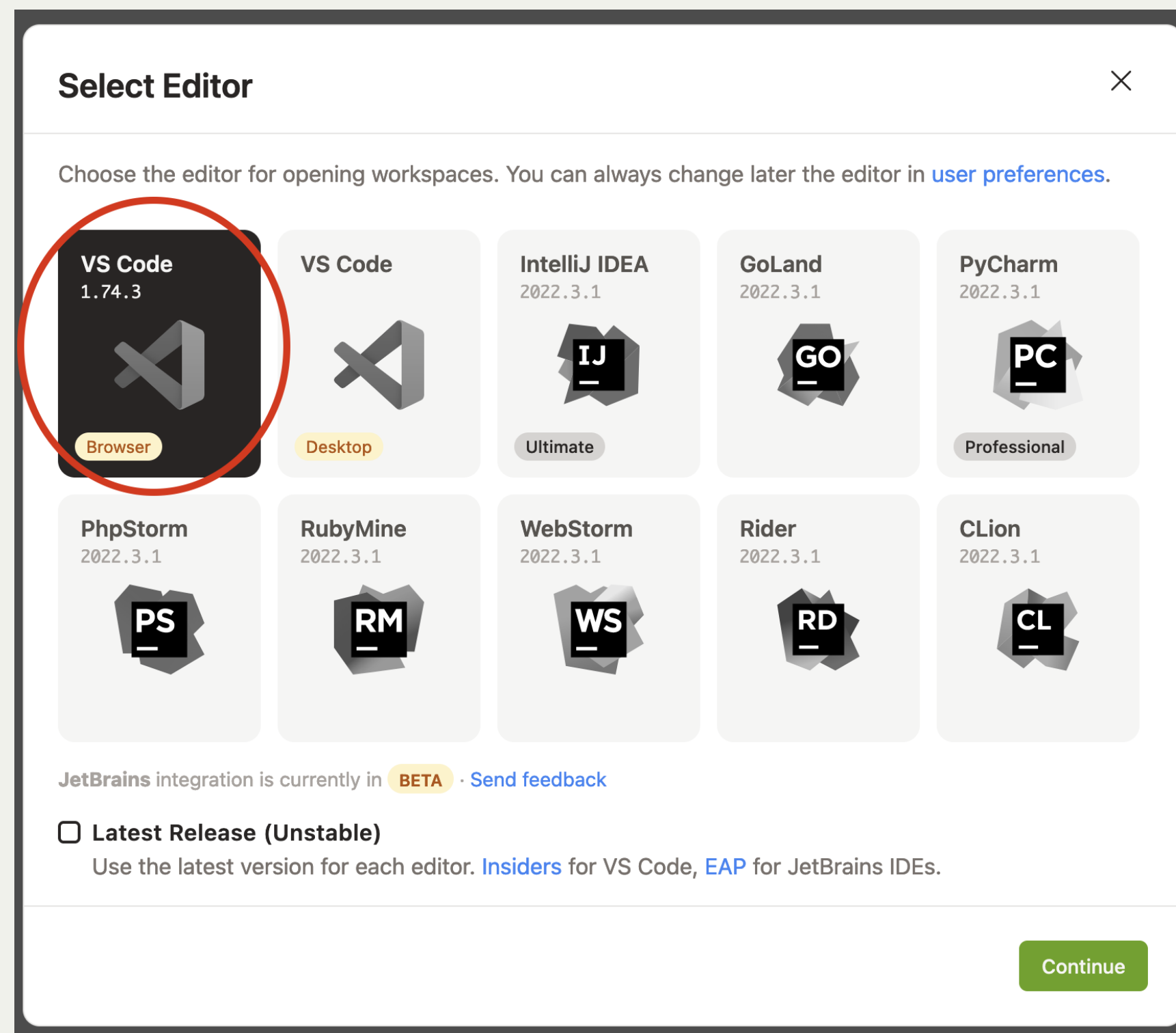
(201) 555-0123

Send Code via SMS

⚠ Passez à la slide suivante avant d'aller plus loin

Choix de l'Éditeur de Code

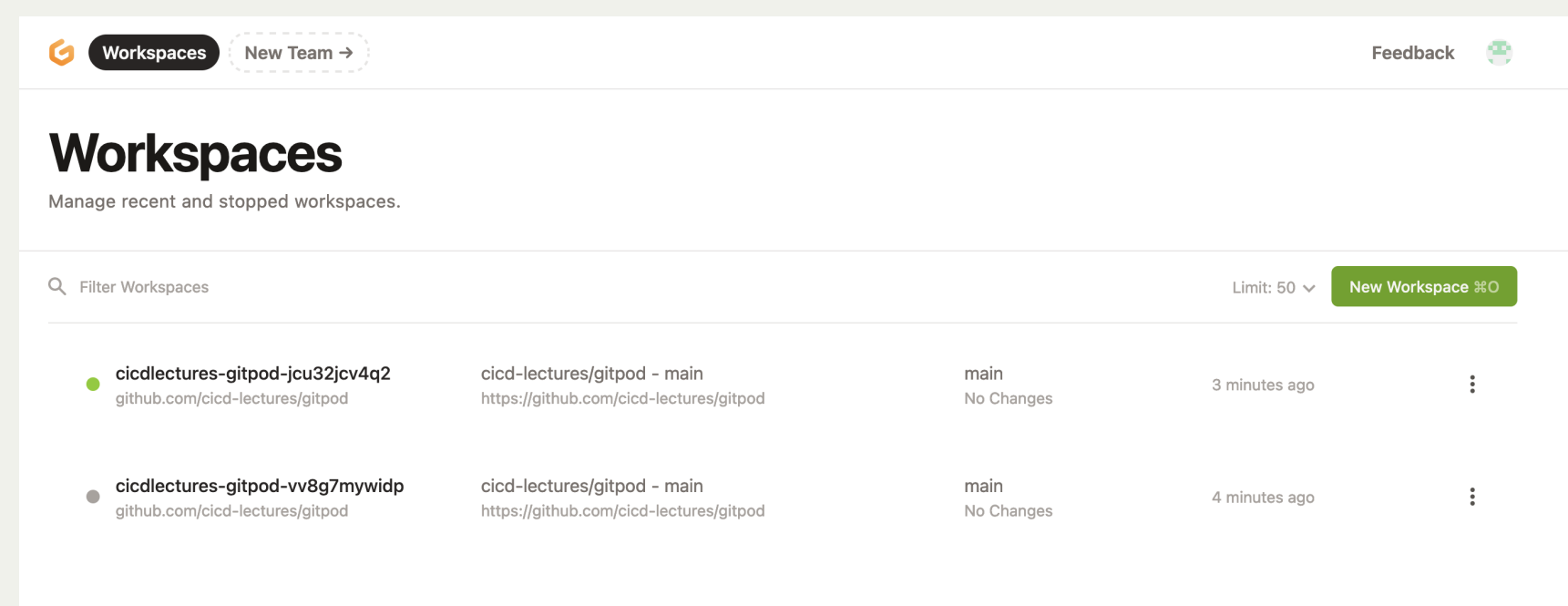
Choisissez l'éditeur "VSCode Browser" (la première tuile) :



⚠️ Passez à la slide suivante avant d'aller plus loin

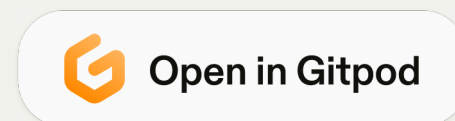
Workspaces GitPod

- Vous arrivez sur la page listant les "workspaces" GitPod :
- Un workspace est une instance d'un environnement de travail virtuel (C'est un ordinateur distant)
- ⚠️ Faites attention à réutiliser le même workspace tout au long de ce cours⚠️



Démarrer l'environnement GitPod

Cliquez sur le bouton ci-dessous pour démarrer un environnement GitPod personnalisé:



Après quelques secondes (minutes?), vous avez accès à l'environnement:

- Gauche: navigateur de fichiers ("Workspace")
- Haut: éditeur de texte ("Get Started")
- Bas: Terminal interactif
- À droite en bas: plein de popups à ignorer (ou pas?)

Source disponible dans :  <https://github.com/dduportal/esgi-gitpod>

Checkpoint

- Vous devriez pouvoir taper la commande `whoami` dans le terminal de GitPod:
 - Retour attendu: `gitpod`
- Vous devriez pouvoir fermer le fichier "Get Started" ...
 - ... et ouvrir le fichier `.gitpod.yml`

On peut commencer !

Ligne de commande

Remise à niveau / Rappels

CLI

- 📖 CLI == "Command Line Interface"
- 📖 "Interface de Ligne de Commande"

REPL

Pour les théoriciens et curieux :

-  REPL == "Read–eval–print loop"

https://en.wikipedia.org/wiki/Read%E2%80%93eval%E2%80%93print_loop

Anatomie d'une commande

```
ls --color=always -l /bin
```

[Copy](#)

- Séparateur : l'espace
- Premier élément (`ls`) : c'est la commande
- Les éléments commençant par un tiret – sont des "options" et/ou drapeaux ("flags")
 - "Option" == "Optionnel"
- Les autres éléments sont des arguments (`/bin`)
 - Nécessaire (par opposition)

Manuel des commande

- Afficher le manuel de <commande> :

```
man <commande> # Commande 'man' avec comme argument le nom de ladite commande
```

- Navigation avec les flèches haut et bas
 - Tapez / puis une chaîne de texte pour chercher
 - Touche n pour sauter d'occurrence en occurrence
- Touche q pour quitter

 Essayez avec `ls`, chercher le mot `color`

- 🐞 La majorité des commandes fournit également une option (`--help`), un flag (`-h`) ou un argument (`help`)
- Google c'est pratique aussi hein !






Raccourcis

Dans un terminal Unix/Linux/WSL :

- CTRL + C : Annuler le process ou prompt en cours
- CTRL + L : Nettoyer le terminal
- CTRL + A : Positionner le curseur au début de la ligne
- CTRL + E : Positionner le curseur à la fin de la ligne
- CTRL + R : Rechercher dans l'historique de commandes

🎓 Essayez-les !



Commandes de base 1/2

- `pwd` : Afficher le répertoire courant
 -  Option `-P` ?
- `ls` : Lister le contenu du répertoire courant
 -  Options `-a` et `-l` ?
- `cd` : Changer de répertoire
 -  Sans argument : que se passe t'il ?
- `cat` : Afficher le contenu d'un fichier
 -  Essayez avec plusieurs arguments
- `mkdir` : créer un répertoire
 -  Option `-p` ?

Commandes de base 2/2

- `echo` : Afficher un (des) message(s)
- `rm` : Supprimer un fichier ou dossier
- `touch` : Créer un fichier
- `grep` : Chercher un motif de texte





Arborescence de fichiers 1/2

- Le système de fichier a une structure d'arbre
 - La racine du disque dur c'est / :  `ls -l /`
 - Le séparateur c'est également / :  `ls -l /usr/bin`
- Deux types de chemins :
 - Absolu (depuis la racine): Commence par / (Ex. `/usr/bin`)
 - Sinon c'est relatif (e.g. depuis le dossier courant) (Ex `./bin` ou `local/bin/`)

 linux directory structure

Source

Arborescence de fichiers 2/2

- Le dossier "courant" c'est . :  `ls -l ./bin # Dans le dossier /usr`
- Le dossier "parent" c'est .. :  `ls -l ../ # Dans le dossier /usr`
- ~ (tilde) c'est un raccourci vers le dossier de l'utilisateur courant :  `ls -l ~`
- Sensible à la casse (majuscules/minuscules) et aux espaces : 

```
ls -l /bin
ls -l /Bin
mkdir ~/"Accent tué"
ls -d ~/Accent\ tué
```

Copy

Un language (?)

- Variables interpolées avec le caractère "dollar" \$:

```
echo $MA_VARIABLE
echo "$MA_VARIABLE"
echo ${MA_VARIABLE}

# Recommendation
echo "${MA_VARIABLE}"

MA_VARIABLE="Salut tout le monde"

echo "${MA_VARIABLE}"
```

Copy

- Sous commandes avec \$ (<command>) :

```
echo ">> Contenu de /tmp :\n$(ls /tmp)"
```

Copy

- Des `if`, des `for` et plein d'autres trucs (<https://tldp.org/LDP/abs/html/>)

Codes de sortie

- Chaque exécution de commande renvoie un code de retour (🚩 "exit code")
 - Nombre entier entre 0 et 255 (en **POSIX**)
- Code accessible dans la variable **éphémère** `$?` :

```
ls /tmp
echo $?

ls /do_not_exist
echo $?

# Une seconde fois. Que se passe-t'il ?
echo $?
```

Copy

Entrée, sortie standard et d'erreur

 cli ios

```
ls -l /tmp
echo "Hello" > /tmp/hello.txt
ls -l /tmp
ls -l /tmp >/dev/null
ls -l /tmp 1>/dev/null

ls -l /do_not_exist
ls -l /do_not_exist 1>/dev/null
ls -l /do_not_exist 2>/dev/null

ls -l /tmp /do_not_exist
ls -l /tmp /do_not_exist 1>/dev/null 2>&1
```

Copy

Pipelines

- Le caractère "pipe" | permet de chaîner des commandes
 - Le "stdout" de la première commande est branchée sur le "stdin" de la seconde
- Exemple : Afficher les fichiers/dossiers contenant le lettre d dans le dossier /bin :

```
ls -l /bin
```

```
ls -l /bin | grep "d" --color=auto
```

Copy

Exécution 1/2

- Les commandes sont des fichier binaires exécutables sur le système :

```
command -v cat # équivalent de "which cat"
```

Copy

```
ls -l "$(command -v cat)"
```

- La variable d'environnement `$PATH` liste les dossiers dans lesquels chercher les binaires
 - 💡 Utiliser cette variable quand une commande fraîchement installée n'est pas trouvée

Exécution 2/2

- Exécution de scripts :
 - Soit appel direct avec l'interpréteur : `sh ~/monscript.txt`
 - Soit droit d'exécution avec un "shebang" (e.g. `#!/bin/bash`)

```
$ chmod +x ./monscript.sh  
  
$ head -n1 ./monscript.sh  
#!/bin/bash  
  
$ ./monscript.sh  
# Exécution
```

Copy

Work in progress

🚧 Revenez plus tard