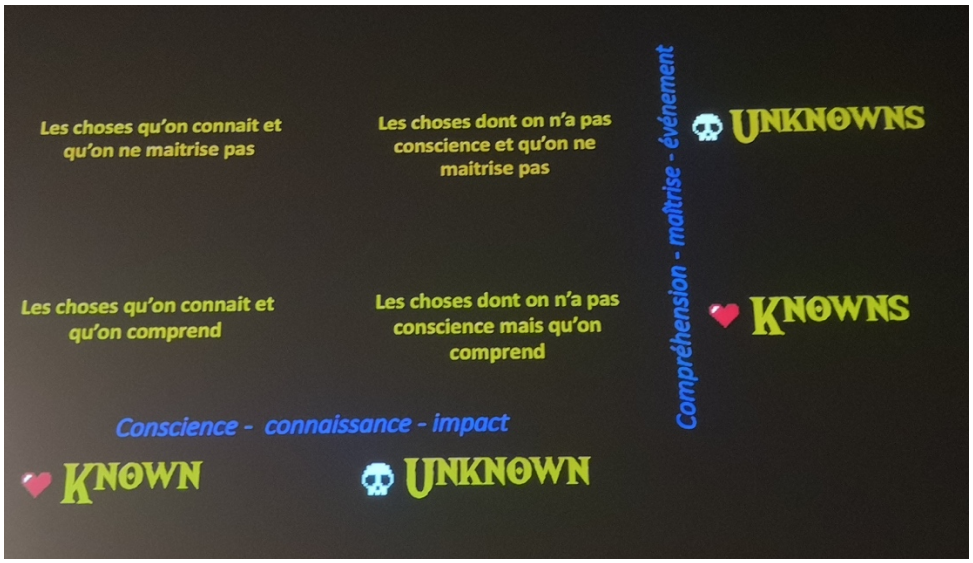
Observabilité, Mythes, réalité et Chaos

Speakers : Benjamin Gakic (Oui.SNCF)
Format : Conférence
Date : 18 avril 2019

Benjamin est ingénieur du Chaos Engineering chez Oui.SNCF.

Sa présentation commence par une définition du **Monitoring** : répéter de manière régulière un processus de test ou de surveillance d'un composant afin d'obtenir très rapidement et simplement une vision précise des évènements ou des anomalies sur la période analysée. On répète, on répète, on répète.

Tableau très intéressant des Known / Unkown / Unknows / Knowns :



Les choses qu'on connaît et qu'on ne maitrise pas : exemple du disque, on met du monitoring pour savoir s'il se remplit. On ne sait pas quand çà arrivera, ni où çà arrivera.

3 étapes de **gestion d'un incident** :

1. **Monitoring** : détecter et alerter un humain, savoir qu'une panne survient
2. **Diagnostique** : comprendre comment est survenue la panne et ce qu'elle implique. L'observabilité peut permettre de diagnostiquer.
3. La **résolution** : corriger un bug, auto-repair, ajouter des patterns de résilience.

Si on ne détecte pas ? On pense que tout se passe bien, mais l'application ne fonctionne pas et on perd de l'argent. A combien de développeur est-ce arrivé de devoir patcher une application pour ajouter une ligne de log afin de diagnostiquer le problème ?

Qu'est-ce qu'un **bon indicateur** ?

1. **Pertinent** : information compréhensible, comportementale et représentative et votre application ou système
2. **Utilisable** : suffisamment précis et facilement interprétable par le plus grand nombre

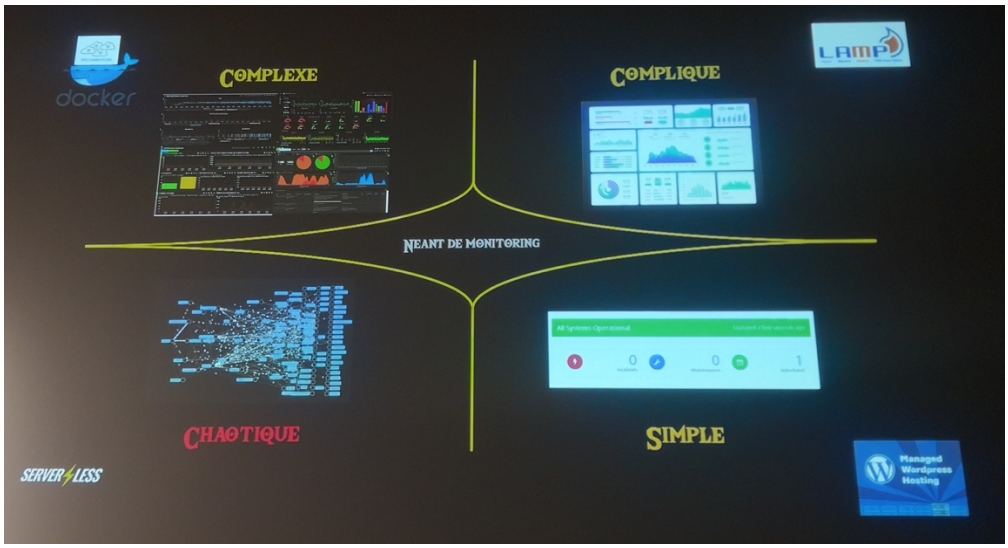
3. **Utilisé** : utile et jugé comme tel au quotidien. De nombreux dashboards ne servent qu'à faire joli et ne sont pas utilisés.

Attention à l'interprétation des indicateurs. On peut faire dire l'inverse à un indicateur. Les politiques le savent très bien.

Avec l'augmentation de la complexité des systèmes, le monitoring ne suffit plus.

Le diagramme ci-dessous présente 4 situations :

- Cas simple : dashboard wordpress
- Compliqué : LAMP n'est pas si facile que çà
- Complexe : cloud, docker … beaucoup de chose, incorrélable par une seule personne
- Chaotique : exemple du serverless



Heureusement, l'**observabilité** est là pour nous aider (cf. photo)
Définition Wikipedia trop compliquée, pas assez parlante.
A quoi çà sert : on a un système observable quand l'équipe peut rapidement et de manière fiable résoudre des problèmes.
Un système est dit observable quand on peut déduire son état des symptômes et des informations qu'il émet.

Définition de Benjamin : capacité du système à exposer des informations, que l'on peut choisir d'exploiter, et permettant d'appréhender son fonctionnement et son état.
L'observabilité est destinée aux humains.

L'observabilité va permettre de mettre en évidence des Unknowns – Unknowns : les choses dont on n'a pas conscience et qu'on ne maitrise pas.

L'observabilité qu'on fait déjà :

- Le monitoring
- Instrumentation du code, logs applicatifs ou technique, indicateurs métiers générés dans le code

- Faire de la corrélation d'évènements
- Outil d'APM (Application Performance Monitoring)

Exemple de code Java et des méthodes de logs en début / fin de méthodes + ajout d'instrumentation.
OpenTracing va normaliser les échanges.

Plus on regarde quelque chose, plus on remarque de choses : exemple de la porte avec carreaux, poignée, double porte …
En fonction de l'endroit où l'on porte son attention, on va retirer de plus en plus d'informations.
Le monitoring : on reste fixé à regarder la porte. On peut zoomer sur un élément de la porte. Si toute l'assemblée se focalise sur la porte, on ne voit pas le reste de la salle. Sauf si quelqu'un détecte de la chaleur et peut donc s'apercevoir qu'un feu est allumé à l'autre bout de la salle.
L'observabilité doit se calquer sur le comportement humain.

Known – Knowns : Les choses qu'on connait et qu'on comprend. Ce sont les tests qui permettent de connaître son système.

Chaos Engineering

Définition : discipline de l'expérimentation sur un système distribué afin de renforcer la confiance du système à résister à des conditions turbulentes en production.

Permet d'adresser les Unknown – Knowns : les choses dont on n'a pas conscience mais qu'on comprend. Effet de sérendipité : à l'origine de toutes les grandes découvertes (ex : Pasteur découvre les antibiotiques, pomme de Newton …)

Retour d'expérience à Oui.SNCF

Chaos Monkey : tout fonctionnait (résilience OK), mais les outils de monitoring ne l'ont pas détecté
Chaos Kong : casser un data center. 1000 façons de faire tomber un Datacenter.

Objectifs du test :
- Validation du respect des procédures et observation des réactions humaines
- Valider la résilience

Périmètre :
- Récurrence : 3 expérimentations avec 1,5 mois d'intervalle
- Chaos progressif : implémentation de la panne cachée, passage d'une date et heure connue à une expérimentation surprise

Résultat espéré :
- Tout va bien
- Procédures respectées

Implémentation : on supprime la ligne Internet du Datacenter et on coupe le lien réseau du DataCenter. Cela déclenche le feu.

Dès la 1ière expérimentation, tous les flux ont été basculés en moins de 10 secondes.
Pas de détection qu'un datacenter et tombé : on a juste remarqué qu'une base de données n'était plus répliquée (sérédendipité)
Non-respect des procédures car manque de prise au sérieux : c'est un exercice, mais on a vraiment coupé le datacenter et il s'agissait d'un vrai incident.

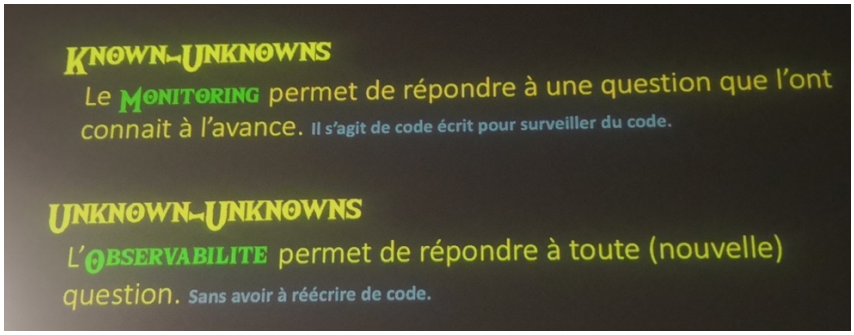
Lors de la 3ième expérimentation, le data center a très vite été réparé. Mais on a oublié de réactiver le traffic.
Outil Netflix Vizceral : affiché en permanence dans leur salle de supervision.

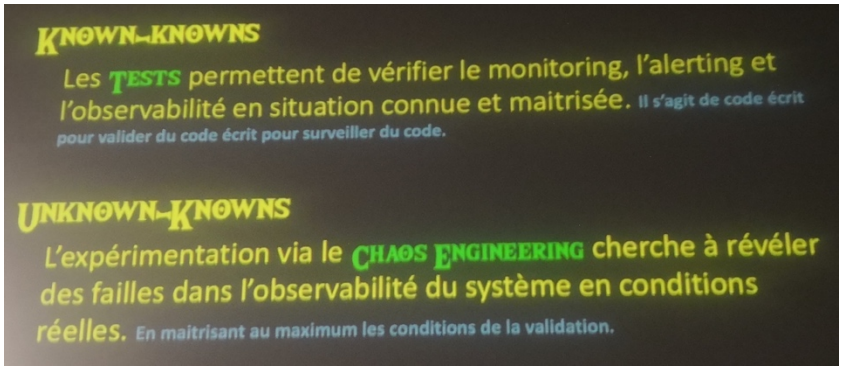
Le chaos et l'observabilité est l'affaire de tous.
Pour arrêter un DataCenter, il faut avoir l'aval de la direction. Chez Oui.SCNF, elle est convaincue de l'intérêt du chaos engeneering.

Conclusion

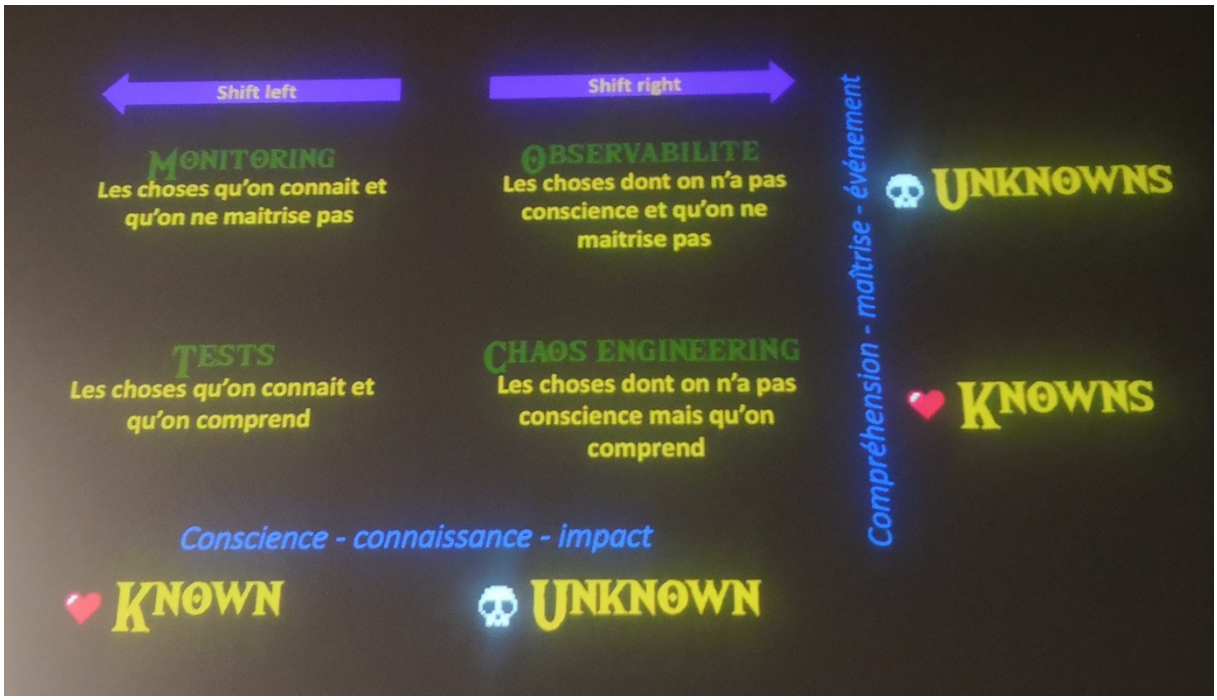
- Le chaos engineering est certainement le meilleur outil de validation de l'observabilité
- « Sans observabilité vous n'avez pas de Chaos Engineering, juste du chaos » (Charity Moor)
- La grande force du Chaos Engineering aura surtout été de provoquer un effet de sérendipité sur l'Observabilité du système
- Pour le Chaos comme l'observabilité, la vérité est ailleurs que dans les environnements de tests. Certaines sociétés (de grandes banques) interdisent aux développeurs de regarder ce qui se passe en Production. C'est un crime. Comment corriger les choses ou les prémunir ?

Les environnements de tests sont tout de même intéressants :
- Le **shift left** est une pratique introduite il y'a quelques années par l'agilité et enjoignant à effectuer les tests au plus tôt dans la chaîne de développement pour gagner de l'argent
- Le **shift right** est une nouvelle pratique initiée par le Chaos Engineering et proposant de vérifier au plus tard et au plus près de la réalité, à savoir en production, la validité de tout travail effectué en amont. On y voit une nécessité de tester en production. Sinon à la moindre problématique on ne sait pas y faire et ça coute très cher.





Le speaker est convaincu que le chaos enginerring est une discipline qui va se démocratiser, au même titre que le Craftmanship, le DevOps ou bien encore l'Agile.
Nécessité de chercher du lien entre les différentes disciplines. Permet de s'adapter au changement. La pratique centrale est le DevOps : les dévs payés pour sortir le plus de features et les Ops pour avoir l'environnement le plus table possible. Il faut commencer par là.



Livres recommandés :

- Mettre en œuvre Devops, Alain Sacquet et Christophe Rochefolle
- Ebook sur l'Observability : https://www.d2si.io/ebook-observability

3 citations :

- « Le chaos est la voie de la transformations », Jean-Pierre Gagné => moteur qui permet de s'améliorer
- « Le changement n'est jamais douloureux. Seule la résistance au changement est douloureuse », Bouddha => remet en cause les compétences de certaines équipes

- « Certaines personnes peuvent lire Guerre et Paix et penser que c'est une simple histoire d'aventures. D'autres peuvent lire les ingrédients sur un papier de chewing-gum et découvrir les secrets de l'univers. », Lex Luthor