Déroulement du codeLab

## « Ça part en prod ! » Testez la résilience de votre appli par le chaos.

### Introduction (15 minutes)

*"Bon, on a développé en interne une super application de gestion des compétences en architecture micro-service, on a fait une super recette fonctionnelle (sur les cas passants, faute de temps.. ça devrait suffire !), on a une couverture de TU au top, on a des super tests d'intégration, … Bref, on est super confiant pour partir en prod !*

*Mais au fait, quelqu’un a regardé si notre application allait tenir la charge ? Si elle était tolérante aux pannes ? Petit doute… Sait-on vraiment si notre application va être résiliente ou pas ?"*

* [SLIDE 1] Qu’est-ce que la résilience ?
* [SLIDE 2] Présentation du concept de Chaos Engineering
* [SLIDE 3] Présentation du scénario (cf abstract et pitch ci-dessus)
* [SLIDE 4] Présentation de notre application : fonctionnel et architecture
* Présentation du besoin : challenger le produit sur des aspects de résilience en utilisant des outils du monde du chaos engineering
* [SLIDE 5] {CODELAB} Installation des outils : java, maven, docker4windows ou docker sur linux, cluster Kubernetes en local
* Récupération des différents dépôts : application, tests de charges et déploiements

### Chapitre 1 (30 minutes)

Objectif : **Mutation testing**, théorie et pratique en utilisant **Pit Test** et **Descartes** (installation et utilisation de l’outil, analyse, correction de TUs en suivant les résultats des analyses)

*"Un stagiaire a touché au code source. Depuis, l'application bugue !!*

*Pourtant, les tests unitaires passaient au vert ! Et la couverture de tests est à 100% sur l'ensemble de notre code métier !*

*A un mois de la MEP, doit t'on blâmer le stagiaire ou l'auteur des tests unitaires ?*

*Comment aurait-on pu éviter d'avoir des TU inutiles ?”*

* [SLIDE 6] Explication mutation testing
* [SLIDE 7] {CODELAB} Ajout des dépendances maven pit test + descartes
* Scénario à affiner :
  + {CODELAB} lancement analyse pit test
  + [SLIDE 8] Pit test c’est moche, mais c’est puissant
  + [SLIDE 9] {CODELAB} Atelier de détection d'erreurs
  + {CODELAB} correction de TUs

### Chapitre 2. 30 minutes

Objectif : **Chaos engineering** et **tests de charge**, théorie et pratique en utilisant **Chaos Monkey** et **Gatling** (installation et utilisation des l’outil, simulation de pannes avec chaos monkey, tirs de références et tirs de charges avec gatling). Analyse en récupérant des métriques.

*"On a des doutes sur la tolérance aux pannes de notre application, en fait personne n'a jamais regardé !*

*Comment peut on faire un état des lieux de la situation ?"*

* explication chaos-monkey / gatling
* {CODELAB} lancement de FuSIIon en local avec les images docker, en configurant chaos-monkey en mode coupure ( simulation de pannes)
* {CODELAB} tir de référence avec gatling et constat : les services tombent tous très vite.

### Chapitre 3. 30 minutes

Objectif : **Résilience**, théorie et pratique en utilisant **Kubernetes** avec Kube, tirs de charges avec Gatling pour constater la résilience de notre application

*"Le constat est fait : ok, notre applicatif répond.. mais est vraiment très sensibles aux pannes !*

*Comment rendre notre application vraiment résiliente ?*

*Comment palier à ce genre de problèmes ?”*

* explication rapide de K8S et du gain en resilience
* {CODELAB} démarrage de FuSIIon en local avec kube
* {CODELAB} constat avec un nouveau tir Gatling : ça marche beaucoup mieux

### Conclusion : 15 minutes

*"Nous voilà rassurés, notre application est tolérante aux défaillances techniques, on va pouvoir aller en production beaucoup plus confiant!*

*En plus, on a même pu challenger nos TU et récupérer des métriques au passage !”*

* debrief sur les outils utilisés
* quelques principes du chaos engineering

## Prérequis :

Outils : maven, java 8, docker et kubernetes en local ( Docker for Windows ou Minikube)

Niveau requis : pour les développeurs intermédiaires, pas de prérequis de connaissances docker & kube, c'est un plus mais c'est surtout un support, pour la partie mutation testing ce sera plus simple si on a pratiqué java un peu