


CODELAB 

Pierre GAULTIER
Sullivan PINEAU
Paul ROYE



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

Poste de travail

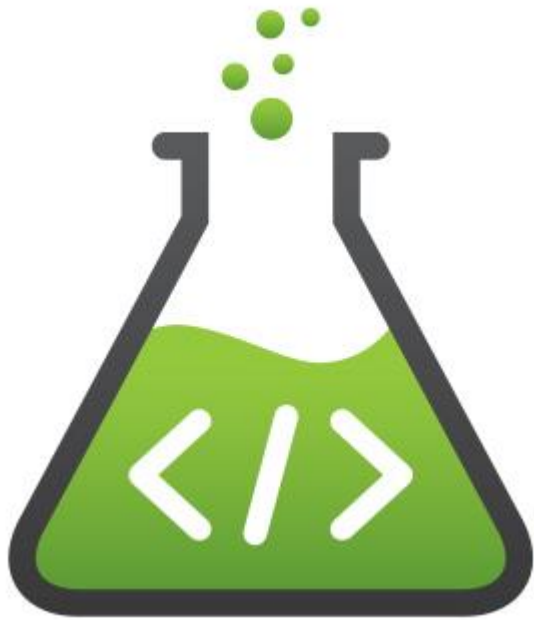


Installation en local :

- JDK 1.8
- Maven
- Docker
- Kubernetes

Poste de travail

Configuration optimale Docker / Kube



Settings

General

Shared Drives

Advanced

Network

Proxies

Daemon

Kubernetes

Advanced

Limit the resources available to Docker Engine.

CPUs: 4



Memory: 6656 MB



Swap: 2048 MB



CPU	RAM	TP2	TP3
2	2Gb	✓	✖
3	4Gb	✓	?
4	6Gb	✓	?
4	8Gb	✓	✓

« Ca part en prod ! »

Testez la résilience de votre appli par le chaos.

Pierre GAULTIER, Sullivan PINEAU, Paul ROYE



Vous êtes ici



<https://tiny.cc/chaos-sii>

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

Pierre GAULTIER, Sullivan PINEAU, Paul ROYE



Ce qui ne nous tue pas...

Résilience

« La résilience est la capacité pour un individu à faire face à une situation difficile ou génératrice de stress. [...] Faculté à « rebondir », à vaincre des situations traumatiques. »

Psychologies.com



Un merveilleux malheur

Chaos Engineering

« Le Chaos Engineering est la discipline de l'expérimentation sur un système distribué afin de renforcer la confiance dans la capacité du système à résister à des conditions turbulentes en production. »

Netflix – Principles of Chaos

Chaos Engineering

« Le Chaos Engineering est la discipline de l'**expérimentation par la destruction ou la dégradation** partielle des composants d'une infrastructure de production en vue de **vérifier sa résilience** »

Benjamin Gakic

« Ca part en prod ! »

Testez la résilience de votre appli par le chaos.

Pierre GAULTIER, Sullivan PINEAU, Paul ROYE



Ca part en prod !



Scénario

"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 ?"



Objectifs

- Introduction au Chaos Engineering
- Manipulation d'outils sur une application concrète :
 - Mutation testing
 - Injection de pannes
 - Tirs de charge
- Mise en place d'une solution pour améliorer la résilience de notre application
- Débriefing

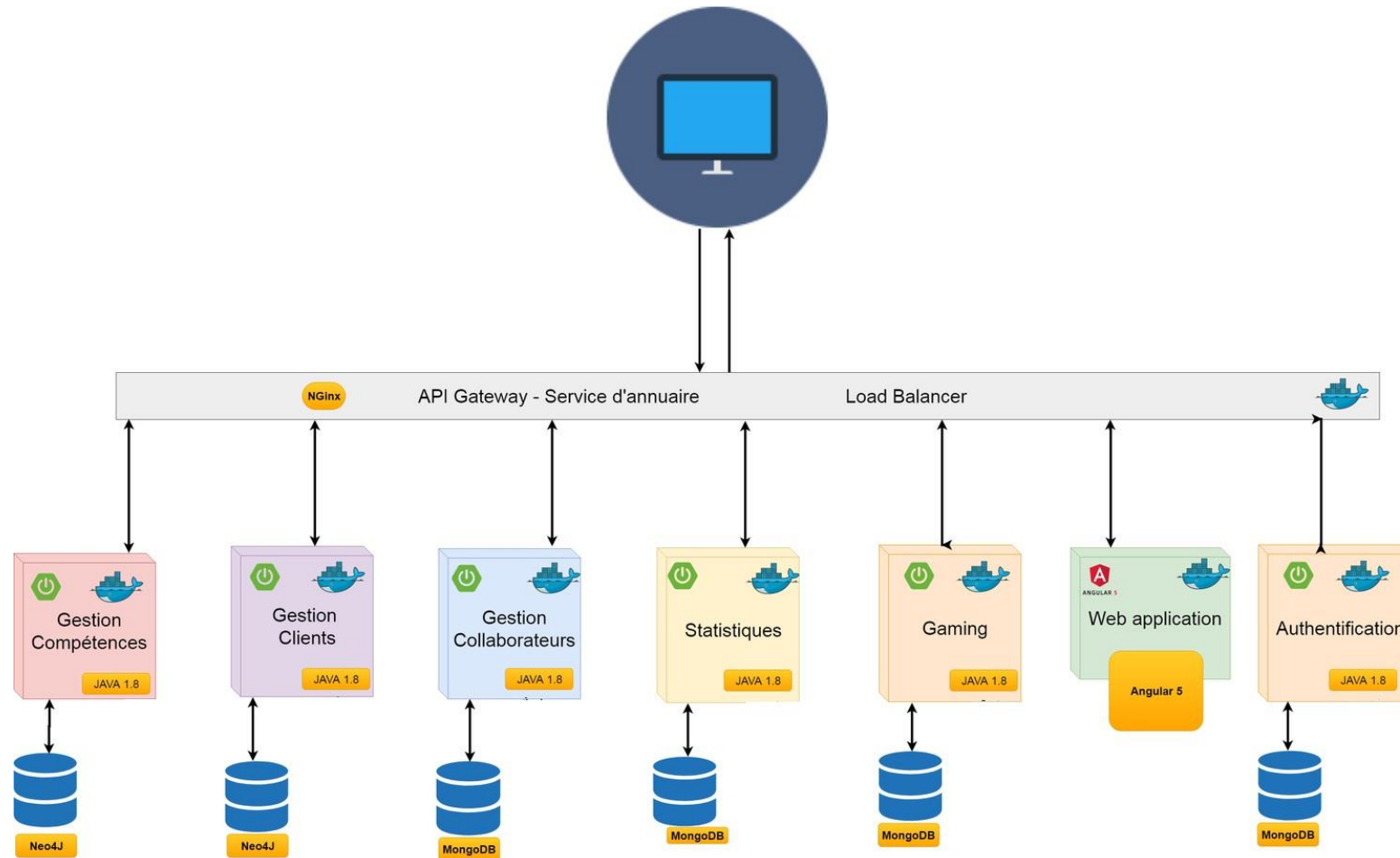
« Ca part en prod ! »

Testez la résilience de votre appli par le chaos.

Pierre GAULTIER, Sullivan PINEAU, Paul ROYE



Notre application



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

Chapitre 1 : Mutation testing



Scénario

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

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

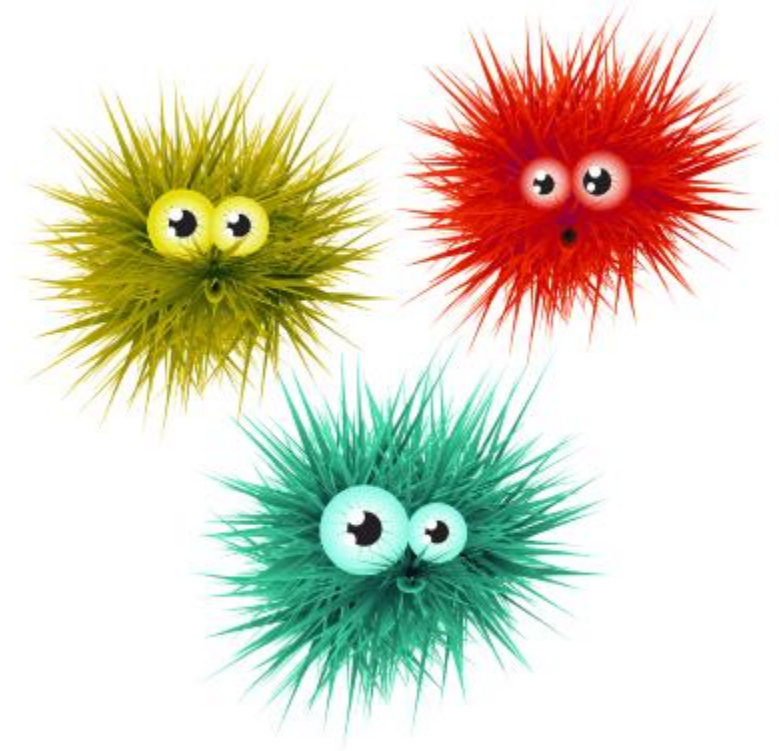
A un mois de la MEP, doit t'on blâmer le stagiaire ou l'auteur des tests unitaires ? Comment aurait-on pu éviter d'avoir des TU inutiles ?”



Les mutants, bientôt parmi nous

Mutation testing

1. Des mutations sont générées dans votre code source
2. Vos tests sont lancés
3. Ce que l'on souhaite :
 - Que les mutants soient « tués »
 - > Que les TU associés à la mutation ne passent plus
4. Si un mutant a survécu :
 - > Le TU associé passe



La qualité de vos tests peut être jugée au regard du pourcentage de mutants tués : **mutation coverage**

Hands On ! TU et couverture de code

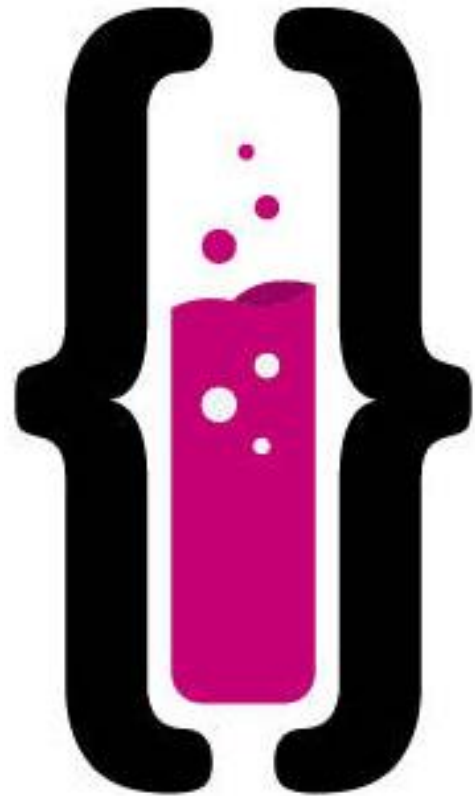


TP 1 :

- Tests unitaires Junit
- Couverture de code avec JaCoCo
- Mutation coverage avec PIT

CompetenceService

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed Cxty	Missed Lines	Missed Methods
actualiserStatistique(String)	<div><div></div></div>	57 %	<div><div></div></div>	66 %	1 3	8 19	0 1
ajouterTagueSur(String, TagueSurJSON)	<div><div></div></div>	61 %	<div><div></div></div>	50 %	2 3	2 12	0 1
competencesUnCollaborateur(String)	<div><div></div></div>	64 %	<div><div></div></div>	50 %	1 2	4 15	0 1
getDetailCompetence(String)	<div><div></div></div>	53 %	<div><div></div></div>	50 %	1 2	1 5	0 1
getStatistiqueSenioriteCompetences(String)	<div><div></div></div>	0 %	<div><div></div></div>	n/a	1 1	4 4	1 1
lambda\$actualiserStatistique\$9(Statistique, DonneeStatistique)	<div><div></div></div>	0 %	<div><div></div></div>	n/a	1 1	3 3	1 1
creerApprendreCompetence(String, String, MentionApprendreSur)	<div><div></div></div>	85 %	<div><div></div></div>	50 %	3 4	2 13	0 1
creerInteretCompetence(String, String, MentionInteretPour)	<div><div></div></div>	85 %	<div><div></div></div>	50 %	3 4	2 13	0 1
findObjectifsByAkCollaborateur(String)	<div><div></div></div>	0 %	<div><div></div></div>	n/a	1 1	2 2	1 1
lambda\$actualiserStatistique\$8(Competence)	<div><div></div></div>	0 %	<div><div></div></div>	n/a	1 1	1 1	1 1
modifierApprendreCompetence(String, String, MentionApprendreSur)	<div><div></div></div>	83 %	<div><div></div></div>	50 %	1 2	1 8	0 1
modifierInteretCompetence(String, String, MentionInteretPour)	<div><div></div></div>	83 %	<div><div></div></div>	50 %	1 2	1 8	0 1
findLikeCompetence(String, int)	<div><div></div></div>	84 %	<div><div></div></div>	50 %	1 2	1 7	0 1
tagsParNom(String, int)	<div><div></div></div>	84 %	<div><div></div></div>	50 %	1 2	1 6	0 1
modifierCompetence(Competence)	<div><div></div></div>	77 %	<div><div></div></div>	50 %	1 2	1 5	0 1
modifierTag(Tag)	<div><div></div></div>	77 %	<div><div></div></div>	50 %	1 2	1 5	0 1
supprimerTag(String)	<div><div></div></div>	77 %	<div><div></div></div>	50 %	1 2	1 6	0 1
lambda\$creerInteretCompetence\$17()	<div><div></div></div>	0 %	<div><div></div></div>	n/a	1 1	1 1	1 1
lambda\$creerInteretCompetence\$16()	<div><div></div></div>	0 %	<div><div></div></div>	n/a	1 1	1 1	1 1
lambda\$creerApprendreCompetence\$15()	<div><div></div></div>	0 %	<div><div></div></div>	n/a	1 1	1 1	1 1
lambda\$creerApprendreCompetence\$14()	<div><div></div></div>	0 %	<div><div></div></div>	n/a	1 1	1 1	1 1
lambda\$getStatistiqueSenioriteCompetences\$5(SenioriteSur)	<div><div></div></div>	0 %	<div><div></div></div>	n/a	1 1	1 1	1 1
lambda\$competencesUnCollaborateur\$2(CompetencesParCollaborateurJSON)	<div><div></div></div>	85 %	<div><div></div></div>	50 %	1 2	0 1	0 1
ajouterCompetentSur(String, CompetentSurJSON)	<div><div></div></div>	100 %	<div><div></div></div>	58 %	5 7	0 32	0 1
createObjectif(ObjectifJSON)	<div><div></div></div>	100 %	<div><div></div></div>	100 %	0 7	0 29	0 1
enleverCharSpeciaux(String)	<div><div></div></div>	100 %	<div><div></div></div>	n/a	0 1	0 20	0 1
createCompetence(Competence)	<div><div></div></div>	100 %	<div><div></div></div>	100 %	0 2	0 9	0 1
ajoutObjectifsCollaborateur(String, List)	<div><div></div></div>	100 %	<div><div></div></div>	75 %	1 3	0 10	0 1
accentToRegex(String)	<div><div></div></div>	100 %	<div><div></div></div>	n/a	0 1	0 10	0 1
createTag(Tag)	<div><div></div></div>	100 %	<div><div></div></div>	100 %	0 2	0 7	0 1
senioriteSemaine(String, String)	<div><div></div></div>	100 %	<div><div></div></div>	100 %	0 2	0 10	0 1
lambda\$ajoutObjectifsCollaborateur\$13(String, Collaborateur, String)	<div><div></div></div>	100 %	<div><div></div></div>	100 %	0 2	0 6	0 1
lambda\$createObjectif\$11(Objectif, String, Integer)	<div><div></div></div>	100 %	<div><div></div></div>	n/a	0 1	0 7	0 1



« Ca part en prod ! »

Testez la résilience de votre appli par le chaos.

Pierre GAULTIER, Sullivan PINEAU, Paul ROYE



PIT : moche et mutant

PIT



```
/**
 * permet de creer une compétence dans le repository correspondant
 *
 * @param competence,
 *         la compétence à créer
 * @return Competence, la compétence créée
 * @throws CompetenceExistanteException
 * @throws JsonProcessingException
 * @throws AmqpException
 */
public Competence createCompetence(Competence competence) throws CompetenceExistanteException {
    LOGGER.info("SERVICE: Appel[createCompetence] - {}", competence);

    1 competence.setAkCompetence("ak_" + competence.getNom());

    1 competence.setAkCompetence(enleverCharSpeciaux(competence.getAkCompetence()));

    1 if(competenceRepository.findOneByAkCompetence(competence.getAkCompetence()).isPresent())
        throw new CompetenceExistanteException();
    }
    this.competenceRepository.save(competence);
    Notification notification = new Notification(Type.TOUS, null, "Nouvelle compétence ");
    1 this.rabbitTemplate.convertAndSend(fanoutExchange.getName(), "", mapper.writeValueAsString(notification));
    1 return competence;
}
```

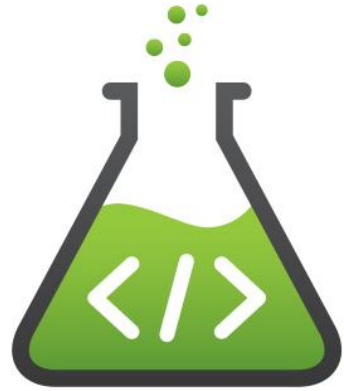
« Ca part en prod ! »

Testez la résilience de votre appli par le chaos.

Pierre GAULTIER, Sullivan PINEAU, Paul ROYE



Hands On ! EXTREME MUTATION



Objectif atelier :
Extreme Mutation avec Descartes



Aller plus loin :
Chasse aux mutants : Correction de TU

Couverture de code VS Mutation testing



JaCoCo

```
public CompetentSur ajouterCompetentSur(String akCollaborateur, (
    LOGGER.info("SERVICE: Appel[ajouterCompetentSur] - {}", {})", a
    Optional<Collaborateur> collab = collaborateurRepository.find
```

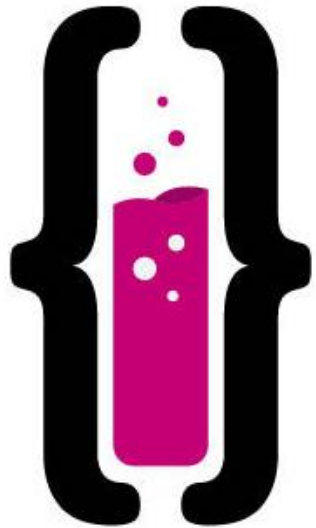
```
    if (!collab.isPresent()) {
        Collaborateur c = new Collaborateur(akCollaborateur);
        collab = Optional.ofNullable(c);
        collaborateurRepository.save(collab.get());
    }
```

PIT

```
public CompetentSur ajouterCompetentSur(String akCollaborateur, Compete
    LOGGER.info("SERVICE: Appel[ajouterCompetentSur] - {}", {})", akC
    Optional<Collaborateur> collab = collaborateurRepository.findOr
```

```
1    if (!collab.isPresent()) {
        Collaborateur c = new Collaborateur(akCollaborateur);
        collab = Optional.ofNullable(c);
        collaborateurRepository.save(collab.get());
    }
```


Couverture de code VS Mutation testing



JaCoCo

```
//Vérifie que la séniorité n'existe pas déjà et que l'évaluation est
♦ if(!senioriteSurRepository.existByAkCollaborateurAndAkCompetence(akCo
    nouvelleEvaluation = true;
    SenioriteSur senioriteSur = new SenioriteSur(collab.get(), compet
♦ if(competentSurJson.getDateDebut() != null) {
    senioriteSur.setDateDebut(competentSurJson.getDateDebut());
}
senioriteSurRepository.save(senioriteSur);
}
```

PIT

```
//Vérifie que la séniorité n'existe pas déjà et que l'évaluation est su
3 if(!senioriteSurRepository.existByAkCollaborateurAndAkCompetence(akCollaborat
    nouvelleEvaluation = true;
    SenioriteSur senioriteSur = new SenioriteSur(collab.get(), competence
1 if(competentSurJson.getDateDebut() != null) {
1     senioriteSur.setDateDebut(competentSurJson.getDateDebut());
}
senioriteSurRepository.save(senioriteSur);
}
```

Chapitre 2 : Tolérance aux pannes



Scénario

"On a des doutes sur la tolérance aux pannes de notre application, en fait personne n'a jamais regardé !" Comment est ce qu'on peut faire un état des lieux de la situation ?"



Toolbox

Chaos Monkey

Provoquer des pannes en environnement réel et de vérifier que le système informatique continue à fonctionner.



Gatling

Gatling est un outil open-source de test de charge et de performance pour applications web.

Chaos Monkey

Installation sur un projet spring boot

```
<dependency>  
  <groupId>de.codecentric</groupId>  
  <artifactId>chaos-monkey</artifactId>  
  <version>2.0.2</version>  
</dependency>
```

Configuration :

```
chaos:  
  monkey:  
    enabled: true  
    assaults:  
      level: 100  
      killApplicationActive: true
```

Sur fond blanc



« Ca part en prod ! »

Testez la résilience de votre appli par le chaos.

Pierre GAULTIER, Sullivan PINEAU, Paul ROYE



Gatling

```
val scn = scenario("BasicSimulation")
  .exec(http("authentication")
    .post(":8080/login")
    .body(StringBody("{\"username\" : \"pgaultier\", \"password\" : \"password\"}"))
    .check(header("Authorization").saveAs("token")))
  .pause(2)
  .exec(http("Collaborateurs")
    .get(":8083/collaborateurs/pgaultier")
    .header("Authorization", "${token}")
    .check(status.is(session => 200)))
  .pause(2)
  .exec(http("Competence/all")
    .get(":8081/competences")
    .header("Authorization", "${token}")
    .check(status.is(session => 200)))
```

```
setUp(
  scn.inject(
    rampUsers(400) during (240 seconds) protocols(httpProtocol)
  ).assertions(
    global.successfulRequests.percent.gt(80),
    forAll.failedRequests.percent.lt(5)
  )
)
```

Sur fond blanc



« Ca part en prod ! »

Testez la résilience de votre appli par le chaos.

Pierre GAULTIER, Sullivan PINEAU, Paul ROYE



Hands on ! TP 2

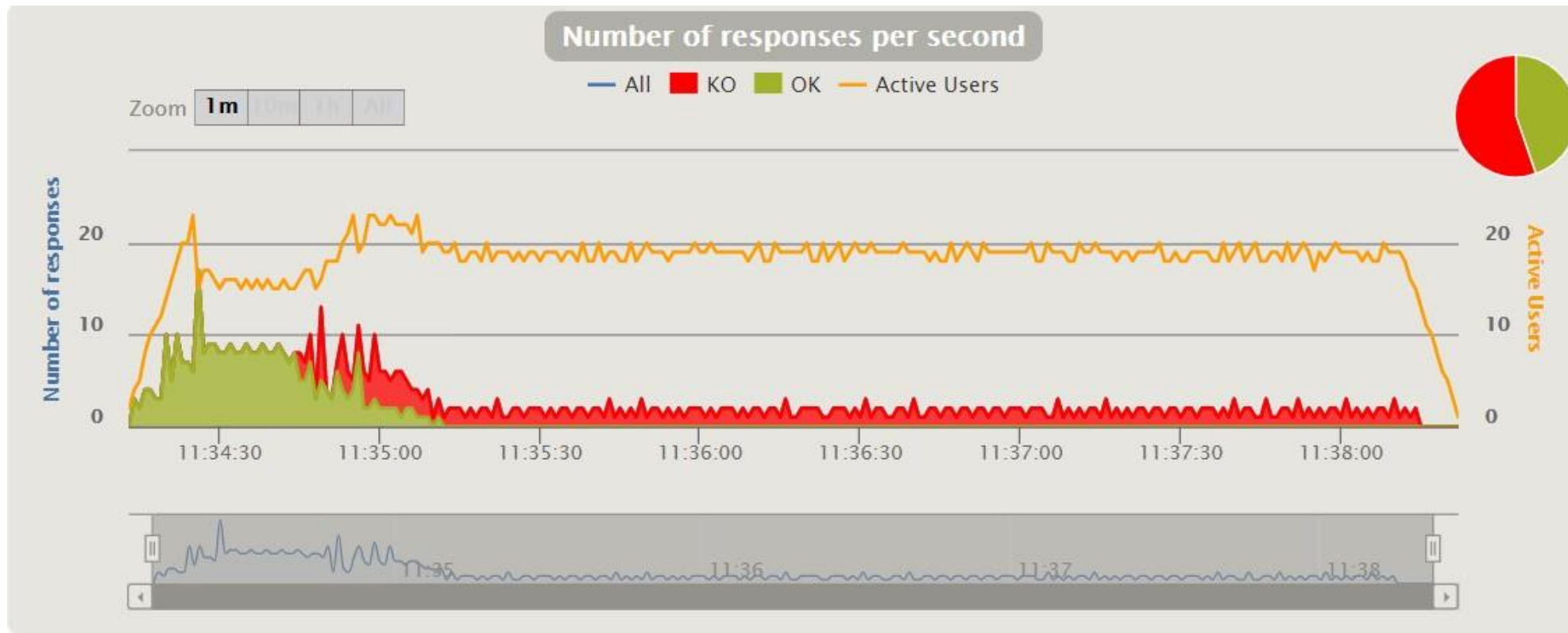


TP 2 :

- Paramétrage du Chaos Monkey
- Démarrage de l'application FuSIlon (Docker)
- Lancement d'un tir de charge
- Analyse des résultats

Analyse du premier tir de charge

Résultat du tir de charge:

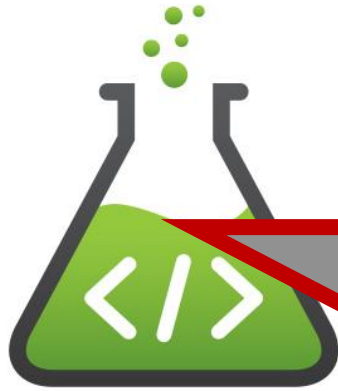


« Ca part en prod ! »
Testez la résilience de votre appli par le chaos.

Pierre GAULTIER, Sullivan PINEAU, Paul ROYE



Constat



Objectif principal : Faire un premier constat sur la résilience de notre

**Taille de
police,
lisibilité**

Alors

- Confrontance avec de la
- Réalité, des exceptions, ...
- Utiliser des assertions dans les tests de charges

Chapitre 3 : Comment devenir résilient ?

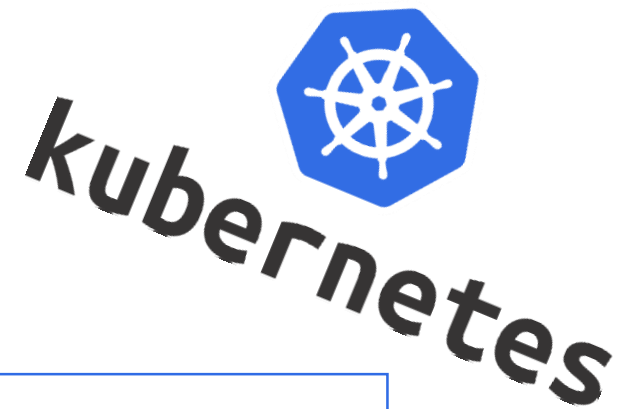


Scénario

"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 ?"



Kubernetes à la rescousse



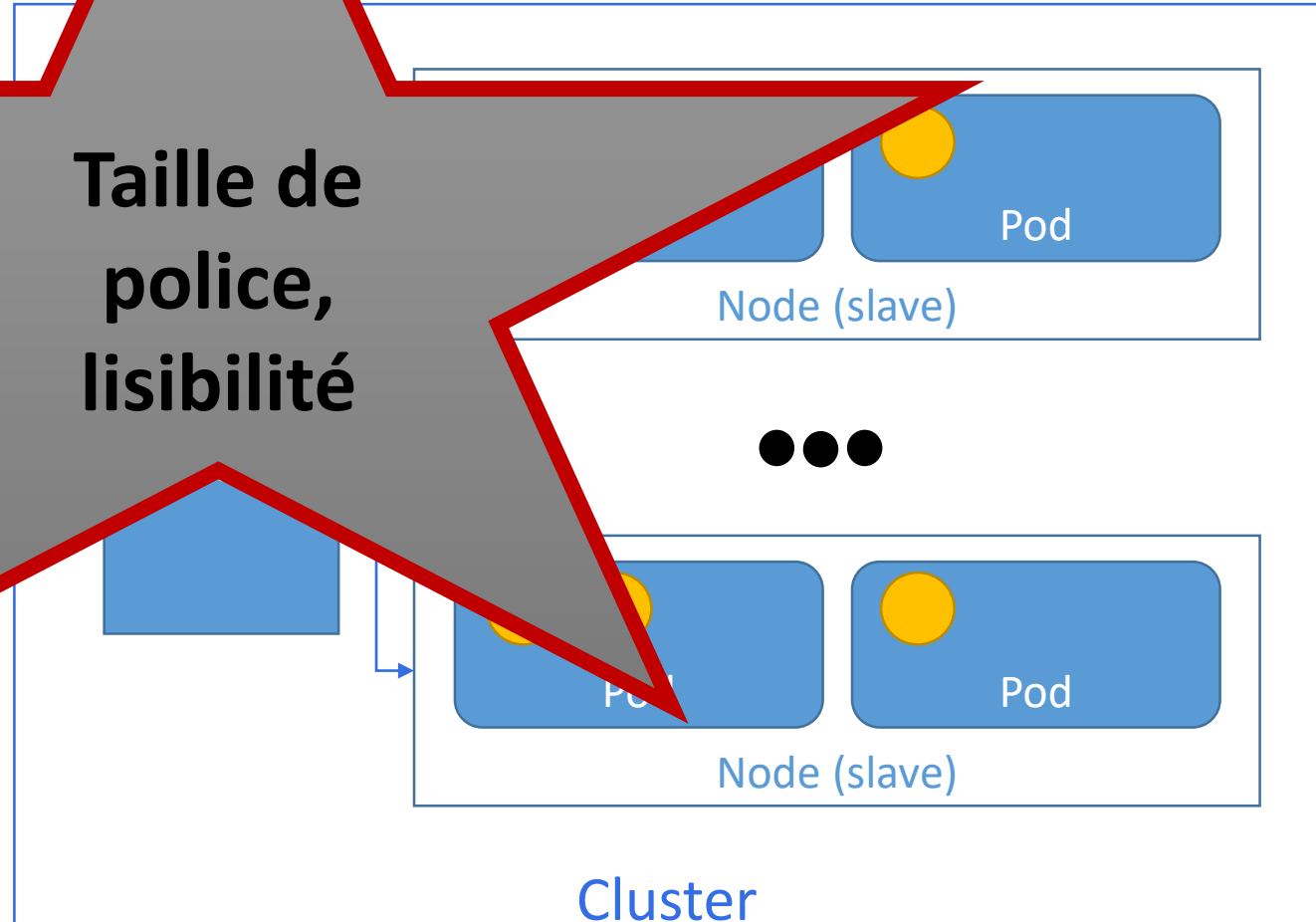
Orchestrateur de conteneur

- Garantie l'intégrité des applications,
- Assure le monitoring,
- Optimise les ressources,
- Gère le stockage, réseau ...



Kubectl/APIs/Dashboard

**Taille de
police,
lisibilité**



« Ca part en prod ! »

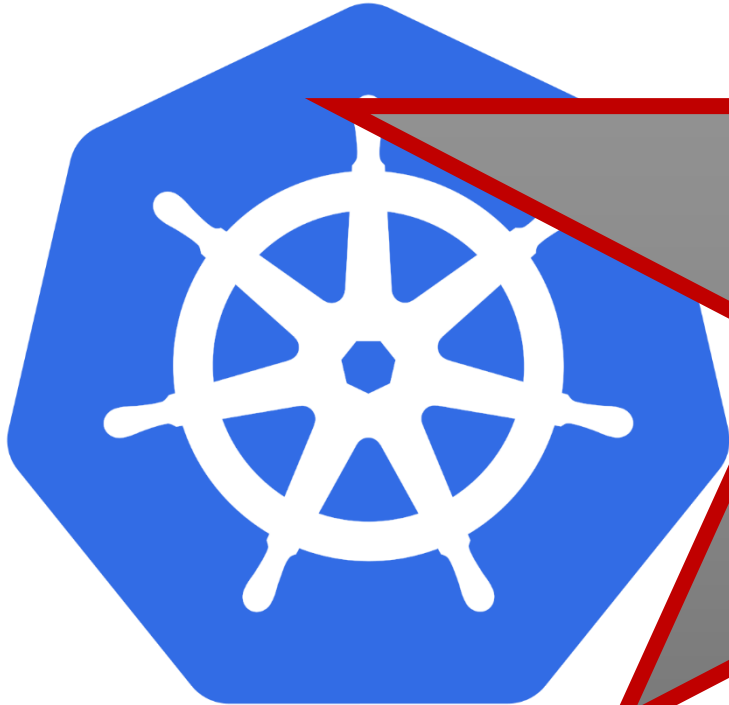
Testez la résilience de votre appli par le chaos.

Pierre GAULTIER, Sullivan PINEAU, Paul ROYE



Paramétrage du Chaos Monkey

Configuration pour les quatre fichiers du répertoire « deployments-fusion » :



Fond blanc

```
profiles-authentication
PROFILES_ACTIVE
monkey
KEY_ENABLED
KEY_LEVEL
- name: CHAOS_MONKEY_KILL_APPLICATION_ACTIVE
  value: "true"
- name: SPRING_DATA_MONGODB_PASSWORD
  valueFrom:
    secretKeyRef:
```

« Ca part en prod ! »

Testez la résilience de votre appli par le chaos.

Pierre GAULTIER, Sullivan PINEAU, Paul ROYE



FuSIlion avec Kubernetes

Lancement de l'application via la commande :



```
NANTES-0169+Sullivan@NANTES-0169 /c/DEV/sources/SII-CodeLab-Chaos/CodeLab-Chaos-TP/TP3-kubernetes-yaml (master)
$ ./run.sh

-----
Lancement du script run.sh
-----
1/6 - create namespace Done
2/6 - create rbac Done
3/6 - create secrets Done
4/6 - create statefulsets Done
5/6 - create default Done
6/6 - deployments Done

-----
ALL DONE
-----
Context "docker-desktop" modified.
NAME                                READY   STATUS    RESTARTS   AGE
fusiion-authentification-b77947d64-skmst  0/1     Pending   0           0s
fusiion-gestion-clients-6ffd6dcdd4-hjkkd  0/1     Pending   0           0s
fusiion-gestion-collaborateurs-54f56cc59b-xc42x  0/1     Pending   0           0s
fusiion-gestion-competences-6f5ff7c898-fthd4  0/1     Pending   0           0s
mongo-0                                    0/1     ContainerCreating 0           1s
neo4j-core-0                              0/1     ContainerCreating 0           1s
nginx-ingress-574f74c78-ml5cp             0/1     ContainerCreating 0           0s
rabbitmq-0                                0/1     ContainerCreating 0           1s
```

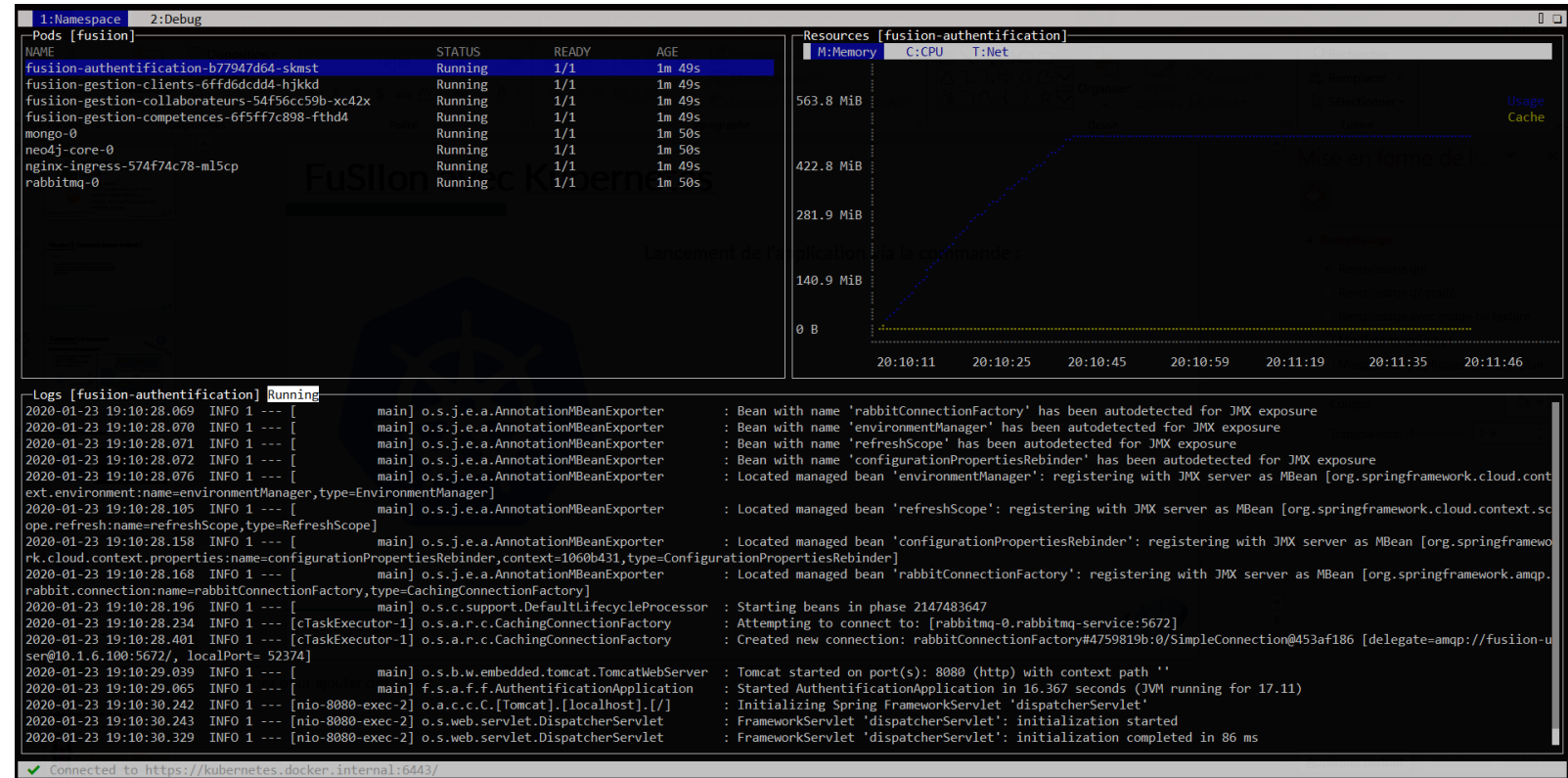
« Ca part en prod ! »

Testez la résilience de votre appli par le chaos.

Pierre GAULTIER, Sullivan PINEAU, Paul ROYE



Lancement du monitoring :



Hands on ! TP 3

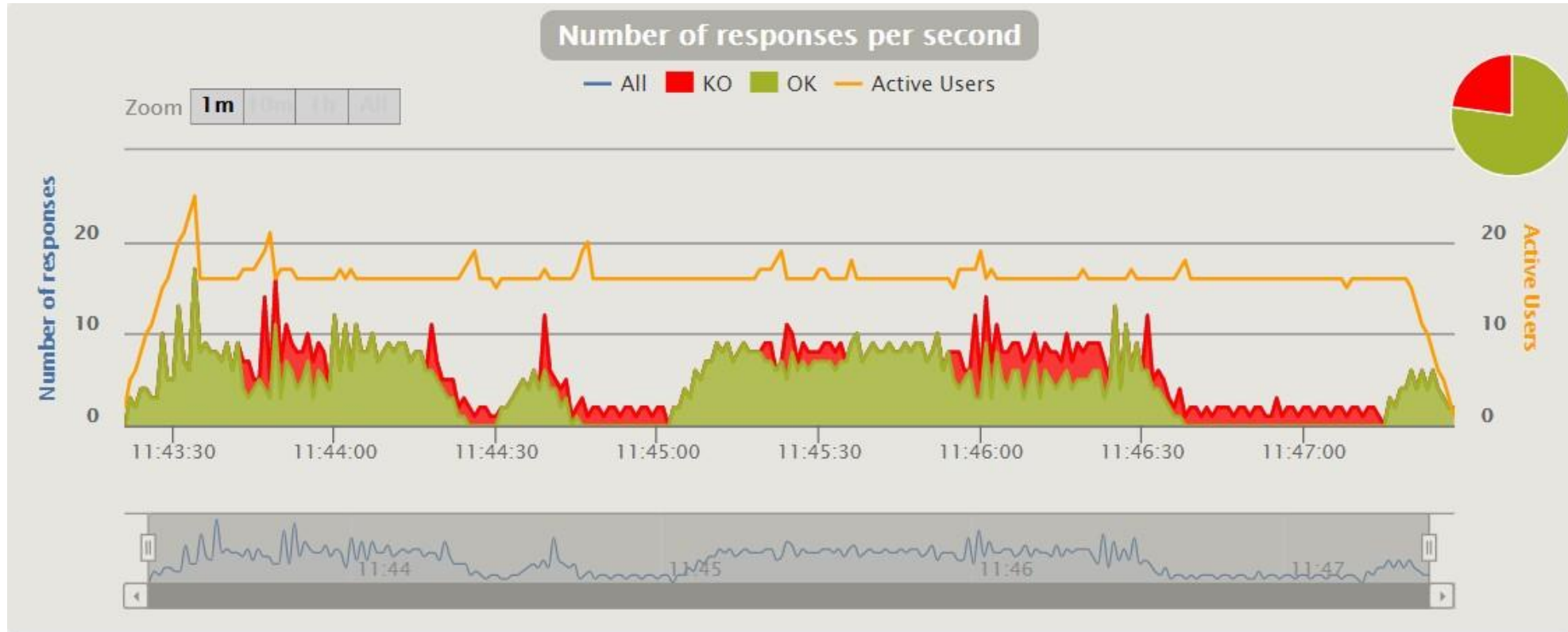
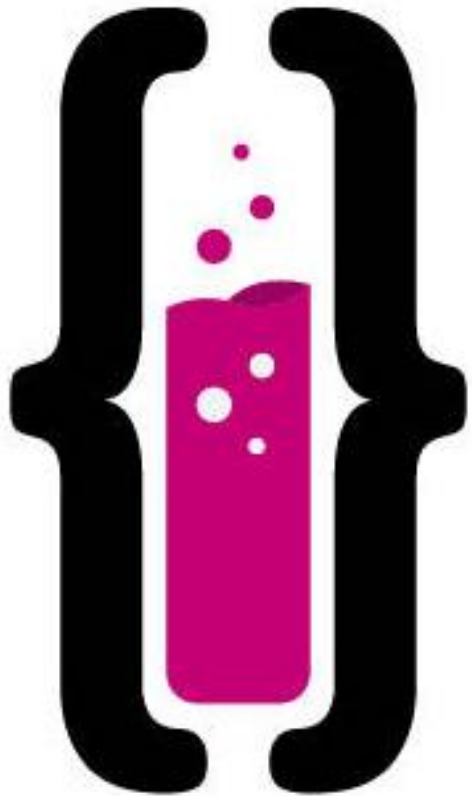


TP 3 :

- Paramétrage du Chaos Monkey
- Démarrage de l'application FuSIlon (Kubernetes)
- Lancement d'un nouveau tir de charge
- Analyse des résultats
- Bonus : augmenter le nombre de réplicas

Analyse du second tir de charge

Résultat du tir de charge avec 1 réplicas par service :



« Ca part en prod ! »

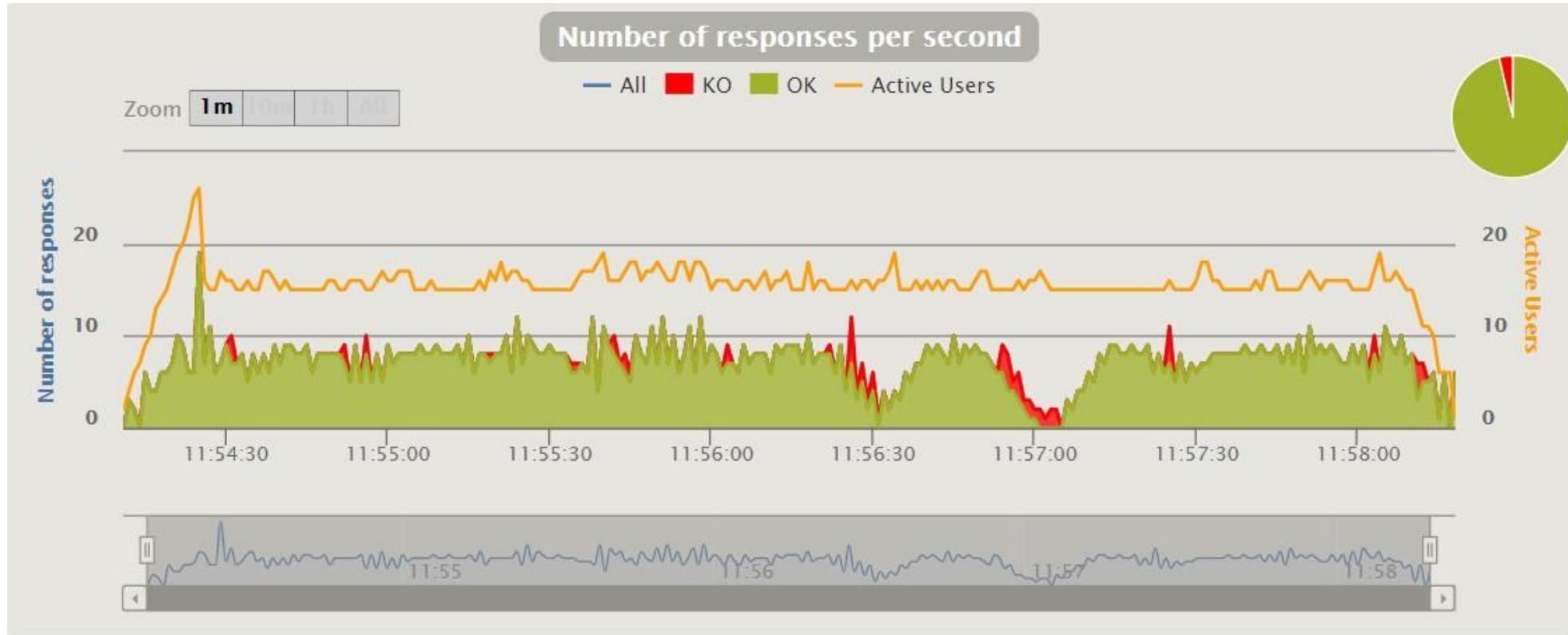
Testez la résilience de votre appli par le chaos.

Pierre GAULTIER, Sullivan PINEAU, Paul ROYE



Analyse du second tir de charge

Résultat du tir de charge avec 3 réplicas par service :

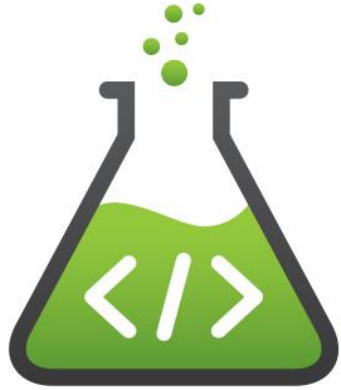


« Ca part en prod ! »
Testez la résilience de votre appli par le chaos.

Pierre GAULTIER, Sullivan PINEAU, Paul ROYE



Constat



Objectif atelier : Faire un second constat de la résilience de notre applicatif



Aller plus loin :

- Tunning des yaml kubernetes
- Outils de supervision avancés

Conclusion



Scénario

"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 de charge et de performance au passage ! "



Conclusion

- Analyse et amélioration de notre patrimoine de TU
- Récupération de métriques de charge et de performances
- Mise à l'épreuve de la tolérance aux pannes de notre applicatif
- Mise en place d'une solution technique pour palier aux problèmes rencontrés



Fin

Questions ?

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

Pierre GAULTIER, Sullivan PINEAU, Paul ROYE



Références

Hands On :

<https://github.com/SII-Codelab-Chaos>

Sources :

<https://codecentric.github.io/chaos-monkey-spring-boot/>

<https://github.com/gatling/gatling-maven-plugin-demo>

<https://pitest.org/>

<https://kubernetes.io/>

<https://www.redhat.com/fr/topics/containers/what-is-kubernetes>

