Engineer

Ingénierie du chaos

tester la résilience avec des pannes injectées sous contrôle

Ce que c’est

L’ingénierie du chaos consiste à injecter volontairement, sous contrôle, des défaillances du monde réel dans un système en fonctionnement pour découvrir s’il les supporte. Les Principes de l’ingénierie du chaos, court document auquel renvoient la plupart des praticiens, la définissent comme « la discipline consistant à expérimenter sur un système afin de renforcer la confiance » dans sa capacité à résister aux conditions turbulentes de la production. Le mot important est expérimenter. Une expérience de chaos ne vérifie pas une fonction par rapport à une spécification : elle pose une question sur le système entier et accepte la réponse, quelle qu’elle soit.

Sa justification commence par les systèmes distribués. Chaque service peut fonctionner correctement, et leurs interactions produire malgré tout des résultats imprévus. Des événements rares du monde réel, comme un serveur qui plante ou une liaison réseau coupée, aggravent la situation. Les Principes énumèrent les faiblesses cachées à cet endroit : mauvais réglages de repli lorsqu’un service est indisponible, délais d’attente mal réglés qui amplifient les nouvelles tentatives, dépendances submergées par leurs appelants, et composant défaillant qui en entraîne plusieurs autres. Rien de tout cela n’apparaît quand chaque partie est testée isolément.

Dans la version des Principes, une expérience comporte quatre étapes :

- Définir l’état stable par une sortie mesurable du système indiquant un comportement normal, comme le débit, le taux d’erreur ou un percentile de latence.
- Formuler l’hypothèse que cet état stable persistera dans un groupe témoin comme dans un groupe expérimental.
- Introduire des variables reflétant des événements réels : serveurs qui plantent, disques qui tombent en panne, connexions réseau coupées.

- Comparer les sorties mesurées des groupes et chercher des éléments contredisant la prédiction.

  Plus l'état stable est difficile à perturber, plus l'équipe peut avoir confiance dans le système. Lorsque l'hypothèse est réfutée, l'équipe a trouvé une faiblesse encore petite, sous surveillance, plutôt qu'au cours d'une panne nocturne.

L'outil le plus connu est Chaos Monkey de Netflix, qui termine aléatoirement des instances de machines virtuelles et des conteneurs en production. Son README le présente comme un exemple d'outil suivant les Principes, avec pour objectif déclaré d'amener les ingénieurs à construire des services résilients aux défaillances d'instances.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce que l'ingénierie du chaos ? » pour éclairer une décision réelle. Nous pouvons vous aider à relier ce concept au système que vous construisez ou achetez. Une supervision et des alertes qui signalent les incidents tant qu'ils restent mineurs, avec les chiffres qui montrent à quelle fréquence ils le restent.

Découvrir nos prestations Contactez-nous →

Vérifiez les hypothèses derrière le terme

Appuyez-vous sur l'explication et les réalisations liées pour formuler les questions de votre projet. Nous pouvons examiner les hypothèses avec vous avant le choix d'un outil. Premier tableau de bord d'observabilité — première vue de l'entreprise sur ses propres systèmes

Premier tableau de bord d'observabilité Contactez-nous →

Pourquoi c'est important

Redondance, basculement et mécanismes de repli sont des promesses sur ce qui se passe en cas de casse, et une promesse jamais testée reste une hypothèse. Une instance de secours qui n'a jamais reçu de trafic, une politique de nouvelle tentative qui n'a jamais rencontré de dépendance lente, une alerte qui ne s'est jamais déclenchée : chacune peut fonctionner, mais le découvrir lors de la première défaillance réelle coûte cher. L'ingénierie du chaos déplace cette découverte à un moment choisi par l'équipe.

Elle teste aussi le système plutôt que ses parties. Les tests unitaires et d'intégration vérifient des comportements auxquels quelqu'un a déjà pensé. Les Principes distinguent explicitement les deux : les expériences de chaos vérifient que le système fonctionne effectivement, au lieu de chercher à valider comment il fonctionne. Elles se concentrent donc sur les sorties perceptibles par un utilisateur, comme le taux d'erreur et la latence, plutôt que sur les mécanismes internes d'un composant.

Les éléments probants se trouvent en production. Le comportement dépend de l'environnement et des profils de trafic. Les Principes soutiennent que seul l'échantillonnage du trafic réel permet de saisir de façon fiable le chemin des requêtes. Le livre Site Reliability Engineering de Google aboutit à la même conclusion en partant de l'autre côté : une charge synthétique peut ne pas révéler où les composants cèdent réellement. Une fois le service protégé contre la surcharge, il suggère donc d'exercer des défaillances sur une partie limitée du service réel, avec une capacité de réserve prête si les protections échouent et qu'un basculement manuel devient nécessaire.

Enfin, elle change les incitations. Le README de Chaos Monkey l'exprime simplement : confronter plus souvent les ingénieurs aux défaillances leur donne une raison de construire des services résilients. Une équipe dont les instances peuvent être terminées n'importe quel jour ouvré conçoit son système en conséquence dès le départ, au lieu de découvrir le manque le jour où une machine meurt sans prévenir.

À quoi cela ressemble en pratique

Une expérience de chaos est écrite avant son exécution, comme un plan de test. En voici une pour une boutique en ligne fictive, dont tous les nombres sont inventés pour l'exemple :

```
Experiment:   the payment provider slows down
Steady state: checkout success rate and p99 checkout latency, per 5 min
Hypothesis:   with 300 ms added to calls to the payment provider, the
              experimental group's success rate stays within 0.2 points
              of the control group's
Groups:       2% of checkout traffic in each, assigned at random
Variable:     300 ms of latency on the payment client, nothing else
Abort if:     success rate gap above 1 point for 2 minutes in a row
Window:       30 min, working hours, on-call engineer told in advance
Result:       hypothesis disproved; retries tripled the calls to the
              provider and p99 latency passed the 1 s timeout
Follow-up:    a retry budget per request; rerun the experiment after
```

L'hypothèse réfutée est ici le résultat utile. La faiblesse est découverte sur 2 % du trafic pendant quelques minutes, avec une condition d'abandon déjà convenue. La suite consiste en une modification technique plutôt qu'en des excuses.

La variable elle-même est souvent injectée avec des outils ordinaires. Sous Linux, la discipline de mise en file d'attente netem émule les propriétés des réseaux réels, notamment retard, perte, duplication et corruption. Par exemple :

```
# experiment 1: add 100 ms of delay to every packet leaving eth0
tc qdisc add dev eth0 root netem delay 100ms
tc qdisc del dev eth0 root

# experiment 2: drop one packet in a thousand, at random
tc qdisc add dev eth0 root netem loss 0.1%
tc qdisc del dev eth0 root
```

La ligne `del` est importante : elle appartient à une étape de nettoyage exécutée que l'expérience réussisse, échoue ou plante à mi-chemin. Netem accepte aussi une valeur `seed` pour reproduire les pertes aléatoires. Son manuel précise que, pour obtenir des résultats de performance TCP réalistes, netem doit être placé à l'entrée de l'hôte récepteur.

À l'échelle de machines entières, Chaos Monkey illustre des réglages initiaux prudents. À l'installation, `enabled` vaut false et `leashed` vaut true : les terminaisons sont seulement simulées et journalisées. Lorsqu'il est activé, il traite des groupes d'instances, par application, stack ou cluster, avec un intervalle moyen de deux jours entre terminaisons par groupe et un minimum d'un jour, uniquement les jours ouvrés, entre 09:00 et 15:00 dans le fuseau configuré. Comptes, régions et stacks peuvent être exclus. La configuration prévoit aussi un emplacement pour un vérificateur signalant si une panne est en cours. Les implémentations fournies pour ce rôle ne font rien : l'opérateur doit donc fournir cette vérification. La liste de tests de défaillance en production du livre SRE est tout aussi concrète : réduire vite ou lentement le nombre de tâches, perdre d'un coup la capacité d'un cluster et supprimer silencieusement le trafic destiné aux backends, y compris ceux qui ne sont pas critiques mais dont le silence peut bloquer un frontend.

Sur ce site, l'étude de cas de septembre 2026 expliquant comment le bon fonctionnement du blocage des intrusions a été démontré à chaque exécution n'est pas une expérience de chaos au sens complet, mais en reprend la structure. Chaque exécution de durcissement bloque une adresse d'un bloc réservé par les normes à la documentation, relit la règle résultante dans le noyau au lieu de se fier au seul état du service, puis lève le blocage dans une étape de nettoyage exécutée que la vérification réussisse ou échoue. L'adresse de test n'appartenant à personne, un blocage résiduel après une exécution échouée ne peut exclure un

vrai réseau. La page expose aussi le coût : écrire une règle en production à chaque exécution est une intervention réelle dans un système actif, acceptée parce que l'autre solution s'était déjà révélée pire. L'implémentation de sauvegarde et de test de restauration applique cette idée à la récupération dans sa conception : un traitement mensuel restaure un fichier connu dans un emplacement temporaire et compare ses octets. Le relevé d'implémentation du 2026-10-27 indique que les minuteurs étaient désactivés faute de dépôt configuré et ne consigne aucune restauration achevée en conditions réelles. Le traitement existe dans le code ; un exercice de récupération réussi reste un résultat distinct à fournir.

Où les problèmes surviennent

Casser sans hypothèse. Tuer un serveur au hasard pour regarder ce qui se passe est une démonstration, pas une expérience. Écrivez d'abord l'état stable, l'hypothèse et la condition d'abandon. Si l'équipe ne voit pas son état stable en temps réel sur un tableau de bord, c'est la première chose à corriger. L'expérience peut attendre.

Observer les mécanismes internes plutôt que les sorties. L'utilisation du CPU d'un hôte ou la longueur d'une file renseigne peu sur ce que les utilisateurs ont remarqué. Les Principes demandent des sorties mesurables du système entier, comme débit, taux d'erreur et percentiles de latence, car c'est ce que vise l'hypothèse.

Un rayon d'impact sans limite. Les Principes acceptent certains effets négatifs à court terme, mais confient à l'ingénieur la responsabilité d'en contenir les conséquences. Commencez par une instance ou un petit pourcentage du trafic, gardez un interrupteur d'abandon fonctionnant sans l'outil qui a causé les dégâts, et ne lancez jamais une expérience pendant un incident en cours.

Ne jamais quitter la préproduction. Un environnement de préproduction est un bon premier pas, mais un mauvais dernier, car son trafic, ses données et son voisinage ne sont pas ceux de la production. Une fois l'expérience exécutée sans problème avant la production, passez à une petite part du trafic réel au lieu de prendre les résultats de préproduction pour la réponse définitive.

Ne l'exécuter qu'une fois. Un système ayant survécu à une expérience en mars a subi de nombreuses modifications depuis. Les Principes décrivent les expériences manuelles comme exigeantes en temps de travail et intenables à terme. Ils

demandent de les automatiser pour qu'elles s'exécutent en continu, afin qu'une régression de résilience apparaisse la semaine même de son introduction.

Ne faire que tuer des machines. Les plantages francs sont le cas facile, car quelque chose les détecte. Les Principes citent aussi des défaillances logicielles, comme des réponses mal formées, et des événements qui ne sont pas des défaillances, comme une pointe de trafic ou une mise à l'échelle. Le livre SRE ajoute un backend qui accepte les requêtes sans jamais répondre, ce qui peut encore nuire à un frontend quand les requêtes ont un long délai avant expiration. Priorisez les variables selon leur impact probable ou leur fréquence.

Des constats sans suite. Une hypothèse réfutée est une faiblesse désormais identifiée. Donnez-lui un responsable et une priorité dans le même outil de suivi que le reste, puis relancez l'expérience après la livraison du correctif pour confirmer que l'état stable tient désormais.

Pour aller plus loin

- Principes de l'ingénierie du chaos — définition, quatre étapes d'une expérience et cinq principes avancés, dernière mise à jour en mars 2019.
- Chaos Monkey — la documentation Netflix de cet outil de terminaison d'instances, qui nécessite Spinnaker.
- Chaos Monkey : configuration du comportement avec Spinnaker — fréquence des terminaisons, groupes et exceptions, avec les valeurs prédéfinies.
- Chaos Monkey : format du fichier de configuration — chaque réglage et sa valeur prédéfinie, y compris le mode leashed et la plage horaire ouvrée.
- Traiter les défaillances en cascade — chapitre 22 du livre Site Reliability Engineering de Google, dont les tests jusqu'à la défaillance et les tests de panne en production.
- tc-netem(8) — la page de manuel Linux de l'émulateur réseau utilisé pour injecter retard, perte et corruption.



Glossaire

Choisissez selon vos contraintes réelles

Présentez le parcours de déploiement, les risques opérationnels et les personnes d'astreinte. Nous pouvons définir un changement avec des vérifications, une procédure de reprise et une responsabilité claire.

Consulter les tarifs Contactez-nous →

Rendez la terminologie concrète

Demandez comment « Qu'est-ce que l'ingénierie du chaos ? » s'applique à votre organisation. Un exemple concret tiré de votre système actuel suffit pour engager la discussion.

Découvrir nos prestations

welcome@engineer.company +45 50 10 61 84 Copenhague, Danemark

Emportez cette page avec vous

Fiche de contact

Créé avec soin et amour par D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Droit d'auteur © 2025 – aujourd'hui · Engineer ApS · Tous droits réservés.

Imprimé depuis https://engineer.company/fr/glossary/chaos-engineering/