Examen *DevOps & Continuous Testing*

- Date: 5 mai 2017
- Durée: 2 heures
- **Aucun document autorisé; barème donné à titre indicatif.**

Vous devez répondre aux questions posées dans le sujet sur la copie d'examen fournie, avec d'éventuelles feuilles intercalaires. Vous ne pouvez pas sortir de la salle d'examen durant la première heure (avant 14h30), ni durant le dernier quart d'heure (après 15h15). Lisez bien tout le sujet avant de commencer à répondre aux questions. Les parties 1 et 2 sont prévues pour représenter environ 45 minutes de travail. Soyez synthétique dans vos réponses (quitte à utiliser des listes à puces), nous n'attendons pas un roman mais une analyse pertinente et précise. N'hésitez pas à mobiliser des compétences acquises dans d'autres cours pour répondre aux questions d'un point de vue global.

Toute fraude identifiée sera transmise au conseil de discipline de l'UNS.

Partie 1: *Question de recul* (5 points)

En 1996, la premier lancement de la fusée Ariane 5 se solde par sa destruction au bout de 37 secondes, un échec dû à une erreur dans le code de la centrale inertielle, qui donne au calculateur de vol les informations sur la position et les mouvements de la fusée. Cette erreur a été attribuée à une mauvaise retranscription des spécifications du programme Ariane 4 vers Ariane 5, un délai d'attente des gyroscopes qui était passé de 60 à 80 secondes. Un simple petit bug, qui aurait quand même coûté 500 millions de dollars...

Quelle réflexion un tel problème vous inspire-t-elle quant à la stratégie de test à mettre en oeuvre sur un projet informatique afin de se prémunir au mieux de ce genre de risque?

Partie 2: *Docker Compose* (3 points)

Durant les TDs, vous avez créé trois images Docker, une pour le client, une pour le serveur de The Cookie Factory, et enfin une pour le serveur de paiement simulant la banque. Puis vous avez assemblé au moins deux de ces images au sein d'un groupe avec Docker Compose.

Pourquoi grouper des images docker dans un ensemble avec Compose ? Quels avantages en tirer, mais aussi quelles limitations ?

Partie 3: Étude de cas (12 points)

Afin de limiter les risques de fraude aux élections, quatre étudiants décident de lancer une startup qui fournira des cartes d'électeurs munies de QR code unique. Dans les bureaux de vote, les assesseurs utiliseront une application sur leur smartphone qui scanne le QRCode d'un électeur, et envoie via Internet à un serveur central l'identifiant de cet électeur, en temps réel, afin d'empêcher ce numéro de prétendre à un vote dans un autre bureau, comme ce peut être le cas avec des cartes en papier. Ainsi, si une carte a été délivrée en double par erreur, l'électeur ne peut tout de même pas l'utiliser deux fois.
Il leur faut donc développer un système composé d'un serveur et d'une application mobile. Afin de fixer les idées, admettons que le serveur soit développé en java, et offre une seule API "*boolean puisJeVoter(string identifiantDElecteur, string identifiantDeBureauDeVote)*", qui répond oui au premier appel, puis non aux appels suivants avec le même identifiant. Ce serveur stocke dans une base de donnée la liste des électeurs ayant voté.
Bien entendu, changer la date des élections n'est pas une option, il faut donc qu'au jour J l'application livrée fonctionne correctement.
Cependant, ces quatre étudiants venant de régions différentes, ils choisissent d'emménager chacun dans leur région d'origine, et donc travaillent à distance.

1. Décrivez comment ces quatre étudiants organisent leur développement afin de collaborer au mieux et permettre à chacun de développer à son rythme.
2. L'état, commanditaire du système, veut être tenu au courant régulièrement de l'avancement du développement. Comment faire pour lui garantir que le développement aussi bien du serveur que de l'application mobile suit son cours, et reste conforme aux spécifications ?
3. Que proposez-vous pour tester l'application mobile et le serveur séparément, puis ensemble ? Comment garantir la qualité du code produit ?
4. Le système va servir de manière très épisodique, cependant les jours d'élection, la charge sera importante et on ne peut pas se permettre une panne. Comment passer d'un code fonctionnellement correct à une application déployée à l'échelle nécessaire, quelle stratégie de déploiement et comment l'implémenter ?