**A. État des lieux**

**1. Présentation de l'agence**

Fondée en 2007 à Paris, Furious Ducks est une agence web avant-gardiste spécialisée dans les technologies open source. Portée par des valeurs d'innovation, de transparence et d'excellence technique, l'agence s'est forgée une réputation solide dans la création de solutions digitales performantes et accessibles.

Sous la direction visionnaire de M. Guido Brasletti, Furious Ducks a connu une croissance constante, atteignant aujourd'hui un effectif de 45 collaborateurs passionnés. Notre équipe pluridisciplinaire rassemble des développeurs, designers, chefs de projets et experts UX qui partagent une même passion : créer des expériences digitales mémorables et fonctionnelles.

**Informations juridiques :**

* Dénomination sociale : Furious Ducks
* Siège social : 14 rue de la Paix, 75002 Paris
* Statut juridique : SAS (Société par Actions Simplifiée)
* Capital social : 100 000€
* N° RCS : Paris B 498 235 698
* N° SIREN : 498 235 698
* N° SIRET : 498 235 698 00021
* Code APE/NAF : 6201Z (Programmation informatique)
* N° TVA intracommunautaire : FR 23 498 235 698

**2. Présentation de la problématique**

M. Brasletti a identifié une désorganisation croissante au sein de l'agence, entraînant une perte d'efficacité notable dans la production et la livraison des projets. Cette situation résulte d'une accumulation de problématiques techniques et organisationnelles :

* **Absence de versionning standardisé** : Chaque équipe utilise ses propres méthodes de gestion de versions, conduisant à des conflits fréquents et des pertes de code.
* **Environnements de développement hétérogènes** : Les configurations différentes entre postes de développement provoquent des erreurs du type "ça marche sur ma machine".
* **Déploiements manuels et risqués** : L'absence d'automatisation des déploiements engendre des erreurs humaines fréquentes et des temps d'indisponibilité.
* **Backups irréguliers et non testés** : Les sauvegardes sont réalisées de façon ad hoc sans procédure de vérification.
* **Tests non systématiques** : L'absence de tests automatisés conduit à des régressions fréquentes et des bugs découverts tardivement.
* **Documentation technique insuffisante** : Les projets manquent de documentation, rendant difficile leur reprise par d'autres équipes.
* **Absence de métriques** : Sans surveillance des performances, il est impossible d'anticiper les problèmes ou d'optimiser les ressources.

Pour répondre à ces problématiques, M. Brasletti attend les livrables suivants :

* Un workflow de production industrialisé basé sur Docker
* Une transition méthodologique vers l'Agile
* La mise en place d'un système d'intégration et de déploiement continus
* Des procédures standardisées pour toutes les étapes de production
* Un système de métriques et de surveillance
* Des solutions de tests et de backup automatisés

**3. Méthodologie de projet actuelle**

Actuellement, Furious Ducks utilise le modèle en V pour la gestion de ses projets. Cette méthodologie séquentielle divise le projet en phases distinctes (analyse des besoins, spécifications, conception, développement, tests unitaires, tests d'intégration, validation, déploiement), avec une correspondance entre chaque phase descendante et sa phase ascendante de tests.

Cette approche présente plusieurs limitations pour une agence web moderne :

* **Rigidité face aux changements** : Une fois les spécifications validées, toute modification devient coûteuse et complexe à intégrer.
* **Détection tardive des bugs** : Les tests intervenant tardivement dans le cycle, les problèmes sont découverts à un stade avancé du projet.
* **Livraison tardive de valeur** : Le client ne voit le résultat qu'en fin de projet, créant des attentes parfois déçues.
* **Communication limitée** : Les silos entre équipes de conception et de développement créent des malentendus techniques.
* **Planification difficile** : Les estimations initiales sont souvent optimistes, entraînant des retards systématiques.
* **Manque d'adaptabilité** : Face à l'évolution rapide des technologies web, cette méthode manque de flexibilité.

Ces contraintes deviennent particulièrement problématiques pour les projets web nécessitant agilité et itérations rapides, comme le futur projet du jeu-concours pour Thé Tip Top.

**B. Méthodologie de projet à venir**

Pour moderniser son approche de gestion de projet, nous préconisons l'adoption de la méthodologie Scrum, complétée par des pratiques DevOps.

**Présentation de Scrum :** Scrum est un framework Agile structuré autour de cycles de développement courts appelés "sprints" (généralement 2 semaines). Cette méthode repose sur trois rôles clés (Product Owner, Scrum Master, équipe de développement), des événements définis (Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospective) et des artefacts (Product Backlog, Sprint Backlog, Incrément).

**Avantages pour Furious Ducks :**

* **Livraison régulière de valeur** : Chaque sprint produit un incrément fonctionnel
* **Flexibilité face aux changements** : Adaptation possible des priorités entre chaque sprint
* **Transparence accrue** : Visibilité constante sur l'avancement via les review et burndown charts
* **Amélioration continue** : Identification et résolution rapide des problèmes grâce aux rétrospectives
* **Satisfaction client** : Implication régulière des parties prenantes via les démonstrations

**Mise en place pratique :**

* Constitution d'équipes pluridisciplinaires de 5-7 personnes
* Formation des Product Owners et Scrum Masters
* Implémentation de rituels Scrum (Daily stand-up meetings, Sprint Planning, Retrospectives)
* Utilisation de Jira comme outil de gestion des backlog et sprints
* Intégration avec le workflow CI/CD pour automatiser les tests et déploiements
* Mise en place de tableaux Kanban digitaux pour visualiser l'avancement

**Outils recommandés :**

* **Jira** pour la gestion des backlogs et sprints
* **Confluence** pour la documentation collaborative
* **Miro** pour les ateliers d'idéation à distance
* **Slack** pour la communication d'équipe

Cette approche Agile sera complétée par des pratiques DevOps pour assurer la continuité entre développement et exploitation des projets.

Je peux continuer avec les sections suivantes si cela vous convient. Préférez-vous que je poursuive avec les sections C, D et E, ou souhaitez-vous des ajustements sur cette première partie ?

**C. Introduction au mouvement DevOps et DevSecOps**

**DevOps et DevSecOps : définition et principes**

Le DevOps est une culture organisationnelle et technique qui vise à décloisonner les équipes de développement (Dev) et d'exploitation (Ops). Cette approche favorise la collaboration, l'automatisation des processus et l'amélioration continue pour livrer des applications plus rapidement et avec une meilleure qualité. Le DevOps s'articule autour de cinq piliers fondamentaux : la collaboration, l'automatisation, la mesure, le partage et l'amélioration continue.

Le DevSecOps enrichit cette approche en y intégrant la sécurité (Sec) dès les premières phases du cycle de développement. Au lieu de considérer la sécurité comme une étape finale ou un frein, elle devient partie intégrante du processus de développement, ce qui permet d'identifier et de corriger les vulnérabilités beaucoup plus tôt dans le cycle.

**Avantages pour Furious Ducks**

L'adoption du DevOps et DevSecOps présente plusieurs avantages significatifs pour l'agence :

* **Réduction du Time-to-Market** : Cycles de développement et de déploiement plus courts
* **Amélioration de la qualité** : Tests automatisés systématiques pour réduire les bugs
* **Sécurité renforcée** : Intégration des tests de sécurité dès le début du développement
* **Fiabilité accrue** : Déploiements plus prévisibles et moins risqués
* **Collaboration optimisée** : Suppression des silos entre équipes
* **Satisfaction client** : Livraisons plus fréquentes et de meilleure qualité

**Topologies DevOps retenues**

Pour Furious Ducks, nous préconisons l'adoption d'une topologie évolutive en trois phases :

**Phase 1 : Équipe DevOps dédiée (6 premiers mois)** Une équipe spécialisée DevOps sera formée pour mettre en place l'infrastructure, les outils et les procédures. Cette équipe servira de centre d'expertise et formera les autres équipes aux pratiques DevOps. Cette approche permet une transition en douceur sans perturber les projets en cours.

**Phase 2 : Modèle d'équipes avec champions DevOps (6-18 mois)** Chaque équipe projet intégrera un "champion DevOps" responsable de l'application des pratiques et de la liaison avec l'équipe DevOps centrale. Cette structure permet de diffuser progressivement la culture DevOps tout en maintenant une expertise centralisée.

**Phase 3 : Modèle d'équipes autonomes (18+ mois)** À terme, les pratiques DevOps seront intégrées dans chaque équipe projet, qui deviendra autonome pour gérer l'ensemble du cycle de vie des applications. L'équipe DevOps centrale évoluera vers un rôle de gouvernance et d'innovation, gérant les plateformes communes et définissant les standards.

Cette évolution progressive permettra à Furious Ducks d'adopter la culture DevOps tout en maintenant sa productivité et en minimisant les risques liés au changement organisationnel.

**D. Analyse du workflow CI/CD**

**1. Présentation Workflow CI/CD**

**L'Intégration Continue (CI)** est une pratique de développement logiciel où les membres d'une équipe intègrent fréquemment leur travail, généralement plusieurs fois par jour. Chaque intégration est vérifiée par des builds automatisés et des tests pour détecter rapidement les erreurs.

**Le Déploiement Continu (CD)** est l'extension logique de l'intégration continue. Il s'agit d'automatiser le processus de livraison d'applications vers les environnements de production après avoir passé avec succès les phases de tests.

**Contraintes du workflow pour Furious Ducks :**

* Compatible avec différents types de projets (sites statiques, CMS, applications mobiles)
* Entièrement basé sur Docker
* Utilisation exclusive de technologies open source ou libres
* Sécurisation des accès avec différents niveaux de permissions
* Automatisation des backups et restaurations
* Adaptabilité aux méthodologies Agile
* Documentation accessible pour les nouveaux collaborateurs

**Tâches à automatiser via le workflow :**

1. **Gestion du code source** : Contrôle de versions, revues de code, protection des branches
2. **Compilation et build** : Construction des livrables pour différentes plateformes
3. **Tests automatisés** : Tests unitaires, d'intégration, fonctionnels et de sécurité
4. **Optimisation des assets** : Minification, compression des images et du code
5. **Déploiement** : Mise en ligne automatisée sur les environnements de développement, préproduction et production
6. **Surveillance** : Métriques de performance et détection des anomalies
7. **Feedback** : Notification des résultats des builds et des tests
8. **Backups** : Sauvegarde régulière des données et du code

L'automatisation de ces tâches permettra de réduire les erreurs humaines, d'accélérer les livraisons et d'améliorer la qualité globale des projets de l'agence.

**2. Choix techniques**

Pour réaliser notre workflow CI/CD, nous avons sélectionné les services suivants :

**Système de Contrôle de Version (SCM) : GitLab Community Edition**

* Justification : Solution open source complète intégrant gestion de code, CI/CD, gestion des issues et wiki
* Bonnes pratiques : Adoption de GitFlow comme workflow de branches (master, develop, feature/, release/, hotfix/)
* Configuration : Serveur dédié avec 4 CPU, 8 Go RAM, 1 To SSD

**Serveur d'Intégration Continue : Jenkins**

* Justification : Outil open source mature, hautement personnalisable avec un large écosystème de plugins
* Bonnes pratiques : Configuration as Code avec JenkinsFiles dans chaque projet
* Configuration : Serveur dédié avec 8 CPU, 16 Go RAM, 500 Go SSD

**Gestion de conteneurs : Docker et Docker Compose**

* Justification : Standard de l'industrie pour la conteneurisation, permettant la portabilité et l'isolation
* Bonnes pratiques : Images légères, multi-stage builds, principe d'immutabilité
* Configuration : Utilisation sur tous les serveurs d'environnement

**Orchestration de conteneurs : Kubernetes**

* Justification : Gestion avancée des déploiements, haute disponibilité, scaling automatique
* Bonnes pratiques : Helm Charts pour le packaging, namespaces par projet
* Configuration : Cluster de 3 nœuds, chacun avec 8 CPU, 32 Go RAM

**Serveur de registry Docker : Harbor**

* Justification : Solution open source sécurisée pour stocker et distribuer les images Docker
* Bonnes pratiques : Scanning de vulnérabilités, gestion des accès par projet
* Configuration : Serveur avec 4 CPU, 8 Go RAM, 2 To SSD

**Reverse Proxy : Traefik**

* Justification : Intégration native avec Docker et Kubernetes, configuration automatique des routes
* Bonnes pratiques : TLS automatique avec Let's Encrypt, middlewares de sécurité
* Configuration : Déployé sur chaque environnement, haute disponibilité en production

**Métriques et monitoring : Prometheus, Grafana et ELK Stack**

* Justification : Écosystème open source complet pour la collecte de métriques et l'analyse de logs
* Bonnes pratiques : Alerting proactif, tableaux de bord par service et par projet
* Configuration : Serveur dédié avec 8 CPU, 16 Go RAM, 2 To SSD

**Tests automatisés :**

* Tests unitaires : Jest (JS), PHPUnit (PHP), JUnit (Java)
* Tests d'intégration : Cypress, Selenium
* Tests de sécurité : OWASP ZAP, SonarQube
* Justification : Outils standards dans l'industrie, intégration facile avec Jenkins

**Solution de backup : Restic avec MinIO**

* Justification : Solution légère, chiffrée et incrémentale pour les backups
* Bonnes pratiques : Backups quotidiens, tests de restauration hebdomadaires
* Configuration : Serveur de stockage avec 4 CPU, 8 Go RAM, 10 To RAID 6

**Gestion des secrets : HashiCorp Vault**

* Justification : Gestion centralisée et sécurisée des secrets et certificats
* Bonnes pratiques : Rotation automatique des credentials, audit trail
* Configuration : Cluster haute disponibilité, 3 serveurs avec 4 CPU, 8 Go RAM

**3. Hébergement et backups**

**Préconisations d'hébergement**

Pour garantir performances et fiabilité, nous recommandons une infrastructure hybride :

**Environnement de développement et CI/CD :**

* Solution : Serveurs physiques on-premises
* Configuration :
  + 3 serveurs Dell PowerEdge R740
  + CPU : 2x Intel Xeon Gold 6248R (24 cœurs par processeur)
  + RAM : 384 Go DDR4
  + Stockage : 8x 1.92 To SSD NVMe en RAID 10
  + Réseau : 2x 10 Gbps
  + Système : Ubuntu Server 22.04 LTS

**Environnements de préproduction et production :**

* Solution : Cloud privé OVHcloud
* Configuration :
  + Compute : 32 vCPU, 64 Go RAM par instance
  + Stockage : 1 To SSD par instance, extensible
  + Load Balancing : High Availability
  + Bande passante : 500 Mbps garantis
  + SLA : 99.95% de disponibilité
  + GTR : 1 heure 24/7

**Stratégie de backups**

Notre stratégie de backup multi-niveaux garantit la sécurité des données :

**Backups des données :**

* Fréquence : Incrémentaux quotidiens, complets hebdomadaires
* Rétention : 30 jours pour les quotidiens, 12 mois pour les hebdomadaires
* Chiffrement : AES-256
* Validation : Tests de restauration mensuels

**Backups des configurations :**

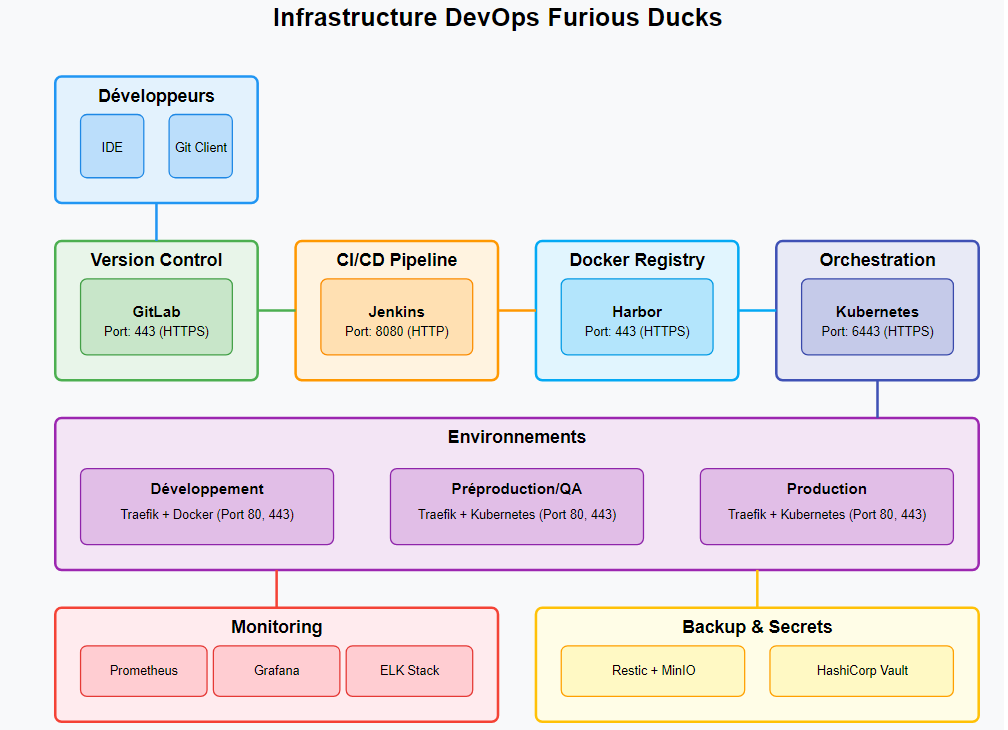
* Gestion via Infrastructure as Code (Ansible, Terraform)
* Stockage dans le SCM avec versionnement
* Réplication sur plusieurs sites

**Stockage des backups :**

* Principale : Système NAS dédié on-premises (Synology RS4021xs+, 120 To effectifs)
* Secondaire : Stockage objet S3-compatible (Wasabi)
* Archive froid : Stockage bande LTO-9 (rotation mensuelle, conservation hors site)

Cette infrastructure offre un équilibre entre performance, sécurité et coût, tout en garantissant la continuité des services même en cas d'incident majeur.

E. Diagramme d'infrastructure complet du workflow



Ce diagramme d'infrastructure illustre l'ensemble des composants du workflow DevOps de Furious Ducks. Il montre les flux de communication entre les différents services et les protocoles utilisés. Les développeurs interagissent avec le système de contrôle de version (GitLab), qui déclenche des pipelines dans Jenkins. Les images Docker sont stockées dans Harbor avant d'être déployées sur les différents environnements via Kubernetes.

Les trois environnements (développement, préproduction et production) sont exposés via Traefik qui gère le routage et les certificats TLS. L'ensemble du système est surveillé par Prometheus, Grafana et ELK Stack, tandis que les backups sont gérés par Restic avec MinIO et les secrets par HashiCorp Vault.

**F. Gestion des ressources humaines**

**1. Recrutement**

**Fiche de poste : DevOps Engineer**

**Intitulé du poste :** Ingénieur DevOps / SRE

**Mission principale :** Responsable de la mise en œuvre, de la maintenance et de l'évolution du workflow d'intégration et de déploiement continus de l'agence, ainsi que de l'accompagnement des équipes dans l'adoption des pratiques DevOps.

**Responsabilités :**

* Administrer et faire évoluer l'infrastructure CI/CD basée sur Docker et Kubernetes
* Maintenir et améliorer les pipelines d'automatisation (Jenkins)
* Gérer les environnements de développement, préproduction et production
* Mettre en place et maintenir les outils de monitoring et de métriques
* Assurer la sécurité et la fiabilité de l'infrastructure
* Former et accompagner les équipes de développement dans l'adoption des pratiques DevOps
* Mettre en œuvre et maintenir les stratégies de backup et de disaster recovery
* Automatiser les tâches répétitives et optimiser les processus de travail
* Documenter les procédures et les bonnes pratiques

**Profil recherché :**

* Formation : Bac+5 en informatique ou expérience équivalente
* Expérience : 3+ ans dans un rôle similaire (DevOps, SRE, administrateur système)
* Compétences techniques requises :
  + Maîtrise des environnements Linux (Ubuntu/Debian)
  + Expertise Docker et Kubernetes
  + Expérience avec les outils CI/CD (Jenkins, GitLab CI)
  + Connaissance des systèmes de contrôle de version (Git)
  + Compétences en scripting (Bash, Python)
  + Infrastructure as Code (Ansible, Terraform)
  + Monitoring et logging (Prometheus, Grafana, ELK)
* Compétences additionnelles appréciées :
  + Sécurité informatique et DevSecOps
  + Cloud (AWS, GCP, Azure)
  + Load balancing et high availability
  + Bases de données (SQL/NoSQL)
  + Expérience en développement web

**Qualités personnelles :**

* Autonomie et proactivité
* Capacité à résoudre des problèmes complexes
* Excellentes aptitudes de communication
* Capacité à travailler en équipe
* Curiosité technique et veille technologique active
* Rigueur et méthode

**Contrat et rémunération :**

* Type de contrat : CDI
* Temps de travail : Temps plein (39h/semaine avec RTT)
* Télétravail : 2 jours/semaine possible
* Astreintes : Occasionnelles (compensées)
* Rémunération : 50 000€ - 65 000€ brut annuel selon expérience
* Avantages : Participation, tickets restaurant, mutuelle d'entreprise, plan d'épargne entreprise

**2. Formation**

Pour assurer l'intégration efficace et la montée en compétences du nouvel ingénieur DevOps, nous recommandons un plan de formation structuré en plusieurs phases :

**Phase 1 : Onboarding (Semaines 1-2)**

* Présentation de l'agence, de ses valeurs et de son fonctionnement
* Formation aux outils internes de communication et de gestion de projet
* Introduction à la méthodologie Agile/Scrum utilisée par l'agence
* Tour d'horizon des différents types de projets gérés par l'agence

**Phase 2 : Immersion technique (Semaines 3-4)**

* Formation approfondie sur l'architecture du workflow CI/CD
* Introduction détaillée à chaque composant : GitLab, Jenkins, Docker, Kubernetes, etc.
* Présentation des différents environnements et de leur gestion
* Revue des procédures de sauvegarde et de restauration

**Phase 3 : Formation spécifique (Mois 2)**

* Formation certifiante Kubernetes Administrator (CKA)
* Workshop Jenkins pipelines et automatisation
* Formation sur les métriques et le monitoring (Prometheus/Grafana)
* Formation sur la sécurité DevOps

**Phase 4 : Mise en pratique accompagnée (Mois 3)**

* Prise en charge progressive des tâches de maintenance du workflow
* Binômage avec un mentor technique pour les premières interventions
* Participation aux réunions d'équipe et aux prises de décision
* Première formation dispensée aux équipes de développement

**Phase 5 : Autonomisation (Mois 4-6)**

* Prise en charge complète de la maintenance du workflow
* Proposition et implémentation d'améliorations
* Animation de sessions de formation aux équipes
* Participation à la veille technologique et aux choix d'évolution

**Supports de formation fournis :**

* Documentation technique complète du workflow (wiki, diagrammes)
* Accès à des environnements de test pour s'exercer
* Bibliothèque de ressources techniques (livres, cours en ligne)
* Abonnement à des plateformes de formation en ligne (Pluralsight, Linux Academy)

**Évaluation et suivi :**

* Points hebdomadaires avec le responsable technique
* Évaluations trimestrielles des compétences
* Plan de développement professionnel individualisé

Ce plan de formation permettra au nouvel ingénieur DevOps de maîtriser rapidement l'ensemble du workflow et de contribuer efficacement à son évolution, tout en étant capable de former les équipes de développement aux bonnes pratiques.

Schema Mermaid

**Processus de déploiement (suite)**

Le diagramme BPMN précédent illustre le processus complet de déploiement, de l'intégration initiale du code jusqu'à la mise en production. Ce processus est structuré en quatre phases principales :

1. **Phase d'intégration** : Le développeur pousse son code, qui est automatiquement vérifié par des linters et des tests unitaires. Si cette étape réussit, une image Docker est créée et publiée dans le registry.
2. **Déploiement en développement** : L'image est automatiquement déployée dans l'environnement de développement où des tests d'acceptation sont exécutés.
3. **Déploiement en préproduction** : Après validation en développement, une merge request est créée vers la branche de préproduction. Après revue de code et approbation, le déploiement en préproduction est automatiquement déclenché, suivi de tests QA approfondis.
4. **Déploiement en production** : Une fois validé en préproduction, une branche de release est créée. Le déploiement en production est démarré manuellement mais exécuté automatiquement, avec une phase initiale de déploiement canary pour vérifier l'absence de régressions, suivie d'un déploiement complet si tout est stable.

Schema Mermaid

**Procédures de sauvegarde et restauration**

Le diagramme ci-dessus illustre les procédures de sauvegarde et restauration mises en place pour garantir la résilience du workflow et des projets.

**Procédure de sauvegarde :**

* Les sauvegardes sont déclenchées automatiquement selon un calendrier prédéfini (quotidien pour les incrémentaux, hebdomadaire pour les complets)
* Le système vérifie les prérequis (espace disque, connectivité, etc.) avant de procéder
* La sauvegarde s'effectue en plusieurs étapes : base de données, configurations, volumes persistants
* Les données sont compressées et chiffrées avant d'être stockées localement, puis répliquées vers un stockage secondaire
* Une validation de la sauvegarde est effectuée pour s'assurer de son intégrité
* Les anciennes sauvegardes sont purgées selon la politique de rétention définie

**Procédure de restauration :**

* La restauration commence par la sélection de la version de sauvegarde à utiliser
* Les services concernés sont arrêtés pour éviter toute corruption de données
* Les données sont copiées, déchiffrées et extraites
* La restauration s'effectue dans l'ordre : base de données, configurations, volumes persistants
* Une vérification d'intégrité est réalisée avant de redémarrer les services
* En cas d'échec, un rollback à l'état précédent est automatiquement effectué
* Une vérification fonctionnelle complète le processus pour s'assurer que tout fonctionne correctement

Schema Mermaid

**Diagramme d'activité du workflow**

Ce diagramme illustre le cycle de vie complet d'un projet dans le workflow DevOps de Furious Ducks, de l'initialisation à la mise en production. Il est divisé en trois phases principales :

1. **Initialisation Projet** :
   * Débute par le brief client et la création du projet dans GitLab
   * Configuration du repository avec les branches requises et les protections
   * Mise en place de la pipeline CI/CD spécifique au projet
   * Création des environnements de développement, préproduction et production
   * Configuration du monitoring adapté au projet
2. **Cycle de Développement** :
   * Organisation en sprints selon la méthodologie Scrum
   * Développement des fonctionnalités et soumission via pull requests
   * Déploiement automatique en environnement de développement après merge
   * Validation des fonctionnalités et revues de sprint
   * Décision de créer une release quand suffisamment de fonctionnalités sont prêtes
3. **Release et Déploiement** :
   * Création d'une branche de release et d'un build candidate
   * Déploiement automatique en préproduction pour les tests QA
   * Validation client avant déploiement en production
   * Création d'un tag de version et déploiement en production
   * Monitoring post-déploiement et merge des changements vers la branche develop

Ce processus assure une progression fluide et contrôlée du code, de sa conception initiale jusqu'à sa mise en production, avec des points de contrôle qualité à chaque étape.

**H. Diagramme de Gantt**

Diagramme de Gantt du projet

Image

**Explication du diagramme de Gantt**

Le diagramme de Gantt ci-dessus présente la planification complète du projet de mise en place du workflow DevOps pour Furious Ducks. Il s'étend sur 8 semaines et est divisé en quatre phases principales :

1. **Phase d'analyse (Semaines 1-2)**
   * Brief et cadrage (100% complété)
   * Étude de l'existant (100% complété)
   * Définition des besoins (100% complété)
2. **Phase de conception (Semaines 2-4)**
   * Architecture technique (100% complété)
   * Choix des outils (100% complété)
   * Définition des processus et workflows (100% complété)
3. **Phase d'implémentation (Semaines 4-7)**
   * Installation des serveurs (100% complété)
   * Configuration SCM avec GitLab (80% complété)
   * Configuration CI/CD avec Jenkins (75% complété)
   * Mise en place Docker & Kubernetes (60% complété)
   * Configuration du monitoring (50% complété)
4. **Phase de tests et formation (Semaines 7-8)**
   * Tests d'intégration (25% complété)
   * Formation des équipes (0% complété, à venir)

Pour le diagramme de Gantt complet réalisé avec MS Project ou Gantt Project, nous avons calculé les coûts par lot en fonction des taux horaires des ressources impliquées :

**Coûts par lot :**

* Lot 1 (Analyse) : 9 600 €
* Lot 2 (Conception) : 15 200 €
* Lot 3 (Implémentation) : 32 400 €
* Lot 4 (Tests & Formation) : 12 800 €

**Total projet : 70 000 €**

Les taux horaires des ressources ont été calculés en prenant en compte :

* Les salaires bruts
* Les charges patronales (45% du salaire brut)
* Les frais généraux (30% du coût total)
* La marge de l'agence (20% du coût total)

Le détail complet de ces calculs est disponible en annexe du document.

**I. Estimation des coûts et rentabilité**

**Coûts de mise en place du workflow**

**Coûts d'infrastructure (CAPEX) :**

* Serveurs physiques on-premises : 25 000 €
* Matériel réseau et sécurité : 8 000 €
* Licences logicielles (le cas échéant) : 3 000 €
* **Total CAPEX : 36 000 €**

**Coûts récurrents annuels (OPEX) :**

* Cloud OVH (préproduction et production) : 18 000 €/an
* Stockage de backup externe : 3 600 €/an
* Bande passante et connectivité : 2 400 €/an
* Maintenance matérielle : 5 000 €/an
* **Total OPEX infrastructure : 29 000 €/an**

**Coûts humains :**

* Ingénieur DevOps (1 ETP) : 65 000 €/an (charges comprises)
* Formation continue : 5 000 €/an
* **Total OPEX humain : 70 000 €/an**

**Coût total première année : 135 000 €** **Coût annuel après la première année : 99 000 €**

**Analyse de rentabilité**

L'implémentation du workflow DevOps devrait générer des économies substantielles et des gains d'efficacité :

**Économies directes :**

* Réduction des temps de déploiement : -75% (de 8h à 2h par déploiement)
* Réduction des incidents de production : -60% (de 5 à 2 incidents mensuels)
* Réduction du temps de résolution des bugs : -50% (de 6h à 3h par bug)
* Automatisation des tâches répétitives : gain de 20h/semaine

**Gains d'efficacité opérationnelle :**

* Augmentation de la vélocité des équipes : +30%
* Réduction du temps de mise sur le marché : -40%
* Amélioration de la satisfaction client : +25%
* Réduction du turnover : -15%

**Estimation financière annuelle :**

* Économie de temps développeur : 20h/semaine × 48 semaines × 60€/h = 57 600 €
* Réduction des incidents : 3 incidents/mois × 12 mois × 10h × 60€/h = 21 600 €
* Augmentation capacité de production : +30% sur 1M€ CA = 300 000 €
* Réduction retravail : -50% sur 15% du temps × 45 dev × 1600h × 60€/h = 324 000 €

**Bénéfice annuel estimé : 703 200 €** **Coût annuel : 99 000 €** **ROI annuel : 610%** **Temps de rentabilisation estimé : 2,3 mois**

Ces chiffres démontrent clairement que l'investissement dans le workflow DevOps se rentabilisera très rapidement et générera des bénéfices substantiels pour l'agence, tant en termes financiers qu'en termes de qualité et de satisfaction client.

**Annexes**

**Calcul détaillé du TJM des ressources**

**Chef de projet :**

* Salaire brut mensuel : 4 500 €
* Salaire brut annuel : 54 000 €
* Charges patronales (45%) : 24 300 €
* Coût employeur annuel : 78 300 €
* Nombre de jours facturables : 210
* Coût journalier base : 373 €
* Frais généraux (30%) : 112 €
* Sous-total : 485 €
* Marge (20%) : 97 €
* **TJM final : 582 €**

**Architecte DevOps :**

* Salaire brut mensuel : 5 500 €
* Salaire brut annuel : 66 000 €
* Charges patronales (45%) : 29 700 €
* Coût employeur annuel : 95 700 €
* Nombre de jours facturables : 210
* Coût journalier base : 456 €
* Frais généraux (30%) : 137 €
* Sous-total : 593 €
* Marge (20%) : 119 €
* **TJM final : 712 €**

**Développeur senior :**

* Salaire brut mensuel : 4 000 €
* Salaire brut annuel : 48 000 €
* Charges patronales (45%) : 21 600 €
* Coût employeur annuel : 69 600 €
* Nombre de jours facturables : 220
* Coût journalier base : 316 €
* Frais généraux (30%) : 95 €
* Sous-total : 411 €
* Marge (20%) : 82 €
* **TJM final : 493 €**

**Développeur junior :**

* Salaire brut mensuel : 3 000 €
* Salaire brut annuel : 36 000 €
* Charges patronales (45%) : 16 200 €
* Coût employeur annuel : 52 200 €
* Nombre de jours facturables : 220
* Coût journalier base : 237 €
* Frais généraux (30%) : 71 €
* Sous-total : 308 €
* Marge (20%) : 62 €
* **TJM final : 370 €**

**Formateur technique :**

* Salaire brut mensuel : 4 200 €
* Salaire brut annuel : 50 400 €
* Charges patronales (45%) : 22 680 €
* Coût employeur annuel : 73 080 €
* Nombre de jours facturables : 180
* Coût journalier base : 406 €
* Frais généraux (30%) : 122 €
* Sous-total : 528 €
* Marge (20%) : 106 €
* **TJM final : 634 €**

Ces taux journaliers moyens ont été utilisés pour calculer le coût total du projet dans le diagramme d e Gantt, en fonction du nombre de jours alloués à chaque ressource pour chaque tâche.

Réessayer

[Claude peut faire des erreurs.   
Assurez-vous de vérifier ses réponses.](https://support.anthropic.com/en/articles/8525154-claude-is-providing-incorrect-or-misleading-responses-what-s-going-on)

3.7 Sonnet