SKALE 5

-The DevOps Company-

Migrate to

Sommaire

Sommaire

Migrate to GCP

Lorsqu'on observe les migrations vers Google Cloud jusqu'en 2022/23, et probablement pour les années suivantes, la grande majorité d'entre elles s'effectuent depuis un Data Center

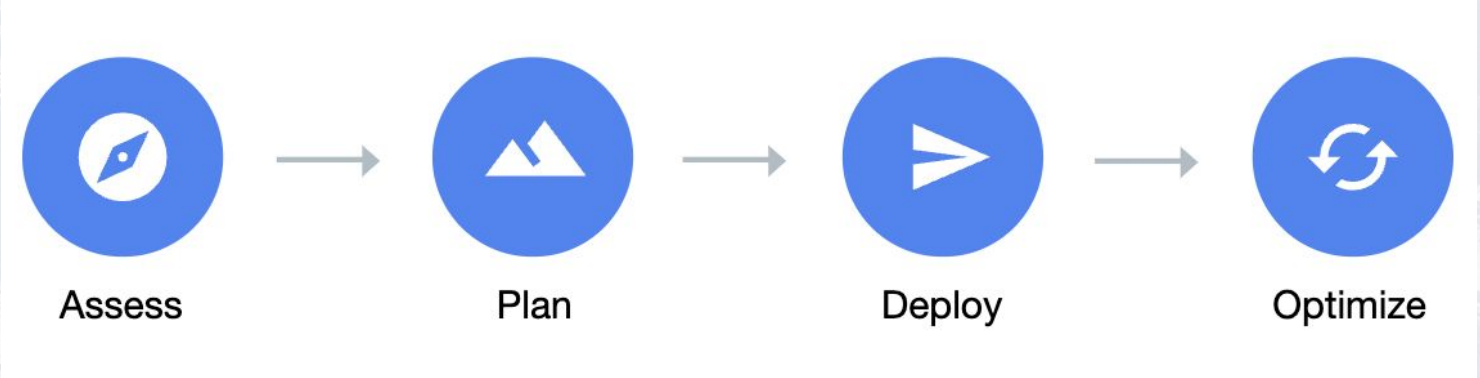
Le transfert d'applications d'un environnement on-premise à un Cloud peut s'avérer difficile, même pour des équipes expérimentées et maîtrisant le patrimoine applicatif. La planification et la préparation de la migration est une étape cruciale dont la réussite repose sur deux piliers :

1. La méthode - suivre un cadre structuré et cadencé.
2. L'expérience - les réflexes et la capacité d'anticipation se construisent sur l'expérience passée.

Chez Skale-5 nous avons les deux piliers ! Parce que nous sommes nés et n'existons que pour cela : migrer et infogérer sur GCP & AWS.

La croissance d'usage des services Cloud Public provient d'applications conçues Cloud natives ou de migrations d'applications existantes. Ce deuxième vecteur d'usage est l'axe principal de croissance des Cloud Providers. On constatera que chacun a construit un framework pour guider avec méthode les modalités d'une migration bien orchestrée.

Google Cloud fragmente sa méthode de migration en quatre phases :



1

Assess - Évaluation

Au cours de cette phase, il convient de cartographier l'existant par une étude approfondie la plus exhaustive possible.
Chez Skale-5, nous subdivisions cette étape en six lots.

Assess / Évaluation

1.1 - Inventaire de l'existant

Il s'agit de commencer la migration en réalisant un état des lieux de l'existant.

. Inventaire des applications

Plusieurs outils du marché ou fournis par Google Cloud permettent de collecter un grand nombre d'informations. Cependant, un travail manuel (parfois long et dépendant du niveau de connaissance interne chez le Client), permettra un regroupement lisible et fiable en vue de concevoir les modalités de migration.

Généralement nous devons identifier les points suivants pour chaque application :

- Composant structurel : VM vmware / VM legacy cause JDK xx.xx / containers spring / containers xxx.
- Ressources système (CPU / RAM).
- Stack technique de dev.
- Emplacement du code source (Github, repo internet, Gitlab).
- Dépendances (en lien avec la liste établie).
- Correspondance avec Services Managés envisageables.
- Modalité de déploiement (manuel / automatisé).

Nom application	Composant structurel	Resources système	Stack technique	Emplacement du code	Dépendances	Services Managés envisageables	Déploiement

. Inventaire des dépendances

Les dépendances sont des composants ou services accessibles par plusieurs applications. Lors de la migration leur incidence est multiple :

- Le composant est-il dissociable des applications ?
  Ex. Montage NFS dont les fichiers sont partagés par l'application A et B.
- Quelle est la complexité de sa duplication sur le Cloud ?
- Quel impact sur les modalités de migration ?
  e: big-bang le jour J ou migration progressive.

Exemple :

Nom application	*NFS*	*Varnish*	*Elastic search*	*BDD*	*Cache*	*FTP*	*Supervisord / Rabbit*	*Filtrage*	*CRON*
bourse	-	oui	-	-	-	-	-	-	-
gateway-partners	-	oui	-	-	-	-	-	IP	-
sante	oui	oui	oui	oui (MySQL + Mongo)	Memcache	-	oui	-	oui
api-tvmag	-	oui (purge)	oui	oui (??)	-	oui	-	-	oui
api-rss	-	oui (purge)	-	oui (MySQL)	-	-	-	IP	-

Assess / Évaluation

(...)

Assess / Évaluation

. Inventaire du matériel

On peut recourir ici à de nombreux outils pour rassembler ces informations. L'entreprise dispose peut-être également d'un inventaire (nommé CMDB dans les process ITIL). L'objectif n'est pas de reproduire l'existant dans le Cloud, mais d'y reporter les informations qui permettent d'obtenir une visibilité suffisemment exhaustive pour comprendre les fonctionnalités portées par certains équipements, notamment les équipements de sécurité et réseau.

Par ailleurs, cet inventaire sera utile pour construire l'évaluation budgétaire dans le Cloud (TCO).

. Inventaire des contraintes Réseaux & Sécurité

Il est important de connaître les contraintes Réseaux et Sécurité qui s'appliquent aux applications concernées par la migration. Cela permettra d'établir un mapping entre ces contraintes et les fonctionnalités natives du Cloud Provider. Certaines contraintes peuvent influer sur le choix de l'architecture Cloud. Attention cependant à ne pas tenter de reproduire la même architecture réseau dans le Cloud, car il faut s'appuyer sur les paradigmes du Cloud (règles de filtrage par tags, Shared VPC vs VPC Peering, etc.).

. Inventaire des sachants

Le savoir est généralement réparti entre plusieurs services et, bien souvent, dans plusieurs cerveaux. Identifier les "sachants" et la documentation existante permet de gagner du temps lorsque des questions se posent et/ou de mettre à contribution les bonnes personnes dans le cadre d'un atelier de préparation.

Généralement nous établissons le tableau suivant :

Nom application	Développé par	Période de dév.	Mintenu par (TMA)	Personne(s) référente(s)	Type de documentation	Lieu /lien documentation

Assess / Évaluation

(...)

Assess / Évaluation

NB : il est possible d'ajouter une colonne portant sur la criticité liée aux applications ou sur l'urgence afin de mesurer le risque ou d'indiquer un ordre de priorité en vue de la phase PLAN.

En corrélation avec les patterns (regroupement d'applications de même typologie technique), il est nécessaire d'établir un regroupement des dépendances, ce qui débouche généralement sur l'élaboration d'un tableau de synthèse du type suivant :

Dépendances ou regroupement de dépendances	Stack technique	Contraintes spécifiques à la migration	Autres contraintes (réglementaire, organisationnelles, etc.)	Nom des applications

A l'issue du processus, ce regroupement doit conduire à identifier les éventuels singletons afin de prendre une décision du type :

- Écarter une application de la migration.
- Forcer son refactoring ou, au contraire ,un Lift & Shift spécifique.
- Repenser les vecteurs de dépendances (regrouper ou dissocier des éléments de dépendance).

Autrement dit, il est impératif de trouver des facteurs de regroupement pour se limiter à 3 ou 4 typologies de regroupement.

Exemple Client :

5 patterns d'Applications :

① **Stateless (standardisées)**

② **API**

③ **AGORA**

④ **Node JS**

⑤ **Full Custom**

+

3 typologies de Migration :

1- GKE ready
Application en containers

2- Move2Cloud
Applications nécessitant un effort raisonnable pour migrer sur GKE + Services Managés (SM)

3- Lift&Shift
Migration VM telle quelle + SM

=

GKE ready : ① app bourse / ④ gateway-partners

Move2Cloud : ② api-tvmag / ② api-rss

Lift&Shift : ⑤ sante (drupal)

Assess / Évaluation

(...)

Assess / Évaluation

1.3 - Évaluation de l'engagement vers l'usage du Cloud

Au-delà des applications, il est généralement utile de cartographier l'usage du Cloud et la mobilisation autour du projet de migration.

Nous utilisons alors deux frameworks :

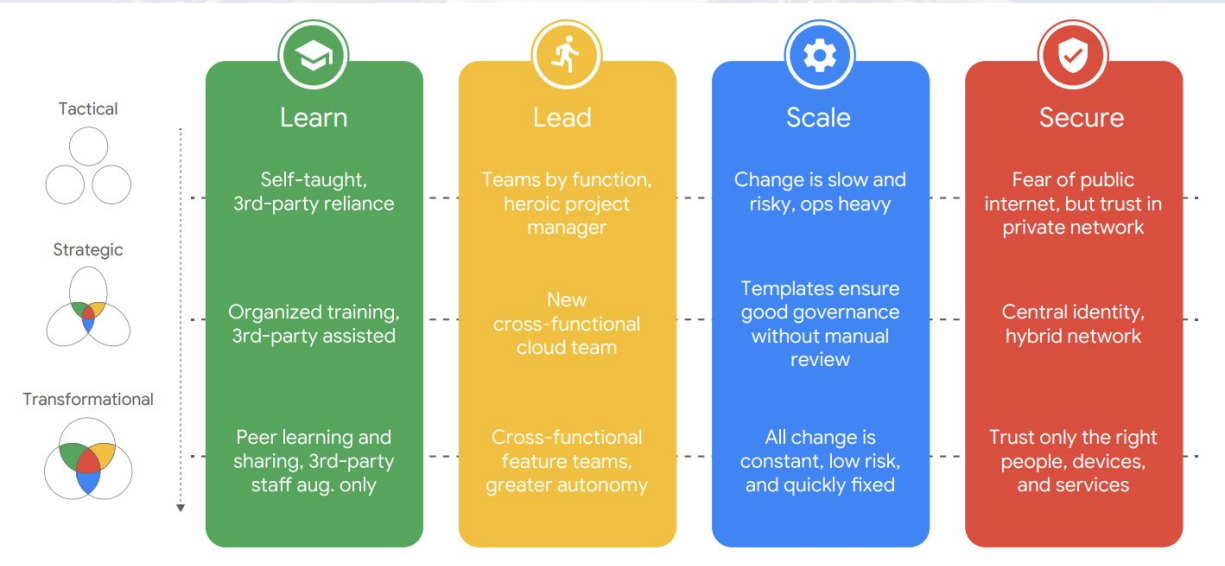
1. Le Google framework pour évaluer le niveau de maturité d'une organisation et son personnel pour s'engager vers une direction résolument axée sur l'usage du Cloud Public.
2. Le Skale-5 framework pour mesurer et partager l'engagement vers l'usage du Cloud de l'entreprise et de ses équipes.

. Framework d'adoption du cloud de Google

Google propose un framework permettant d'organiser cette démarche en décrivant les personnes chargées de la mise en oeuvre et les processus qui les régissent.

Le Google Cloud Adoption Framework construit une structure comportant des rubriques sur les personnes, les processus et la technologie que vous pouvez utiliser pour obtenir une évaluation solide de l'état d'avancement de votre migration vers le Cloud ainsi que des programmes exploitables pour atteindre votre objectif. Elle s'appuie sur l'évolution de Google dans le domaine du Cloud computing et sur ses nombreuses années d'expérience dans l'aide aux clients.

Vous pouvez utiliser ce cadre vous-même pour évaluer l'état de préparation de votre entreprise au Cloud computing et évaluer ce que vous devez faire pour combler les lacunes et développer de nouvelles compétences.



Assess / Évaluation

. Skale-5 Move2cloud framework

Cette approche très pragmatique est plus concise que le framework Google. Il s'agit d'évaluer collectivement :

- la maturité de l'organisation au regard a) d'un usage Cloud Public et des pratiques DevOps, b) de la cible post-migration souhaitée en termes d'autonomie et de déploiement,
- le niveau de risque de la migration lié à la maîtrise du patrimoine applicatif.

Nous prenons toujours le temps de comprendre ce qui motive la migration de nos Clients. Cette démarche permet de situer : a) les connaissances et capacités de notre Client concernant son patrimoine applicatif b) les objectifs post-migration : autonomie des équipes, délégation complète ou partielle, etc.

Bien que nous maîtrisions le déploiement décrit dans le processus "Google Cloud Adoption Framework", nous proposons régulièrement un modèle design by Skale-5 beaucoup plus simple et répondant aux migrations de quelques applications pour 100 à 300 Jours Hommes de migration.

Nous appliquons notre modèle selon 4 axes :

1. **Quel est votre objectif #1 de la migration ?** De l'accès à de l'IaaS pour fermer un DC ou l'adoption profonde des briques de services dont le Serverless ; Il s'agit pour nous de mesurer l'implication de notre Client dans l'usage du Cloud. Entre choix tactique de consommation Infra et une posture de transformation IT orientée Cloud, nous désignerons notre accompagnement de façon ciblée.
2. **Quel niveau de mobilisation ?** Il s'agit d'analyser si des freins existent et si des normes (ITIL ou ISO notamment) peuvent handicaper l'adoption de process DevOps.
3. **Quel niveau d'autonomie interne souhaitez-vous sur vos infrastructures GCP ?** De-là découlent un plan de formation et un cadre de structuration des infras : convention, landingzone documentée, etc.
4. **Quel niveau de maîtrise de votre patrimoine applicatif ?** De façon transparente et partagée avec notre Client, nous souhaitons mesurer la maîtrise technologique et opérationnelle de son patrimoine applicatif : équipe interne, sous-traitants en place ou partis, niveau de documentation, nombre d'incidents, etc.

Un ou deux entretiens suffisent largement à établir la représentation de l'existant et des souhaits du Client. Le compte rendu de ces échanges expose le mapping et les actions clés qu'il convient de mettre en œuvre.

Assess / Évaluation

Exemple :

	Facilité d'engagement vers l'usage du Cloud →				Commentaires / Actions "++" = niveau d'importance
objectif #1	IaaS	Start learn	Accelerate cloud usage	Cloud first	. Convention +++ . Formation avancée ++
Mobilisation	Freinés	Individuelle	Localisée	Engagée globalement	. Niveau d'exigence +++ . Modernisation & replatforming ++
Autonomie souhaitée	Full délégation	Faible invest. DevOps	Infogérance déléguée	Full autonomie	. Travail collectif au quotidien ++ . Construction Monitoring commun ++
Maîtrise patrimoine applicatif	Très faible	Prestations équilibrées interne/externe	Documentation ok et applications récentes	Maîtrise interne majoritairement	. Sécurisation des plans de bascule ++ . Vigilance délais cf. replatforming ou modernisation priorisé vs migration +++

Ici nous illustrons un projet de Migration dont la maîtrise des applications va de pair avec l'objectif d'une bonne autonomie construite sur un partage de code et d'outils exploitables par d'autres prestataires, qu'il s'agisse de l'infogérance ou de développement.

Notons que le barycentre (regroupement des points de couleurs) doit être cohérent. Dans le cas contraire, il s'agit soit d'une erreur d'appréciation, soit d'une incohérence dont il faut immédiatement lever le risque.

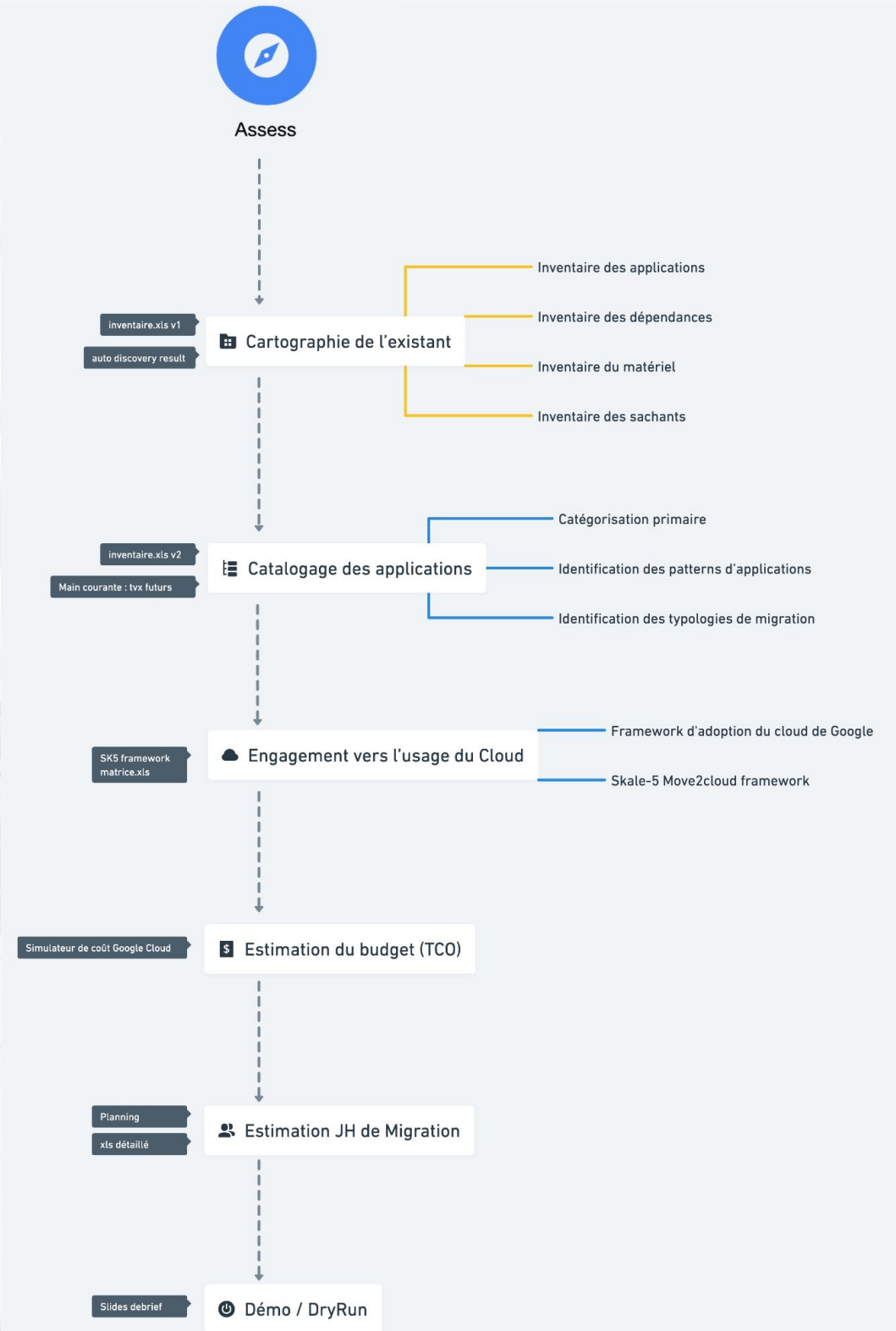
Exemple : Avoir une démarche résolument orientée Cloud (Cloud first) et une faible maîtrise de ses applications.

Assess / Évaluation

(...)

Assess / Évaluation

1.7 - Synoptique de la phase d'ASSESS



2

Plan - Structuration

Au cours de cette phase il s’agit de concevoir et de poser les fondations qui auront une influence majeure sur la suite de la migration et l’usage au quotidien de GCP.

Les principaux objectifs de cette phase sont :
- L'établissement d'une convention de nommage.
- La constitution d'une structuration des ressources et projets.
- La définition des droits et des groupes d’utilisateurs.

Plan / Structuration

2.1 - Établir les identités

Google Cloud utilise des identités pour la gestion de l'authentification et des accès. L'accès à une ressource Google Cloud, quelle qu'elle soit, par un membre de votre organisation, suppose qu'il dispose d'une identité que Google Cloud puisse comprendre.

Trois méthodes sont couramment utilisées pour configurer des identités dans Google Cloud :

- Gérer les identités directement dans Google Cloud.
- Utiliser vos comptes Google Workspace existants.
- Utiliser un fournisseur d'identité (IdP) existant comme Active Directory.

. La gestion des accès

Sans détailler plus avant la typologie des accès, retenons que le modèle de gestion des accès se compose de quatre concepts fondamentaux :

- *Compte principal* : il s'agit d'une entité sur laquelle nous appliquerons des rôles. Ainsi, lorsqu'on attribue un rôle à un compte principal, on lui accorde toutes les autorisations contenues dans ce rôle. Les comptes principaux peuvent être issus de différentes sources : utilisateur Google, un compte de service pour les produits Google Cloud, un groupe Google, voire un compte Google Workspace ou Cloud Identity.
- *Rôle* : correspond à une collection d'autorisations.
- *Autorisation* : détermine les opérations qui sont autorisées sur une ressource.
- *Stratégie IAM* : il s'agit d'appliquer une stratégie d'accès, autrement dit d'assurer le couplage Comptes principaux / rôle.

Une configuration appropriée et une gestion efficace des comptes principaux, des rôles et des autorisations constituent le pilier de votre stratégie de sécurité dans Google Cloud. En outre, cette démarche vous protège et rend auditable votre usage de GCP.

Notons que Google propose de nombreux rôles et que nous déconseillons de construire ses propres rôles car les phases de debug associées seront longues et inévitables …

Notons qu'une typologie d'accès spécifique doit être structurée et déployée. Il s'agit des comptes servant aux accès des applications. Ils sont généralement appelés comptes de services. Un compte de service est une identité que vos programmes et services peuvent utiliser pour s'authentifier et accéder aux ressources Google Cloud.

Plan / Structuration

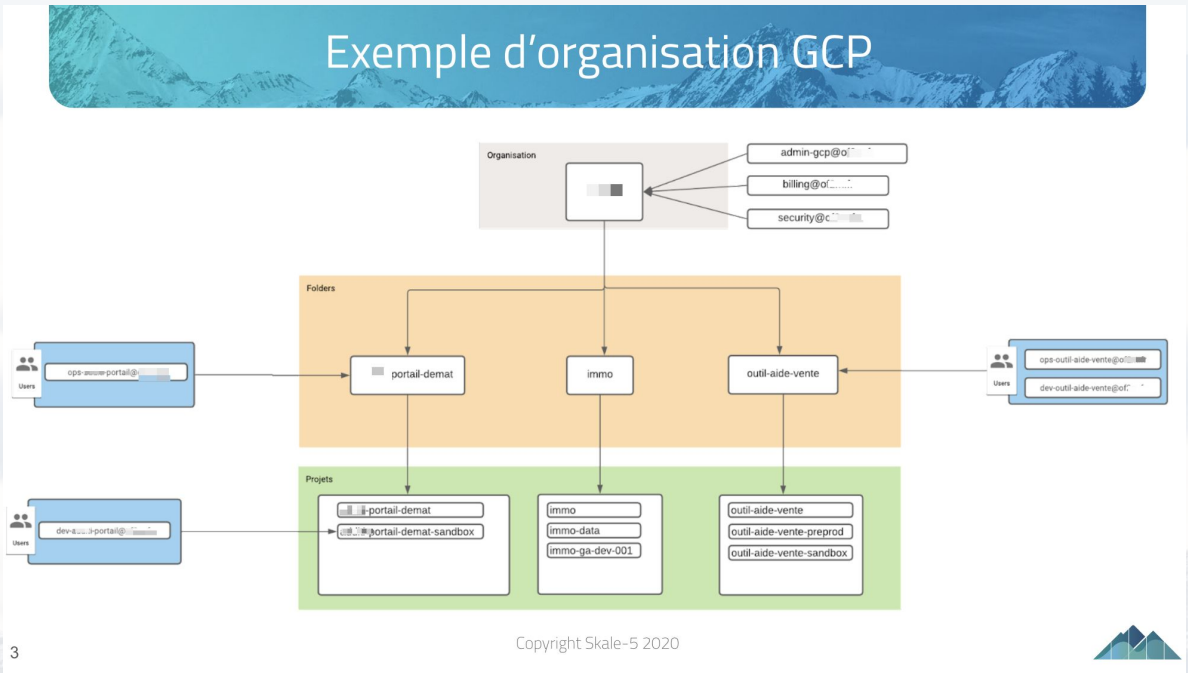
- (..)

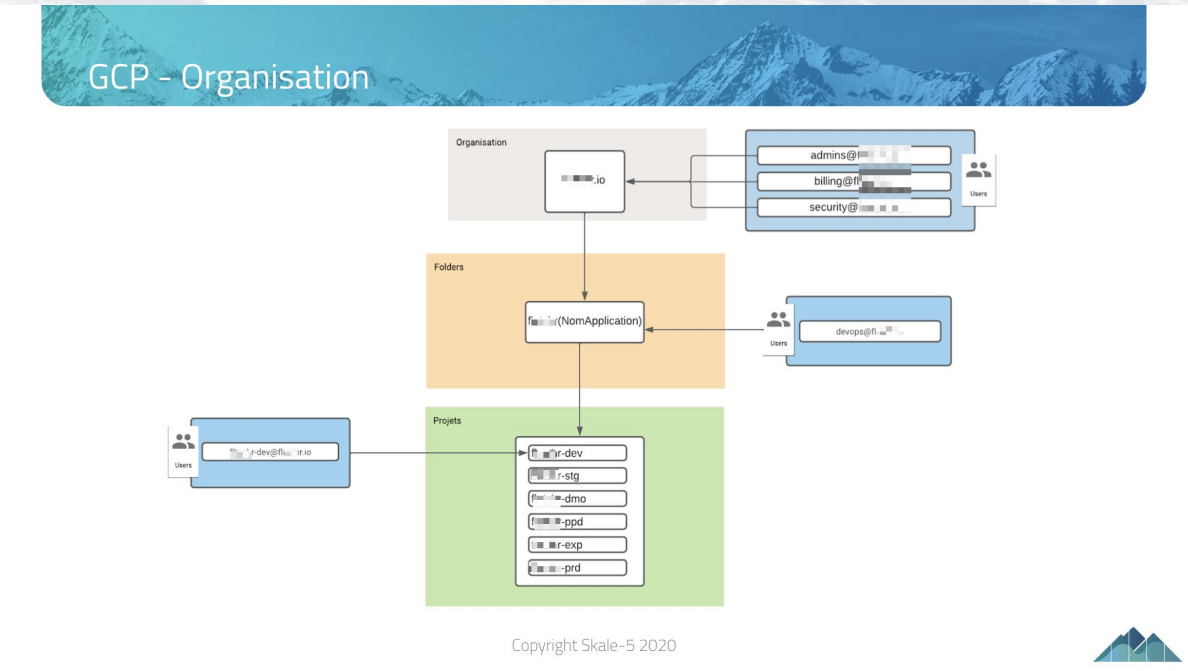
Plan / Structuration

- **Hiérarchie axée sur la fonction et/ou l'application (au niveau des Dossiers).**

Ici l'organisation contient un dossier par Application. Il s'agit d'une structuration souvent déployée par Skale-5 car elle permet de regrouper des projets qui sont gérés par la même équipe.

Exemples réels :





Plan / Structuration

- (..)

Plan / Structuration

2.4 - Gestion de projet

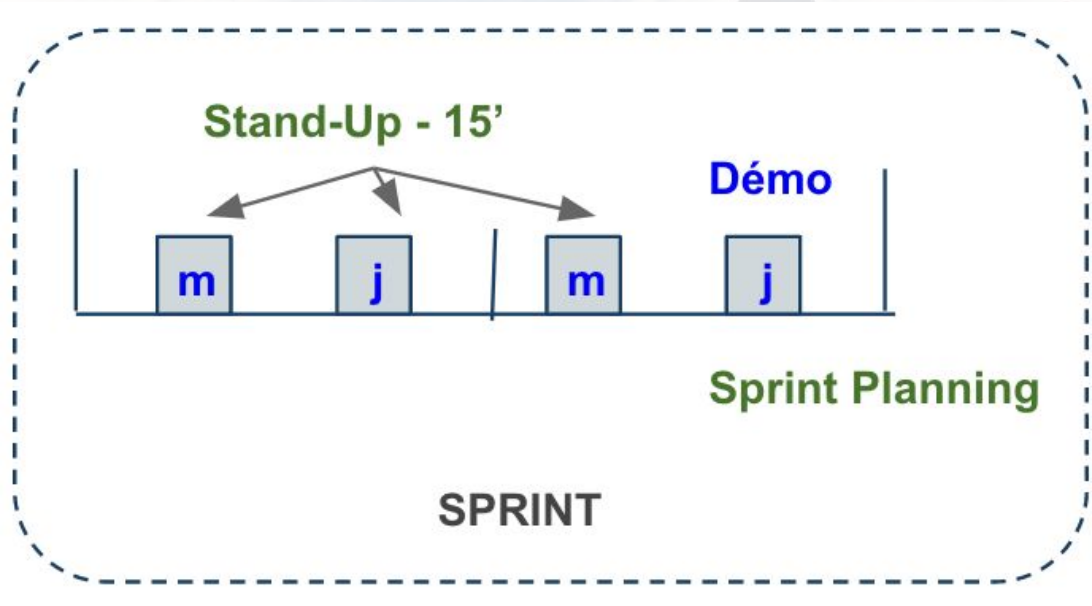
Nous utilisons la méthode AGILE/SCRUM pour structurer et piloter nos projets. Cependant nous nous sommes permis quelques adaptations pour garder de la souplesse et passer au pilotage du projet le temps strictement nécessaire.

Le cadre est le suivant :

- Un sprint dure 2 semaines.
- Une story (une tâche à effectuer) est évaluée en temps (vs complexité).
- Une story ne doit pas excéder 2 jours.
- Une et une seule personne par story …
- Un backlog construit collectivement.

Nous gardons les cérémonies suivantes :

- Sprint planning en début de sprint pour définir les stories du Sprint.
- Une démo en fin de sprint.
- 3 Stand-up par sprint pour supprimer les difficultés d'avancement.

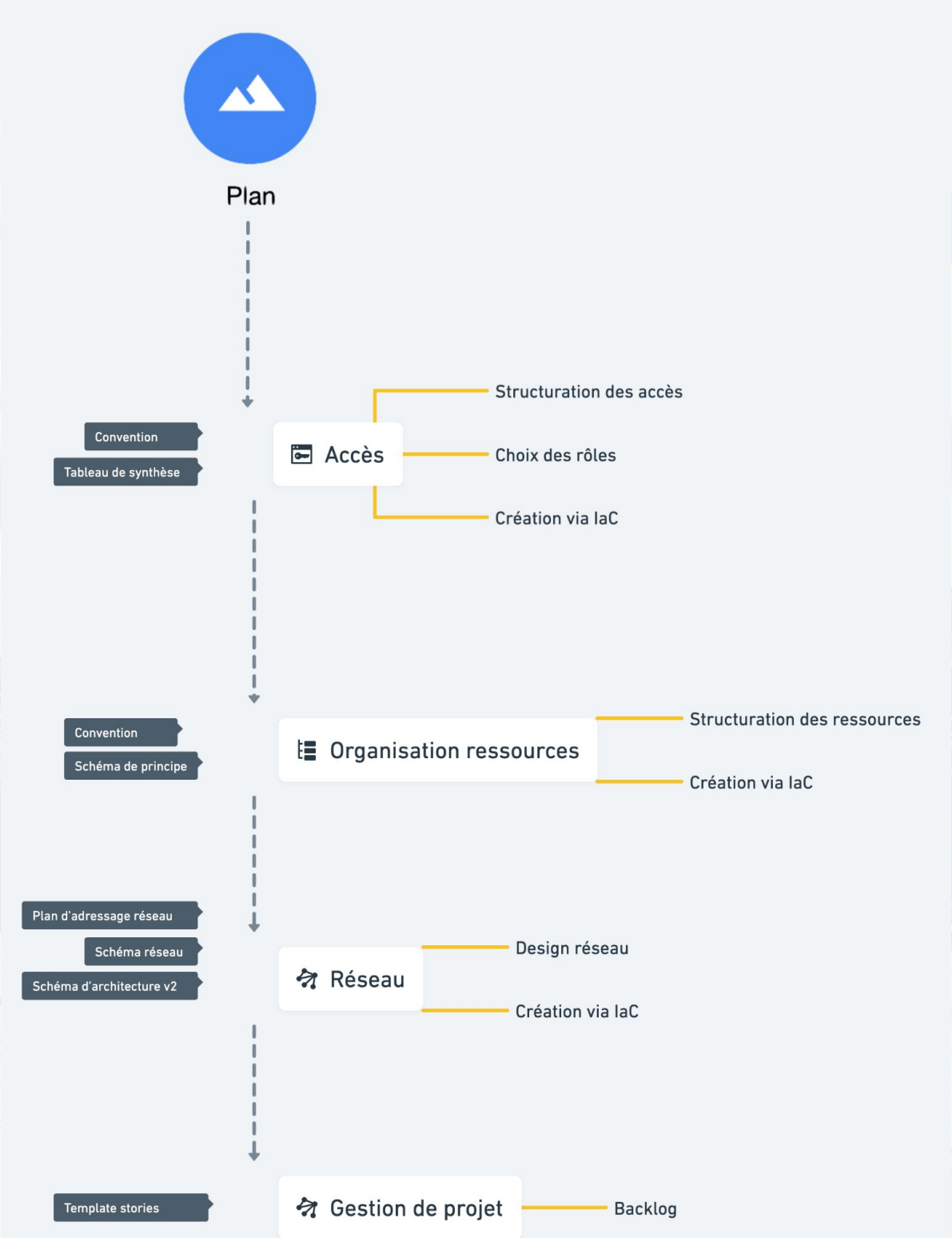


En résumé :

- **Sprint de 2 semaines**
- **100% des Stories dans un outils type Jira, Gitlab, notion.so ou trello**
- **Démo obligatoire en fin de Sprint**
- **3 Stand-Up / semaine**

Plan / Structuration

2.5 - Synoptique de la phase PLAN



3

Deploy - Déploiement

Maintenant que toutes les bases de la migration sont prêtes, il est temps de déployer les premières briques de l'infrastructure.

Cette première mouture ne sera vraisemblablement pas la version définitive, ce qui constitue une raison de plus pour déployer via du code (IaC). Les modifications seront plus faciles. Elles seront surtout à la portée de tous les participants au projet, qu'il s'agisse de Skalers ou des équipes de notre Client. Pilier d'une démarche DevOps, le partage du Cloud est le moyen privilégié pour que l'infrastructure GCP devienne la propriété de chacun sous la responsabilité de tous !

Bien que l'infrastructure soit obligatoirement codée (conviction forte et inaliénable de Skale-5), la phase de déploiement comporte également la mise à jour (et/ou l'installation) des applications. Sur ce point les travaux réalisés en phase d'ASSESS ont d'ores et déjà identifié là où devront être portés les efforts. Tout ne sera pas déployé de façon codée et automatique… La mise en place de CI/CD moderne ou la réalisation d'actions manuelles sont ainsi étroitement liées aux stacks techniques (cf. legacy technique ou de conception).

Deploy / Déploiement

3.1 - Déployer les charges de travail

3.1.1 - Infrastructure as Code

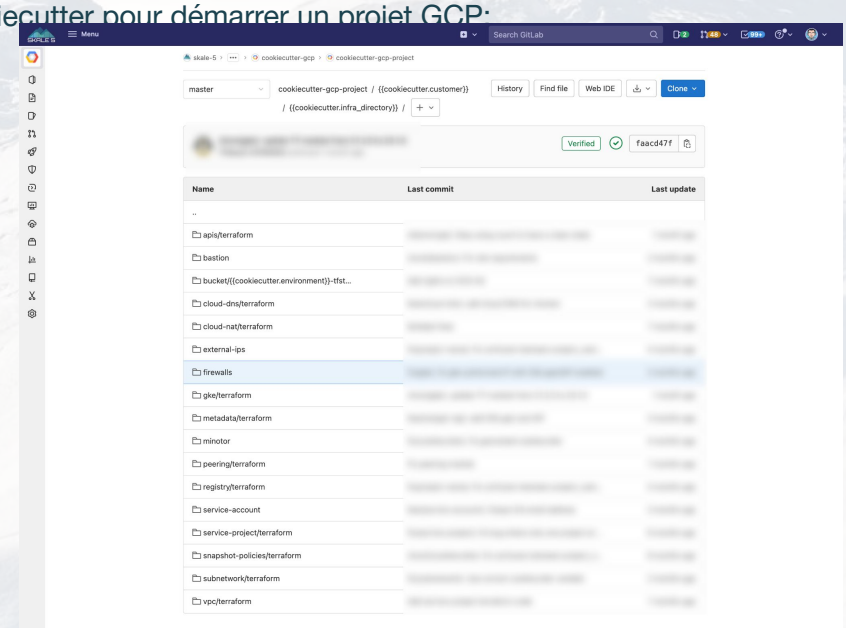
En construisant l'infrastructure via du code on s'assure d'une construction immutable, c'est-à-dire une construction dont la seule définition (au sens de la détermination des caractéristiques) n'existe que dans le code. Autrement dite une modification manuelle sera effacée lors du prochain déploiement…

Cette base de code sera également un pré-requis à l'automatisation des déploiements, par exemple pour la réplication d'environnement ou les mécanismes de scalabilité. Il existe de nombreux outils pour coder l'infrastructure. Chez Skale-5 et nos Clients nous utilisons généralement *Terraform* et *Deployment Manager.*

Notons qu'au-delà des nombreux avantages (impératifs!) de l'IaC, l'utilisation de containers et d'un orchestrateur comme GKE nécessitera la mise en place de processus de déploiement de type GitOps dont l'IaC est une pièce obligatoire.

Chez Skale-5, nous prônons la mutualisation du code et sa réutilisation à chaque fois que c'estpossible. C'est pour cela que nous utilisons des templates de code Terraform, générés via l'outil CookieCutter (moteur de template) : ces templates vont produire du code Terraform standardisé et déjà éprouvé sur d'autres migrations. Ce code Terraform référence lui-même des modules Terraform, soit maintenus par Skale-5 (Registry Terraform privée), soit maintenus par la communauté Terraform (Registry Terraform publique).

Exemple du cookiecutter pour démarrer un projet GCP:

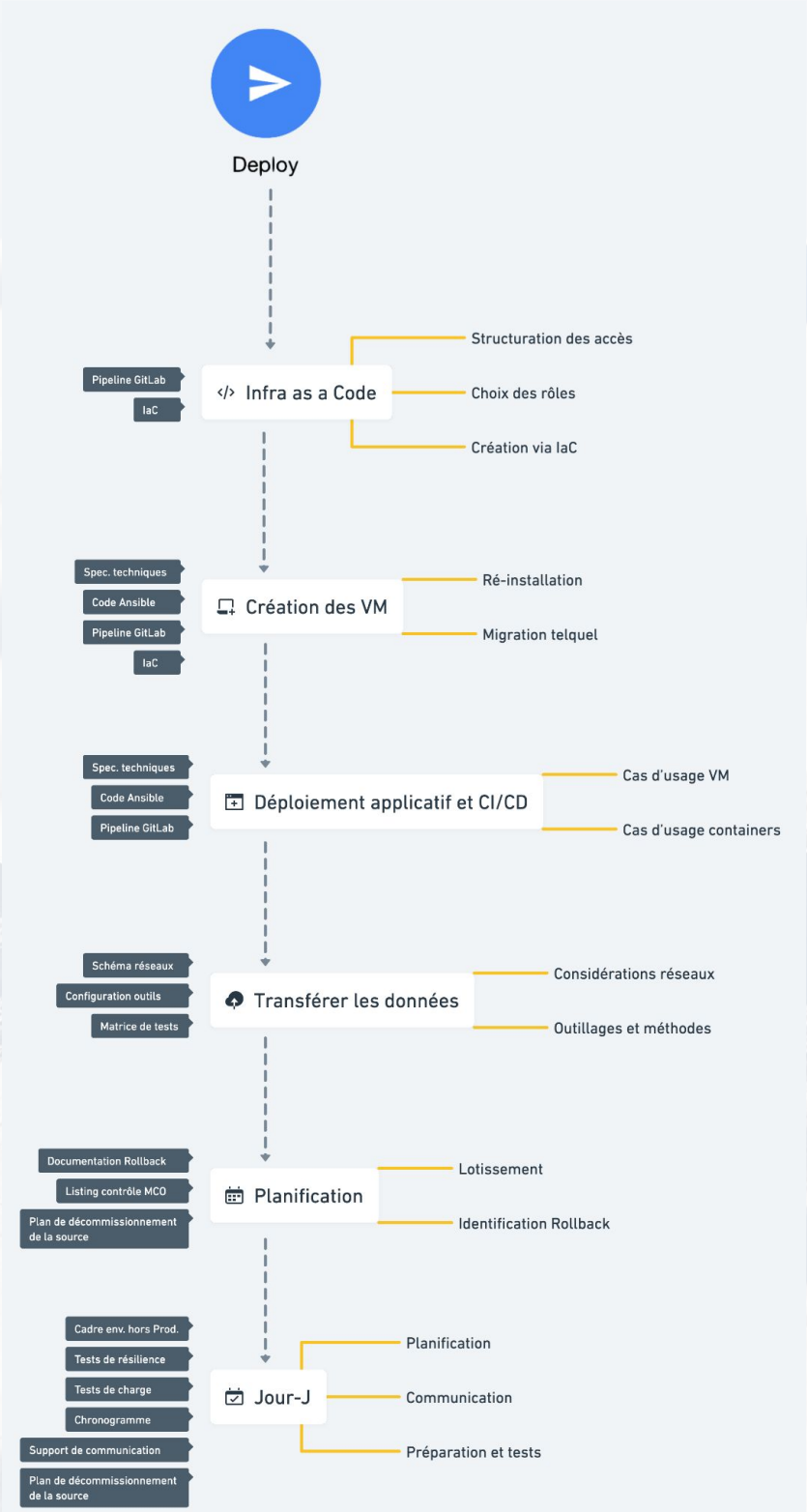


Deploy / Déploiement

(..)

Deploy / Déploiement

3.4 - Synoptique de la phase DEPLOY



4

Optimize - Optimisation

Il s'agit d'une phase de progrès consécutive à la migration. Au cours de celle-ci, ou depuis que nous sommes dans les nouveaux environnements GCP, nous avons identifié des axes d'amélioration. Il est temps de planifier leurs mises en œuvre et de réaliser leurs exécutions.

Notons que cette phase est un jour sans fin : et chacun des items exposés ici sera de nature à construire une nouvelle boucle d’améliorations.

Optimize / Optimisation

4.1 - Évaluation d'une Roadmap post-Migration

L'optimisation n'a de sens que si elle suit des objectifs clairs, objets d'un consensus, et si les actions entreprises dans ces boucles d'optimisation sont mesurées et pilotées.

4.1.1 - Donner une directive macro-cycle

Les roadmaps d'amélioration ne s'apparentent pas à une liste au Père Noël ! Elles s'inscrivent dans un cadre bordé par deux vecteurs contraignants :

1. Le budget que l'on pourra allouer, notamment le nombre de Jours Hommes.
2. Le "pain point" majeur à date.

Généralement nous conseillons un macro-cycle d'un an répondant à un budget et un ou deux points de vigilance / risques prioritaires.

4.1.2 - Identifier les axes prioritaires de progrès

Pour aider à structurer son approche de progrès, prioriser et se focaliser sur des objectifs clairs, Skale-5 a identifié 9 axes de progrès :

1. Optimiser les coûts d'infrastructure (FinOps).
2. Réduire la dépendance aux sachants (Documentation, convention, IaC).
3. Réduire les risques d'erreur et la lassitude des équipes (automatisation et CI/CD).
4. Augmenter la qualité des livraisons (automatisation des tests).
5. Augmenter et maintenir le niveau de sécurité (automatisation de tests, audit).
6. Améliorer la visibilité et la résolution (Monitoring, tracing, supervision).
7. Moderniser les applications vers du Cloud Native (containers, serverless).
8. Améliorer la scalabilité face au trafic ou l'augmentation de features applicatives.
9. Améliorer la résilience (inclus PCA).

Optimize / Optimisation

(...)

Optimize / Optimisation

4.2 - Synoptique de la phase Optimize

SKALE 5
-The DevOps Company-

Migrate to GCP

Vous voulez lire l'intégralité du Livre Blanc ?

Contactez-nous …

contact@skale-5.com