Formation **Pulumi**

Provisionnez et versionnez vos infrastructures AWS, Azure ou GCP en Python ou TypeScript, avec les bons réflexes CI/CD et sécurité.

Pulumi est une plateforme d'Infrastructure as Code qui permet de décrire des ressources cloud dans un vrai langage de programmation (Python, TypeScript, Go) plutôt qu'en DSL dédié. Cette formation de 2 jours vous apprend à structurer vos projets en stacks, provisionner des ressources sur AWS, Azure et GCP via l'API Pulumi, et gérer proprement la configuration, l'état et les secrets. Vous mettez en place un pipeline CI/CD qui valide chaque changement d'infrastructure via pull request, puis abordez les bonnes pratiques de sécurité, de gouvernance et de réutilisation (modules, packages tiers). À l'issue de ces 2 jours, vous êtes capable de porter un projet Pulumi de la première ressource déployée jusqu'à un workflow collaboratif industrialisé.

Durée

2 jours

Objectifs pédagogiques

- Expliquer les apports de l'Infrastructure as Code moderne et positionner Pulumi face aux autres outils IaC
- Structurer un projet Pulumi en stacks, configurations et ressources réutilisables
- Provisionner des ressources sur AWS, Azure et GCP via l'API Pulumi
- Gérer l'état des déploiements et sécuriser les secrets
- Intégrer Pulumi dans un pipeline CI/CD avec validation par pull request
- Appliquer les bonnes pratiques de sécurité, de gouvernance et de réutilisation (modules, packages)

Public

Ingénieurs DevOps, développeurs cloud, administrateurs système et architectes cloud amenés à automatiser le provisionnement d'infrastructures multi-providers.

Prérequis

• Maîtrise de base d'un langage parmi Python, TypeScript ou Go
• Connaissance des fondamentaux d'au moins un cloud (AWS, Azure ou GCP) : services courants et IAM
• Familiarité avec Git et le workflow pull request
• Aisance avec la ligne de commande

Programme de formation

Phase d'inclusion

Fondamentaux Pulumi et Infrastructure as Code

• Infrastructure as Code moderne : enjeux, apports par rapport aux scripts impératifs et aux approches manuelles
• Positionnement de Pulumi : comparaison avec Terraform, CloudFormation et autres outils IaC
• Choix du langage : Python, TypeScript ou Go — critères de décision selon l'équipe et le projet
• Installation et environnement de développement : CLI Pulumi, SDK, intégration IDE (Linux / macOS / Windows)
• Architecture d'un projet Pulumi : projets, stacks, ressources, providers, packages
• Configuration et compréhension des piles (stacks) pour isoler les environnements

Exemples d'activités pratiques :
• Installation et configuration de Pulumi et du cloud provider cible
• Écriture et déploiement d'une première ressource simple (instance EC2 ou App Service Azure)
• Inspection d'un stack et lecture de l'état produit

Provisionner des ressources cloud multi-providers

• API Pulumi pour AWS : services courants (compute, stockage, réseau, IAM)
• API Pulumi pour Azure et Google Cloud : équivalences et spécificités par provider
• Programmation de l'infrastructure : boucles, conditions, fonctions pour générer dynamiquement des ressources
• Composants réutilisables : création de modules pour encapsuler des patterns d'infrastructure
• Dépendances entre ressources et orchestration de l'ordre de création
• Gestion multi-environnements : isoler dev, préprod et prod via la configuration de stacks

Exemples d'activités pratiques :
• Déploiement d'un réseau, d'une base de données et d'un serveur d'application sur le cloud cible
• Paramétrage d'un même projet pour plusieurs environnements via la configuration de stacks
• Extraction d'un composant réutilisable à partir d'une infrastructure existante

Gestion de l'état, des secrets et intégration CI/CD

• Gestion de l'état des déploiements : backends Pulumi Cloud ou self-managed, verrouillage, récupération
• Secrets et données sensibles : chiffrement natif Pulumi, intégration avec les gestionnaires de secrets cloud
• Intégration Git : versioning du code d'infrastructure, workflows collaboratifs
• Pipelines CI/CD : GitHub Actions et GitLab CI pour automatiser `pulumi preview` et `pulumi up`
• Revues de code d'infrastructure : validation des changements via pull request
• Rollbacks, audit des changements et fonctionnalités de débogage

Exemples d'activités pratiques :
• Migration d'un état local vers un backend partagé et gestion d'un secret chiffré
• Mise en place d'un pipeline GitHub Actions qui déclenche `pulumi preview` sur pull request
• Simulation d'un rollback après un déploiement défectueux

Bonnes pratiques, sécurité et usages avancés

• Structuration de projets à grande échelle : organisation du code, conventions, composants partagés
• Policy as Code : définition et application de règles de conformité sur les déploiements
• Sécurité : gestion des droits d'accès, protection des secrets, moindre privilège IAM
• Observabilité : audit, logs de déploiement, suivi des changements
• Écosystème : packages communautaires, intégration d'outils et services tiers
• Limites et choix d'architecture : quand Pulumi est le bon outil, quand il ne l'est pas

Exemples d'activités pratiques :
• Application d'une politique de sécurité bloquant un déploiement non conforme
• Consommation d'un package communautaire pour intégrer un service tiers dans un projet
• Revue croisée d'un projet Pulumi et documentation des choix d'architecture

Moyens et méthodes pédagogiques

- La formation alterne entre présentations des concepts théoriques et mises en application à travers d'ateliers et exercices pratiques (hors formation de type séminaire).
- Les participants bénéficient des retours d'expérience terrains du formateur ou de la formatrice
- Un support de cours numérique est fourni aux stagiaires

Modalités d'évaluation

- **En amont de la session de formation**, un questionnaire d'auto-positionnement est remis aux participants, afin qu'ils situent leurs connaissances et compétences déjà acquises par rapport au thème de la formation.

- **En cours de formation**, l'évaluation se fait sous forme d'ateliers, exercices et travaux pratiques de validation, de retour d'observation et/ou de partage d'expérience, en cohérence avec les objectifs pédagogiques visés.

- **En fin de session**, le formateur évalue les compétences et connaissances acquises par les apprenants grâce à un questionnaire reprenant les mêmes éléments que l'auto-positionnement, permettant ainsi une analyse détaillée de leur progression.