Engineer

Bicep

le langage propre à Azure pour déclarer l'infrastructure

Ce que c’est

Bicep est un langage de Microsoft qui permet de décrire l’infrastructure Azure sous forme de texte : quelles ressources existent, quels noms elles portent, comment chacune est configurée et comment elles se relient. Un fichier Bicep n’énumère aucune étape. Il décrit le résultat final, et Azure Resource Manager, la couche traversée par les requêtes de gestion Azure, détermine ce qu’il faut créer, modifier ou laisser intact pour l’atteindre. Une personne qui demande ce qu’est Bicep veut généralement savoir s’il remplace les clics successifs dans le portail Azure. C’est le cas, et c’est l’essentiel de sa valeur.

Sous cette forme, Bicep est une manière plus lisible d’écrire quelque chose qu’Azure comprenait déjà. Resource Manager accepte depuis longtemps des modèles de déploiement en JSON, complets mais difficiles à lire et encore davantage à examiner. Pendant un déploiement, l’outil Bicep en ligne de commande convertit un fichier `.bicep` en l’un de ces modèles JSON, que Resource Manager déploie exactement comme un modèle écrit à la main. Microsoft précise que Bicep prend en charge chaque type de ressource et chaque version d’API dès leur introduction par le fournisseur de ressources, préversions comprises. Un nouveau service est donc utilisable sans attendre la mise à jour d’une extension. Les modèles JSON existants peuvent être décompilés en Bicep : convertir un ancien parc devient une opération mécanique plutôt qu’une réécriture.

Un petit fichier montre la structure. Les paramètres viennent en premier, puis les ressources, chacune avec un nom symbolique auquel le reste du fichier peut faire référence :

```
param env string
param location string = resourceGroup().location

resource store 'Microsoft.Storage/storageAccounts@2023-05-01' = {
  name: 'st${env}${uniqueString(resourceGroup().id)}'
  location: location
  sku: {
    name: 'Standard_LRS'
  }
  kind: 'StorageV2'
}
```

Lorsqu'une autre ressource mentionne `store.id`, Bicep déduit qu'elle doit attendre le compte de stockage. Personne n'a donc à écrire l'ordre à la main, et Resource Manager déploie en parallèle tout ce qui n'a aucune dépendance. Un fichier peut en appeler un autre comme module, pour décrire une fois « une base de données avec ses règles réseau » et réutiliser cet ensemble partout. Les valeurs propres à chaque environnement résident dans un fichier `.bicepparam` distinct. Microsoft indique explicitement que Bicep est gratuit, à code source ouvert et n'a pas vocation à être un langage généraliste : ses boucles, conditions et fonctions servent uniquement à décrire les ressources.

Comment se compare-t-il aux autres solutions ? Face à ARM JSON, il utilise le même moteur avec un code source bien plus court. Face à Terraform, les différences principales sont la portée et l'état : Terraform couvre de nombreux clouds et conserve son propre fichier d'état, tandis que le processus Azure Resource Manager de Bicep ne conserve pas de fichier d'état propre, car Azure stocke l'état du déploiement. Les extensions de Bicep prennent aussi en charge des ressources extérieures à Resource Manager, dont Kubernetes et Microsoft Graph. Face à Pulumi, qui décrit l'infrastructure dans des langages généralistes comme TypeScript ou Go, Bicep échange de la souplesse contre un langage plus petit, rapide à lire lors d'une revue.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce que Bicep ? » pour éclairer une décision réelle. Nous pouvons vous aider à relier ce concept au système que vous construisez ou achetez. Toute la configuration décrite dans des fichiers, pour la recréer à l'identique sans qu'elle ne dépende de la mémoire d'une seule personne.

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. 1 000 règles RBAC pour sécuriser Linux — 1,000 règles d'accès écrites pour combler une faille de sécurité

1 000 règles RBAC pour sécuriser Linux Contactez-nous →

Pourquoi c’est important

Une infrastructure construite à la main n’a pas d’historique. Un paramètre modifié dans le portail ne laisse aucune justification, personne ne l’examine avant le changement, et reconstruire l’environnement après une panne dépend des souvenirs de chaque clic. Dès que l’infrastructure devient un ensemble de fichiers dans le dépôt, elle bénéficie de ce que possède déjà le code applicatif : une demande de fusion, une personne chargée de la revue, un historique et un moyen de revenir en arrière.

Le deuxième gain est la cohérence des environnements. Développement, préproduction et production construits à partir des mêmes fichiers avec des paramètres différents ne diffèrent que là où ces paramètres l’indiquent. Les déploiements Bicep sont idempotents : selon la formulation de Microsoft traduite ici, un même fichier déployé plusieurs fois donne « les mêmes types de ressources dans le même état ». Un déploiement peut donc être répété sans risque, et créer un environnement revient à diriger la même commande vers un nouveau groupe de ressources.

Le troisième gain est la visibilité avant le changement. L’opération what-if compare un fichier à l’existant et énumère chaque ressource qu’elle créerait, modifierait ou supprimerait avant toute intervention. Associé à une demande de fusion, cet aperçu transforme « je pense que cela ne change que la règle de pare-feu » en une affirmation que la personne chargée de la revue peut vérifier.

Les coûts méritent d’être nommés. Les ressources prises en charge dépendent du processus de déploiement et de ses extensions ; le DNS d’un autre fournisseur ou les ressources d’un second cloud peuvent donc encore nécessiter un outil distinct. Dans le processus Resource Manager, Bicep décrit des ressources Azure ; les paquets et les fichiers à l’intérieur d’une machine virtuelle relèvent de la gestion de configuration, par exemple avec Ansible. Enfin, les règles appliquées par Resource Manager à la suppression et au redéploiement sans configuration contraire surprennent les personnes qui attendent d’un outil d’état souhaité qu’il retire ce que le fichier ne mentionne plus. La quatrième section couvre ce point.

À quoi cela ressemble en pratique

Plusieurs projets Azure de ce site reposent sur Bicep, et trois montrent ce qu'il apporte à une équipe dans la durée.

Sur une plateforme, tout le parc Azure est écrit en Bicep. Chaque environnement provient des mêmes modèles : api et www sous Azure Container Apps, un PostgreSQL Flexible Server, Redis pour le cache, Front Door avec une politique WAF en façade, et le réseau sous-jacent — VNet, NSG, DNS privé — avec Log Analytics raccordé pour le diagnostic et des images récupérées dans Azure Container Registry du projet. Comme l'ensemble est paramétré, créer un environnement ou modifier l'existant passe par une demande de fusion. La page décrit des environnements devenus reproductibles et examinables, ainsi qu'une dérive qui a cessé d'être mystérieuse parce que le code fait foi.

Les mêmes modèles portent le parcours de publication. Les versions passent du développement à la préproduction puis à la production, et Bicep maintient une infrastructure cible identique d'un environnement à l'autre : une compilation promue ne rencontre donc jamais de différence subtile. La configuration propre à chaque environnement est séparée des secrets, ce qui permet à un artefact construit de traverser toutes les étapes sans être reconstruit.

Bicep conserve aussi une partie du plan de reprise. La conservation des sauvegardes est configurée dans Bicep à côté de la base protégée, pour que la restauration à un instant donné soit une propriété du modèle plutôt qu'un réglage autrefois choisi dans le portail. L'environnement entourant la base est lui aussi du code : restaurer des données dans un environnement qu'il faut reconstruire de mémoire n'est guère une reprise. La page nomme ensuite les points restant ouverts : géoredondance désactivée et objectif de délai de reprise proposé plutôt que mesuré. Un modèle montre la configuration ; seul un exercice montre le temps nécessaire à une restauration.

Où les problèmes surviennent

La plupart des difficultés viennent de l'attente d'un outil qui possède tout le contenu d'un groupe de ressources. Sans configuration contraire, Bicep ne gère que ce que le fichier mentionne, et seulement de la façon décrite.

Un réglage modifié dans le portail disparaît au déploiement suivant. Redéployer une ressource réapplique toutes ses propriétés, et toute propriété omise par le modèle revient à sa valeur prédéfinie. Une correction manuelle pendant un incident est donc annulée par la publication ordinaire suivante. Traitez le fichier comme l'état final complet de chaque ressource, inscrivez-y chaque valeur différente de la valeur prédéfinie et faites passer les changements urgents par une demande de fusion. Les paramètres d'interdiction d'une pile de déploiement peuvent bloquer entièrement les modifications extérieures au déploiement.

Une ressource retirée du modèle continue à fonctionner et à coûter de l'argent. Sans choix explicite d'un autre mode, le déploiement est incrémentiel : les ressources du groupe absentes du fichier sont laissées intactes. Supprimer une base du code la laisse donc dans Azure. Microsoft répond par les piles de déploiement. Une pile consigne les ressources qu'elle gère, et son paramètre `actionOnUnmanage` décide si une ressource retirée du fichier est détachée ou supprimée. Le mode complet supprime lui aussi les ressources omises du fichier, mais Microsoft le déconseille et le retire progressivement.

L'aperçu what-if liste des changements que personne n'a effectués. Une propriété omise du fichier, mais remplie par Azure avec une valeur prédéfinie, peut apparaître comme supprimée : Microsoft appelle cela du bruit. Toute propriété définie depuis `reference` est annoncée comme modifiée à chaque passage. Des valeurs comme `utcNow()` ou un paramètre sécurisé n'existent que pendant le déploiement et apparaissent donc non évaluées. Exécutez l'aperçu à chaque demande de fusion, lisez-le comme un brouillon et définissez explicitement les propriétés bruyantes pour faire ressortir les vrais changements :

```
az deployment group what-if \
  --resource-group rg-staging \
  --template-file main.bicep \
  --parameters staging.bicepparam
```

Un mot de passe apparaît dans l'historique de déploiement. Un paramètre chaîne ordinaire est enregistré avec le déploiement, où toute personne pouvant lire l'historique peut le consulter. Marquez les paramètres secrets avec le décorateur `@secure()`, qui exclut leur valeur de l'historique et des journaux. Conservez le secret lui-même dans Key Vault plutôt que dans un fichier de paramètres du dépôt.

Pour aller plus loin

- Qu’est-ce que Bicep ? — la présentation de Microsoft : langage, bénéfices et objectifs qu’il ne cherche pas à couvrir.
- Modules Bicep — diviser un déploiement en fichiers qui s’appellent.
- Paramètres dans les fichiers Bicep — types, décorateurs, valeurs utilisées lorsque les paramètres ne sont pas fournis et éléments protégés dans l’historique par `@secure()` .
- Modes de déploiement — modes incrémentiel et complet, et leur traitement des ressources absentes du fichier.
- Opération what-if de déploiement Bicep — aperçu d’un déploiement, types de changements et sources de bruit connues.
- Piles de déploiement — gérer les ressources comme un ensemble, avec suppression et paramètres d’interdiction.

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 Bicep ? » 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/bicep/