Engineer

Bicep

мова Azure для опису інфраструктури

Що це таке

Bicep — це мова від Microsoft, якою інфраструктуру Azure записують як текст: які ресурси існують, як вони називаються, як налаштований кожен і як вони пов'язані між собою. У файлі Bicep немає кроків. Він описує кінцевий результат, а Azure Resource Manager, шар, через який проходять запити на керування Azure, сам визначає, що створити, що змінити, а що залишити як є. Хто питає «що таке Bicep?», зазвичай хоче знати, чи може він замінити клацання по порталу Azure. Може, і в цьому більша частина його цінності.

Під капотом Bicep — зручніший спосіб написати те, що Azure і так розумів. Resource Manager давно приймає шаблони розгортання у форматі JSON: вони повні, але їх важко читати й ще важче рецензувати. Під час розгортання інструмент командного рядка Bicep перетворює файл `.bicep` на такий JSON-шаблон, і Resource Manager розгортає його так само, як написаний вручну. За словами Microsoft, Bicep підтримує всі типи ресурсів і версії API одразу, щойно постачальник ресурсів їх запроваджує, зокрема й попередні версії, тож новою службою можна користуватися, не чекаючи, поки хтось оновить плагін. Наявні JSON-шаблони можна декомпілювати в Bicep, і перенесення старої інфраструктури стає механічною роботою, а не переписуванням.

Невеликий файл показує форму. Спершу параметри, далі ресурси, кожен із символьним ім'ям, на яке може посилатися решта файлу:

```
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'
}
```

Коли інший ресурс згадує `store.id` , Bicep сам робить висновок, що той має дочекатися облікового запису сховища, тож порядок вручну ніхто не прописує, а Resource Manager паралельно розгортає все, що ні від чого не залежить. Один файл може викликати інший як модуль, і команда описує «базу даних з її мережевими правилами» один раз, а потім використовує всюди. Значення, що відрізняються між середовищами, лежать в окремому файлі `.bicepparam` . Microsoft прямо пише, що Bicep безкоштовний, має відкритий код і не задуманий як мова програмування загального призначення: цикли, умови й функції в ньому є, але лише для опису ресурсів.

Як Bicep співвідноситься з альтернативами? Порівняно з ARM JSON це той самий рушій із набагато коротшим кодом. Порівняно з Terraform різниця в охопленні й стані: Terraform працює з багатьма хмарами й веде власний файл стану, а процес роботи Bicep через Azure Resource Manager не веде власного файлу стану, бо стан розгортання зберігає Azure. Розширення Bicep також підтримують ресурси поза Resource Manager, зокрема Kubernetes і Microsoft Graph. Порівняно з Pulumi, який описує інфраструктуру звичайними мовами програмування на кшталт TypeScript чи Go, Bicep міняє гнучкість на меншу мову, яку швидко прочитати під час рецензування.

Застосуйте поняття до рішення

Застосуйте «Що таке Bicep?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Уся конфігурація записана у файлах, тож її можна відтворити точно, і вона не живе лише в чиїйсь голові.

Переглянути наші послуги Зв'язатися з нами →

Перевірте припущення за терміном

Використайте пояснення та пов'язані роботи, щоб сформулювати запитання для проєкту. Ми можемо перевірити припущення разом до вибору інструмента. Написала 1 000 правил RBAC для Linux-серверів — 1,000 правил доступу написано, щоб усунути вразливість

Написала 1 000 правил RBAC для Linux-серверів Зв'язатися з нами →

Чому це важливо

Інфраструктура, зібрана вручну, не має історії. Налаштування, змінене в порталі, не залишає по собі причини, ніхто не переглядає його до того, як воно стається, а відбудова середовища після поганого дня залежить від того,

чи хтось пам’ятає кожне клацання. Коли інфраструктура — це набір файлів у репозиторії, вона отримує те, що код застосунку вже має: pull request, рецензента, історію й шлях назад.

Друга перевага — однаковість. Development, staging і production, зібрані з тих самих файлів із різними параметрами, відрізняються лише там, де так кажуть параметри. Розгортання Bicep ідемпотентні: за словами Microsoft, той самий файл, розгорнутий багато разів, дає ті самі типи ресурсів у тому самому стані. Тому розгортання можна спокійно повторювати, а нове середовище — це та сама команда, спрямована на нову групу ресурсів.

Третя — видимість до зміни. Операція what-if порівнює файл із тим, що працює, і перелічує кожен ресурс, який вона створила б, змінила б чи видалила б, ще до того, як щось буде зачеплено. Разом із pull request такий попередній перегляд перетворює «здається, це змінює лише правило брандмауера» на твердження, яке рецензент може перевірити.

Ціну теж варто назвати. Підтримка ресурсів залежить від способу розгортання та його розширень, тому для DNS в іншого постачальника чи ресурсів у другій хмарі все ще може знадобитися окремий інструмент. У способі роботи через Resource Manager Bicep описує ресурси Azure; пакети й файли всередині віртуальної машини належать до керування конфігурацією, наприклад Ansible. А поведінка Resource Manager за замовчуванням під час видалення й повторного розгортання дивує тих, хто чекає, що інструмент бажаного стану прибере все, чого файл більше не згадує. До цього ми повернемося нижче.

Як це виглядає на практиці

Кілька проєктів на Azure на цьому сайті тримаються на Bicep, і три з них показують, що він дає команді з часом.

На одній платформі вся інфраструктура Azure описана в Bicep. Кожне середовище виходить із тих самих шаблонів: api та www як Azure Container Apps, PostgreSQL Flexible Server, Redis для кешування, Front Door з політикою WAF попереду й мережа під усім цим (VNet, NSG, приватний DNS), з підключеним Log Analytics для діагностики та образами з Azure Container Registry проєкту. Оскільки все параметризовано, нове середовище чи зміна

наявного — це pull request. Сторінка описує результат так: середовища стали відтворюваними й доступними для перегляду, а дрейф перестав бути загадкою, бо джерелом істини є код.

Ті самі шаблони несуть шлях релізу. Релізи рухаються від development до staging і далі до production, а шаблони Вісер тримають цільову інфраструктуру ідентичною від одного середовища до іншого, тож білд не потрапляє в трохи інше місце. Конфігурація, що різниться між середовищами, зберігається окремо від секретів, і той самий зібраний артефакт проходить усі етапи без перезбирання.

А ще Вісер тримає частину плану відновлення. Політику зберігання резервних копій налаштовано в Вісер поруч із базою даних, яку він захищає, тож відновлення на конкретний момент часу є властивістю шаблону, а не налаштуванням, яке колись хтось клікнув. Середовище навколо бази теж описане кодом, і це важливо: відновити дані в середовище, яке довелося б відбудовувати з пам'яті, — навряд чи відновлення. Далі сторінка називає те, що лишається відкритим: геонадлишковість вимкнено, а цільовий час відновлення запропоновано, а не виміряно. Шаблон показує, що налаштовано; скільки триває відновлення, показує лише тестове відновлення.

Де виникають проблеми

Більшість клопоту з Вісер походить від очікування, що він поводиться як інструмент, який володіє всім у групі ресурсів. За замовчуванням він володіє лише тим, що згадано у файлі, і лише так, як файл каже.

Налаштування, змінене в порталі, зникає під час наступного розгортання. Повторне розгортання ресурсу заново застосовує всі його властивості, а кожна властивість, якої шаблон не містить, повертається до значення за замовчуванням. Виправлення, зроблене вручну під час інциденту, скасовує наступний звичайний реліз. Вважайте файл повним кінцевим станом кожного ресурсу, записуйте в нього кожне нестандартне значення, а термінові зміни робіть через pull request. Налаштування deny у deployment stack можуть узагалі заблокувати зміни в обхід розгортання.

Ресурс, прибраний із шаблону, далі працює й далі коштує грошей. Режим розгортання за замовчуванням — інкрементний: ресурси в групі, яких файл не згадує, залишаються на місці. Видаліть базу даних із коду, і в Azure вона

залишиться. Відповідь Microsoft — deployment stacks. Stack записує ресурси, якими керує, а його параметр `actionOnUnmanage` вирішує, чи ресурс, що випав із файлу, буде від'єднано чи видалено. Режим complete теж видаляє те, чого у файлі немає, але Microsoft не радить ним користуватися й поступово його виводить.

Попередній перегляд what-if показує зміни, яких ніхто не робив. Властивість, якої у файлі немає, але яку Azure під час розгортання заповнює значенням за замовчуванням, може бути показана як видалена, і Microsoft називає це шумом. Будь-яка властивість, задана через функцію `reference`, щоразу показується як змінена, а значення на кшталт `utcNow()` чи захищеного параметра існують лише під час розгортання, тож what-if показує їх необчисленими. Запускайте його на кожен pull request, читайте як чернетку й задавайте шумні властивості явно, щоб справжні зміни було видно:

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

Пароль з'являється в історії розгортань. Звичайний рядковий параметр зберігається разом із розгортанням, і його бачить кожен, хто має доступ на читання до історії. Позначайте секретні параметри декоратором `@secure()`, який тримає значення поза історією й журналами, а сам секрет зберігайте в Key Vault, а не у файлі параметрів у репозиторії.

Що почитати далі

- What is Bicep? — власний огляд від Microsoft: мова, її переваги й те, чим вона не намагається бути.
- Bicep modules — як розбити розгортання на файли, що викликають один одного.
- Parameters in Bicep files — типи, декоратори, значення за замовчуванням і те, що `@secure()` тримає поза історією.
- Deployment modes — інкрементний режим проти complete і що кожен робить із ресурсами, яких файл не згадує.
- Bicep deployment what-if operation — попередній перегляд розгортання, типи змін і відомі джерела шуму.

- Deployment stacks — керування ресурсами як однією одиницею, з видаленням і налаштуваннями deny.

Словник

Обирайте за фактичними умовами

Опишіть шлях розгортання, експлуатаційні ризики та чергову команду. Ми можемо визначити зміну з перевірками, шляхом відновлення та чіткою відповідальністю.

Дивитися, скільки це коштує Зв'язатися з нами →

Зробіть термінологію практичною

Запитайте, як «Що таке Bicep?» стосується вашої організації. Конкретного прикладу з поточної системи достатньо для початку розмови.

Переглянути наші послуги

welcome@engineer.company +45 50 10 61 84 Копенгаген, Данія

Візьміть цю сторінку із собою

Візитівка

Дбайливо зроблено з любов'ю — D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Авторське право © 2025 – дотепер · Engineer ApS · Усі права захищено.

Надруковано з https://engineer.company/uk/glossary/bicep/