Prokopenko Mykola

СУЧАСНІ ПІДХОДИ У УПРАВЛІННІ ІНФРАСТРУКТУРОЮ НА РІЗНИХ ПРОЕКТАХ З ТОЧКИ ЗОРУ DEVOPS ІНЖЕНЕРА. І ПРО ДОЦІЛЬНІСТЬ МІКРОСЕРВІСІВ

Західному світу вигідно співпрацювати з інжнерами з України. Зазвичай форми взаємодії з клієнтами аутсорс та аутстаф. Україна, як і раніше, тримає співвідношення ціна/якість. Ми трохи можемо бути грубішими, але застосовуємо свіжі підходи такі як Infrastructure as a Code та GitOps, про які сьогодні поговоримо.

Infrastructure as a Code

Інфраструктура як код – це підхід до конфігурації інфраструктури за допомогою коду, в якому ми описуємо її бажаний стан. Все завжди починається із ручного підходу. Опис інфраструктури кодом завжди дорожчий, ніж налаштувати все руками. Скрипти та код для інфраструктури окупається з лишком у перспективі. IaC заощаджує потенційний час на документуванні інфраструктури. Цей підхід зменшує можливість накопичення різниці між різними ідентичними оточеннями. Може збільшувати відмовостійкість шляхом підняття частин інфраструктури, що відповідають за обробку важких запитів (але іноді краще використовувати системи оркестрації такі як, наприклад, Kubernetes). Підхід зменшує час так званого disaster recovery (аварійне відновлення).

Мінуси:

- більш трудомісткий;
- Потрібні кваліфіковані фахівці відразу, щоб описати очікуваний стан інфраструктури
- далеко не завжди можна швидко описати інфраструктуру кодом.

Ідемпотентність є принципом інфраструктури як коду.

Сухі цифри на прикладах:

- Конфігурація сервера з nginx, з генерацією сертифіката за допомогою Ansible займає до 2х хвилин. У той же час конфігурація руками може зайняти до 20 хвилин, враховуючи чіткі вимоги і хороші знання інженера.
- створення та налаштування інфраструктури в AWS що складається з Lambda, RDS, SQS, SNS та CloudWatch руками займає близько 1-1,5 години. За допомогою CloudFormation це можна зробити до 4-5 хвилин.

Важливо дивитися на сухі цифри та адекватно оцінювати наявні людські та технічні ресурси.

GitOps

GitOps відносно новий підхід до управління інфраструктурою як код. Цілком тримається на системі контролю версій. Не може існувати без вже описаної інфраструктури кодом. Git репозиторій є єдиним вірним джерелом істини. Дозволяє скорочувати ресурси управління інфраструктурою як кодом. Можна завжди відстежити хто і що зробив(бо все робиться через Git). Доступ до інфраструктури має обмежена кількість людей, а всі необхідні іншим зміни здійснюються тільки через Pull Request'и. Будь-якої миті можна відкотити інфраструктуру до попереднього стану. Мінімізується людський фактор. Легше дотримуватися вимог безпеки. Будь-якої миті часу можна вивантажити стан зміни інфраструктури, щоб порівняти її з еталонним станом в репозиторії.

Мікросервіси це добре, але не завжди доцільно

Бізнес середовище конкуретне і не дружнє.

Мікросервіси це завжди надійніше, і може масштабуватись, навідмінно від монолітної архітектури.

Моноліт і мікросервіси мають місце в нинішніх реаліях.

Плюси моноліту:

- швидкість розробки;
- простота розгортання;
- моноліт розробляється завжди швидше;
- іноді простіше підтримувати.

Мінуси моноліту:

- згодом збільшується час входження до проекту;
- складність масштабування;
- складність упровадження нових технологій.

Плюси мікросервісної архітектури:

- незалежний набір технологій;
- незалежність команд;
- майже нескінченна масштабованість;
- висока відмовостійкість.

Мінуси мікросервісної архітектури:

- складність розробки;
- складність підтримки;
- складність тестування;

Потрібно застосовувати обидва підходи у комплексі. Спочатку розробляється моноліт. Потім по необхідності дробимо на мікросервіси. Програма або сервіс має спочатку пройти обкатку. Якщо запуск проекту/сервісу не задовольняє певним заздалегідь показникам, його подальший розвиток може не мати сенсу.

Все вищесказане є суб'єктивним досвідом. Скільки людей, стільки та ситуацій. Всі клієнти/бізнеси/проекти можуть кардинально відрізнятись підходами до проектування. Всі ці підходи можуть застосовуватись як усі разом, так і частково. Важливо завжди ставати на бік бізнесу - уявіть, що це ваш бізнес, як би Ви надійшли в тій чи іншій ситуації? Гнучкість та вміння адаптуватися – вирішальні фактори у побудові успішної працюючої системи. Важливо дивитися проблеми комплексно. Клієнти не люблять слово "проблема". Будь-який проект та ідея має право на життя, але все потрібно перевіряти.

Перелік посилань

1. https://www.redhat.com/en/topics/automation/what-is-infrastructure-as-code-iac#:~:text=Infrastructure%20as%20Code%20(IaC)%20is,to%20edit%20and%20distribute%20configurations.
2. https://docs.microsoft.com/en-us/devops/deliver/what-is-infrastructure-as-code
3. https://about.gitlab.com/topics/gitops/infrastructure-as-code/
4. https://about.gitlab.com/topics/gitops/
5. https://www.redhat.com/en/topics/devops/what-is-gitops
6. https://www.digitalocean.com/blog/monolithic-vs-microservice-architecture