1) Такі рішення уже існують, тож що нового у твоєму дослідженні?

Так, ви праві, ці інструменти не такий вже і новий тренд. Однак в ЗСУ це не дуже активно використовується, а це можна застосувати для автоматизації розгортання різноманітної необхідної нам інфраструктури, не обов’язково на рівні головного інформаційного центру, а на рівні військової частини (ініціалізувати віртуальні машини, контейнери, а на них вже наприклад розгортати tftp сервер, системи моніторингу Zabbix, Prometheus тощо).

2) Де можна застосувати цю модель у ЗСУ?

Конкретно ця модель доречна для певного відділу веброзробки та супроводження для того, щоб захостити вебсайт, який створять веброзробники і при цьому, щоб фахівці із відділу експлуатації не налаштовували це все вручну, а автоматизовано. Проте насправді це можна застосовувати навіть на рівні військової частини. Відділ зв’язку адмініструє багато сервісів, наприклад системи моніторингу - і це можна також побудувати, описавши інфраструктуру як код.

3) А чому саме такі інструменти IaC були використанні для моделі? Чому така модель? Чому…

Це лише демонстраційна модель, я не можу побудувати певну універсальну модель як панацею під усі випадки життя, оскільки це неможливо.

Звісно, можна було і задати запитання чому Apache, а не Nginx, оскільки другий має вбудований балансувальник навантаження, але знову таки в конкретних ситуаціях підходять різні рішення.

Суть концепції CI/CD та IaC в тому, що його можна використовувати для різних задач та потреб. І відповідно для цього можуть підійти трохи краще ті чи інші інструменти. Моя модель демонструє те, як це можна використовувати для того щоб побудувати необхідну інфраструктуру, щоб задеплоїти вебсайт.

4) Чому саме GitLab CI/CD ? А не наприклад Jenkins

GitLab надає можливість створювати локальний репозиторій, а також це повноцінна DevOps платформа, а отже використовувати один GitLab цілком достатньо для реалізації поставлених задач до моделі.

GitLab CI/CD має вбудований інтерфейс для налаштування конвеєрів CI/CD, тому він є дещо простішим порівняно з Jenkins, який зазвичай вимагає додаткових плагінів та конфігурацій.

Звісно якщо уявити іншу модель, інші задачі для реалізації якої є команда, котра має досвід роботи з Jenkins або у вас є складні вимоги до конвеєрів CI/CD, які важко задовольнити в GitLab CI/CD, Jenkins може бути кращим вибором.

5) Що таке мережевоцентрична війна? Розкажіть детальніше і яке відношення CI/CD та IaC до цього мають

Мережево-центрична війна орієнтована на досягнення інформаційної переваги за допомогою об'єднання військових об’єктів (наприклад, інформація розвідданих з багатьох можливих джерел) у одну інформаційну систему та мережу.

Ведення війни із використанням мережево-центричних принципів фактично неможливе без CI/CD та IaC.

6) Поясни слайд про IaC

Зліва на схемі показано розробників, які пишуть код і надсилають його до системи контролю версій через push або pull запити. Код потрапляє до конвеєра (pipeline), де він збирається та тестується. На правій стороні схеми показано DevOps команду, яка описує інфраструктуру як код (IaC) за допомогою різних інструментів, таких як скрипти, шаблони та політики. Ця інформація потрапляє до платформ для автоматичного розгортання інфраструктури. Після успішного проходження через конвеєр, код і описана інфраструктура розгортаються у середовищах dev (розробка), stage (тестування) та prod (продакшн).

7) Що таке Lean Thinking? Розкажіть в чому його суть і чому ви це використали як додаток?

Це теорія ощадливого виробництва. Ця тема дуже дотична до мого дослідження. Гнучка методологія Agile означає ітераційну розробку маленькими шматками, що забезпечує гнучкість та постійний зворотний зв’язок. Моя робота це фактично перехід від теоретичного осмислення до практичних моментів впровадження. А Agile-підходи у свою чергу запозичені із принципів філософії Toyota Таїті Оно, тому винесла в додаток основоположну інформацію.

8) Що за додаток про досвід роботи GWS? Чому ви це винесли в додаток?

В додатку Б я винесла реальний кейс, досвід роботи GWS. Загалом він розповідає про важливість використання автоматизованих тестів. Зі слів Майка Бленда, котрий долучився до команди GWS, їхня команда потрапила в ситуацію, коли їм було надзвичайно важко внести зміни у вебсервер, також казав, що це дуже вплинуло на розробників, як він каже “страх перед змінами став убивцею мислення, бо вони розуміли наслідки, які може відчути кінцевий користувач при пошуку чогось через їхні помилки”. GWS швидко стала однією з найпродуктивніших команд у компанії. Керівник команди Бхарат Медіратта вважав, що автоматизоване тестування допоможе розв’язати проблему. Результати виявилися вражаючими. Бленд зазначав: “GWS швидко стала однією з найпродуктивніших команд у компанії”.

Абревіатури

**API (Application Programming Interface)** - це набір чітко визначених правил та специфікацій, які описують, як різні програмні компоненти повинні взаємодіяти один з одним. Іншими словами, API визначає, як різні частини програмного забезпечення спілкуються та обмінюються даними.

**REST API (Representational State Transfer Application Programming Interface)** - це архітектурний стиль для розробки вебсервісів, який базується на принципах REST. REST - це набір архітектурних принципів, які визначають, як вебсервіси повинні працювати. У REST API, ресурси, наприклад дані або функціональність, ідентифікуються за допомогою URI (Універсальний ідентифікатор ресурсу). Клієнти взаємодіють із ресурсами, використовуючи HTTP-методи:

GET - для отримання ресурсу

POST - для створення нового ресурсу

PUT - для оновлення ресурсу

DELETE - для видалення ресурсу

Дані зазвичай передаються у форматах, таких як JSON або XML.

**IRC (Internet Relay Chat)** - це протокол і мережа, призначені для обміну текстовими повідомленнями в режимі реального часу. IRC дозволяє користувачам підключатися до різних каналів (кімнат) та спілкуватися один з одним у режимі чату. Кожен канал зазвичай присвячений певній темі або інтересам, що дозволяє користувачам знаходити людей зі схожими інтересами.

**PaaS (Platform as a Service)** - це модель хмарних обчислень, в якій постачальник хмарних послуг надає платформу для розробки, тестування, розгортання та запуску додатків на його інфраструктурі.

У моделі PaaS постачальник хмарних послуг бере на себе відповідальність за управління основною інфраструктурою, операційними системами, міддлвером, засобами розробки та інструментами, необхідними для розробки та розгортання додатків. Розробники можуть зосередитися на створенні додатків, не турбуючись про управління та налаштування базової інфраструктури.

**SaaS (Software as a Service)** - це модель постачання програмного забезпечення, в якій постачальник розробляє та підтримує застосунок, а користувачі отримують доступ до нього через Інтернет за моделлю підписки або оплати за використання.

У моделі SaaS програмне забезпечення не встановлюється на комп'ютерах користувачів, а замість цього запускається на серверах постачальника та доступне через веб-браузер або тонкий клієнт. Користувачі можуть отримати доступ до додатка з будь-якого пристрою, підключеного до Інтернету.

**SLA (Service Level Agreement)**офіційна угода між постачальником послуг та клієнтом, що визначає очікувані рівні послуг та відповідальність сторін.

**SSL (Secure Socket Layer)** – криптографічний протокол для забезпечення зашифрованого або безпечного з’єднання між вебсервером (сайтом) і браузером