Актуальність

У сучасній боротьбі з противником, котрий має значну перевагу в майже усьому нам вкрай необхідно мати перевагу у веденні мережоцентричної війни, таким чином створивши асиметричну перевагу.

Побудова системи з мережоцентричними принципами без використання CI/CD та IaC хоча і буде теоетично можливою, хай і дуже складно, проте на практиці відсутність IaC та CI/CD зведе нанівець ключові переваги мережоцентричного підходу - гнучкість, масштабованість, адаптивність та швидке реагування. Система стане повільною, ненадійною та вразливою до помилок.

Слайд 5

CI/CD це безперевна інтеграція та безперевна доставка.

Частинка CI стосується практики регулярного збирання та тестування коду програми після кожного злиття у спільний репозиторій. Це допомагає виявляти помилки на ранніх стадіях циклу розробки – що є суттєвою перевагою.

Щодо наступних переваг це: можливість частіших релізів, автоматичне тестування, гнучкість та економія часу за рахунок автоматизації.

CD розширює концепцію CI, автоматизуючи процес релізу програмного забезпечення для кінцевих користувачів.

Важливо зауважити, що є 2 види CD: Delivery та Deployment. Різниця в тому, що Deployment (Безперервне розгортання) повністю автоматичний процес, який після усіх тестів сам закидає готовий код у виробниче середовище (prod). А Delivery (Безперевна доставка) не автоматичний, але автоматизований процес, при якому потрібне ручне схвалення DevOps.

Слайд 6

На слайді зображений типовий виробничий процес.

З використанням CI/CD ми зменшуємо його час. Ось приклад, що відбувається, коли розробник закомітив зміни.

Конвеєр розгортання (pipeline) після будь-якої зміни перевіряє, що код успішно інтегрується у середовище, близьке до виробничого.

За такого сценарію час розгортання вимірюється хвилинами або, у гіршому разі, годинами, а карта потоку цінності має виглядати приблизно так, як показано на рисунку.

Слайд 7

На слайді зображена концепція IaC

Інфраструктура як код - це парадигма, в якій вся інфраструктура ІТ-систем (обчислювальні ресурси, сховища даних, мережі тощо) визначається як версійований машинозчитуваний код замість ручного налаштування ресурсів.

Сьогодні більшість проєктів живе в «хмарах». Тобто ресурси — це віртуальні машини, контейнери, балансувальники навантаження.

Утім, побутує міф про небезпеку використання хмарних сервісів для інформаційних систем, пов’язаних із сектором оборони, проте справжній досвід виразно показав, що такі побоювання являються марними. Як приклад можна навести досвід використання системи ситуаційної обізнаності *Delta.*

Слайд 8

Проаналізувавши різні інструменти я обрала саме Ansible та Terraform, зокрема через простоту в розумінні та популярність і відповідно широке ком’юніті. Це важливий аспект, оскільки що для того, щоб це впровадити і щоб це працювало, варто відповідно навчити персонал. А чим складніше нам це зробити - тим нам менш вигідно.

Незважаючи на відмінності в підходах (*Ansible* – для управління конфігураціями, *Terraform* – для управління інфраструктурою), їх часто використовують разом у великих проєктах для досягнення повної автоматизації та реалізації IaC принципів.

Розгляньмо детальніше про їх особливості.

Слайд 9

В контексті Інфраструктури як код, *Terraform* це засіб ініцілізації ресурсів. Засоби ініціалізації ресурсів створюють самі ресурси, тобто віртуальні машини, кластер *Kubernetes*, бази даних, мережі, системи моніторингу, сертифікати SSL і майже будь-який аспект інфраструктури.

На слайді показана візуалізація як Terraform працює. DevOps планує зміни, застосовує їх використовуючи різні провайдери, зокрема як у прикладі, які власне розгортають цю ж інфраструктуру.

Слайд 11

Ansible - це потужний інструмент для автоматизації конфігурації та управління системами.

Базове середовище *Ansible* складається з трьох основних компонентів:

1) Вузол управління (*Control node*) де встановлено *Ansible*.

2) Керований вузол (*Managed node*) - віддалена система або хост, яким керує *Ansible*.

3) Інвентар (*Inventory*) - Список керованих вузлів, які логічно організовані.

Слайд 13

На даному слайді зображена демонстраційна модель, котра ілюструє приклад використання процесів CI/CD та IaC для того, щоб захостити вебсайт.

Зверху на схемі ви можете побачити розробників, які пишуть програмний код. Трохи нижче команду DevOps, котра на контрольній ноді керує конфігураціями Ansible та Terraform для розгортання необхідної інфраструктури для того, щоб захостити сайт, котрий пишуть розробники. Terraform ініцілізує віртуальну машину, власне створюватиме наш managed node. Ansible ж буде мати інвентарному файлі, де ми вказуємо, що адмініструватимемо керуючу ноду і власне сам playbook. Все це буде зберігатися як і програмний код в репозиторії GitLab. А GitLab буде запускати підготовлений конвеєр.

Висновки

Отже, впроваджуючи процеси CI/CD разом із інструментами IaC ми здобуваємо наступні переваги, щодо IaC це:

Ідемпотентність, тобто багаторазове використання коду

Декларативний підхід, тобто ми описуємо бажаний стан конфігурації і ці інструменти постійно цей стан відстежують та контролюють, + це означає, що будь-хто охочий може легко розібратися… щодо наступного Легка відтворюваність – можна не тільки легко ознайомитися, а і просто це відтворити

Мобільність та масштабованість, ця перевага пов’язана із щойно названою (наприклад, якщо ваш відділ географічно розширюватиметься і скажуть відтворити знову таку ж інфраструктуру, то це буде дуже легко зробити)

Ну і звісно це автоматизація і скорочення часу (ви лише один раз описуєте бажаний стан системи, а не кожного разу всі налаштування та зміни вручну)

Додавши сюди процеси CI/CD ми здобуваємо версіонування конфігурацій, таким чином ви не будете боятися впроваджувати оновлення через думку, що щось може зламатися, оскільки ви в будь-який момент можете відкотити до попередньої версії, коли все було добре. Ба більше, так можна простіше тестувати продукт. При надходженні нового коду в систему контролю версій будуть запускатися тестування, при успішному проходженні яких або зміни автоматично надходитимуть у виробниче середовище, або “будуть напоготові”, коли DevOps вручну схвалить це все.