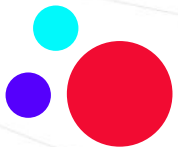


Systemy CIcD

infoShare Academy



HELLO

Karol Kołodziejczyk

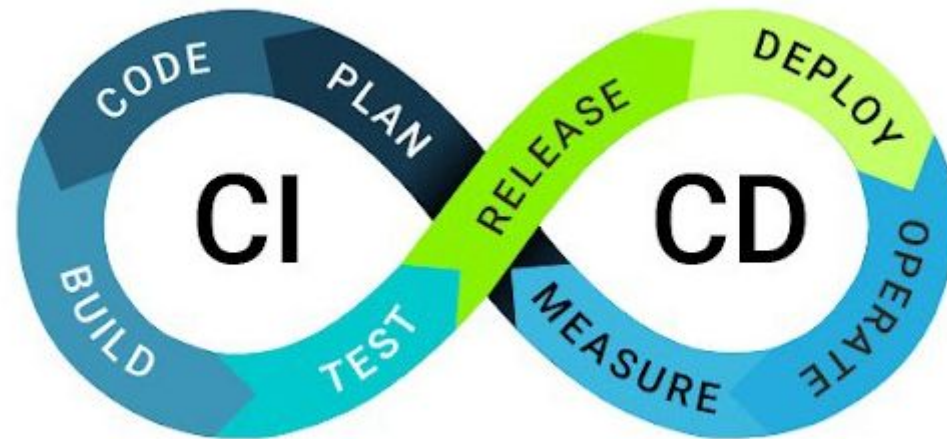
DevOps/Cloud engineer @ 7N
Azure, Azure DevOps, Terraform, Data engineering

- Organizacja
- Wstęp
- Podstawowe zasady i pojęcia
- Narzędzia i krótkie porównanie
- Budowa i główne elementy składowe



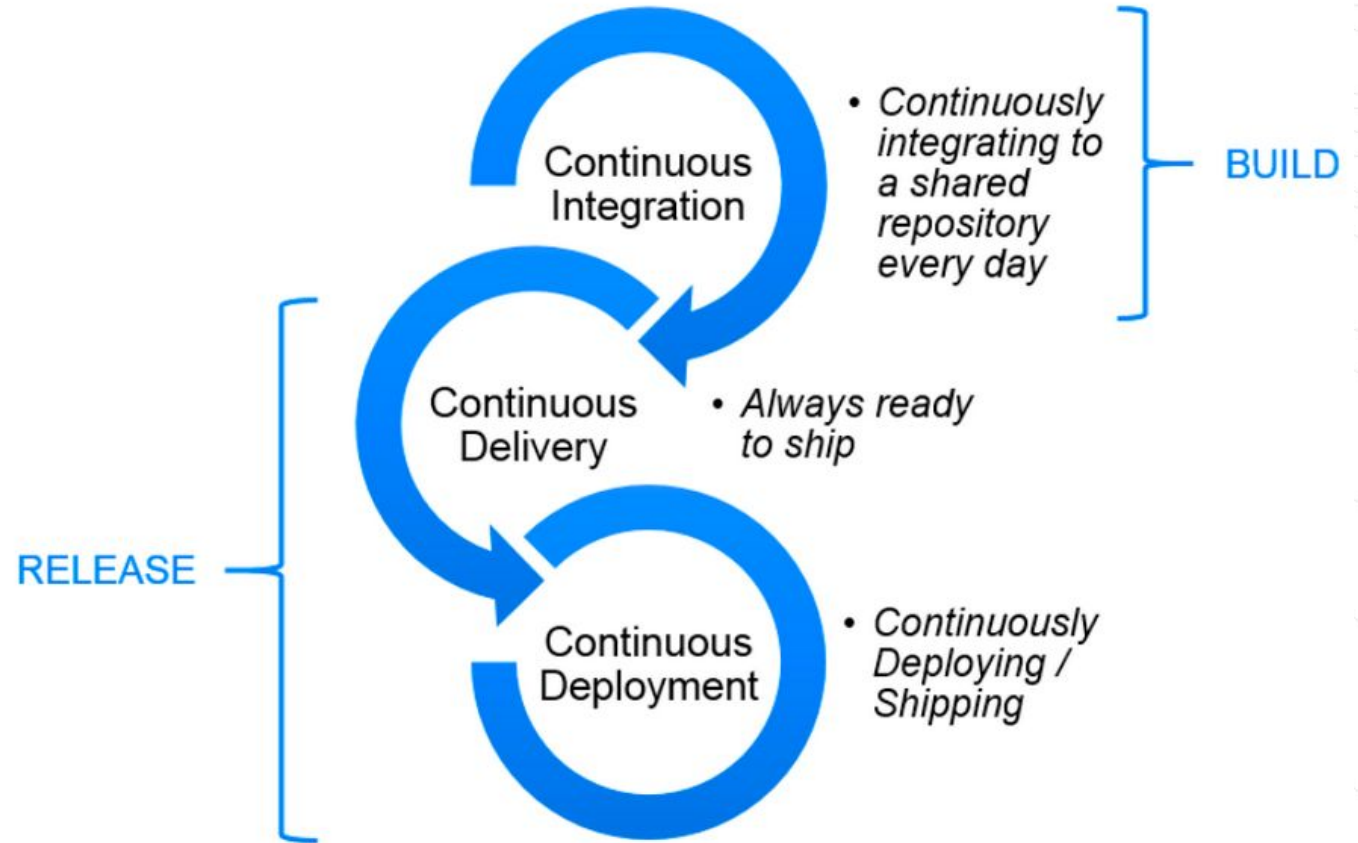
Wstęp

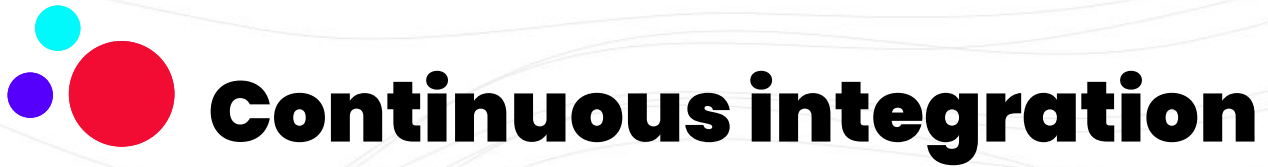
- Wspomagające wytwarzanie oprogramowania
- Automatyzujące powtarzalne i manualne kroki
- Zaprojektowane i stworzone z myślą o integracji i wdrażaniu aplikacji i systemów.



- Automatyzacja powtarzalnych procesów oraz skrócenie czasu wdrożenia zmian do klientów
- Zmniejszenie nakładów pracy (automatyzacja kroków)
- Większa odporność na błędy (testy automatyczne, brak zmian manualnych, kilka środowisk dev/test/production)
- Przyspieszenie wdrożeń

- Continuous integration
- Continuous delivery
- Continuous deployment





Continuous integration

- Problem – duże, rozproszone zespoły
- Test locally -> Pull Request -> Build -> Test -> Merge to master

Zasady:

- Integrowanie często – przyrostowo
- Im większa zmiana, tym większe prawdopodobieństwo problemów
- Testy!




Continuous delivery

- Problem – biznes potrzebuje nowych wersji do sprawdzenia
- Merge to master -> Build, Test -> Publish package -> Release to Dev/Test

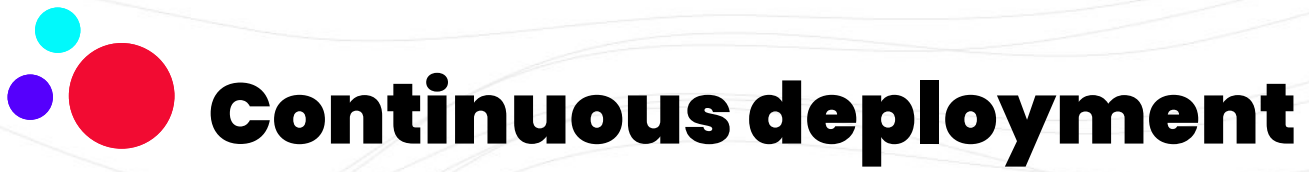
Zasady:

- Budowanie i dostarczanie często – przyrostowo
- Regularne dostarczenie nowych wersji aplikacji
- Częstsze dostarczenie paczek -> częstsza walidacja

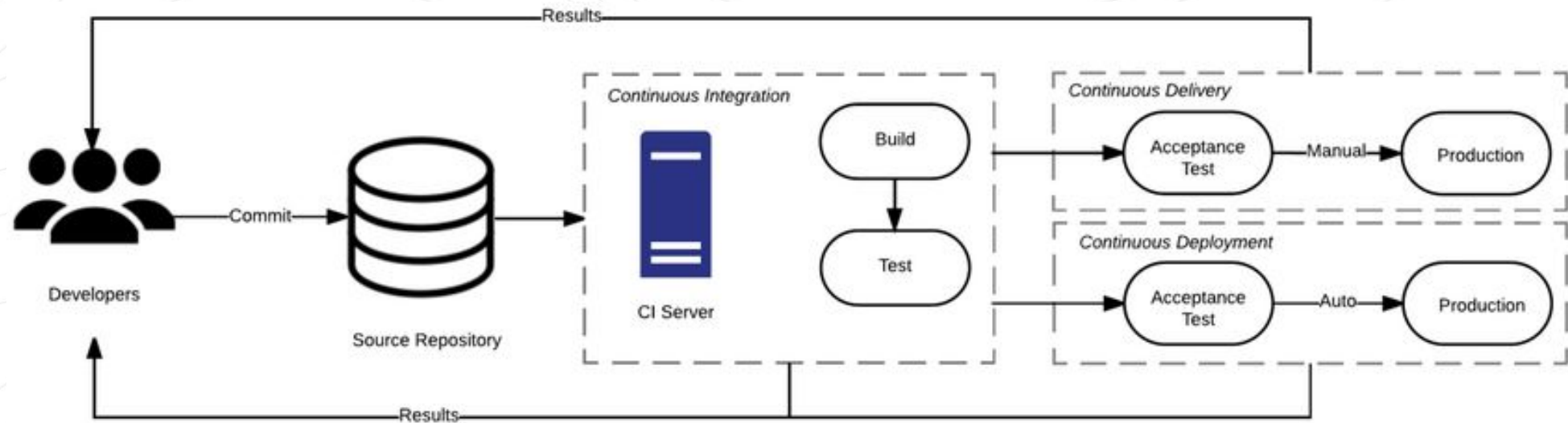


Continuous deployment

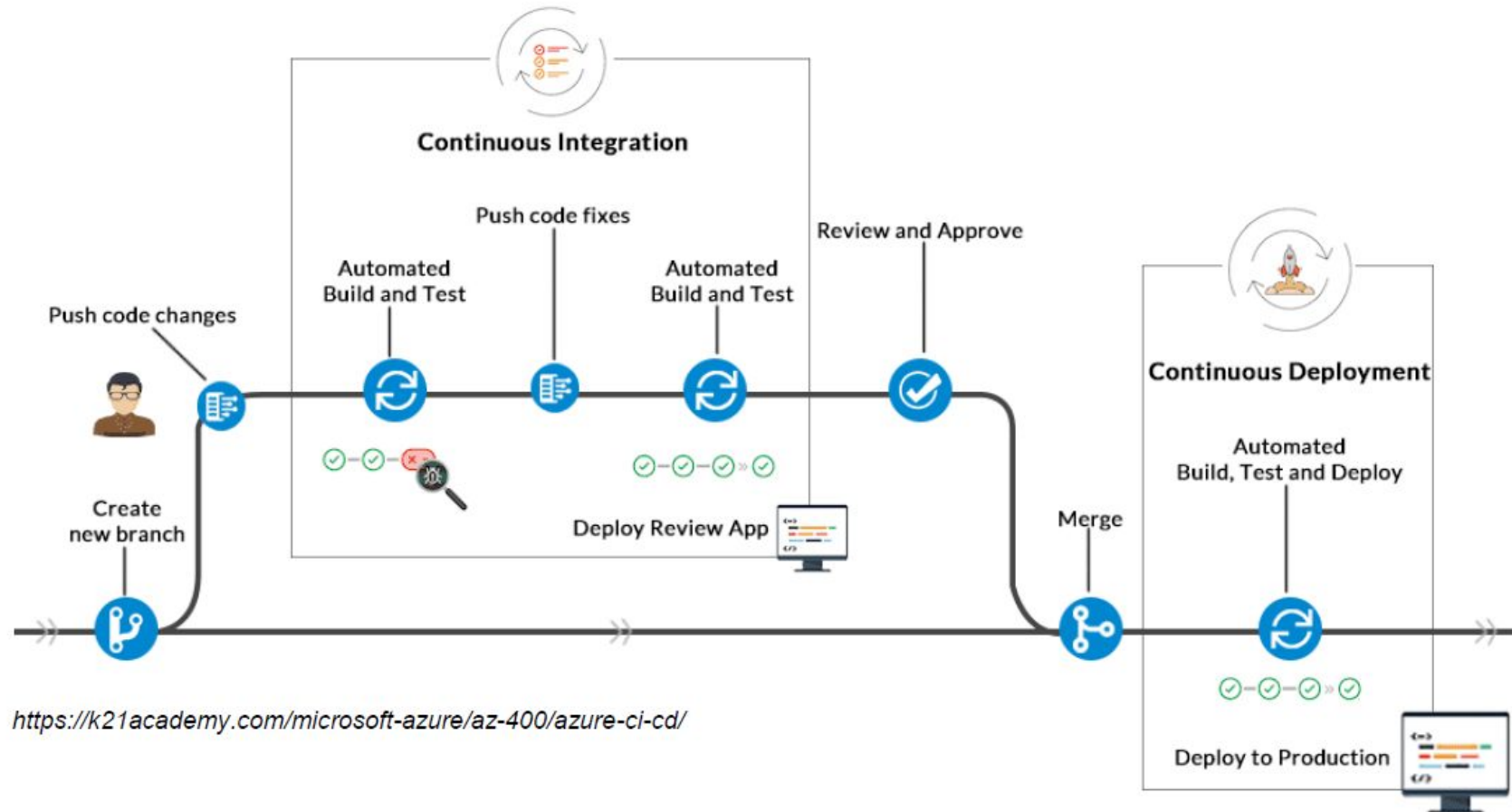
- Automatyzacja całego procesu budowania i wdrożenia na produkcję
- Brak manualnych interwencji
 - Brak ręcznych zgód
 - Automatyczne przywrócenie do działania w razie problemów
- Wymagana dojrzałość systemów i rozwiązań
 - Testy
 - Monitoring
 - Rollback



Continuous deployment



Workflow



<https://k21academy.com/microsoft-azure/az-400/azure-ci-cd/>



Ważne punkty i dobre praktyki

- Planowanie i projektowanie na początku
- Wersjonowanie systemów CI/CD
- Separacja środowisk i ich spójność
- Gate'y – warunkowe uruchamianie
- Testowanie jako kluczowy element
- Częste wdrożenia
- Standaryzacja (template'y)
- Monitoring
- Dokumentacja



Najczęstsze błędne praktyki

- Brak strategii i standaryzacji – tworzenie procesów CI/CD bez wcześniejszego zaprojektowania
- Brak kontroli wersji pipeline'ów
- Brak podziału lub spójności środowisk
- Ręczne wdrażanie zmian
- Brak code review
- Brak testów
- Brak analizy artefaktów – rozmiaru, zawartości, security
- Wdrożenia bez zasad odnośnie branch'y i PRów
- Brak monitoringu lub reakcji na monitoring
- Brak dokumentacji

- Jenkins
- GitLab
- Azure DevOps
- Github Actions
- TeamCity
- CircleCI
- i więcej...



Porównanie narzędzi

	Cena	Hosting	Open source	Źródło pluginów i rozszerzeń	Wbudowane repo GIT	Zalety
Jenkins	Darmowy	Self-hosted	Tak	Internal store	Nie	Popularność, duża ilość pluginów
GitLab	Per user/month	Self-hosted/SaaS	Tak	Internal store	Tak	GIT, Duża ilość pluginów
circleCI	Per month	Self-hosted/SaaS	Nie	Internal store/GitHub	Nie	Prosty i szybki
TeamCity	Per month	Self-hosted/SaaS	Nie	Internal store/GitHub	Nie	Dużo systemów kontroli wersji - integracja
Azure DevOps	Per user/month	Self-hosted/SaaS	Nie	Internal store/GitHub	Tak	Duża ilość pluginów, integracja z Azure, GIT
GitHub Actions	Per user/month	Self-hosted/SaaS	Nie	Internal store/GitHub	Tak	GitHub, pluginy, Azure



Porównanie narzędzi

- Ciekawe źródło

<https://www.czerniga.it/pl/2022/03/27/znajdz-swoje-najlepsze-narzedzie-ci-cd/>



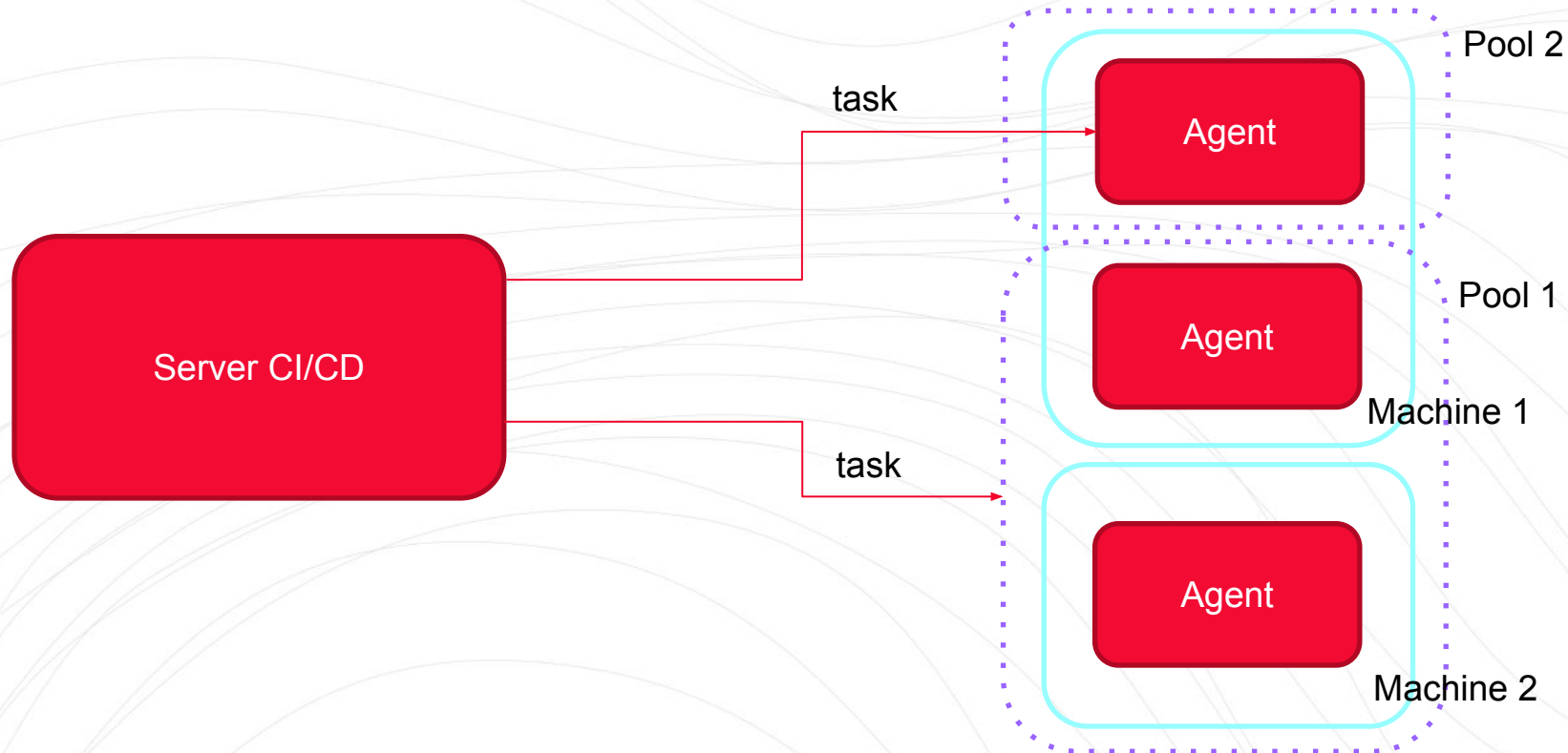
02. Demo i rozgrzewka!

infoShare
ACADEMY



Budowa i główne elementy składowe

- System CI/CD – system do realizacji zadań CI/CD i zarządzania wszystkim dookoła (dostępny, integracje, agenty itp.)
- Agent/node/worker/runner – maszyna/host lub serwis działający na hoście, który jest używany do wykonywania zadań w ramach systemu CI/CD



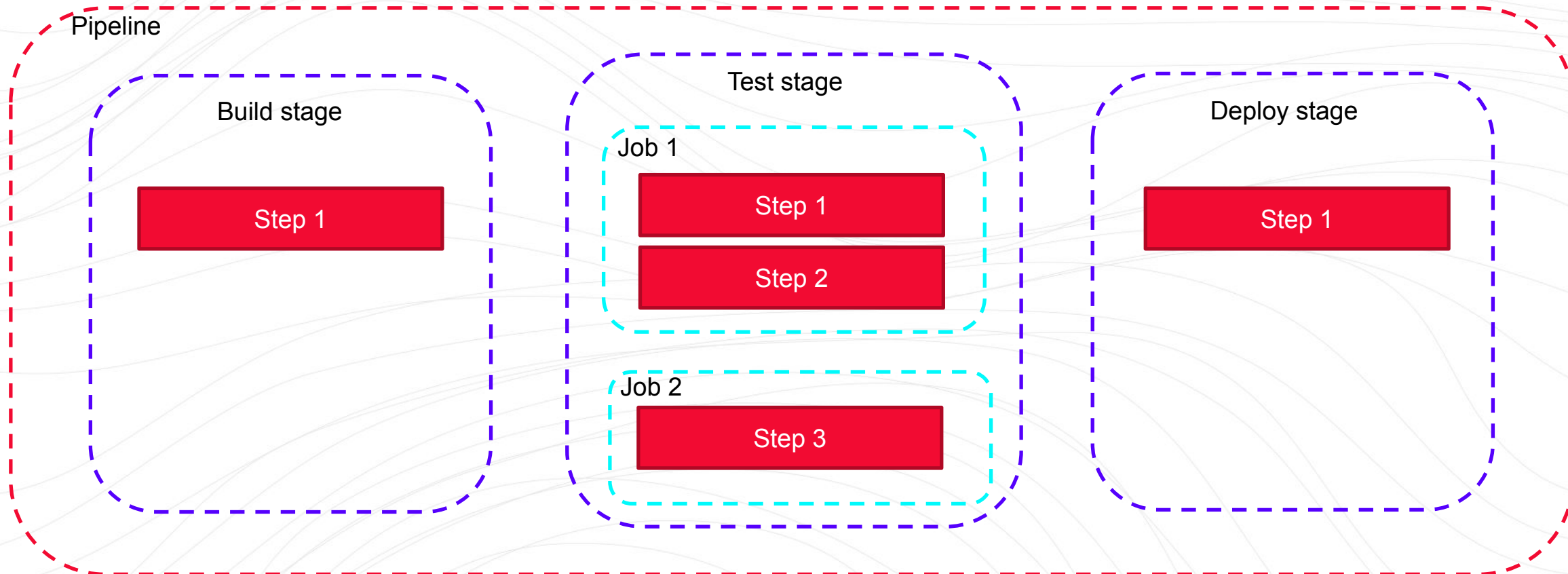


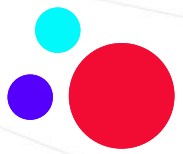
Główne pojęcia

- Pipeline – cały ciąg procesów związanych z aplikacją – najwyższy poziom
- Stage – podzbiór zadań wykonywanych przez pipeline (np. build lub test)
- Job – paczka zadań w ramach jednego etapu
- Step/task – pojedyncze zadanie



Główne pojęcia





Główne pojęcia – elementy dodatkowe

- Trigger – warunek, po którym pipeline jest startowany
- Artefakt – zbiór plików/paczka, która powstaje po przejściu pipeline'a/stage'a – buildu; zwykle tagowane/wersjonowane
- Środowisko – np. zbiór hostów, na który wrzucamy paczkę
- Zmienne – wartości używane w pipeline'ach
- Sekrety – np. hasła, które używamy w pipeline'ach lub aplikacji
- Gate'y – warunkowe uruchamianie kolejnych etapów pipeline'u (np. wynik testu, approval)



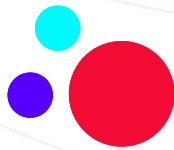
Główne akcje

- Wybór i konfiguracja agenta
- Konfiguracja potrzebnych zmiennych
- Pobranie repozytorium
- Statyczna analiza kodu
- Budowanie aplikacji
- Uruchomienie testów
- Opublikowanie/pobranie artefaktu
- Konfiguracja środowisk/gate'ów i zależności



Integracje

- Uwierzytelnianie (GitHub, Azure AD)
- Połączenia np. do chmury czy na server produkcyjny
- Zewnętrzne serwery GIT
- Zewnętrzne triggery pipeline'ów – webhooki
- Powiadomienia zewnętrznych serwisów – eventy, wiadomości, notyfikacje



Metody budowania pipeline'ow

- Logikę zwykle konfigurujemy za pomocą
- yaml lub JSON
- GUI
- cli



04. Ćwiczenia!

infoShare
ACADEMY



THANK YOU FOR YOUR ATTENTION

infoShareAcademy.com