Semaphore UI w Dockerze: Ansible z przeglądarki

Poziom: operacyjny
Dotyczy: Semaphore UI 2.19, obraz semaphoreui/semaphore:v2.19.12 **Czas: ~35 min**

Efekt: działający kontener Semaphore z własną bazą PostgreSQL, projekt podłączony do repozytorium playbooków w Gitea i pierwszy szablon zadania, który Ola potrafi uruchomić sama, bez konta SSH na żadnym serwerze.

Skróty użyte na tej stronie (8)

Skrót	Co znaczy
AD	Active Directory: katalog Microsoftu z kontami użytkowników, komputerów i grup w firmie
CLI	Command Line Interface: praca przez wpisywanie komend w terminalu
IT	Information Technology: dział albo praca związana z komputerami i infrastrukturą firmy
LDAP	Lightweight Directory Access Protocol: protokół do odpytywania katalogu użytkowników (AD, FreeIPA)
LDAPS	LDAP szyfrowany TLS, zwykle port 636
SSH	Secure Shell: szyfrowane zdalne logowanie do serwera
StartTLS	Start Transport Layer Security: polecenie, które zamienia zwykłe, nieszyfrowane połączenie (np. LDAP na porcie 389) w szyfrowane już po jego nawiązaniu
WWW	World Wide Web: strony i aplikacje otwierane w przeglądarce po HTTP albo HTTPS

Zanim zaczniemy

Z ŻYCIA WZIĘTE

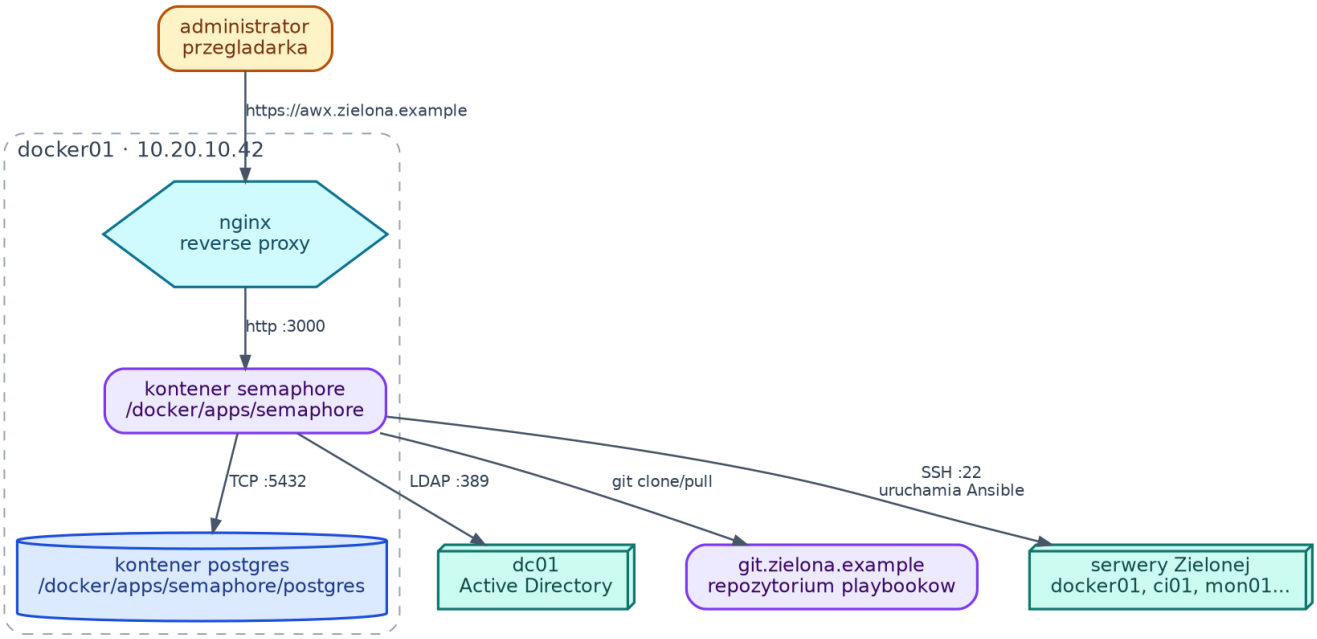
Ola od miesiąca prosi o dostęp „tylko na chwilę” do serwerów, na których stoją jej kontenery, żeby samodzielnie odpalić playbook aktualizujący zależności. Zespół IT nie chce rozdawać kluczy SSH do produkcji trzeciej osobie, ale też nie chce być jedynym, kto potrafi wpisać `ansible-playbook`. Semaphore rozwiązuje oba problemy naraz: Ola dostaje przycisk w

> przeglądarce, klucz SSH zostaje tam, gdzie był.

Komendy wykonujemy z CLI bash jako user z uprawnieniami sudo albo root, na `docker01`, z zainstalowanym Dockerem i wtyczką Compose. Zakładamy, że Ansible i pierwszy inwentarz już istnieją, zgodnie ze stroną Instalacja Ansible i pierwszy inwentarz Zielonej, i że playbooki leżą w repozytorium Gitea.

> W naszym środowisku dane aplikacji trzymamy w `/docker/apps/<nazwa>`, a plik Compose w `/docker/compose/<nazwa>/compose.yaml`. To konwencja, nie wymóg Dockera: można ją zmienić, byle w całej firmie była jedna. Katalog `/docker` to osobny wolumen `vgDocker/lvDocker`, żeby pełny dysk aplikacji nie zatrzymał systemu.

Wymaganie	Uwaga
Docker Engine i wtyczka Compose	sprawdzamy `docker compose version`
Wolny port `3000` na `docker01`	reverse proxy nginx stawiamy przed nim tak samo jak przed innymi aplikacjami Zielonej
Repozytorium z playbookami w Gitea	patrz Pierwsze repozytorium: od pustego projektu do pierwszego pusha
Klucz SSH z poprzedniej strony	prywatna połowa trafia do magazynu poświadczeń Semaphore, nie do repozytorium



Architektura Semaphore na docker01: nginx, kontener semaphore i postgres, połączenia do AD, Gitea i serwerów Zielonej (kliknięcie otwiera pełny rozmiar)

Przebieg

Compose z Semaphore i własną bazą

Zakładamy katalogi zgodnie z konwencją ścieżek:

```
sudo mkdir -p /docker/apps/semaphore/postgres /docker/apps/semaphore/config
sudo mkdir -p /docker/compose/semaphore
```

Plik `/docker/compose/semaphore/.env`, obok `compose.yaml`, nie w nim samym. Klucz szyfrowania kluczy generujemy sami, nie kopiujemy żadnego przykładu z dokumentacji:

```
openssl rand -base64 32
```

```
SEMAPHORE_DB_PASSWORD=HasloDoBazy
SEMAPHORE_ADMIN_PASSWORD=HasloAdministratora
SEMAPHORE_ACCESS_KEY_ENCRYPTION=WygenerowanyKluczBase64
```

Plik `/docker/compose/semaphore/compose.yaml`:

```
services:
  postgres:
    image: postgres:16
    container_name: semaphore-postgres
    restart: always
    environment:
      - POSTGRES_USER=semaphore
      - POSTGRES_PASSWORD=${SEMAPHORE_DB_PASSWORD}
      - POSTGRES_DB=semaphore
    volumes:
      - /docker/apps/semaphore/postgres:/var/lib/postgresql/data

  semaphore:
    image: semaphoreui/semaphore:v2.19.12
    container_name: semaphore
```

```
    restart: always
    ports:
      - "3000:3000"
    environment:
      - SEMAPHORE_DB_DIALECT=postgres
      - SEMAPHORE_DB_HOST=postgres
      - SEMAPHORE_DB_PORT=5432
      - SEMAPHORE_DB_USER=semaphore
      - SEMAPHORE_DB_PASS=${SEMAPHORE_DB_PASSWORD}
      - SEMAPHORE_DB=semaphore
      - SEMAPHORE_ADMIN=admin
      - SEMAPHORE_ADMIN_NAME=admin
      - SEMAPHORE_ADMIN_EMAIL=admin@zielona.example
      - SEMAPHORE_ADMIN_PASSWORD=${SEMAPHORE_ADMIN_PASSWORD}
      - SEMAPHORE_ACCESS_KEY_ENCRYPTION=${SEMAPHORE_ACCESS_KEY_ENCRYPTION}
      - TZ=Europe/Warsaw
    volumes:
      - /docker/apps/semaphore/config:/etc/semaphore
    depends_on:
      - postgres
```

Pierwsze uruchomienie:

```
cd /docker/compose/semaphore
docker compose up -d
```

◫ ZRZUT EKRANU DO UZUPEŁNIENIA

Zrzut do uzupełnienia: pierwszy ekran logowania pod `http://docker01:3000`, formularz z polami „Username” i „Password”.

Projekt podłączony do Gitea

W interfejsie zakładamy nowy projekt i wskazujemy repozytorium z playbookami: adres `git.zielona.example/zespol-it/automatyzacja-zielona.git`, gałąź `main`. Semaphore klonuje repozytorium przy każdym uruchomieniu zadania, więc treść playbooka zawsze pochodzi z ostatniego commita, nie z tego, co ktoś wkleił kiedyś do formularza.

◫ ZRZUT EKRANU DO UZUPEŁNIENIA

Zrzut do uzupełnienia: ekran „New Project” z wypełnionym polem repozytorium Git i wybraną gałęzią `main`.

Klucze w magazynie Semaphore

W sekcji poświadczeń dodajemy klucz SSH wygenerowany na stronie Instalacja Ansible i pierwszy inwentarz Zielonej, typ „SSH Key”, wklejając prywatną część klucza. Playbook odwołuje się do tego poświadczenia przez nazwę, a osoba uruchamiająca zadanie, na przykład Ola, nie widzi ani nie potrzebuje samego klucza.

Inwentarz dodajemy tym samym plikiem `inventory.yml` co na poprzedniej stronie, wskazując go z repozytorium zamiast wklejać ręcznie, żeby obie strony (Ansible z linii poleceń i Ansible z Semaphore) korzystały z jednego źródła prawdy.

Szablon zadania

Nowy szablon zadania łączy trzy elementy: projekt, inwentarz i klucz SSH z kroków wyżej, plus konkretny playbook z repozytorium, na przykład `aktualizacja.yml`. Zapisany szablon staje się przyciskiem „Run”, który Ola klika bez znajomości linii poleceń.

◫ ZRZUT EKRANU DO UZUPEŁNIENIA

Zrzut do uzupełnienia: ekran „New Task Template” z wypełnionymi polami Playbook, Inventory i SSH Key, wskazującymi na `aktualizacja.yml`.

Harmonogram

Pole harmonogramu przyjmuje standardowy pięciopolowy zapis cron. Przykład: uruchomienie playbooka porządkującego stare pliki tymczasowe codziennie o drugiej w nocy:

```
0 2 * * *
```

◫ ZRZUT EKRANU DO UZUPEŁNIENIA

Zrzut do uzupełnienia: ekran „Schedules” z dodanym wpisem `0 2 * * *` przypisanym do szablonu zadania.

Logowanie LDAP z AD

Semaphore umie uwierzytelniać przez LDAP, w tym przez Active Directory, ale konfiguruje się to plikiem `config.json`, nie samym interfejsem. Po pierwszym uruchomieniu plik leży w `/docker/apps/semaphore/config/config.json` (zamontowanym jako `/etc/semaphore` w kontenerze) i dopisujemy do niego sekcję LDAP:

```
{
  "ldap_enable": true,
  "ldap_needtls": true,
  "ldap_server": "dc01.ad.zielona.example:389",
  "ldap_binddn": "CN=svc-semaphore,OU=Serwisowe,DC=ad,DC=zielona,DC=example",
  "ldap_bindpassword": "HasloKontaSerwisowego",
  "ldap_searchdn": "OU=Uzytkownicy,DC=ad,DC=zielona,DC=example",
  "ldap_searchfilter": "(sAMAccountName=%s)",
  "ldap_mappings": {
    "dn": "dn",
    "mail": "mail",
    "uid": "sAMAccountName",
    "cn": "displayName"
  }
}
```

Po zapisaniu pliku restartujemy kontener, żeby wczytał nową konfigurację:

```
docker compose restart semaphore
```

`ldap_binddn` wskazuje konto serwisowe w AD, którym Semaphore przeszukuje katalog, nie konto osoby logującej się. Dokładne wymaganie co do szyfrowania połączenia (`ldap_needtls`, port 389 ze StartTLS czy 636 z LDAPS) sprawdzamy w dokumentacji Semaphore dla naszej wersji, bo bywa zależne od konfiguracji samego `dc01`.

Dlaczego Semaphore, a nie AWX

Pełne porównanie obu narzędzi, niezależnie od Zielonej, stoi na stronie AWX i Semaphore. Tutaj tylko powód konkretnej decyzji:

Kryterium w Zielonej	Semaphore	AWX

Gdzie by dziś stanął	jeden dodatkowy kontener na już istniejącym `docker01`	potrzebuje Kubernetesa albo wielu współpracujących usług; jedyny kandydat to laboratoryjny `k3s01`
Kto go utrzyma	dwuosobowy zespół IT, bez dedykowanego administratora platformy	zwykle wymaga kogoś, kto na stałe pilnuje samej platformy, nie tylko playbooków
Kiedy Zielona po niego sięgnie	już teraz, do codziennego uruchamiania playbooków	gdy `k3s01` przejdzie z laboratorium do produkcji

Weryfikacja

Sprawdzamy, że oba kontenery działają:

```
docker compose ps
```

Przykład:

```
NAME                IMAGE                              STATUS
semaphore           semaphoreui/semaphore:v2.19.12     Up 3 minutes
semaphore-postgres  postgres:16                        Up 3 minutes
```

Sprawdzamy, że interfejs WWW odpowiada:

```
curl -I http://localhost:3000
```

Przykład:

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
```

Uruchamiamy zapisany szablon zadania z przeglądarki i sprawdzamy, że log zadania kończy się statusem „Success”, tak samo jak wcześniej kończyło się uruchomienie tego samego playbooka z linii poleceń.

Wycofanie zmian

Zatrzymanie i usunięcie kontenerów, bez kasowania danych:

```
docker compose down
```

Pełne usunięcie razem z danymi i historią zadań:

```
docker compose down -v
sudo rm -rf /docker/apps/semaphore
```

Gdy coś nie zadziała

Objaw	Przyczyna	Naprawa
Kontener `semaphore` restartuje się w pętli	baza `semaphore-postgres` jeszcze nie wystartowała	`docker compose logs postgres`, czekamy albo sprawdzamy hasło w `.env`
Zadanie kończy się `Permission denied (publickey)`	do szablonu podpięto złe poświadczenie albo klucz publiczny nie trafił na maszynę docelową	sprawdzamy przypisany klucz SSH w szablonie i powtarzamy `ssh-copy-id` z poprzedniej strony
Logowanie LDAP zwraca błąd połączenia	`dc01` nieosiągalny z kontenera albo zła wartość `ldap_needtls`	`docker compose exec semaphore sh` i próba połączenia do `dc01.ad.zielona.example:389` z wnętrza kontenera
Harmonogram nie uruchamia zadania o właściwej porze	zmienna `TZ` kontenera nie ustawiona albo ustawiona na inną strefę niż oczekiwana	ustawiamy `TZ=Europe/Warsaw` w `compose.yaml` i restartujemy kontener

□□ Ćwiczenia

> Zadania sprawdzające.

1. Wyjaśniamy, dlaczego prywatny klucz SSH trafia do magazynu poświadczeń Semaphore, a nie do repozytorium Gitea razem z playbookiem.

Odpowiedź

Repozytorium widzi cały zespół, a klucz prywatny daje pełny dostęp do maszyn docelowych. Magazyn poświadczeń Semaphore pozwala playbookowi odwołać się do klucza przez nazwę, bez ujawniania jego treści osobie uruchamiającej zadanie, nawet jeśli ta osoba ma prawo kliknąć „Run”.

1. Zespół chce, żeby Ola mogła uruchamiać wyłącznie jeden konkretny playbook, bez dostępu do reszty projektów w Semaphore. Opisujemy, jakim mechanizmem to

ograniczyć.

Odpowiedź

Osobny projekt w Semaphore ze swoim własnym zestawem poświadczeń, inwentarzy i szablonów zadań, do którego dodajemy Olę jako jedynego użytkownika spoza zespołu IT, zamiast dawać jej dostęp do głównego projektu z całą resztą automatyzacji.

1. Piszemy wpis harmonogramu, który uruchamia szablon zadania w każdą sobotę o 3:00.

Odpowiedź

`0 3 * * 6` (minuta 0, godzina 3, każdy dzień miesiąca, każdy miesiąc, dzień tygodnia 6, czyli sobota).

Powiązane: Instalacja Ansible i pierwszy inwentarz Zielonej · AWX i Semaphore · Ansible, AWX, Semaphore, Rudder, Terraform: co do czego · Pierwsze repozytorium: od pustego projektu do pierwszego pusha · Konwencja Dockera w Zielonej

Źródła: semaphoreui.com/docs, rozdział „Docker" (zmienne środowiskowe i ścieżka `/etc/semaphore`) · semaphoreui.com/docs, rozdział „LDAP and Active Directory” · semaphoreui.com/docs, rozdział „Schedules” (format cron)

Revision #22
Created 2026-09-28 17:15:25 UTC by Tomasz Konieczny
Updated 2026-10-26 14:13:39 UTC by Tomasz Konieczny