DOCKER W 1 DZIEŃ

OD PODSTAW, PO PROJEKTOWANIE I PRAKTYCZNE ZASTOSOWANIA

WSTĘP

Rozpoczynając od zarysu naszych celów, warto zaznaczyć, że głównym zamiarem tej książki jest dostarczenie czytelnikom kompleksowego źródła wiedzy o Dockerze – narzędziu, które zrewolucjonizowało sposób, w jaki deweloperzy i administratorzy systemów pracują z aplikacjami. Docker, będący platformą do konteneryzacji aplikacji, pozwala na łatwe i szybkie pakowanie, dystrybucję oraz uruchamianie aplikacji w izolowanych środowiskach zwanych kontenerami. Wiedza ta jest niezbędna dla osób, które chcą efektywnie wdrożyć i wykorzystywać konteneryzację w swoich projektach, zarówno na małą jak i dużą skalę.

Przedstawienie samej idei konteneryzacji, w tym przypadku realizowanej przez Docker, jest kluczowym punktem wyjścia. Kontenery Docker sprawiają, że aplikacje działają w sposób niezależny od środowiska, co znacząco ułatwia procesy developerskie, testingowe oraz deployment. Wyjaśnienie, jak dokładnie Docker izoluje aplikacje od środowiska, wymaga zagłębienia się w mechanizmy takie jak cgroups czy namespaces, które są używane w systemach operacyjnych typu Linux do izolacji zasobów.

Znaczącą część książki poświęcamy na opisanie Docker Engine, czyli serwerowej części oprogramowania, która odpowiada za uruchamianie kontenerów oraz zarządzanie nimi. Szczegółowo analizujemy każdą składową tej technologii, począwszy od instalacji Docker Engine na różnych systemach operacyjnych, aż po konfigurację i optymalizację dla specyficznych wymagań. Skupiamy się również na koncepcji obrazów Docker, które są szablonami dla kontenerów. Czytelnik dowie się, jak tworzyć, modyfikować oraz zarządzać obrazami, jak również jak wykorzystywać rejestr obrazów Docker Hub oraz prywatne rejestry.

Bardzo ważnym aspektem, na który zwracamy uwagę, jest praca z Dockerfile, plikiem, w którym zdefiniowane są instrukcje potrzebne do zbudowania obrazu Docker. Znajomość składni Dockerfile i umiejętność tworzenia efektywnych plików budujących jest kluczowa dla efektywnego wykorzystania Docker. Czytelnik nauczy

się stosować instrukcje takie jak FROM, RUN, COPY, czy ENTRYPOINT, a także zrozumie, jak wpływają one na rozmiar i wydajność końcowego obrazu.

Nie pomijamy również tematu sieciowania w Dockerze. Wyjaśniamy, jak Docker zarządza siecią wewnętrzną kontenerów, jak umożliwia komunikację między nimi, a także jak kontenery mogą komunikować się ze światem zewnętrznym. Szczególną uwagę przykładamy do możliwości konfiguracji sieciowych, takich jak bridge, host, czy overlay, które odpowiadają za różne aspekty sieciowania kontenerów.

W ramach praktycznych zastosowań Docker, omawiamy tworzenie skomplikowanych aplikacji składających się z wielu serwisów za pomocą Docker Compose. Jest to narzędzie, które pozwala na definiowanie i uruchamianie wielokontenerowych aplikacji Docker za pomocą jednego pliku YAML. Czytelnicy zrozumieją, jak używać Compose do zarządzania cyklem życia całego stosu aplikacji, od uruchomienia, przez aktualizację, aż po zatrzymanie wszystkich serwisów.

Kwestią o podstawowym znaczeniu dla użytkowników korporacyjnych jest bezpieczeństwo kontenerów. Wnikliwie omawiamy praktyki zapewniające bezpieczeństwo aplikacji uruchamianych w Dockerze, w tym zarządzanie dostępem, skanowanie obrazów pod kątem znanych podatności oraz aspekty dotyczące bezpiecznego przechowywania i zarządzania sekretami.

Bazując na tych wszystkich elementach, przechodzimy do pokazania zaawansowanych technik, takich jak orkiestracja kontenerów przy użyciu narzędzi takich jak Docker Swarm czy Kubernetes. Orkiestracja jest niezbędna w zarządzaniu rozbudowanymi systemami kontenerów, gdzie niezbędne jest ich automatyczne skalowanie, samonaprawa i zarządzanie. Czytelnik poznaje zarówno teoretyczne podstawy orkiestracji, jak i konkretne przykłady zastosowania w praktyce, wraz z omówieniem różnic i przypadków użycia obu rozwiązań.

Książka nie zapomina o aspekcie ciągłej integracji i ciągłego dostarczania (CI/CD), pokazując jak konteneryzacja wpisuje się w te procesy. Wyjaśniamy, jak Docker może być wykorzystany do automatyzacji procesów budowania i testowania aplikacji, a także jak wspiera on automatyczne wdrażanie aplikacji w środowiskach produkcyjnych. Na realnych przykładach prezentujemy konfigurację pipeline'ów w narzędziach takich jak Jenkins, GitLab CI/CD czy GitHub Actions, które wykorzystują Docker w cyklu życia oprogramowania.

Na koniec skupiamy się również na monitorowaniu i logowaniu w kontekście kontenerów Docker. Przyglądamy się narzędziom takim jak Docker stats, cAdvisor, Prometheus czy ELK Stack (Elasticsearch, Logstash, Kibana), które pozwalają na śledzenie wydajności i stanu zdrowia kontenerów, jak również na agregowanie i analizę logów z aplikacji działających w kontenerach.

Wnikliwa analiza każdego z tych aspektów ma za zadanie nie tylko nauczyć użytkownika, jak korzystać z Docker, ale również wyjaśnić dlaczego pewne rozwiązania są bardziej preferowane od innych, jak rozwiązywać typowe problemy oraz jak wykorzystać Docker do optymalizacji i automatyzacji pracy z aplikacjami. W efekcie, czytelnik powinien posiadać solidne podstawy do samodzielnego wdrożenia konteneryzacji w swoich projektach oraz do dalszego rozwijania swojej wiedzy w tym obszarze. Celem jest, aby po przyswojeniu zawartości tej książki, czytelnik czuł się komfortowo w świecie kontenerów i był w stanie wykorzystać Docker do poprawy jakości, wydajności oraz skalowalności swoich aplikacji.

PRAKTYCZNE PODEJŚCIE I WYMAGANIA WSTĘPNE

Rozpoczynając pracę z Dockerem, kluczowe jest zrozumienie, czym są kontenery oraz jakie wymagania wstępne muszą być spełnione, aby rozpocząć korzystanie z tej technologii. Kontenery Docker to lekkie, przenośne, samowystarczalne pakiety oprogramowania, które zawierają wszystko, co jest potrzebne do uruchomienia aplikacji: kod, środowisko uruchomieniowe, biblioteki systemowe oraz ustawienia. Umożliwiają one tworzenie izolowanych środowisk, które działają w sposób spójny na różnych platformach.

Aby rozpocząć pracę z Dockerem, należy przede wszystkim upewnić się, że używany system operacyjny jest kompatybilny. Docker wspiera większość popularnych dystrybucji systemu Linux, w tym Ubuntu, Debian, Fedora oraz CentOS. Ponadto, dostępne są edycje dla systemów macOS i Windows, choć w przypadku tego ostatniego wymagana jest edycja Pro, Enterprise, lub Education z obsługą Hyper-V i Containers. W przypadku systemów macOS i Windows, Docker Desktop zapewnia zintegrowane środowisko, które znacznie ułatwia zarządzanie kontenerami.

Instalacja Docker na systemie Linux wymaga uprawnień administratora oraz dostępu do terminala. Proces instalacji na różnych dystrybucjach może się nieznacznie różnić, ale generalnie polega na dodaniu repozytorium Docker do listy źródeł oprogramowania systemu, po czym można zainstalować pakiet `docker-ce` (Community Edition) lub `docker-ee` (Enterprise Edition) przy pomocy menedżera pakietów, jak `apt` w przypadku Ubuntu czy `dnf` w przypadku Fedora.

Na systemach macOS i Windows, proces instalacji jest uproszczony dzięki instalatorom graficznym. Wystarczy pobrać i uruchomić Docker Desktop, a program przeprowadzi użytkownika przez proces konfiguracji. Po zainstalowaniu, Docker Desktop dostępny jest w zasobniku systemowym i oferuje łatwy dostęp do konsoli Docker, zarządzania kontenerami i ustawień.

Niezależnie od systemu operacyjnego, warto zaznajomić się z wymaganiami sprzętowymi. Docker, chociaż lekki w porównaniu do wirtualnych maszyn, nadal wymaga pewnych zasobów systemowych. Zaleca się, aby komputer miał przynajmniej 4GB pamięci RAM oraz odpowiednio dużo miejsca na dysku twardym dla obrazów i kontenerów Docker, które mogą zajmować znaczącą ilość przestrzeni w miarę ich tworzenia i używania.

Po zainstalowaniu Docker, ważnym krokiem jest nauka pracy z interfejsem wiersza poleceń (CLI). Docker CLI jest potężnym narzędziem, które pozwala na zarządzanie cyklem życia kontenerów: od uruchamiania i zatrzymywania, po monitorowanie i logowanie. Komendy takie jak `docker run`, `docker stop`, czy `docker rm` są fundamentalne i pozwalają na interakcję z kontenerami. Ważne jest także zrozumienie, co to są obrazy Docker (Docker images), z których powstają kontenery. Komendy takie jak `docker pull` pozwalają na pobieranie obrazów z Docker Hub, co jest pierwszym krokiem do uruchomienia kontenera.

Równie istotne co umiejętność pracy z CLI jest zrozumienie plików Dockerfile. Dockerfile to tekstowy plik konfiguracyjny, który opisuje, jak zbudować obraz Docker, zawierając kolejne instrukcje, takie jak kopiowanie plików, ustawienie zmiennych środowiskowych, instalowanie pakietów czy ustawienie komendy uruchomieniowej. Umiejętność pisania Dockerfile pozwala na tworzenie niestandardowych obrazów, które są dokładnie dostosowane do potrzeb aplikacji.

Kolejnym istotnym elementem praktycznej pracy z Dockerem jest zrozumienie i wykorzystanie Docker Compose. Jest to narzędzie, które pozwala na definicję i uruchamianie aplikacji wielokontenerowych. Docker Compose korzysta z pliku YAML do konfiguracji usług, które mają współpracować, dzięki czemu cały stack aplikacyjny można uruchomić jednym poleceniem. Umożliwia to szybkie i łatwe wdrażanie złożonych aplikacji, które mogą zawierać bazy danych, kolejki wiadomości i inne zależności.

Praca z Dockerem to także znajomość sieci i mechanizmów przechowywania danych. Kontenery Docker domyślnie są izolowane od siebie i od hosta w zakresie sieci i systemu plików. Docker oferuje jednak różne strategie sieciowe, takie jak bridge, host, none czy overlay, które pozwalają na komunikację między kontenerami, a także z hostem. Ponadto, ważne jest zrozumienie pojęć woluminów i bind mounts, które pozwalają na trwałe przechowywanie danych poza cyklem życia kontenera.

Należy również pamiętać o bezpieczeństwie. Kontenery mogą być potencjalnym wektorem ataku, jeśli nie zostaną odpowiednio zabezpieczone. Praktyki takie jak regularne aktualizacje obrazów, minimalizacja liczby procesów działających w kontenerze, zarządzanie uprawnieniami i dostępem do kontenerów oraz monitorowanie logów to kluczowe aspekty zapewnienia bezpieczeństwa kontenerów.

Ostatnim, ale nie mniej ważnym aspektem pracy z Dockerem, jest utrzymanie i aktualizacja środowisk. Aby uniknąć problemów związanych z "drift" konfiguracji, czyli różnic między środowiskami deweloperskimi, testowymi i produkcyjnymi, warto korzystać z narzędzi CI/CD (Continuous Integration/Continuous Delivery), które umożliwiają automatyzację procesów wdrażania. Integracja Docker z popularnymi narzędziami takimi jak Jenkins, GitLab CI czy GitHub Actions pozwala na utrzymanie aktualności i spójności aplikacji w różnych środowiskach.

SPIS TREŚCI

1. DOCKER - PIERWSZE KROKI

CZYM JEST DOCKER?

Definicja i zastosowanie konteneryzacji

Docker to nowoczesna platforma do konteneryzacji, która umożliwia programistom i administratorom systemów pakowanie, dystrybucję oraz uruchamianie aplikacji w izolowanych środowiskach zwanych kontenerami. Konteneryzacja, będąca kluczowym pojęciem w świecie Docker'a, jest procesem kapsułkowania aplikacji wraz ze wszystkimi jej zależnościami w lekkim, przenośnym i samowystarczalnym kontenerze, który jest uruchamiany na pojedynczym hostowym systemie operacyjnym.

Zasadniczą zaletą konteneryzacji jest izolacja zasobów, co oznacza, że każdy kontener działa niezależnie od innych i od systemu gospodarza, posiadając własny system plików, swoje biblioteki, nie dzieląc pamięci ani innych zasobów z innymi kontenerami ani z hostem. Dzięki temu można uniknąć problemów z kompatybilnością i konfliktami między aplikacjami wynikającymi z różnic w wymaganiach dotyczących bibliotek i zależności. Konteneryzacja pozwala także na łatwe przenoszenie aplikacji pomiędzy różnymi środowiskami, od deweloperskich maszyn lokalnych po serwery produkcyjne, niezależnie od platformy.

Rozpowszechnienie się konteneryzacji, a w szczególności Docker'a, zrewolucjonizowało sposób tworzenia, wdrażania i skalowania aplikacji. Umożliwia ona deweloperom skupienie się na pisaniu kodu, bez potrzeby zajmowania się szczegółami konfiguracji środowiska uruchomieniowego. Nie bez znaczenia jest również fakt, że kontenery są znacznie lżejsze niż tradycyjne maszyny wirtualne, gdyż dzielą one kernel systemu operacyjnego hosta, a nie wymagają jego własnej, dedykowanej instancji. To sprawia, że można na pojedynczym hoście uruchomić znacznie więcej kontenerów niż maszyn wirtualnych, co jest kluczowe w kontekście efektywnego wykorzystania zasobów.

W praktyce, stosowanie konteneryzacji z Docker'em wiąże się z użyciem plików Dockerfile, które są przepisami na stworzenie obrazu kontenera. Obraz taki zawiera w sobie wszystko, co potrzebne do uruchomienia aplikacji: kod, środowisko uruchomieniowe, biblioteki, zmienne środowiskowe i pliki konfiguracyjne. Gdy

obraz zostanie zbudowany, może być on przesłany do rejestru, takiego jak Docker Hub lub prywatnego rejestru, skąd można go pobrać i uruchomić na dowolnym systemie, który obsługuje Docker'a. To znacząco ułatwia proces CI/CD (Continuous Integration/Continuous Delivery), ponieważ zmiany w kodzie mogą być automatycznie budowane, testowane i wdrażane w postaci gotowych kontenerów.

Docker również umożliwia zarządzanie wieloma kontenerami za pomocą Docker Compose, które pozwala na definicję i uruchomienie wielokontenerowych aplikacji za pomocą jednego pliku YAML. Z kolei dla złożonych aplikacji rozproszonych i mikroserwisów, Docker współpracuje z platformami orchestracji kontenerów, takimi jak Kubernetes, które automatyzują wdrożenie, skalowanie i zarządzanie kontenerami.

Stosując Docker'a, zyskuje się również większą kontrolę nad zasobami. Mechanizmy takie jak limity CPU, ograniczenia pamięci, czy też grupy kontroli dostępu (ang. access control groups) pozwalają na dokładne zarządzanie tym, jak aplikacje wykorzystują zasoby systemu gospodarza. To sprawia, że zarówno deweloperzy, jak i administratorzy systemów mają pełniejszą kontrolę nad środowiskiem uruchomieniowym aplikacji, co prowadzi do większej stabilności i przewidywalności działania systemu.

Zastosowania konteneryzacji są wszechstronne i obejmują takie obszary jak:

1. Ułatwienie i przyspieszenie procesu deweloperskiego poprzez szybkie tworzenie izolowanych środowisk deweloperskich.

2. Zapewnienie spójności środowiska na wszystkich etapach pipeline'u wytwarzania oprogramowania: od deweloperki, przez testy, aż po produkcję.

3. Możliwość łatwego skalowania aplikacji w odpowiedzi na zmieniające się obciążenie, co jest szczególnie ważne w przypadku aplikacji webowych i mikroserwisów.

4. Zwiększenie gęstości wdrożeń i wykorzystania zasobów poprzez możliwość uruchamiania większej liczby aplikacji na tej samej infrastrukturze.

5. Uproszczenie i automatyzacja procesów wdrażania i zarządzania aplikacjami za pomocą narzędzi do orchestacji kontenerów.

Na zakończenie warto zauważyć, że konteneryzacja z Docker'em otworzyła nowe możliwości w dziedzinie rozwoju oprogramowania i zarządzania aplikacjami. Uprościła wiele procesów, pozwoliła na osiągnięcie wyższej efektywności w wykorzystaniu zasobów oraz przyczyniła się do poprawy przenośności i spójności środowisk uruchomieniowych aplikacji. Zalety te sprawiają, że Docker jest obecnie jednym z najbardziej popularnych narzędzi w obszarze DevOps i kontynuuje swój rozwój w świecie szybko rozwijających się technologii chmurowych i automatyzacji.

PODSTAWOWE ZALETY UŻYCIA DOCKERA

Docker zrewolucjonizował sposób, w jaki programiści i administratorzy systemów myślą o wdrażaniu i skalowaniu aplikacji. Kluczowe zalety, które przemawiają za wykorzystaniem tego narzędzia, obejmują standardyzację środowisk, mobilność aplikacji, izolację, skalowalność, szybkość oraz łatwość zarządzania. Każdy z tych aspektów ma głęboki wpływ na cykl życia aplikacji, od developmentu po produkcję.

Standardyzacja środowisk jest jednym z najbardziej oczywistych atutów Dockera. Gdy programiści tworzą aplikacje w swoich lokalnych środowiskach deweloperskich, często zdarza się, że oprogramowanie działa na jednym komputerze, ale napotyka problemy w innym miejscu. Docker eliminuje tę przeszkodę poprzez konteneryzację, która umożliwia uruchomienie aplikacji w dokładnie takim samym środowisku niezależnie od lokalizacji. Taki kontener zawiera wszystkie niezbędne zależności i konfigurację, co gwarantuje, że aplikacja będzie działać identycznie na każdym komputerze, który obsługuje Docker.

Mobilność aplikacji to kolejny istotny atut, który wypływa z natury konteneryzacji. Kontenery Docker są przenośne pomiędzy różnymi systemami i platformami chmurowymi. To oznacza, że użytkownicy mogą łatwo przenosić aplikacje z lokalnych środowisk deweloperskich do testów, a następnie do produkcji bez potrzeby modyfikowania kodu czy konfiguracji. Zamiast zajmować się różnicami między środowiskami, zespoły mogą skupić się na rozwoju i doskonaleniu samych aplikacji.

Izolacja jest kolejną fundamentalną zaletą Dockera. Każdy kontener działa niezależnie, dzieląc jądro systemu operacyjnego hosta, ale izolując procesy aplikacji od innych kontenerów i systemu. Dzięki temu, jeśli jedna aplikacja zawiedzie, nie wpływa to na działanie innych, co jest szczególnie istotne w środowiskach, gdzie

wiele aplikacji działa na pojedynczym hoście. Izolacja zwiększa bezpieczeństwo, ponieważ ewentualne luki w jednej aplikacji nie są bezpośrednio zagrożeniem dla innych uruchomionych usług.

Skalowalność jest nieodzowną cechą w obecnych czasach, gdzie obciążenie systemów może znacząco się zmieniać. Docker wspiera skalowanie horyzontalne, czyli dodawanie więcej instancji kontenera, aby obsłużyć większe obciążenie. Taki model jest niezwykle wydajny, ponieważ kontenery można szybko uruchomić i zatrzymać, co pozwala na dynamiczne dostosowanie ilości zasobów do bieżących potrzeb. Oznacza to, że aplikacje można skalować w zależności od potrzeb, co jest szczególnie ważne dla aplikacji internetowych o zmiennej liczbie użytkowników.

Szybkość działania i wdrożeń jest kolejnym elementem, który sprawia, że Docker stał się tak popularny. Kontenery uruchamiają się błyskawicznie, ponieważ nie wymagają uruchamiania pełnego systemu operacyjnego, jak ma to miejsce w przypadku maszyn wirtualnych. Szybki start i stop kontenerów ułatwiają szybkie iteracje, testowanie oraz automatyzację procesów CI/CD (Continuous Integration/Continuous Delivery). To przyspiesza cykl życia oprogramowania, umożliwiając częstsze wypuszczanie nowych wersji aplikacji i szybsze wprowadzanie zmian.

Łatwość zarządzania jest nie mniej ważna. Docker oferuje potężne narzędzia do zarządzania cyklem życia kontenerów, w tym Docker Compose do definiowania i uruchamiania wielokontenerowych aplikacji Docker oraz Docker Swarm czy Kubernetes do orkiestracji kontenerów w większej skali. Te narzędzia umożliwiają prostą automatyzację, monitorowanie i zarządzanie kontenerami oraz usługami w nich działającymi, co znacząco upraszcza operacje, szczególnie w rozległych, rozproszonych systemach.

Przyjrzyjmy się bliżej jak Docker radzi sobie z zależnościami aplikacji, co jest częstym problemem w tradycyjnych środowiskach. W przeszłości zarządzanie zależnościami bywało skomplikowane, wymagając koordynacji między zespołami deweloperskimi, operacyjnymi i wsparcia. Kontenery Docker rozwiązują ten problem, pakując zależności wewnątrz kontenera wraz z aplikacją. Oznacza to, że wszystko, co potrzebne do uruchomienia aplikacji, jest zawarte w jednym, samowystarczalnym pakiecie. To eliminuje konieczność ręcznego konfigurowania środowiska pod każdą aplikację oraz pozwala na łatwą migrację i aktualizacje.

Zastosowanie Dockera ma także znaczący wpływ na redukcję kosztów. Konteneryzacja pozwala na znacznie lepsze wykorzystanie zasobów sprzętowych w porównaniu do maszyn wirtualnych, które wymagają dedykowanego systemu operacyjnego i zasobów dla każdej instancji. W przypadku Dockera, wiele kontenerów może współdzielić te same zasoby systemu operacyjnego hosta, co pozwala na uruchomienie większej liczby aplikacji na tym samym sprzęcie, tym samym obniżając koszty infrastruktury. Ponadto, minimalizuje to ilość wymaganej konserwacji i aktualizacji, co dodatkowo przekłada się na oszczędności czasu i pieniędzy.

Docker oferuje również zaawansowane funkcje, takie jak zarządzanie woluminami i sieciami, które umożliwiają zarządzanie danymi i komunikacją między kontenerami. Volumeny Docker pozwalają na trwałe przechowywanie danych poza cyklem życia kontenera, co jest krytyczne dla aplikacji wymagających zachowania stanu. Zarządzanie siecią umożliwia tworzenie izolowanych sieci dla kontenerów, co pozwala na bezpieczne komunikowanie się aplikacji między sobą, a także z zewnętrznym światem.

Bezpieczeństwo, choć nie jest wyłączną zaletą Dockera, zostało znacząco poprawione dzięki mechanizmom izolacji oraz regularnym aktualizacjom obrazów kontenerów. Docker pozwala na stosowanie zasad najmniejszych uprawnień, co oznacza uruchamianie aplikacji tylko z niezbędnymi uprawnieniami, minimalizując tym samym ryzyko ataków. Ponadto, możliwość szybkiego aktualizowania kontenerów w przypadku wykrycia luk bezpieczeństwa w oprogramowaniu znacząco skraca czas reakcji na potencjalne zagrożenia.

Reasumując, Docker jest technologią, która zapewnia znaczącą elastyczność i efektywność w procesie rozwoju, wdrażania i zarządzania aplikacjami. Korzyści płynące z jego użycia są różnorodne i mają szeroki wpływ na różne aspekty infrastruktury IT, co czyni go atrakcyjnym rozwiązaniem dla firm każdej wielkości. Umożliwienie tworzenia spójnych i izolowanych środowisk dla aplikacji, szybkie wdrożenia, łatwa skalowalność, efektywność kosztowa i silne wsparcie dla procesów DevOps to tylko niektóre z powodów, dla których Docker stał się tak ważnym narzędziem w nowoczesnej informatyce.

PRZEGLĄD EKOSYSTEMU DOCKERA

Docker to nie tylko samodzielna aplikacja, ale cały ekosystem narzędzi, usług i produktów, które wspólnie tworzą kompleksową platformę do tworzenia, uruchamiania i zarządzania kontenerami. Ekosystem Dockera obejmuje oryginalne narzędzie Docker Engine, narzędzia CLI takie jak Docker Compose, Docker Swarm, jak również usługi dostępne w chmurze takie jak Docker Hub i Docker Cloud.

Rozpocznijmy od Docker Engine, serca ekosystemu. Jest to aplikacja serwerowa, która używa natywnych technologii systemu operacyjnego, takich jak cgroups i namespaces w Linuxie, do izolowania procesów. Docker Engine działa w tle na maszynie hosta i zajmuje się budowaniem, uruchamianiem oraz dystrybucją kontenerów. Umożliwia on użytkownikom interakcję z kontenerami przy użyciu Docker API lub za pomocą wiersza poleceń poprzez Docker CLI.

Następnie mamy Docker CLI, zestaw poleceń wiersza poleceń, które pozwalają użytkownikom na interakcję z Docker Engine. CLI umożliwia wykonywanie wielu różnych operacji, takich jak budowanie obrazów kontenerów z plików Dockerfile, uruchamianie kontenerów z tych obrazów, przesyłanie obrazów do repozytorium, pobieranie obrazów, zarządzanie kontenerami i obrazami oraz wiele więcej.

Docker Compose to kolejne narzędzie w ekosystemie, które ułatwia definiowanie i uruchamianie aplikacji wielokontenerowych. Używa plików YAML do konfiguracji usług aplikacji, sieci i woluminów. Compose pozwala na uruchomienie całej aplikacji zdefiniowanej w jednym lub więcej plikach Compose jednym poleceniem, co znacznie upraszcza procesy deweloperskie i automatyzację.

Docker Swarm to narzędzie do klastrowania i orkiestracji kontenerów, które umożliwia zarządzanie grupą maszyn Docker Engine i wdrażanie aplikacji na nich jako usług. Swarm wykorzystuje standardowe API Dockera, co oznacza, że narzędzia, które już działają z Dockerem, mogą działać z Swarmem bez zmian. Umożliwia to łatwe skalowanie aplikacji w kontenerach poprzez dodawanie więcej maszyn do klastra.

Docker Hub jest publicznym repozytorium obrazów kontenerów, które pozwala na ich udostępnianie. Docker Hub oferuje także zautomatyzowane buildy, które automatycznie budują obrazy z kodu źródłowego przechowywanego w

repozytoriach, takich jak GitHub czy Bitbucket. Jest to ogromnie przydatne, gdy chcemy szybko udostępnić nasze obrazy innym użytkownikom lub zautomatyzować proces wdrażania aplikacji.

Docker Cloud to usługa zapewniająca hostowany rejestr, który umożliwia współpracę zespołów, automatyzację wdrażania, oraz zarządzanie kontenerami na dużą skalę. Z Docker Cloud użytkownicy mogą zarządzać swoimi kontenerami poprzez interfejs webowy lub API, co sprawia, że zarządzanie skomplikowanymi aplikacjami staje się łatwiejsze.

Również Docker Store, podobnie jak Docker Hub, umożliwia znalezienie i udostępnianie obrazów kontenerów, ale skupia się na certyfikowanych i zaufanych zawartościach, co jest ważne dla przedsiębiorstw szukających gwarancji bezpieczeństwa i wsparcia dla używanych obrazów.

Nie można zapominać o Docker for Mac i Docker for Windows, które to narzędzia umożliwiają łatwą instalację i uruchomienie środowiska Docker na komputerach Mac i Windows. Te wersje Docker są dostosowane do specyfiki systemów operacyjnych, co oznacza, że użytkownicy tych systemów mogą korzystać z Dockera bez konieczności konfiguracji dodatkowych maszyn wirtualnych czy środowisk Linux.

W ekosystemie Dockera istotne jest też pojęcie Dockerfile, czyli tekstowego dokumentu zawierającego wszystkie polecenia, które użytkownik mógłby wywołać w wierszu poleceń, aby zbudować dany obraz. Dockerfile jest jak przepis na stworzenie obrazu kontenera i często jest częścią kodu źródłowego aplikacji w systemach kontroli wersji.

Pod kątem zaawansowanego zarządzania i orkiestracji kontenerów, Docker integruje się także z narzędziami zewnętrznymi takimi jak Kubernetes, który jest obecnie standardem w branży do zarządzania kontenerami w produkcji. Choć Kubernetes nie jest produktem stworzonym przez Docker Inc., to obie technologie współpracują ze sobą, dając użytkownikom większą elastyczność i możliwości w zarządzaniu skomplikowanymi, rozproszonymi aplikacjami.

Ekosystem Dockera jest wciąż w dynamicznym rozwoju, a społeczność wokół tej technologii aktywnie pracuje nad nowymi narzędziami i ulepszeniami. Regularne

aktualizacje, wsparcie społeczności, jak i inwestycje ze strony przedsiębiorstw w integrację i rozwój produktów opartych o Docker świadczą o jego stabilności i kluczowej roli w świecie nowoczesnego rozwoju oprogramowania oraz operacji IT. Warto na bieżąco śledzić nowinki w ekosystemie Dockera, gdyż w dynamicznym świecie technologii kontenerowej, innowacje pojawiają się bardzo często, oferując nowe możliwości i rozwiązania dla programistów, operatorów i przedsiębiorstw na całym świecie.