Docker a kontejnerizace

Kontejnerizace je metoda virtualizace na úrovni operačního systému, která umožňuje zabalit aplikaci a všechny její závislosti (knihovny, konfigurační soubory, frameworky) do jednoho izolovaného balíčku zvaného **kontejner**. Tento přístup zaručuje, že aplikace poběží naprosto stejně a spolehlivě bez ohledu na to, zda je spuštěna na vývojářově notebooku, testovacím serveru nebo v produkčním cloudu.

Nejznámější a nejrozšířenější platformou pro vytváření a správu kontejnerů je **Docker**. Ačkoliv Docker nebyl první technologií svého druhu (předcházely mu např. LXC kontejnery v Linuxu), díky své jednoduchosti a vývojářsky přívětivému nástrojovému vybavení způsobil v roce 2013 naprostou revoluci ve vývoji softwaru (tzv. DevOps kultura) a vyřešil letitý problém: *„U mě na počítači to funguje!“*

Princip: Kontejnery vs. Virtuální stroje (VM)

Pro pochopení geniality kontejnerů je nutné je porovnat s tradičními virtuálními stroji.

V klasické virtualizaci (např. VMware, VirtualBox) sedí na fyzickém hardwaru vrstva zvaná **Hypervizor**. Pokud chcete spustit tři izolované aplikace, musíte vytvořit tři virtuální stroje. Každý tento stroj musí obsahovat svůj vlastní, plnohodnotný operační systém (Guest OS). To znamená obrovskou redundanci – systém spotřebuje gigabyty RAM a desítky gigabytů na disku jen pro samotný běh tří kopií stejného operačního systému.

[Image of Docker container architecture vs Virtual Machine]

Kontejnery fungují odlišně. Nesimulují celý hardware, ale sdílejí jádro (Kernel) hostitelského operačního systému. Docker Engine běží přímo na hostitelském OS a každému kontejneru přiděluje izolovaný prostor (pomocí linuxových funkcí *Namespaces* a *Cgroups*). Kontejner tak obsahuje pouze samotnou aplikaci a nezbytné knihovny. Výsledkem je, že kontejner má velikost v řádech megabytů a nastartuje za zlomek sekundy.

Architektura a klíčové pojmy Dockeru

Ekosystém Dockeru se skládá z několika základních stavebních kamenů, které na sebe logicky navazují:

1. Dockerfile

Jednoduchý textový soubor obsahující sekvenci příkazů (recept), jak se má aplikace sestavit. Definuje výchozí bod (např. čistý operační systém Alpine Linux nebo prostředí Pythonu), kopíruje zdrojové kódy, instaluje závislosti a určuje, jaký příkaz se má spustit při startu.

2. Docker Image (Obraz)

Když spustíte sestavení (build) nad Dockerfilem, vznikne Image. Je to statická, **pouze pro čtení (read-only)** šablona s vaší aplikací. Image je složen z vrstev, což umožňuje efektivní sdílení dat na disku (pokud máte 10 obrazů založených na stejném základu Ubuntu, stáhne se a uloží Ubuntu jen jednou).

3. Docker Container (Kontejner)

Kontejner je běžící instance Docker Image. Můžete si to představit jako objekt vytvořený ze třídy v programování. Z jednoho obrazu můžete spustit tisíce identických kontejnerů. Kontejner má svou vlastní izolovanou paměť, souborový systém a síťové rozhraní.

4. Docker Registry / Docker Hub

Centrální repozitář (katalog) pro ukládání a sdílení Docker Images. Funguje podobně jako GitHub pro zdrojové kódy. Docker Hub je obří veřejná knihovna, odkud si můžete stáhnout hotové a oficiální obrazy databází (MySQL, PostgreSQL), webových serverů (Nginx, Apache) nebo analytických nástrojů.

Výhody a nevýhody kontejnerizace

Výhody

- **Absolutní přenositelnost (Portability):** Pokud kontejner funguje u vás, bude fungovat kdekoliv, kde běží Docker.
- **Rychlost:** Spuštění kontejneru netrvá minuty jako start Windows/Linuxu ve virtuálním stroji, ale sekundy či milisekundy.
- **Hustota a efektivita:** Na jednom serveru můžete spustit stovky kontejnerů místo jednotek virtuálních strojů, čímž drasticky šetříte náklady na hardware.
- **Verzování a rollback:** Vždy víte, z jaké verze obrazu kontejner běží. Pokud nová verze aplikace obsahuje chybu, bleskově spustíte kontejner ze staršího obrazu.

Nevýhody

- **Bezpečnostní rizika sdíleného jádra:** Protože všechny kontejnery sdílejí jeden kernel hostitelského OS, kritická zranitelnost v jádře může teoreticky umožnit útočníkovi „uniknout“ z kontejneru a ohrozit ostatní.
- **Efemérní (pomíjivá) povaha:** Data uvnitř kontejneru jsou zničena, jakmile kontejner smažete. Pro trvalé uložení dat (např. u databází) je nutné složitěji mapovat tzv. **Docker Volumes** (svazky) na hostitelský disk.
- **Není to na všechno:** Monolitické desktopové aplikace s bohatým grafickým rozhraním (GUI) nejsou pro kontejnerizaci vhodné.

Orchestrace: Když už Docker nestačí

Pokud provozujete jednotky kontejnerů, postačí vám základní nástroj **Docker Compose**. Pokud však stavíte rozsáhlou architekturu mikroslužeb (Netflix, Spotify, banky) a potřebujete spravovat, aktualizovat a load-balancovat desetitisíce kontejnerů rozprostřených přes stovky serverů, samotný Docker je bezmocný.

Pro tyto účely se používají **Orchestrátory**. Absolutním průmyslovým standardem se stal systém **Kubernetes (K8s)** (původně vyvinutý společností Google), který se stará o to, aby vaše kontejnery neustále běžely, automaticky je restartuje při pádu a přidává další v případě náhlé špičky v návštěvnosti.

Rychlé srovnání: Kontejner vs. Virtuální stroj

Vlastnost	**Virtuální stroj (VM)**	**Kontejner (Docker)**
Izolace	Úplná (Hardwarová)	Částečná (Na úrovni OS)
Operační systém	Každá VM má vlastní těžký Guest OS	Všechny sdílejí jádro Host OS
Velikost na disku	Gigabyty (GB)	Megabyty (MB)
Doba spuštění	Minuty	Sekundy / Milisekundy
Využití zdrojů	Extrémně náročné	Vysoce efektivní
Typické nasazení	Běh odlišných OS na jednom serveru	Mikroslužby, CI/CD pipeline, webové appky