Docker je platforma pro správu kontejnerizovaných aplikací, které umožňují izolaci prostředí a přenositelnost mezi různými systémy.

Orchestrace kontejnerů, jako Kubernetes a Docker Swarm, se stará o automatizaci nasazení, škálování a správu kontejnerů.

Grafika se dělí na rastrovou (pixelovou) a vektorovou; převody mezi formáty, jako JPEG, PNG (rastrová) a SVG, EPS (vektorová), jsou běžné, přičemž rastrovou grafiku lze převádět mezi formáty snadno, zatímco převod na vektorovou je složitější.

NoSQL databáze zahrnují různé typy, jako klíč-hodnota (Redis), dokumentové (MongoDB), sloupcové (Cassandra) a grafové (Neo4j), které jsou flexibilní, škálovatelné a ideální pro práci s velkými objemy nestrukturovaných dat.

**Lehká virtualizace a Docker**

Lehká virtualizace je technologie, která umožňuje provozovat izolovaná prostředí v rámci jednoho operačního systému. Oproti plné virtualizaci nepoužívá hypervizor a nenapodobuje celý operační systém, ale sdílí jádro s hostitelem, čímž dosahuje vyšší efektivity a nižší režie.

**Výhody oproti plné virtualizaci:**

1. Nižší nároky na výkon a paměť
2. Rychlejší spouštění a restart kontejnerů
3. Snadnější správa, nasazení a škálování aplikací
4. Menší režie oproti plné virtualizaci

**Nevýhody oproti plné virtualizaci:**

1. Menší izolace mezi kontejnery (problémy se zabezpečením při špatné konfiguraci)
2. Omezená podpora různých operačních systémů (všechny kontejnery sdílí jádro hostitele)
3. Závislost na hostitelském jádře – omezení pro některé aplikace

**Základní pojmy Dockeru:**

* **Docker Image** – šablona obsahující všechny potřebné součásti pro běh aplikace (knihovny, závislosti, konfigurační soubory)
* **Dockerfile** – skript definující kroky pro vytvoření Docker image (např. instalační příkazy, kopírování souborů, nastavení proměnných prostředí)
* **Docker Kontejner** – běžící instance Docker image, izolovaná a samostatná jednotka s vlastními procesy a závislostmi

**Orchestrace kontejnerů a Kubernetes:**

* Nástroje pro orchestraci kontejnerů, jako je **Kubernetes**, slouží k automatizaci správy kontejnerů, jejich nasazení, škálování a monitorování. Umožňují lepší využití zdrojů a zajišťují vysokou dostupnost aplikací.

**Funkce Kubernetes:**

* Automatické škálování aplikací podle vytížení
* Load balancing pro efektivní rozdělení provozu mezi kontejnery
* Automatická obnova selhaných kontejnerů
* Možnost správy kontejnerů napříč více servery
* Správa síťové komunikace mezi kontejnery
* Kubernetes umožňuje běh rozsáhlých kontejnerových aplikací v cloudových i on-premise prostředích, což je dnes klíčové pro moderní IT infrastrukturu.

**Převody jednotlivých typů a druhů 2D a 3D vektorové i rastrové grafiky mezi sebou**

* Grafika se dělí na **rastrovou** a **vektorovou**, přičemž každý typ má své výhody a omezení. Možnosti převodů mezi těmito formáty závisí na účelu a požadované kvalitě výstupu.

**Převod rastrové grafiky na vektorovou:**

* Použití automatizovaného trasování (např. Adobe Illustrator, Inkscape)
* Manuální překreslení pomocí vektorových nástrojů
* Výhody: Možnost neomezeného škálování bez ztráty kvality
* Nevýhody: Ztráta detailů při automatizovaném trasování, vyšší pracnost u složitých obrázků

**Převod vektorové grafiky na rastrovou:**

* Export do rastrového formátu (např. PNG, JPEG, BMP, TIFF)
* Možnost nastavení rozlišení pro zachování kvality
* Výhody: Přesný výstup vhodný pro tisk a web
* Nevýhody: Ztráta možnosti škálování bez ztráty kvality

**Převod mezi 2D a 3D grafikou:**

* Konverze 2D obrázků na 3D modely pomocí software (např. Blender, Adobe Dimension)
* Použití výškových map pro generování 3D objektů
* Možnost extruze vektorových tvarů do 3D prostoru
* Výhody: Větší flexibilita v designu a vizualizacích
* Nevýhody: Náročnost na výpočetní výkon, potřeba přizpůsobení pro různé použití

**Převody mezi různými formáty grafiky:**

* **Rastrové formáty:** PNG (bezztrátový), JPEG (ztrátová komprese), TIFF (pro profesionální tisk)
* **Vektorové formáty:** SVG (web), AI (Adobe Illustrator), EPS (profesionální tisk)
* **3D formáty:** OBJ, STL (3D tisk), FBX (animace, herní enginy)

Výběr správného formátu závisí na účelu použití a požadované kvalitě výstupu

**SQL vs. NoSQL databáze**

SQL (Structured Query Language) byl vyvinut v 70. letech společností IBM jako standard pro správu relačních databází. Jeho široké přijetí vedlo ke vzniku mnoha komerčních a open-source databází.

**Základní rozdíl mezi SQL a NoSQL databázemi:**

* SQL databáze jsou relační, používají tabulky se strukturou (schématem) a dotazovací jazyk SQL.
* NoSQL databáze jsou nestrukturované, nevyžadují pevně definované schéma a umožňují větší flexibilitu v ukládání dat.

**Proč jsou SQL databáze častější?**

1. Dobře zavedený standard, široká podpora a dokumentace
2. Silná integrita dat díky ACID vlastnostem (Atomicita, Konzistence, Izolace, Trvalost)
3. Vhodné pro komplexní dotazy a transakční systémy
4. Náklady na vytváření databází jsou vysoké, proto se využívají zavedené standardy

**Datové modely NoSQL databází:**

* Klíč-hodnota (Key-Value) – jednoduchá struktura podobná hash tabulkám (např. Redis, Riak)
* Sloupcové úložiště – ukládání dat ve sloupcích namísto řádků (např. Apache Cassandra, HBase)
* Dokumentové úložiště – data jsou uložena ve formátu JSON nebo BSON (např. MongoDB, CouchDB)
* Grafové databáze – data jsou reprezentována jako graf s uzly a hranami (např. Neo4j, ArangoDB)

**Kdy je NoSQL lepší volbou?**

1. Big Data a distribuované systémy – zvládají velké objemy dat a horizontální škálování
2. Reálné aplikace v cloudu – flexibilita bez pevného schématu (např. Firebase, DynamoDB)
3. Sociální sítě a analytika – grafové databáze jsou ideální pro mapování vztahů a analýzu dat

NoSQL databáze jsou tedy vhodné pro specifická použití, kde je důležitá škálovatelnost, výkon a flexibilita.