

DevOps, 2. časť: Nasadenie v prostredí cloudu, orchestrácia služieb, stratégie a best practices

adam.puskas@uxtweak.com

Vývoj progresívnych webových aplikácií

Lektor: Ing. Adam Puškáš

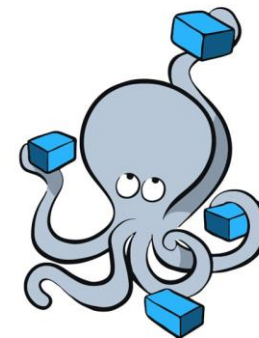
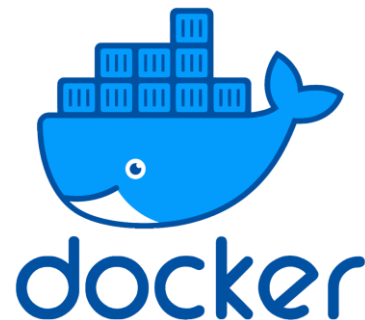
Vedúci kurzu: Ing. Eduard Kuric, PhD.

29.11.2022

Na minulej prednáške...

- Mikroslužby ako architektúra moderných webových aplikácií
 - Kohéznosť, zapuzdrenosť, nasaditeľnosť, škálovateľnosť, testovateľnosť...
- Hardvérové architektúry - x86 vs. ARM
- Virtualizácia a virtuálne stroje - flexibilita
- Softvérové kontajnery \leftrightarrow mikroslužby
- Docker - vývoja, ladenie, nasadzovanie softvérových kontajnerov

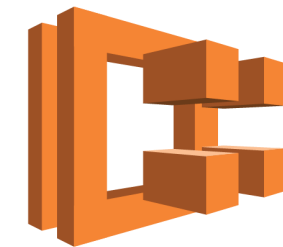
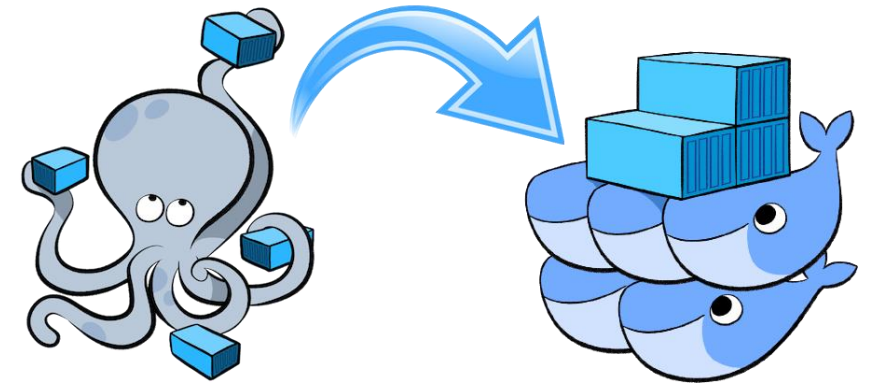
vmware®



docker
Compose

Agenda dnešnej prednášky

- Nasadenie v prostredí cloudu - úvod, terminológia
- Orchestrácia softvérových kontajnerov (služieb)
 - **Docker Swarm**
 - Kubernetes (úvod)
- Stratégie nasadzovania a best practices
 - Amazon Web Services
 - Nasadenie na AWS a škálovanie (AWS ECS)



AWS ECS

Cloud computing

- Realizácia výpočtov a uloženie dát vo vzdialenom centre (cloud)
- Poskytovanie IT infraštruktúry prostredníctvom siete Internet
- Vlastnosti:
 - **Agilita** (agility) - prístup k širokému spektru služieb podľa potreby:
 - Výpočty (compute)
 - Úložisko dát (block / object storage)
 - Relačná databáza (SQL)
 - Logovanie a analytika (logging and analytics)
 - Strojové učenie (machine learning)
 - Bezpečnosť (firewall) a mnohé iné



Cloud computing /2

- Vlastnosti cloud computingu (pokrač.):
 - **Elasticita** (elasticity) - flexibilné škálovanie prostriedkov podľa biznis potreby
 - **Abstrakcia** (abstraction) - vysokoúrovňové rozhranie nad výpočtovými prostriedkami - rýchlejšie a jednoduchšie nasadenie (v porovnaní s “bare-metal”)
 - využívajú sa virtuálne stroje (VM) a softvérové kontajnery
 - **Samoobsluha** (self-service) - cloud poskytuje samoobslužný prístup k infraštruktúre prostriedkov (IaaS)
 - **Cenová flexibilita** (cost flexibility) - kontrola nad výdavkami na jemnejšej úrovni granularity (často model “pay-as-you-go”)

Cloud z pohľadu organizácie dátového centra

- Public cloud:
 - Prostriedky poskytovateľa cloudu (napr. AWS) sú **zdieľané** medzi klientov cloudu (firmy využívajúce AWS ako poskytovateľa)
 - Napr. 1 fyzický stroj je pomocou virtualizácie rozdelený medzi viacerých klientov
- Private cloud:
 - Prostriedky poskytovateľa cloudu sú **vyhradené** pre jedného klienta
 - Ak je to potrebné z pohľadu bezpečnosti (legislatívy), výkonu, špec. potrieb
- On-premise cloud:
 - Cloudové riešenie je nasadené priamo **u klienta**
 - Poskytuje sa softvérová platforma na nasadenie, napr. licencovaný DMS

Cloud z pohľadu typu poskytovanej služby

- **Software-as-a-Service (SaaS)**
 - Poskytuje sa hotový produkt, vyvíjaný, **nasadený a prevádzkovaný jeho poskytovateľom** (napr. DMS hostovaný firmou, ktorá ho vyvíja)
- **Platform-as-a-Service (PaaS)**
 - Poskytuje sa vzdialená platforma, umožňujúca napr. nasadiť vlastnú aplikáciu pre klientov **bez nutnosti správy podpornej infraštruktúry** (napr. AWS LightSail)
 - Nemusím priamo riešiť konfiguráciu siete, virtuálnych strojov, Dockra...
- **Infrastructure-as-a-Service (IaaS)**
 - Poskytujú sa vzdialené prostriedky v podobe **blokov pre vyskladanie si vlastnej infraštruktúry** (napr. Azure / AWS / GCP - výpočty, úložisko, sieť, monitoring...)

Cloud computing - zhrnutie a kontexty využitia

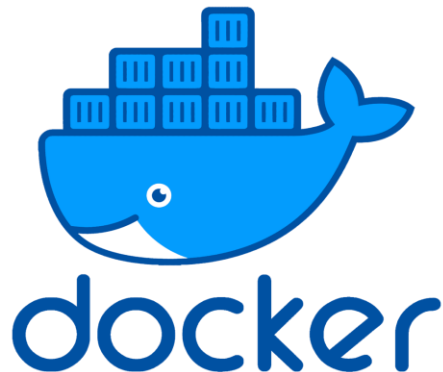
- Public cloud + PaaS (Platform-as-a-Service):
 - Malý tím, ktorý má málo, resp. žiadne skúsenosti s (Dev)Ops
 - “Odbremenie” sa od infraštruktúry, sústredenie sa na **vývoj produktu**
 - Nevýhody: menšia kontrola nad nákladmi, PaaS časom nemusí stačiť...
- Public / private cloud + IaaS (Infrastructure-as-a-Service):
 - Stredne veľký až väčší (DevOps) tím, ktorý dokáže spravovať infraštruktúru
 - **Agilita** využívania služieb, **elasticita** škálovania, cenová **flexibilita**...
 - Nevýhoda: s narastajúcimi potrebami tímu rapídne rastú náklady

Cloud computing - zhrnutie a kontexty využitia /2

- On-premise cloud
 - Veľká, nadnárodná spoločnosť, príp. štátna inštitúcia - sama sa stáva **poskytovateľom cloudu** (pre interných / externých klientov)
 - Maximálna kontrola nad prevádzkovanou infraštruktúrou
 - Nevýhody - nižšia flexibilita a agilita, vysoká prvotná investícia, bezpečnostné implikácie
- Aj v kontexte cloud computingu je veľmi dôležité zvážiť **kontext využitia**
 - Veľkosť a skúsenosti tímu, očakávania rastu, aktuálne a budúce potreby...

Orchestrácia softvérových kontajnerov (služieb)

- Architektúra založená na mikroslužbách:
 - Systém môže pozostávať z desiatok (stoviek) mikroslužieb
 - Každá mikroslužba môže mať desiatky (stovky) inštancií - replík (podľa potreby)
 - Ako takýto systém efektívne koordinovať?
- **Orchestrácia** = správa, koordinácia, konfigurácia, nasadenie a škálovanie systému na báze mikroslužieb (softvérových kontajnerov)



+

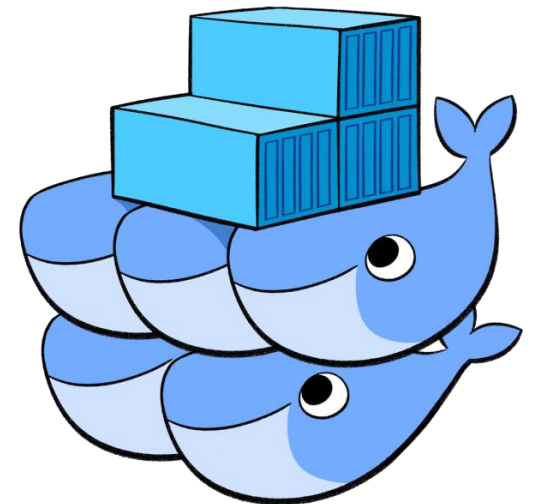


Orchestrácia softvérových kontajnerov (služieb) /2

- Centralizovaný proces, zabezpečuje ho **orchestrátor**
 - “Master” nodes však môže byť viac (prevencia “single point of failure”)
- Orchestrátor je komplement, nie náhrada softvérových kontajnerov
 - Napr. Kubernetes môže spravovať Docker kontajnery, ale tiež kontajnery na báze LXC / containerd a pod.
- Známe orchestrátory (orchestration engines):
 - **Docker Swarm**
 - **AWS ECS**
 - Kubernetes

Orchestrácia kontajnerov - Docker Swarm

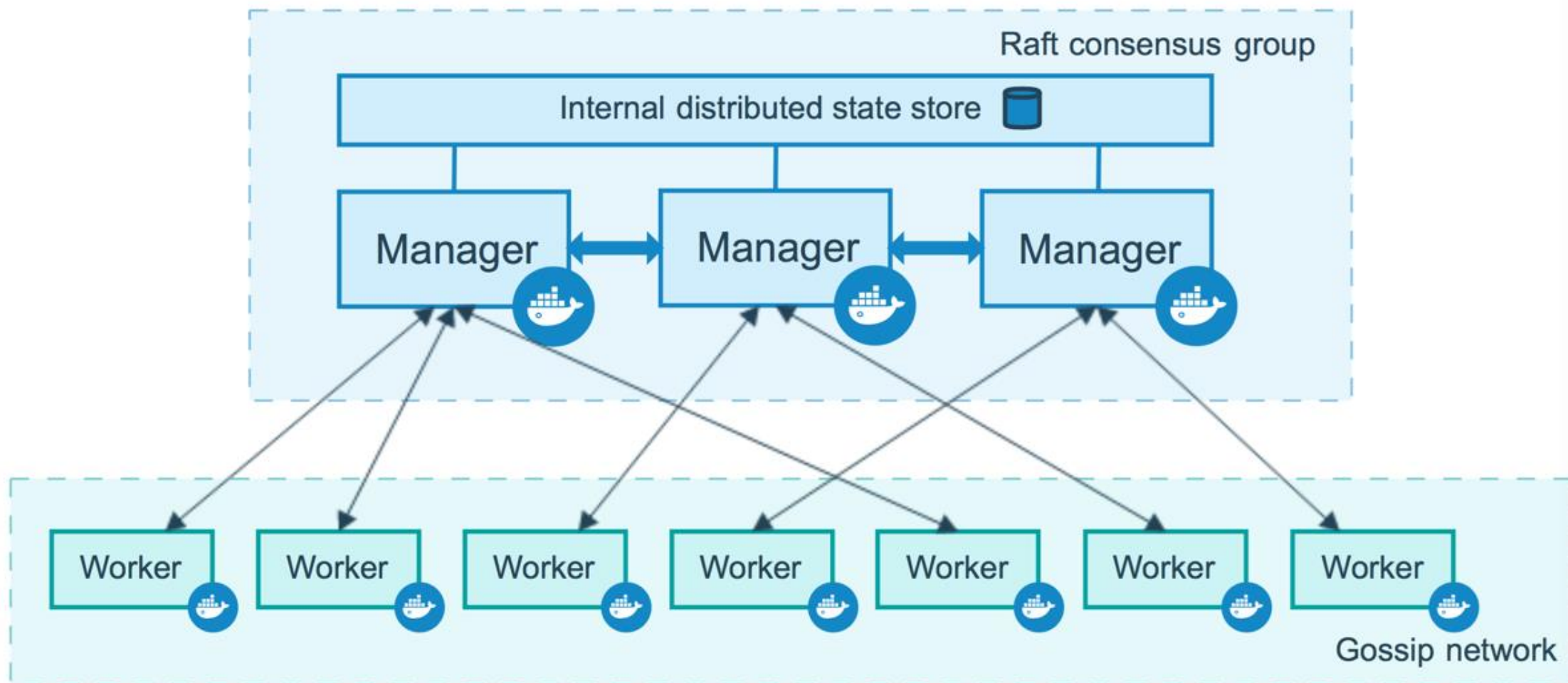
- Jednoduchý a výkonný **orchestrátor softvérových kontajnerov**
- Priama súčasť Docker Engine (“Swarm mode”)
- Základné koncepty:
 - Tzv. swarm pozostáva z 1 - N uzlov (**nodes**)
 - Uzol môže byť riadiaci (**manager**), pracovný (**worker**), príp. oboje
 - Prechod od kontajnerov k službám (**services**)
 - Služba má 1 - N úloh (**replicas / tasks**)
 - Docker udržiava želaný stav služieb
 - Spúšťanie, reštartovanie, škálovanie...



Docker Swarm - Uzol (node)

- Uzol (node)
 - Inštancia Docker engine, súčasť “swarmu”
 - Rola manažéra (**manager node**) - **plánuje a deleguje úlohy** (tasks / replicas) na dostupné worker nodes
 - Rola pracovníka (**worker node**) - **vykonáva úlohy** pridelené manager nodes, reportuje stav úloh
 - Manager node **distribuuje repliky úloh** medzi worker nodes podľa potreby
 - Manager node zabezpečuje orchestráciu, škálovanie a koordináciu v klastri (cluster)

Docker Swarm - Uzol (node) /2



Čítajte viac: <https://docs.docker.com/engine/swarm/how-swarm-mode-works/nodes/>

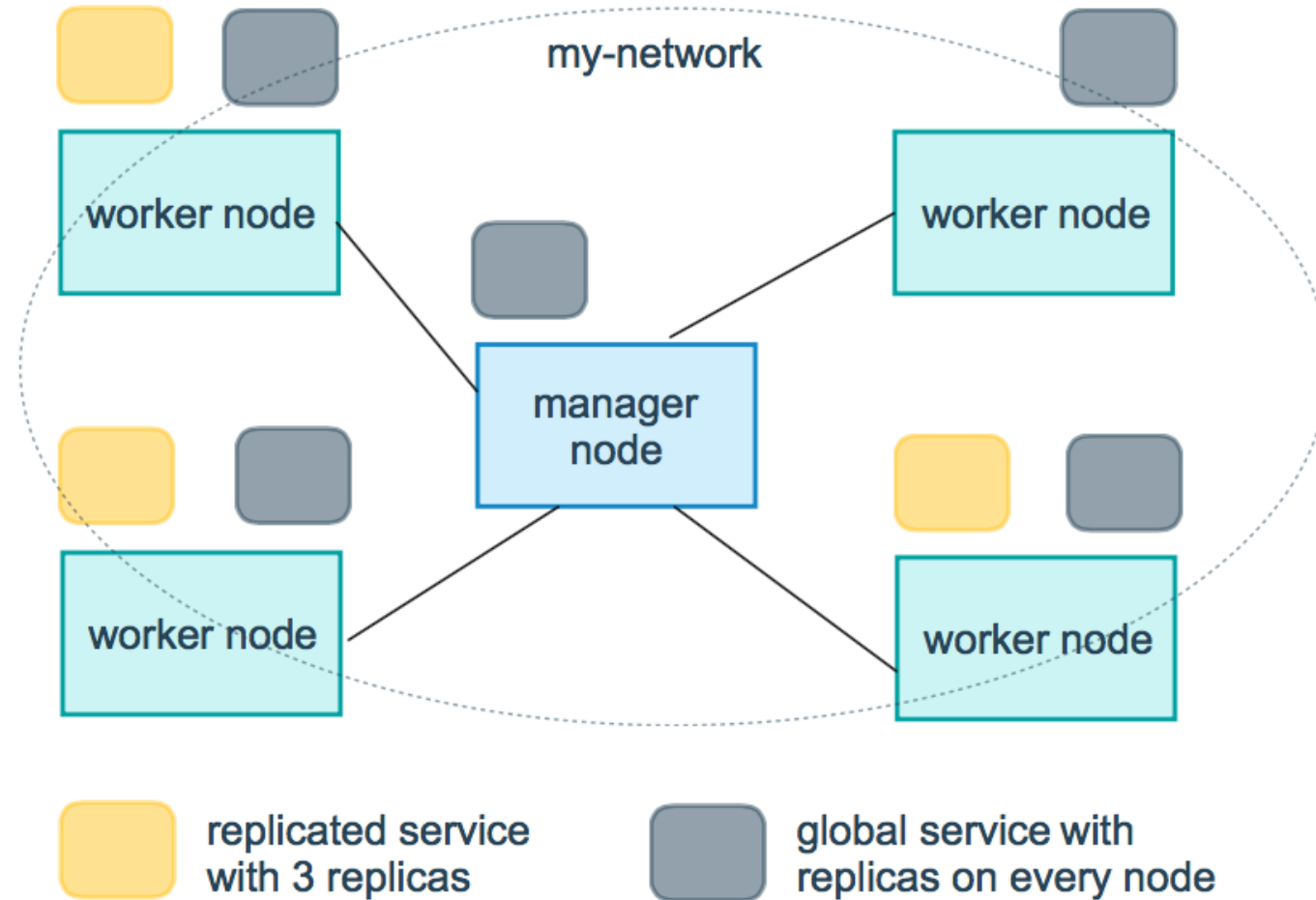
Docker Swarm - Úlohy a služby (tasks and services)

- Úloha (task):
 - **Inštancia Docker kontajnera**
 - Umiestnená na niektorý worker node podľa pokynu manager node
 - Má svoj **stav** - bežiaca (running), zlyhaná (failed) - worker node ho reportuje manager node
- Služba (service):
 - Definícia služby na vyššej úrovni abstrakcie - **skladá sa z úloh** (tasks / replicas)
 - Koreňová (root) jednotka, ktorú **definujeme v konfigurácii** Docker Swarm

Docker Swarm - Úlohy a služby (tasks and services) /2

- Služba (service) (pokrač.):
 - Konfigurácia služby v štýle docker-compose:
 - Docker image
 - Premenné prostredia
 - Volumes
 - ...
 - Môže bežať v móde repliky (replicated service) alebo globálne (global service)
 - **Replicated service** = špecifikujeme **počet replík**, ktoré manager node distribuuje na dostupné worker nodes
 - **Global service (démon)** = **každý** dostupný node spustí **práve 1 inštanciu** úlohy

Docker Swarm - Úlohy a služby (tasks and services) /3



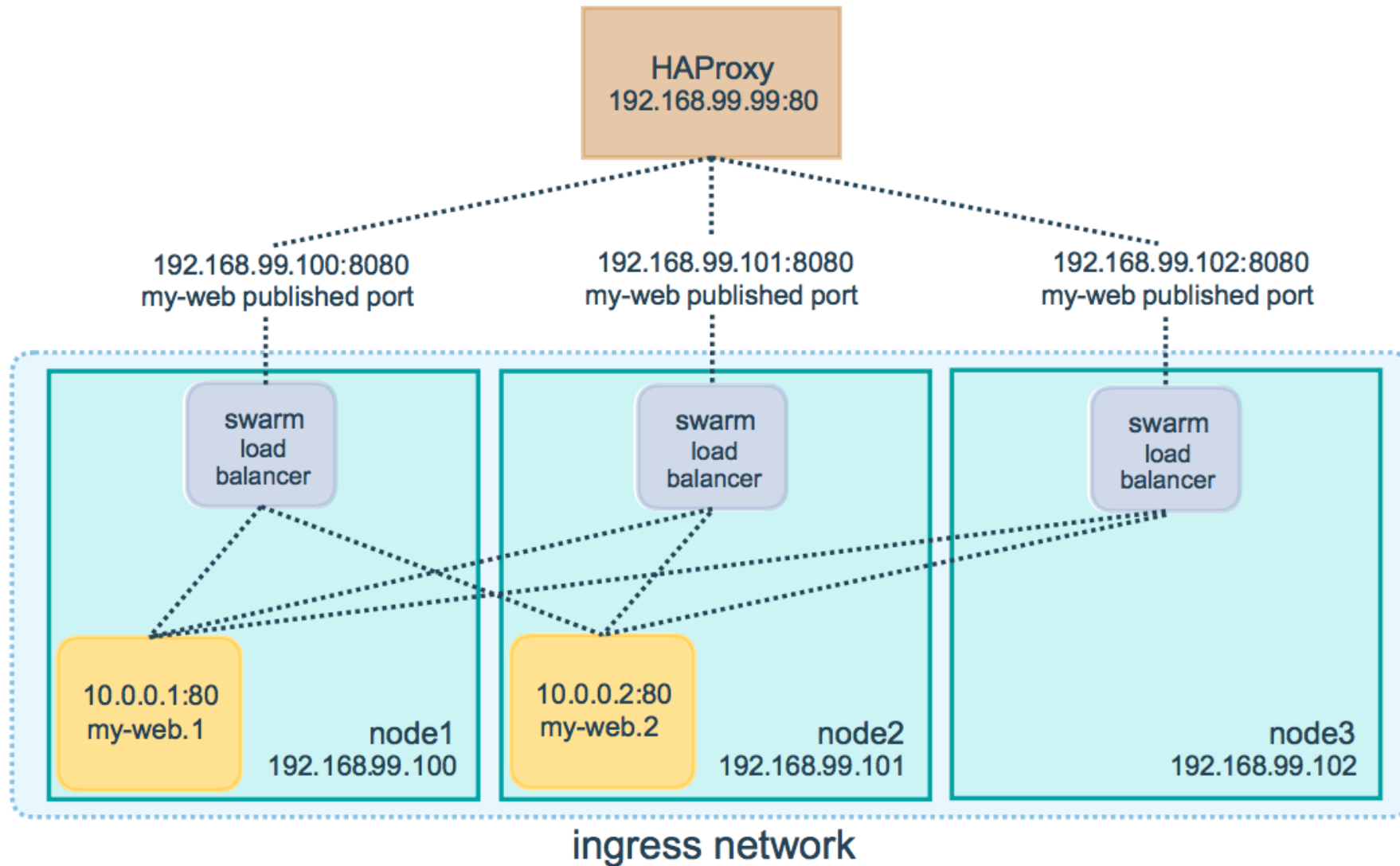
Docker Swarm - Úlohy a služby (tasks and services) /4

- Služba (service) (pokrač.):
 - Má pridelené **systémové prostriedky** (na úrovni inštancie) - limit pre CPU a RAM
 - **Rezervácia (reservations)** - “mäkký” limit (hodnotu možno presiahnuť)
 - **Limit** - “tvrdý” limit (po presiahnutí bude úloha ukončená)
 - Má nastavenú **politiku reštartovania**
 - no, on-failure, always, unless-stopped
 - Má nastavené **obmedzenia pre umiestnenie** (placement constraints)
 - Podľa role: `node.role == manager`
 - Podľa hostname: `node.hostname == a01.myapp.dev`
 - ...

Docker Swarm - distribúcia záťaže (load balancing)

- Využíva sa tzv. **ingress load balancing**:
 - Definujeme port pre vystavenie služby (PublishedPort)
 - Alebo manager node **pridelí port automaticky** (z rozsahu 30000-32767)
- Služba je vďaka tomuto mechanizmu prístupná na definovanom porte z **akéhokoľvek node** v rámci Swarm klastra
 - Dokonca aj v prípade, ak na danom node úloha (replika) služby nebeží
- Každá služba má pridelený **DNS záznam** (service discovery)
 - Volanie služby podľa DNS mena - distribúcia záťaže medzi nodes
- Ingress sieť umožňuje jednoduchú integráciu s **externými load balancerami**

Docker Swarm - distribúcia záťaže (load balancing) /2



Docker Stack

- Nadstavba Docker Swarm - jednoduchá inicializácia **Swarm klastra** pomocou konfiguračného YAML súboru
- Syntax je nadstavbou docker-compose.yml
 - Fungovať bude aj bežný docker-compose.yml
- Direktíva **deploy**:
 - **Mód** nasadenia služby (mode: global / replicated), **počet replík** (replicas)
 - Politika reštartovania (restart_policy)
 - Pridelenie prostriedkov (resources)
 - Obmedzenia pre umiestnenie (placement constraints)

Docker-compose.yml - ukážka

version: '3.5'

services:

wordpress:

image: arm64v8/wordpress

restart: unless-stopped

ports:

- 8080:80

environment:

HOST: 0.0.0.0

volumes:

- wordpress:/var/www/html

volumes:

wordpress:

name: wordpress-volume

Docker Stack YAML - ukážka

version: '3.5'

services:

wordpress:

image: arm64v8/wordpress

ports:

- 8080:80

environment:

HOST: 0.0.0.0

volumes:

- wordpress:/var/www/html

volumes:

wordpress:

name: wordpress-volume

driver: local

driver_opts:

type: nfs

o: nfsvers=4,addr=192.168.1.120,rw

device: "/mnt/disk1"

deploy:

mode: replicated

replicas: 6

update_config:

parallelism: 2

delay: 10s

restart_policy:

condition: on-failure

resources:

limits:

cpus: "0.8"

memory: "150M"

reservations:

cpus: "0.2"

memory: "50M"

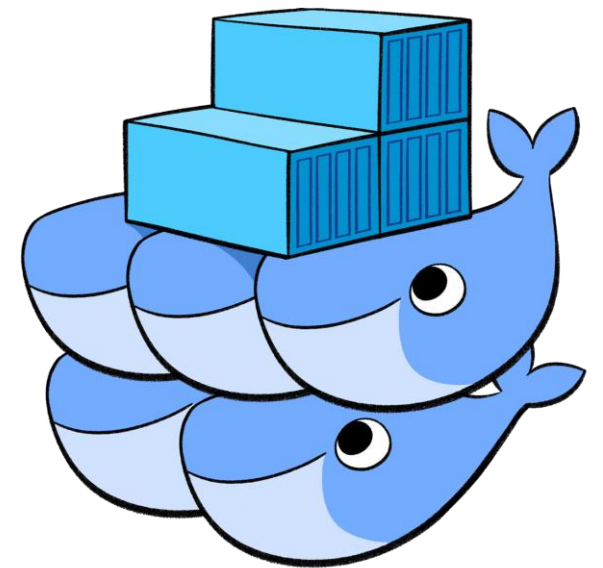
placement:

constraints:

[node.hostname == a01.mywp.dev]

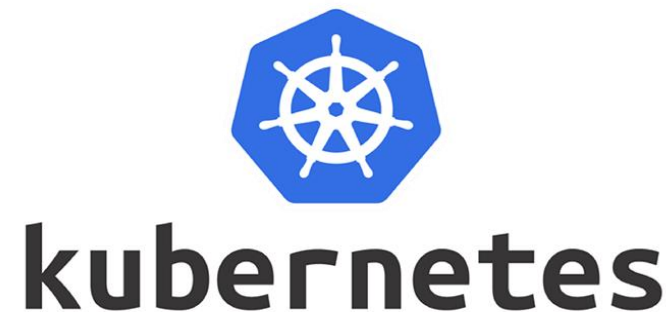
Docker Swarm - zhrnutie

- Orchestračný engine integrovaný v Dockri
- Hierarchia nodes (master / worker) → services → tasks (replicas)
- Decentralizovaný dizajn
 - M master nodes ← ingress network → N worker nodes
- Ingress distribúcia záťaže (load balancing)
- Objavenie služby - služby majú pridelený DNS názov (service discovery)



Ďalšie orchestrátory - Kubernetes (základné pojmy)

- Kubernetes Cluster: 1 - N uzlov (nodes)
- Kubernetes Node:
 - riadiaci uzol (kmaster) - obvykle exkluzívna rola (nevykonáva inštanciu služby)
 - pracovný uzol (worker node)
- Kubernetes **Pod**:
 - Atomický prvok pre správu Kubernetesom
 - 1 - N kontajnerov (služieb)
 - Prepojenie s ďalšími Pods cez virtuálnu sieť
 - Zdieľanie prostriedkov v rámci Pod (napr. volumes, CPU)

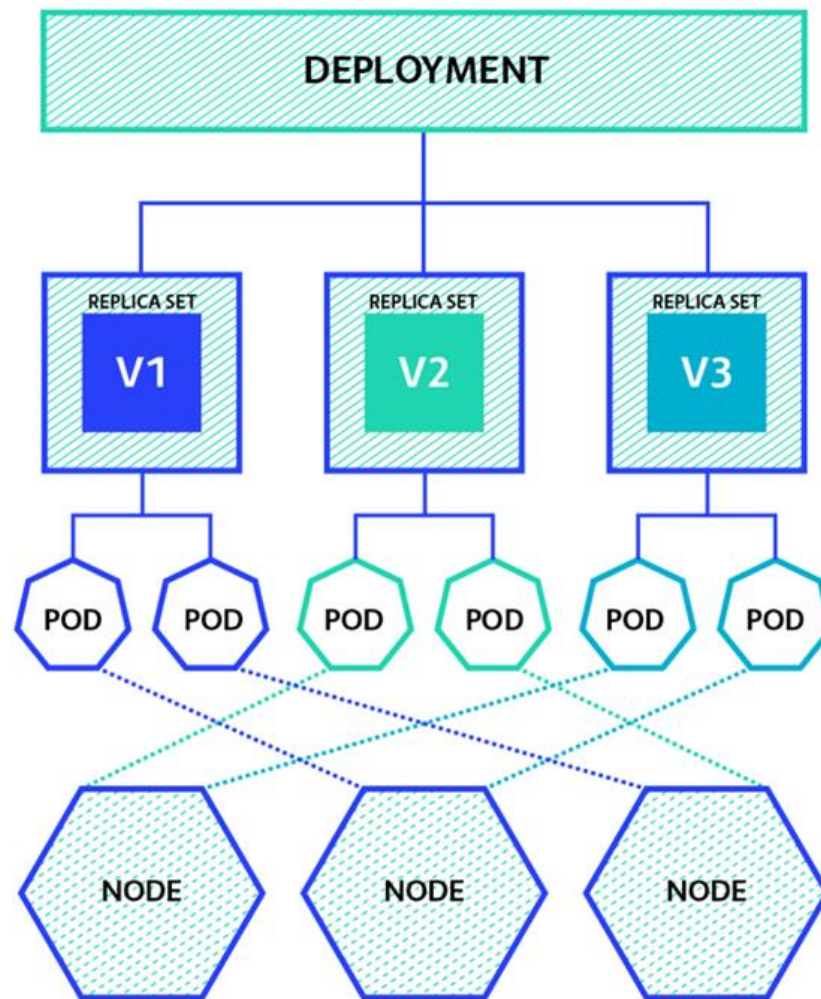
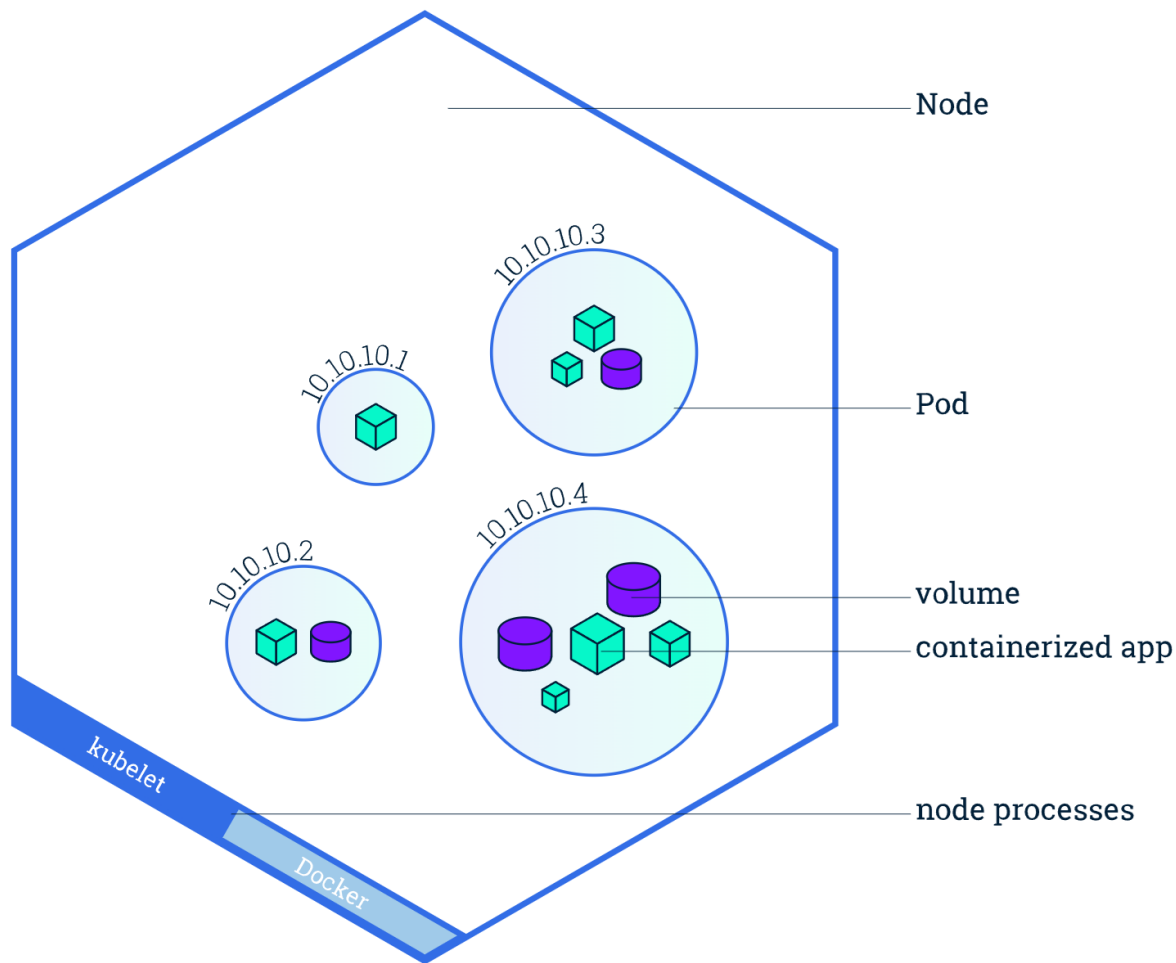


Ďalšie orchestrátory - Kubernetes (základné pojmy) /2

- Kubernetes ReplicaSet:
 - správa viacerých inštancií (replík) Podu - distribúcia Podov na jednotlivé Nodes
- Kubernetes Service:
 - Abstrakcia nad Pod-om s prideleným menom DNS - vystavenie služby (service discovery)
- Cloud provideri poskytujú tzv. **manažovaný (managed) Kubernetes**:
 - Digital Ocean (DOKS), Azure (AKS), AWS (EKS), Google Cloud (GKE) - abstrakcia nad “čistým” Kubernetesom

Čítajte viac: [Tutorials | Kubernetes](#)

Kubernetes - Node, Pod a Deployment (hierarchia)



Čítajte viac: [Viewing Pods and Nodes | Kubernetes](#) ,
[How Kubernetes Deployments Work – The New Stack](#)

Porovnanie Docker Swarm vs. Kubernetes

- Kubernetes:
 - + Nasadenie najkomplexnejších systémov na báze mikroslužieb - “veľkí hráči” (Google, Spotify, Pinterest...)
 - + Veľká flexibilita, konfigurovateľnosť, komunita
 - + Automatické škálovanie, natívna podpora containerd, treťostranné integrácie...
 - Zložitý systém, strmšia krivka učenia
- Docker Swarm:
 - + Natívna súčasť Docker Engine, jednoduchá inštalácia a konfigurácia
 - + Postačujúci na orchestráciu stredne veľkých systémov
 - Limitovaná funkcionálnosť v porovnaní s Kubernetesom (napr. auto-scaling)

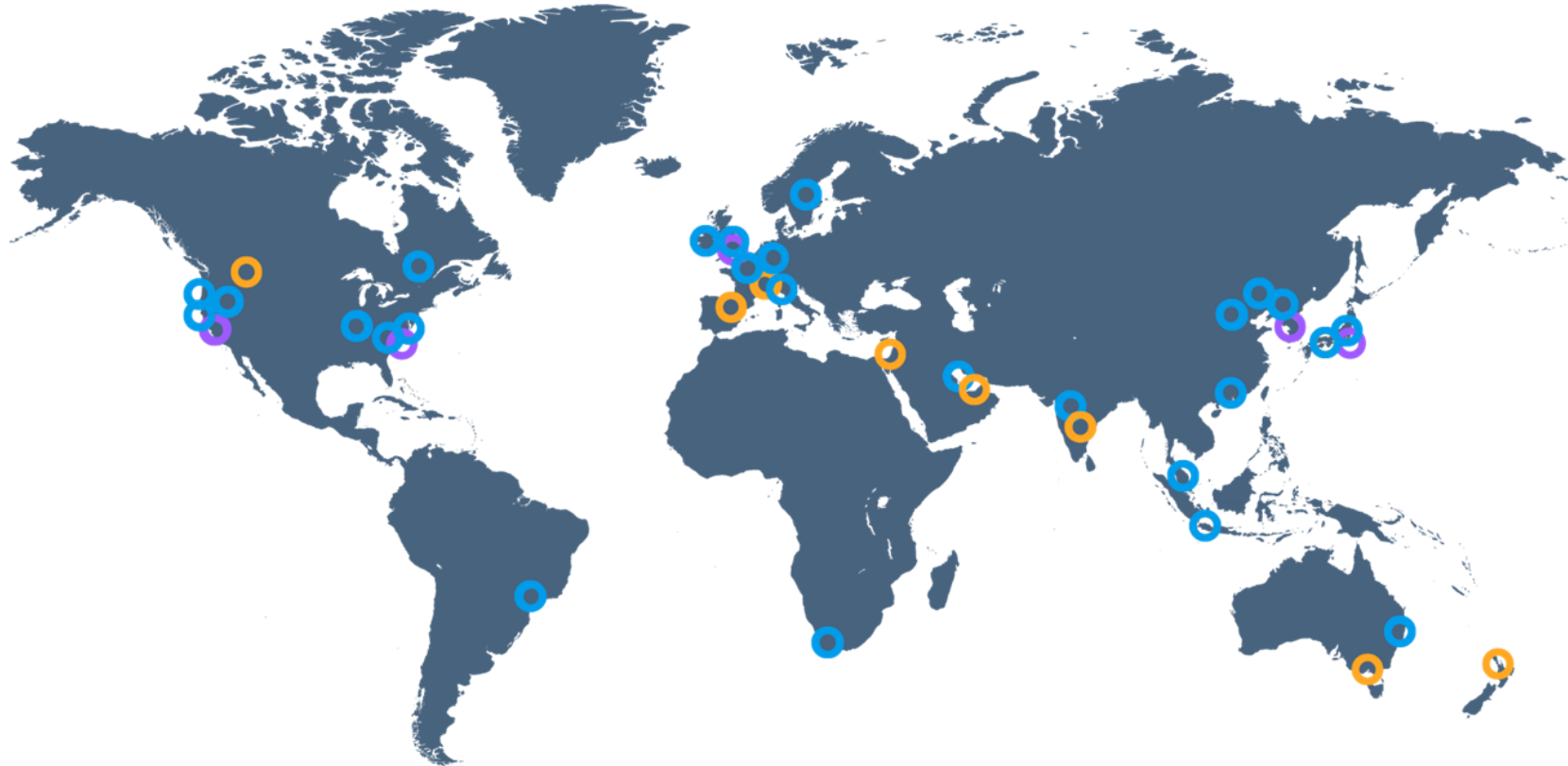
Amazon Web Services (AWS)



- Najväčší (najviac rozšírený)* cloud provider typu IaaS na svete:
 - 84 zón dostupnosti (**Availability Zones**) naprieč 26 regiónmi sveta - EU, US...
 - Výpočty (**Compute**) - AWS EC2 (VPS), AWS ECS / EKS (SW kontajnery), AWS Lambda (tzv. serverless výpočty)
 - Databázy (**Database**) - AWS RDS (SQL - Postgres, MariaDB), AWS ElastiCache (in-memory - Redis), AWS DynamoDB (key-value - “NoSQL”)
 - Úložisko (**Storage**) - AWS EBS (disk - block storage), AWS EFS (sieťový, škálovateľný súborový systém), AWS S3 (“bucket” - object storage)
 - Bezpečnosť (**Security**) - AWS WAF (firewall web. aplikácií), AWS Shield (DDoS)
 - Monitoring (**Monitor**) - AWS CloudWatch (logy, metriky, alarmy) a i.

* Zdroj údajov: <https://www.zdnet.com/article/the-top-cloud-providers-of-2021-aws-microsoft-azure-google-cloud-hybrid-saas/>

Amazon Web Services (AWS) - zóny dostupnosti služieb



Nasadenie webovej aplikácie na AWS - čo treba?

1. **Server (VPS)** - ideálne s podporou SW kontajnerov, orchestrácie:
 - a. **AWS EC2** - inštancia **VPS**, vrátane OS (Amazon Linux / Debian...), Docker Engine
 - b. **AWS ECS** - **orchestrácia kontajnerov**, úzka integrácia s AWS EC2
=> ECS umožňuje vytvoriť inštanciu EC2 so všetkým potrebným (sprievodca)

2. **Úložisko (block / object storage)** - podľa kontextu:
 - a. **AWS EBS** - virtuálny disk (**block storage**), pripojiteľný k serveru (EC2 inštancii)
=> Všeobecné úložisko dát - **databáza (SQL / NoSQL)**, **raw data** - výpočty, ML...
 - a. **AWS S3** - tzv. vedierko (bucket - **object storage**), analógia Google Drive
=> Úložisko pre profilové **obrázky, prílohy (multimédiá)**; server logy, zálohy

Nasadenie webovej aplikácie na AWS - čo treba? /2

3. **Databázu (SQL)** - voliteľné, tzv. “manažovaná” (managed) databáza

a. **AWS RDS - SQL databáza** pod správou AWS (napr. PostgreSQL)

=> AWS za vás (za poplatok) vyrieši **nasadenie, zálohy, upgrady, replikáciu, monitoring...**

=> Odbremenenie od réžie spojenej s prevádzkou DB, **bezpečnosť**, garancie

4. **Sieť pre distribúciu obsahu (CDN)** - voliteľné, vhodné pre object storage

a. **AWS CloudFront** - CDN s uzlami po celom svete pre **optimálnu distribúciu (statického) obsahu**, často v kombinácii s **AWS S3 (object storage)**

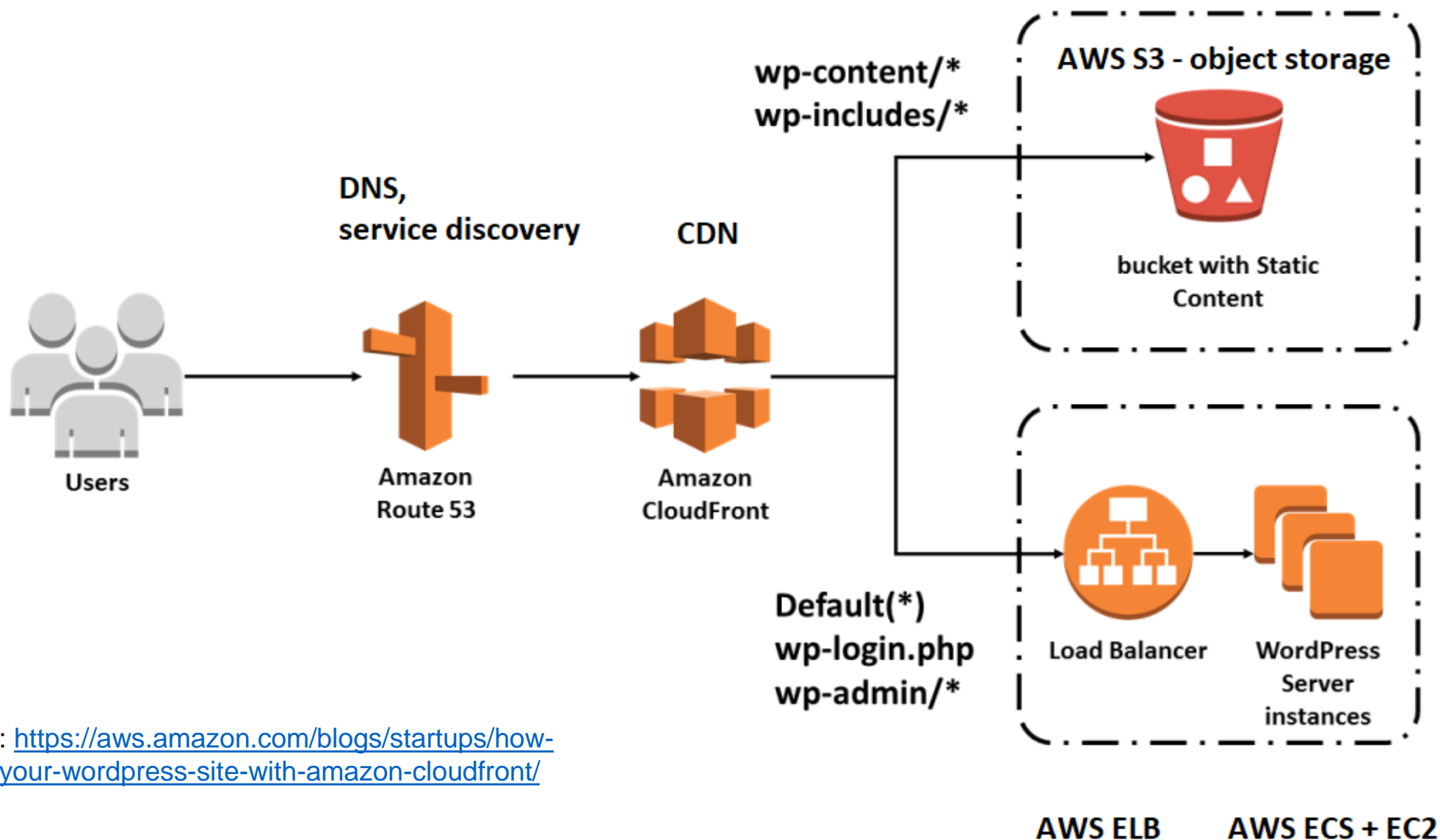
=> Optimálna distribúcia obsahu podľa **geolokácie** (edge servers)

=> **Bezpečnosť** (ochrana pred DDoS), **zníženie latencie, úspora nákladov**

Nasadenie webovej aplikácie na AWS - čo treba? /3

- 5. **Podporné služby** - voliteľné, monitoring vysoko odporúčaný:
 - a. **AWS CloudWatch** - monitoring logov zo servera, **metriky** (záťaž), **alarmy**, **triggre**
=> napr. je možné naviazať metriku na akciu auto-škálovania (AWS ECS)
 - a. **AWS VPC** - konfigurácia **virtuálnej siete** (subnets, route tables, gateways...)
 - b. **AWS Route 53** - služba **DNS**, **service discovery**
 - c. **AWS ELB** - distribúcia záťaže (**load balancer**) pre webové aplikácie
 - d. **AWS SES** - **mailový server** (podpora SMTP, REST API)
 - e. **AWS SNS** - **notifikácie** v systéme AWS (**webhooky**, **maily**, **AWS Lambda**)
 - f. **AWS Lambda** - tzv. “**serverless**” **vykonávanie kódu** (funkcie)

Nasadenie webovej aplikácie na AWS - príklad (Wordpress)

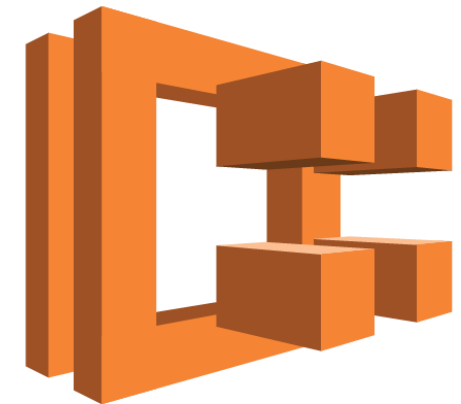


Adaptované z: <https://aws.amazon.com/blogs/startups/how-to-accelerate-your-wordpress-site-with-amazon-cloudfront/>

Nasadenie na AWS ECS (postup)

1. Vytvorenie **ECS klastra**

- Typ klastra (EC2 Linux + Networking)
- Typ inštancie (t3.micro, on-demand), OS (Amazon Linux), kľúč pre SSH prístup
 - Možnosť jednoducho vygenerovať RSA kľúčový pár cez EC2 konzolu (webové rozhranie)
- Špecifikácia virtuálnej siete (VPC) - vytvorenie novej
 - Nastavenie bloku CIDR, subsietí
- Nastavenie bezpečnostnej skupiny - otvorené porty
 - Predvolene: plný prístup k inštancii na porte 80
- Vytvorenie klastra (1 inštancia)



AWS ECS

Vytvorenie ECS klastra (príklad)

The screenshot shows the AWS Management Console interface. At the top, there is a dark navigation bar with the AWS logo, a 'Services' menu, and a search bar. Below this, the left sidebar contains a list of services: 'New ECS Experience', 'Amazon ECS' (highlighted with a red box), 'Clusters' (highlighted with a red box), 'Task Definitions', 'Account Settings', 'Amazon EKS', 'Clusters', and 'Amazon ECR'. The main content area is titled 'Clusters' and contains a description of an Amazon ECS cluster. Below the description, there are two buttons: 'Create Cluster' (highlighted with a red box) and 'Get Started'. At the bottom of the main content area, there is a 'View' section with 'list' and 'card' options.

aws Services Search for services, features, blogs, docs, and more [Alt+S]

☐ New ECS Experience
Tell us what you think

Amazon ECS

Clusters

Task Definitions

Account Settings

Amazon EKS

Clusters

Amazon ECR

Clusters

An Amazon ECS cluster is a regional grouping of one or more container instances on which you Clusters may contain more than one Amazon EC2 instance type.

For more information, see the [ECS documentation](#).

Create Cluster Get Started

View

Vytvorenie ECS klastra (príklad) /2

Select cluster template

The following cluster templates are available to simplify cluster creation. Additional configuration and integrations can be added later.

Networking only ⓘ

Resources to be created:

Cluster
VPC (optional)
Subnets (optional)

ⓘ For use with either AWS Fargate (Windows/Linux) or with External instance capacity.

EC2 Linux + Networking

Resources to be created:

Cluster
VPC
Subnets
Auto Scaling group with Linux AMI

EC2 Windows + Networking

Resources to be created:

Cluster
VPC
Subnets
Auto Scaling group with Windows AMI

*Required

Cancel

Next step

Vytvorenie ECS klastra (príklad) /3

Configure cluster

Cluster name*

test-cluster

☐

Create an empty cluster

Instance configuration

Provisioning Mode

☒

On-Demand Instance

With On-Demand Instances, you pay for compute capacity by the hour, with no long-term commitments or upfront payments.

☐

Spot

Amazon EC2 Spot Instances let you take advantage of unused EC2 capacity in the AWS cloud. Spot Instances are available at up to a 90% discount compared to On-Demand prices.

[Learn more](#)

EC2 instance type*

t3.micro

☐

Manually enter desired instance type

Vytvorenie ECS klastra (príklad) /4

Number of instances* ⓘ

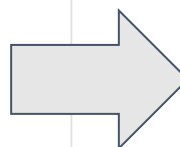
EC2 AMI ID* ⓘ

Root EBS Volume Size (GiB) ⓘ

Key pair ⓘ ⓘ

3) You will not be able to SSH into your EC2 instances without a key pair. You can create a new key pair in the [EC2 console](#). 1)

2)



Networking

Configure the VPC for your container instances to use. A VPC is an isolated portion of the AWS cloud populated by AWS objects, such as Amazon EC2 instances. You can choose an existing VPC, or create a new one with this wizard.

VPC ⓘ ⓘ

CIDR block ⓘ

Subnet 1 ⓘ ⓘ

Subnet 2 ⓘ

[+ Add more subnets.](#)

Security group ⓘ ⓘ

Security group inbound rules

CIDR block	Port range	Protocol
<input type="text" value="0.0.0.0/0"/>	<input type="text" value="80"/>	<input type="text" value="tcp"/>

Plný prístup na porte 80 (!)

Vytvorenie ECS klastra (príklad) /5

Container instance IAM role

The Amazon ECS container agent makes calls to the Amazon ECS API actions on your behalf, so container instances that run the agent require the `ecsInstanceRole` IAM policy and role for the service to know that the agent belongs to you. If you do not have the `ecsInstanceRole` already, we can create one for you.

Container instance IAM role

ecsInstanceRole



For container instances to receive the new ARN and resource ID format, the root user needs to opt in for the container instance IAM role. Opt in and try again.

Tags

Key

Value

Add key

Add value

CloudWatch Container Insights

CloudWatch Container Insights is a monitoring and troubleshooting solution for containerized applications and microservices. It collects, aggregates, and summarizes compute utilization such as CPU, memory, disk, and network; and diagnostic information such as container restart failures to help you isolate issues with your clusters and resolve them quickly. [Learn more](#)

CloudWatch Container Insights ☐ Enable Container Insights

*Required

Cancel

Previous

Create

Nasadenie na AWS ECS (postup) /2

2. Vytvorenie **definície úlohy (task definition)**:

- Docker image (link na Docker Hub alebo AWS ECR - privátny Docker repozitár)
- Názov úlohy, hostname
- Premenné prostredia (environment variables)
- Pripojené zdroje - Docker volumes / bind-mounts (mount points)
- Pridelenie zdrojov - CPU a pamäť
 - “Mäkký” limit - rezervácia (soft limit)
 - “Tvrdý” limit (hard limit)
- Logovanie (driver: Docker, AWS CloudWatch)
- Obmedzenia (constraints, napr. umiestnenie na podskupinu nodes)

Vytvorenie definície úlohy - príklad

Task definition name*

my-test-app



Task role

Select a role...



Optional IAM role that tasks can use to make API requests to authorized AWS services. Create an Amazon Elastic Container Service Task Role in the [IAM Console](#)

Network mode

<default>



If you choose <default>, ECS will start your container using Docker's default networking mode, which is Bridge on Linux and NAT on Windows. Windows tasks support the <default> and awsvpc network modes.

Requires compatibilities



EC2



FARGATE



EXTERNAL

Vytvorenie definície úlohy - príklad /2

Container name*

Image*

Private repository authentication* ☐

Memory Limits (MiB)

Soft limit ▼	<input type="text" value="512"/>	✕
Hard limit ▼	<input type="text" value="768"/>	✕

Define hard and/or soft memory limits in MiB for your container. Hard and soft limits correspond to the `memory` and `memoryReservation` parameters, respectively, in task definitions.

ECS recommends 300-500 MiB as a starting point for web applications.

Port mappings

Host port	Container port	Protocol
<input type="text" value="80"/>	<input type="text" value="8080"/>	<input type="text" value="tcp"/>

[+ Add port mapping](#)

Vytvorenie definície úlohy - príklad /3

Volumes

Use a volume configuration to add volumes

Name	data-volume
Volume type	Bind Mount
Source path	/data/myapp

Mount points

Source volume

data-volume

Container path

/myapp/persist

Read only

☐

+ Add mount point

Environment variables

You may also designate AWS Systems Manager Parameter Store keys or ARNs using the 'valueFrom' field. ECS will inject the value into cor

Key		Value		
DEBUG		Value	▼	false
HOST		Value	▼	0.0.0.0
PORT		Value	▼	8080

Nasadenie na AWS ECS (postup) /3

3. Vytvorenie **služby** pre ECS:

- Spôsob nasadenia služby - EC2 (využívame IaaS, AWS poskytuje aj PaaS)
- Názov služby (identifikátor)
- Výber definície úlohy (vytvorili sme v predchádzajúcom kroku)
- Výber ECS klastra na nasadenie (klaster má 1 - N nodes)
- Typ služby:
 - **Replika** (N inštancií distribuovaných medzi nodes) + počet úloh / replík (tasks)
 - **Démon** (1 inštancia na každom node)
- Zdravie služby:
 - Minimálny a maximálny "stav" - počet inštancií služby, ktoré ECS udržiava

Nasadenie služby - AWS ECS (postup) /4

3. Vytvorenie **služby** pre ECS (pokrač.):

- Stratégia v prípade chyby pri nasadení (deployment circuit breaker)
- Spôsob nasadenia - rolling update, Blue/Green deployment
- Distribúcia medzi nodes:
 - Rovnomerne medzi zóny dostupnosti (AZ balanced spread)
 - 1 úloha na node (One task per host)
 - ...
- Distribúcia záťaže:
 - Naviazanie load balancera (Application Load Balancer)
 - Perióda kontroly zdravia po nasadení služby (Health check grace period)
 - ...

Vytvorenie služby - príklad

Launch type ☐ FARGATE ☒ **EC2** ☐ EXTERNAL ?

[Switch to capacity provider strategy](#) ?

Task Definition

Family

my-test-app ▼

Revision

1 ▼

Cluster test-cluster ▼ ?

Service name my-test-service ?

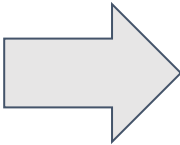
Service type* ☒ **REPLICA** ☐ DAEMON ?

Number of tasks 6 ?

Minimum healthy percent 100 ?

Maximum percent 200 ?

Deployment circuit breaker Enabled with rollback ▼ ?



Vytvorenie služby - príklad /2

Deployments

Choose a deployment option for the service.

Deployment type*



Rolling update



Blue/green deployment (powered by AWS CodeDeploy)



This sets AWS CodeDeploy as the deployment controller for the service. A CodeDeploy application and deployment group are created automatically with [default settings](#) for the service. To change to the rolling update deployment type after the service has been created, you must re-create the service and select the "rolling update" deployment type.

Task Placement

Lets you customize how tasks are placed on instances within your cluster. Different placement strategies are available to optimize for availability and efficiency.

Placement Templates

AZ Balanced Spread



[Edit](#)

This template will spread tasks across availability zones and within the availability zone spread tasks across instances. [Learn more](#)

Strategy: spread(attribute:ecs.availability-zone), spread(instanceId)

Vytvorenie služby - príklad /3

Health check grace period

60



Load balancing

An Elastic Load Balancing load balancer distributes incoming traffic across the tasks running in your service. Choose an existing load balancer, or create a new one in the [Amazon EC2 console](#).

Load balancer
type*



None

Your service will not use a load balancer.



Application Load Balancer

Allows containers to use dynamic host port mapping (multiple tasks allowed per container instance). Multiple services can use the same listener port on a single load balancer with rule-based routing and paths.



Network Load Balancer

A Network Load Balancer functions at the fourth layer of the Open Systems Interconnection (OSI) model. After the load balancer receives a request, it selects a target from the target group for the default rule using a flow hash routing algorithm.



Classic Load Balancer

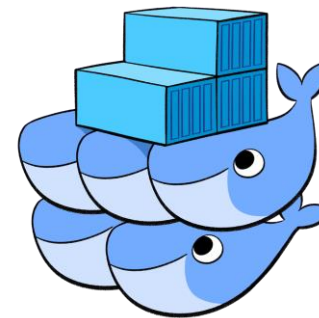
Requires static host port mappings (only one task allowed per container instance); rule-based routing and paths are not supported.

Čítajte viac: <https://docs.aws.amazon.com/AmazonECS/latest/developerguide/getting-started.html>

Zhrnutie

- Cloud computing - výpočty a uloženie dát vo vzdialenom centre (cloud)
- Orchestrácia kontajnerov - správa, koordinácia, konfigurácia, nasadenie a škálovanie systému mikroslužieb

- Docker Swarm, Kubernetes, AWS ECS



- Nabudúce:
 - Nasadenie v prostredí cloudu - časť 2 (CDN, replikácia a zálohovanie dát)
 - DevOps ako pojem, CI/CD, správa verzií (Git), automatizácia (Jenkins)
 - Základy testovania aplikácií (TDD), základy bezpečnosti