

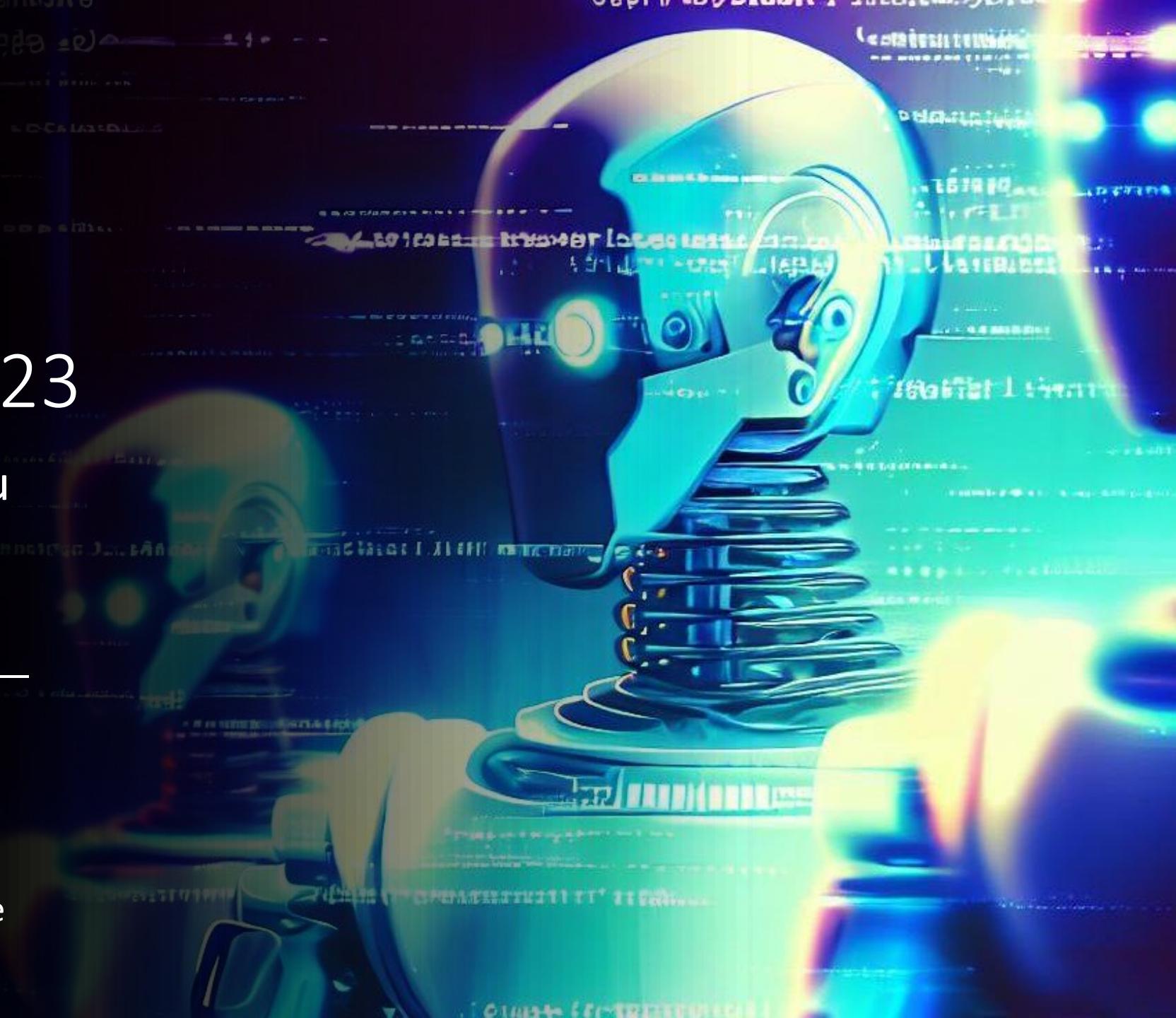
Cloudná akce 2023

Update cloudařova mozku

Tomáš Kubica

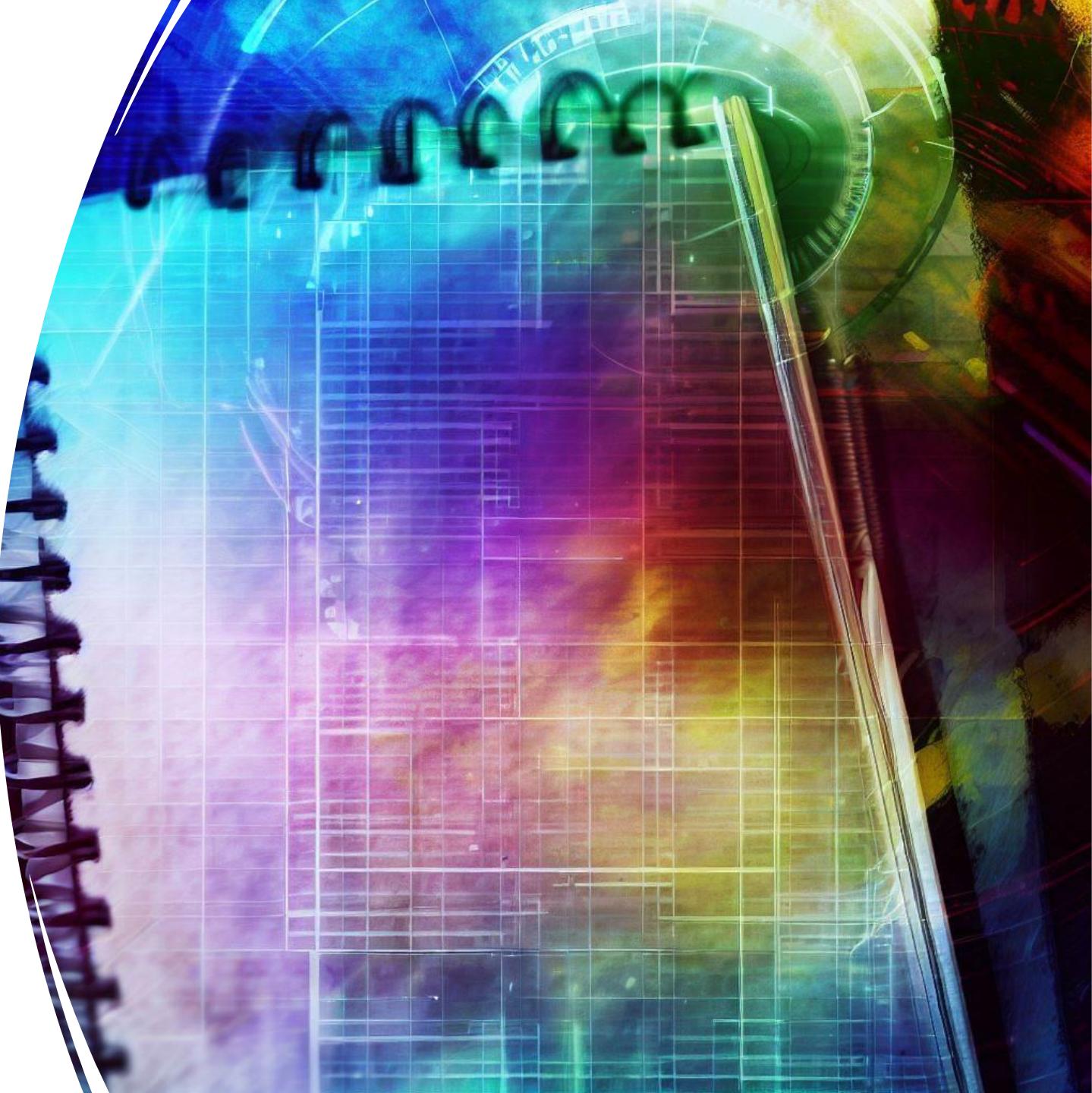
Cloud Solution Architect v
Microsoftu

GitHub.com/tkubica12/cloudnaakce



Agenda

- Cloud native
- Bezpečnost cloutu a identit
- Umělá inteligence (AI)
- Osobní tipy jak zůstat relevantní



Agenda

- **Cloud native**
 - Infrastructure as Code ve velkém a pořádně
 - GitOps
 - Progressive delivery
 - Next-gen: Kubernetes bez Kubernetes
 - Chaos engineering
- Bezpečnost cloudu a identit
- Umělá inteligence (AI)
- Osobní tipy jak zůstat relevantní



Agenda

- Cloud native
- **Bezpečnost cloutu a identit**
 - Decentralizovaná identita
 - Federované workload identity
 - Když nemusím věřit – Confidential Computing, homomorfní šifrování a differential privacy
- Umělá inteligence (AI)
- Osobní tipy jak zůstat relevantní



Agenda

- Cloud native
- Bezpečnost cloutu a identit
- **Umělá inteligence (AI)**
 - Tah 37, 4-hodinový velmistr, žárovky a inovátoři, AI-assisted humans
 - Praktická použití velkých jazykových modelů jako je ChatGPT
 - Naše budoucnost s AI
- Osobní tipy jak zůstat relevantní



Agenda

- Cloud native
- Bezpečnost cloutu a identit
- Umělá inteligence (AI)
- **Osobní tipy jak zůstat relevantní**



Agenda

- Cloud native
 - Infrastructure as Code ve velkém a pořádně
 - GitOps
 - Progressive delivery
 - Next-gen: Kubernetes bez Kubernetes
 - Chaos engineering
- Bezpečnost cloudu a identit
- Umělá inteligence (AI)
- Osobní tipy jak zůstat relevantní



Infrastructure as Code



Správa a kompletní životní cyklus počítačové infrastruktury pomocí strojově čitelných definičních souborů, ke kterým se lze chovat stejně, jako k počítačovému kódu – verzování, testování, řízení změn.



Deklarativní desired state



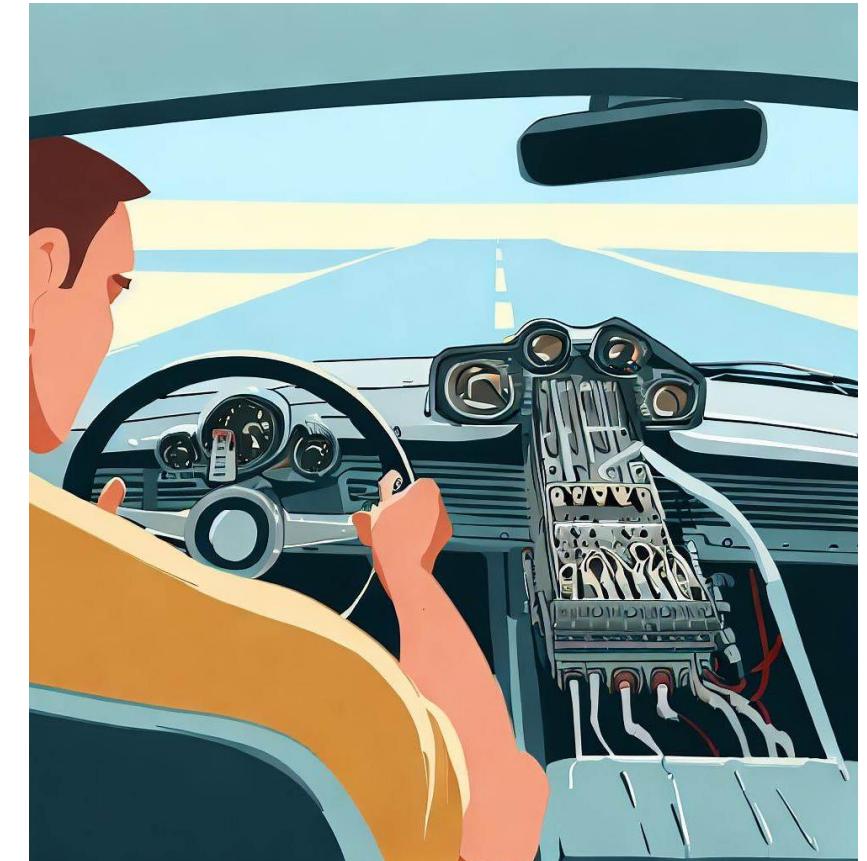
Řeknete jaký má být výsledný stav systému a neřešíte jakými kroky se dostat ze stávajícího k požadovanému. Opak imperativního přístupu, kdy řešíte pořadí operací (např. skript).

Hlavní výhody Infrastructure as Code

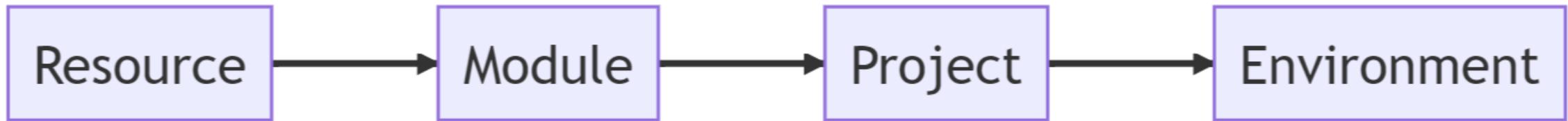
- Opakovatelné výsledky bez ohledu na to kdo a kdy to udělá (na rozdíl od lidí a jejich copy and paste)
- Konzistentní a trvanlivé (na rozdíl od screenshotů)
- Jednotka práce, kterou lze předat s jistotou výsledku (na rozdíl od release notes)
- Řízení verzí (kdo co změnil s možností se vrátit do známého funkčního stavu)
- Řízení změn přes Git (schvalování, bezpečností kontrola, automatizované testy)
- Vždy aktuální a konzistentní spustitelná dokumentace (dokumentaci uvede robot přesně do reálného světa)
- Schopnost okamžitě startovat nová prostředí a instance (agilita, automatizace testů, nižší náklady za občasně využívaná prostředí)

Vrstvy abstrakce

- Do Not Repeat Yourself (DRY) princip
- Vrstvy abstrakce zahrnují firemní znalosti, doporučené praktiky z projektů, závěry specialistů.
- Vytvářejte znovupoužitelné moduly pro nasazení standardizovaných komponent:
 - Zapouzdřete firemní preferované konfigurace
 - Určete „smart defaults“ ať můžou nováčci začít s rozumnou konfigurací bez nutnosti pochopit všechny detaily
- Moduly ať jsou univerzální, použitelné ve většině projektů (např. cílité na 95% případů).
- Verzujte moduly tak, aby projekty mohly aplikovat nové verze podle svých potřeb kdy jim to vyhovuje (ale mějte deprecation policy).
- Instance projektů v jednotlivých prostředích ať mají možnost unikátních vstupů

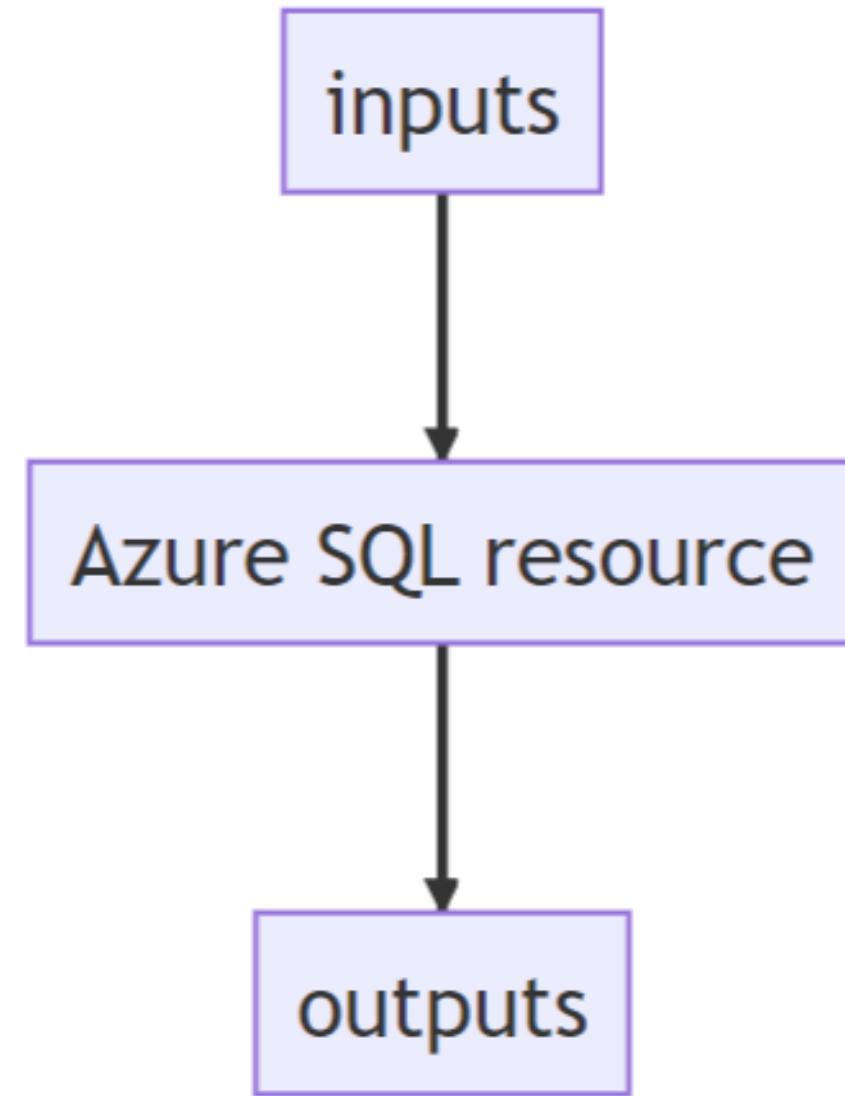


Vrstvy abstrakce



Jednotlivý zdroj

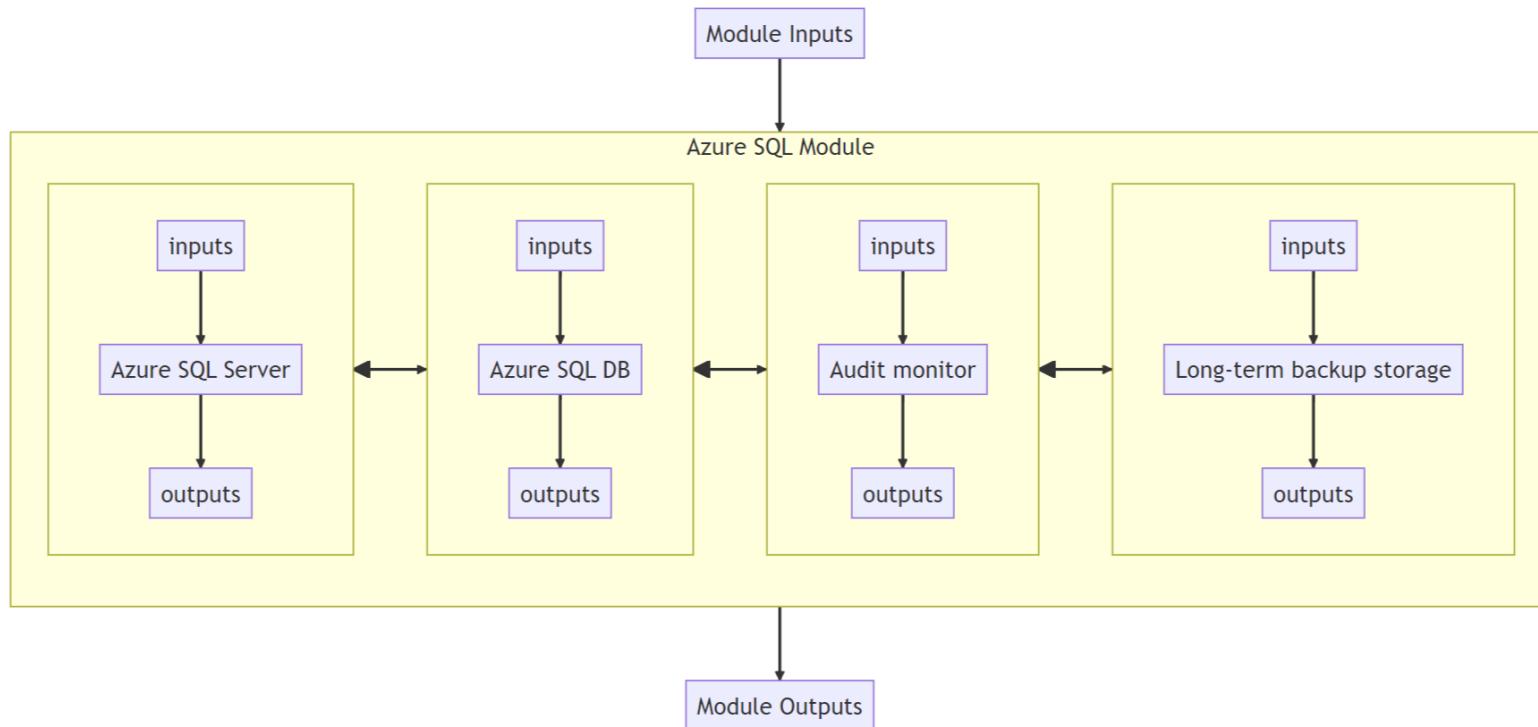
- Vstupy – parametry zdroje (SKU, jméno, region, ...)
- Výstupy – výsledné ID, URL, klíč, ...



Vyšší úroveň abstrakce (Module)

- Jeden nebo více zdrojů (např. NIC, disk a VM)
- Zabudované best practice (některá rozhodnutí uděláte za uživatele modulu, například zabezpečení hodné vaší firmy)
- Abstrakce (například jeden přepínač pro nastavení komplexního chování jako je enableAudit nebo enableHa)
- Smart defaults (pokud nevíte a neřeknete, dostanete rozumný základ)
- Znovupoužitelné napříč projekty a prostředími

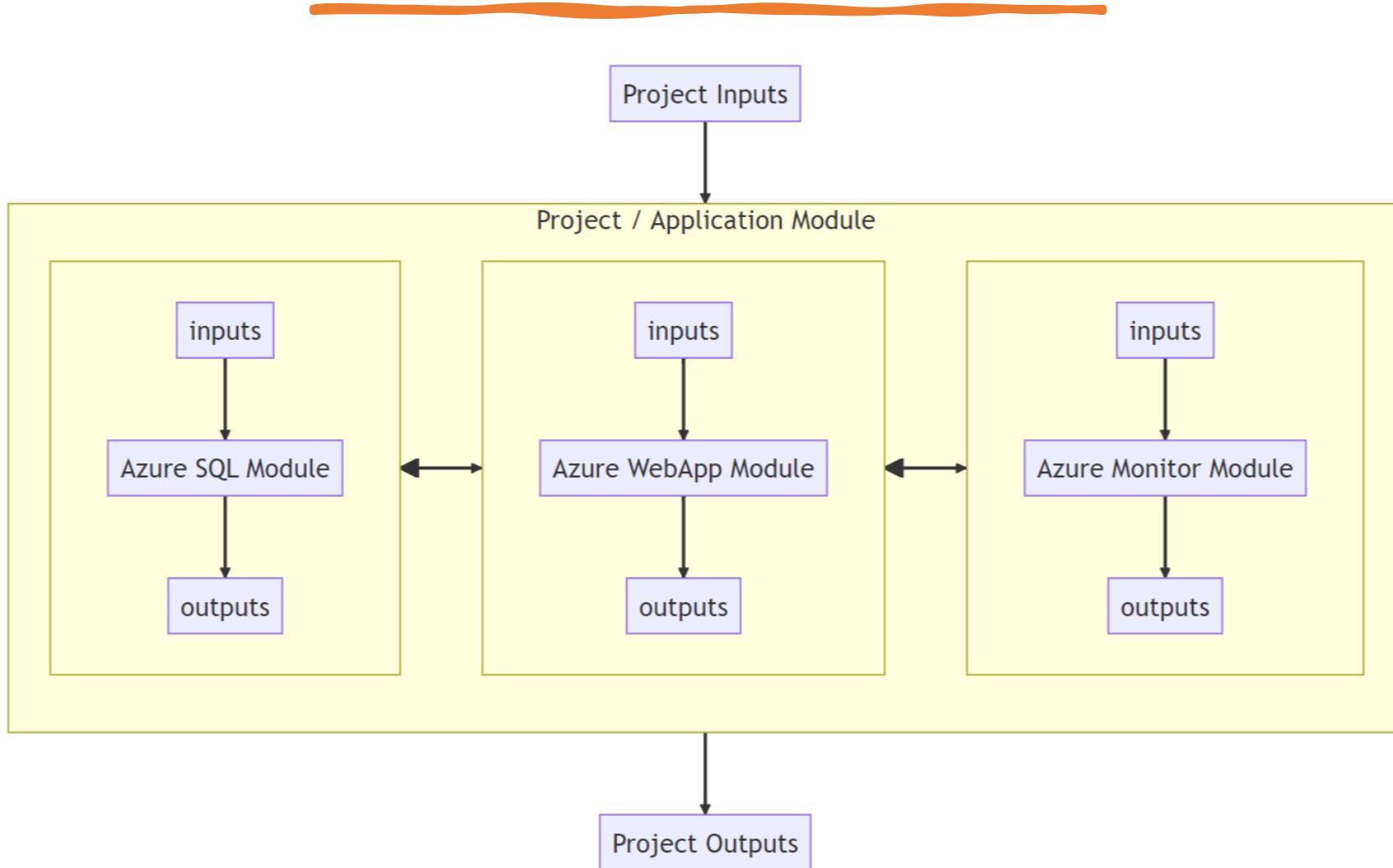
Vyšší úroveň abstrakce (Module)



Projekt jako jednotka nasazení

- Zdroje a moduly použité dohromady k vytvoření specifického řešení aplikace nebo projektu.
- Využívá firemních modulů jako stavebních bloků.
- Topologie a rozhodnutí specifická pro projekt (např. použitá databáze nebo compute platforma).
- Vstupy umožňují upravitelnost pro jednotlivá prostředí/instance.

Projekt jako jednotka nasazení

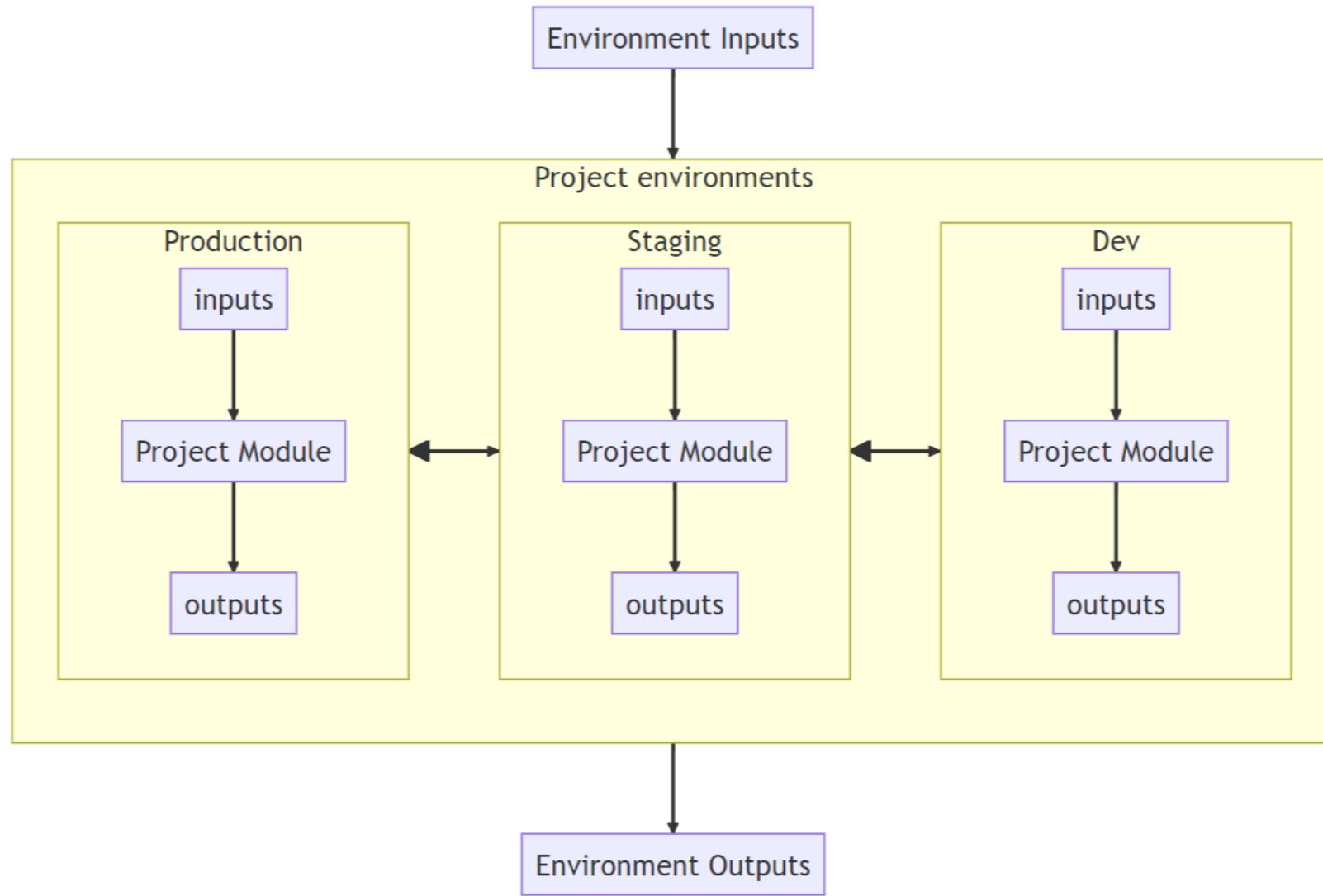


Prostředí a/nebo instance

- Prostředí je např. dev, test, uat, prod a instance je například EU, US, zákazník1.
- Žije ve svém adresáři a jsou různé varianty jak to řešit:
 - Kopírování (není DRY, ale je to jednoduché, přehledné a customizovatelné)
 - Použití pouze vstupů ala tfvars (jednoduché, ale složitější evoluce kódu)
 - GitFlow a branchování (složité, moc nedoporučuji)
 - Terragrunt, který se o to postará
 - Další vrstva modulů (env -> projekt -> moduly)



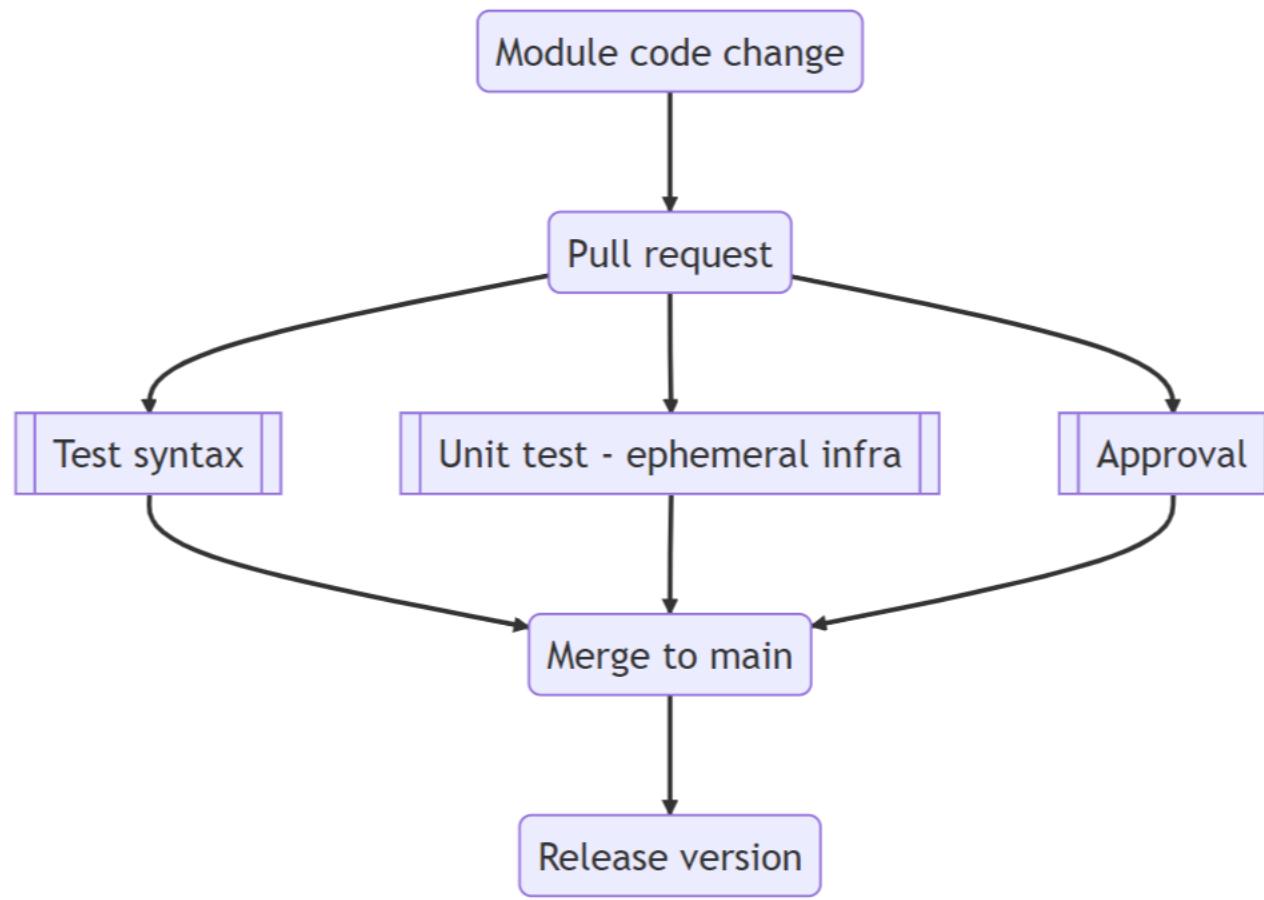
Prostředí a/nebo instance



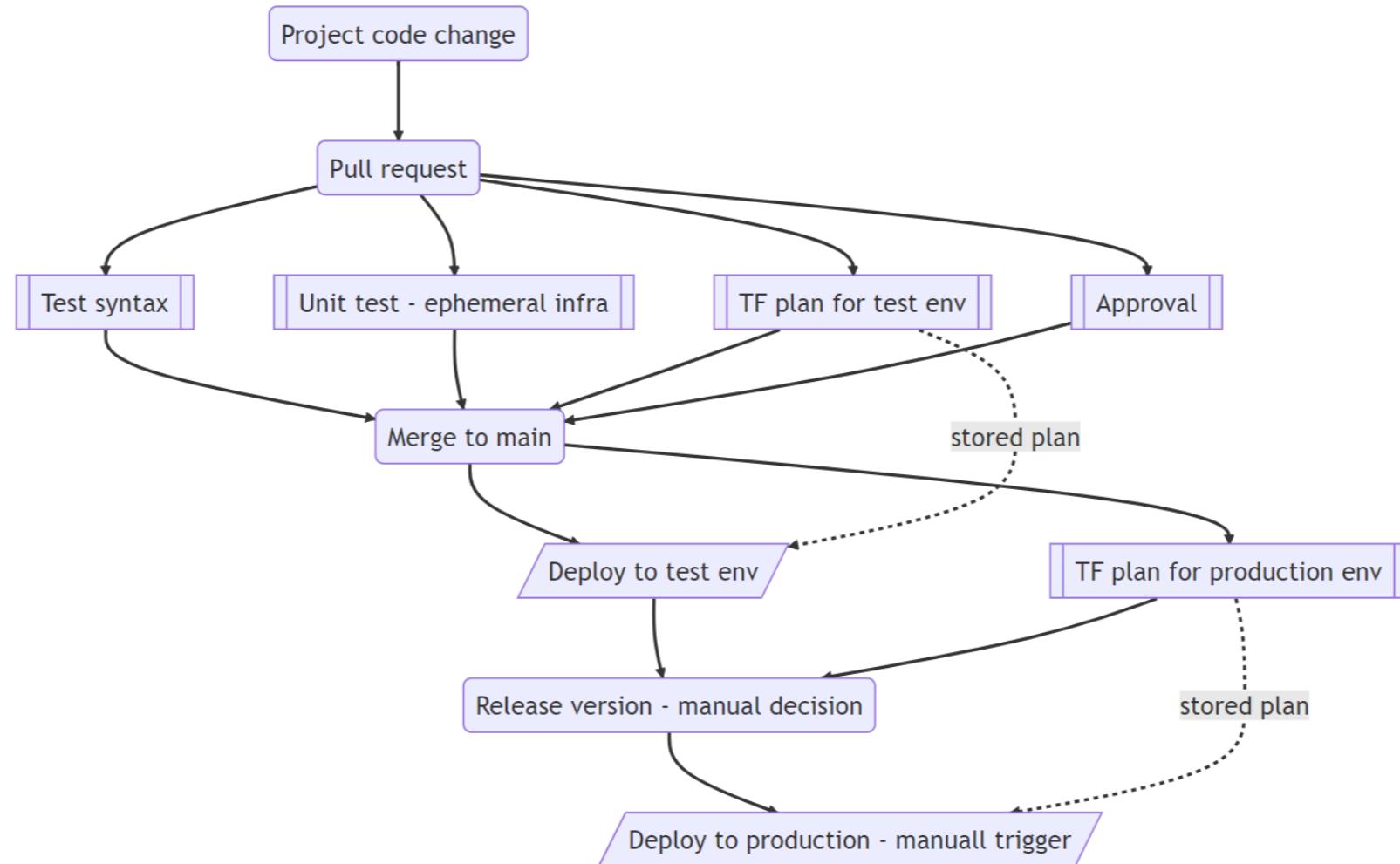
GitOps procesy

- No work without reason – vždycky vytvořte work item (třeba GitHub Issue) a změny asocujte s ní
- Navrhujte změny přes Pull Request proti main větvi
- Spouštějte automatizované testy v rámci PR a merge, kterými ověřte syntaxi, ujistíte se, že je to nasaditelné (například nasadím do ephemeral prostředí), vytvořte plán změn (diff oproti reálně infrastruktúre), spusťte smoke testy (nasadíte a ověřte funkčnost), projděte bezpečnostní politiky
- V průběhu schvalování PR vždy projděte plánované změny a uložte je, aby byly použity při pozdějším nasazování, aby byla jistota, že při nasazení se provedou jen schválené změny (v mezičase se například mohla infrastruktura změnit ručně)
- Zvažte zjednodušené trunk-based řešení se staging a produkcí
- Všechny moduly a projekty mají semantické verzování s použitím Git tagů (např. GitHub Release), takže uživatelé mohou svůj kód zamknout na konkrétní verzi.
- Můžete mít monorepo a tam všechno nebo (jednodušší na stažení a přispívání), repo pro každý modul a projekt (jednodušší na verzování a CI/CD) nebo nejčastěji kombinaci (jedno repo pro všechny firemní moduly, jedno repo pro landing zone, repa pro jeden a více projektů).
- Referencujte moduly přes repo a tag – nekopírujte (není to DRY) ani neodkazujte cestou (pak neverzujete, takže možná krom ephemeral testů).

Řízení změn v modulu

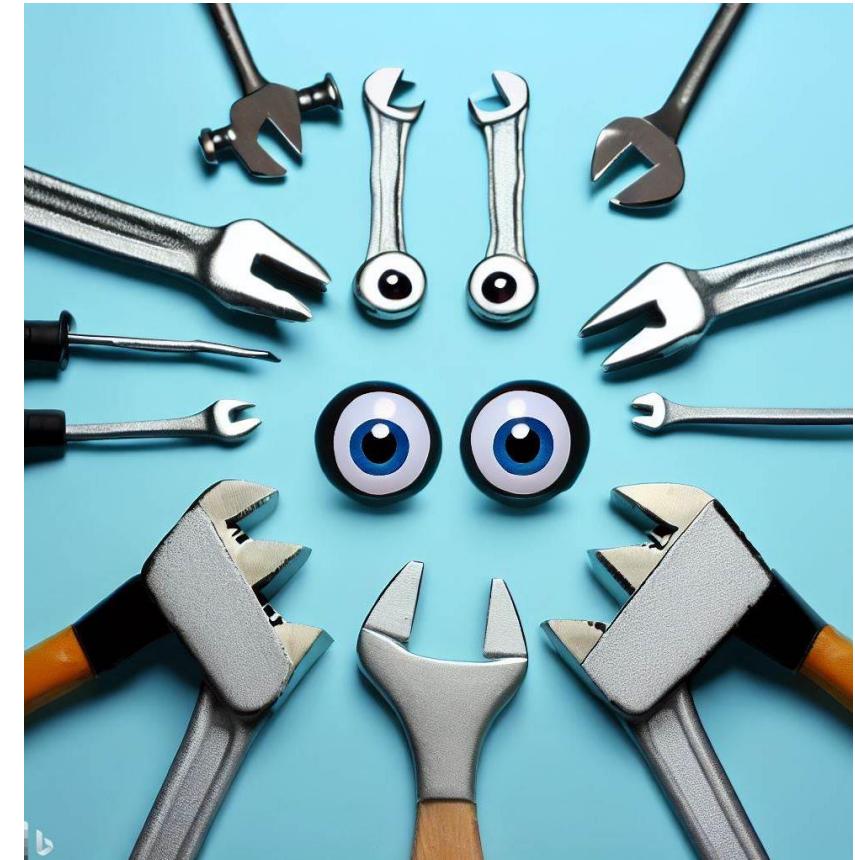


Řízení změn v projektech



Terraform vs. other tools

- **Terraform** – nejrozšířenější, multi-cloud, výborná podpora cloudů. Některé providery se dělají ručně (mohou být pozadu ve funkcích), některé se generují (nemusí být tak čitelné). Vlastní DSL (ale velmi příjemné).
 - Podpora plnohodnotného programovacího jazyka přes CDK (reakce na Pulumi?)
 - Rekoncilační smyčka a GitOps není úplně nativní (ale je Kubernetes operátor pro Terraform cloud - reakce na Crossplane?)
- **Pulumi** – používá plnohodnotný programovací jazyk (skvělé pro vývojáře, méně pak pro provozáky), autogenerované zdroje (je stále aktuální), multi-cloud.
- **Crossplane** – myšlenkový vůdce v GitOps pull-based rekoncilační smyčce zabudované v Kubernetes, takže automaticky dědí jeho robustnost a ekosystém (např. Flux, ArgoCD, Etcd, ...). Skvělé pro lidi používající Kubernetes, ale podstatně méně pak pro provozáky, governance, síťáře.
- **Ansible** – populární převážně imperativní (byť role jsou deklarativní) nástroj pro správu OS a síťářinu, ale lze použít i pro cloudovou infrastrukturu. Osobně pro nedostatečnou deklarativnost preferuji jiné (byť pro OS configuration management naopak Ansible rád).
- **Nativní nástroje** – většinou na výběr šablony, CDK a service operator do Kubernetes



Agenda

- Cloud native
 - Infrastructure as Code ve velkém a pořádně
 - **GitOps**
 - Progressive delivery
 - Next-gen: Kubernetes bez Kubernetes
 - Chaos engineering
- Bezpečnost cloudu a identit
- Umělá inteligence (AI)
- Osobní tipy jak zůstat relevantní



GitOps

- Git je centrálním zdrojem pravdy – nejen kódu/předpisu, ale i jeho konkrétních instancí/prostředí (všechny konfigurační parametry musí žít rovněž v Gitu).
- Git procesy typu Pull Request se řídí změny.
- GitOps nevyžaduje pull-based přístup a rekoniliační smyčku, ale dává s ní nejlepší smysl.
- Tajnosti, hesla ani klíče tam ale nepatří! Použijte nějaký Vault nebo sealed secrets.



Ukázka

Agenda

- Cloud native
 - Infrastructure as Code ve velkém a pořádně
 - GitOps
 - **Progressive delivery**
 - Next-gen: Kubernetes bez Kubernetes
 - Chaos engineering
- Bezpečnost cloudu a identit
- Umělá inteligence (AI)
- Osobní tipy jak zůstat relevantní





Ukázka

Agenda

- Cloud native
 - Infrastructure as Code ve velkém a pořádně
 - GitOps
 - Progressive delivery
 - **Next-gen: Kubernetes bez Kubernetes**
 - Chaos engineering
- Bezpečnost cloudu a identit
- Umělá inteligence (AI)
- Osobní tipy jak zůstat relevantní





Ukázka

Agenda

- Cloud native
 - Infrastructure as Code ve velkém a pořádně
 - GitOps
 - Progressive delivery
 - Next-gen: Kubernetes bez Kubernetes
 - **Chaos engineering**
- Bezpečnost cloudu a identit
- Umělá inteligence (AI)
- Osobní tipy jak zůstat relevantní



Cesta k anti-fragile systémům

Fragile

Nepříznivé okolnosti systém poškodí

Robust

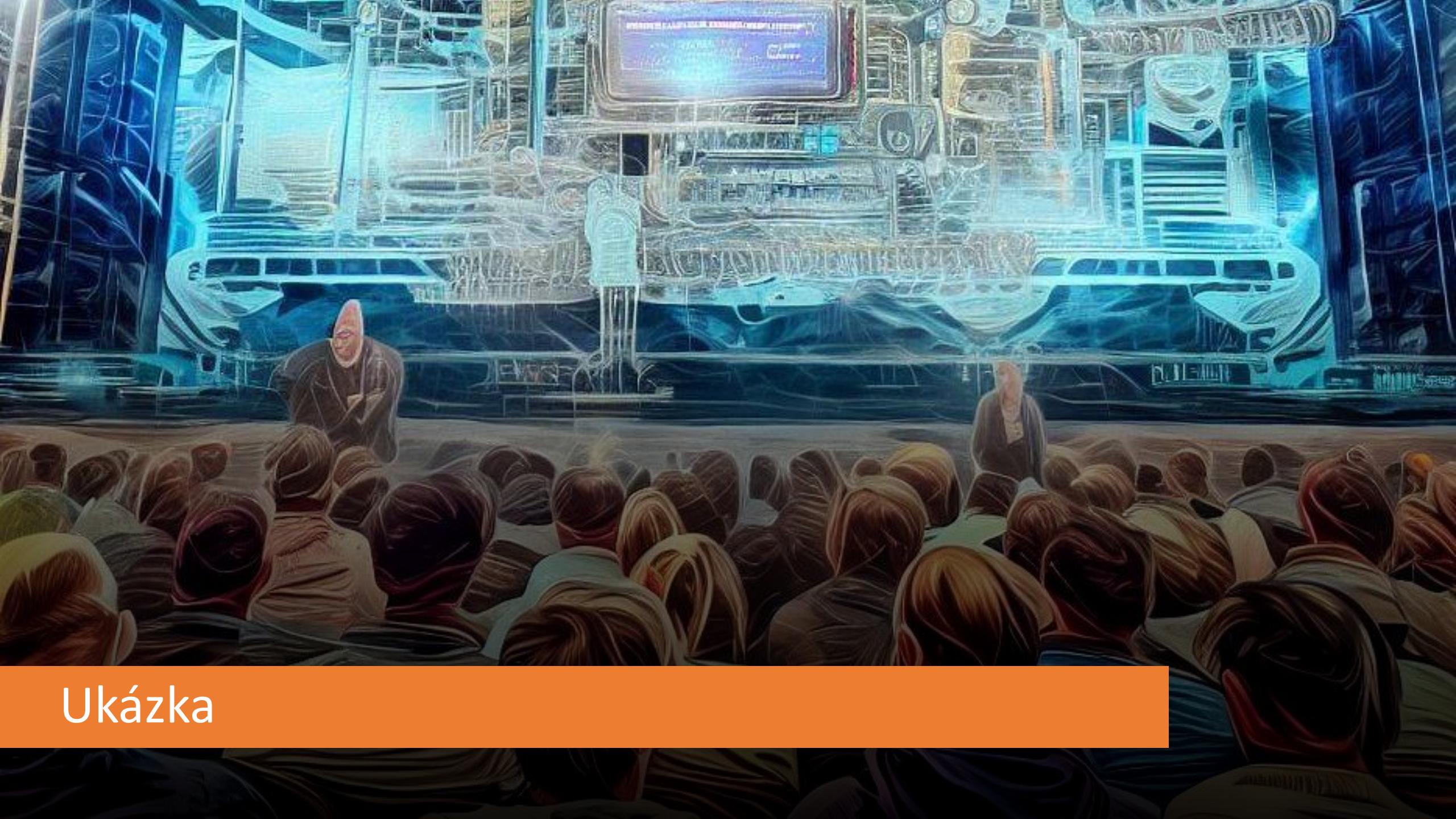
Systém nereaguje na některé nepříznivé okolnosti, ale jiné ho poškodí

Resillient

Systém se dokáže z nepříznivých okolnost vzpamatovat a obnoví se na původní hodnotu

Anti-fragile

Nepříznivé okolnosti zvyšují hodnotu systému, jsou mu prospěšné



Ukázka

Agenda

- Cloud native
- **Bezpečnost cloutu a identit**
 - Decentralizovaná identita
 - Federované workload identity
 - Když nemusím věřit – Confidential Computing, homomorfní šifrování a differential privacy
- Umělá inteligence (AI)
- Osobní tipy jak zůstat relevantní



Agenda

- Cloud native
- Bezpečnost cloutu a identit
 - **Decentralizovaná identita**
 - Federované workload identity
 - Když nemusím věřit – Confidential Computing, homomorfní šifrování a differential privacy
- Umělá inteligence (AI)
- Osobní tipy jak zůstat relevantní





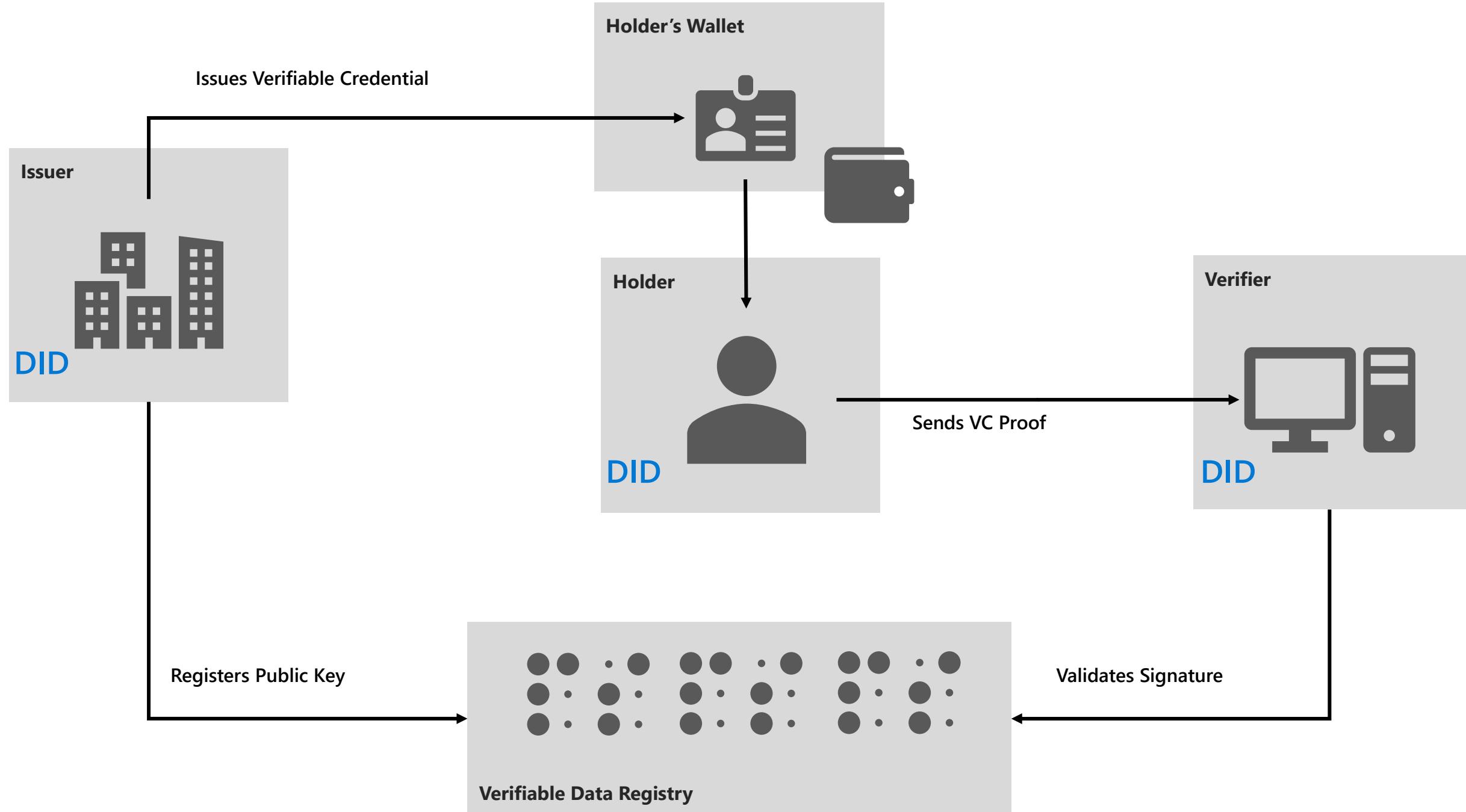
V peněžence
nosíte doklady,
které vám někdo
vystavil (stát,
banka,
zaměstnavatel,
kino, kavárna) a vy
rozhodujete komu
a kdy je ukážete.



V digitálním světě ale vaše údaje a identita leží na cizím serveru.

Proč self-sovereign identitu?

- Identitu (DID) mám fyzicky u sebe, ne na nějakém serveru poskytovatele.
- Důvěryhodnost identit všech subjektů může být distribuovaná například v ledger (může jít třeba o konsorcium bank nebo cech obchodníků) nebo i plně decentralizovaná ve veřejném blockchainu.
- Ostatní mi subjekty (Issuer) mi vystavují „osvědčení“ ve formě Verifiable Credentials.
- Sám se rozhoduji komu (Verifier) kartu ukážu (Verifiable Presentation) a jaká políčka na ní budou vidět (jaké informace chci sdílet). Já mám logy komu co jsem dal (vs. všechno má můj poskytovatel identity a ví, kde nakupuji nebo kdy chodím do kina).
- Digitální wallet (Holder) umí ještě další kouzla a může to být ten, který používáte třeba na passwordless ověření (Microsoft Authenticator)
- Nemá „full-mesh“ problém, protože ten kdo si ověřuje nemusí kontaktovat vydavatele osvědčení.



Agenda

- Cloud native
- Bezpečnost cloutu a identit
 - Decentralizovaná identita
 - **Federované workload identity**
 - Když nemusím věřit – Confidential Computing, homomorfní šifrování a differential privacy
- Umělá inteligence (AI)
- Osobní tipy jak zůstat relevantní



Workload identity a hesla

- Přístup do DB na lokální účet a heslo? Masakrózně špatně.
- Přístup do DB přes identitní systém? Okej, ale jak řešit bezpečné doručení (a rotaci) hesla do aplikace?
- Jeden systémový účet na všechno, ať to máme snadné? Nic moc, chceme least privilege a identity, jejichž životní cyklus je provázaný s aplikací.



Tam, kde běží vaše aplikace často existuje způsob, jak mít **workload identitu bez použití klíče** – systém pro mě vygeneruje platný token a s tím se mohu ověřit vůči službě.

Příklad: Azure, Google, AWS, GitHub, Kubernetes, ...



Díky federaci pak
můžete vyměnit
jeden token za
druhý.

Příklady

- Aplikace běží v Kubernetes v onprem. Ten je v namespace prod pro účet myapp1 federovaný s Azure Active Directory managed identitou, která má přístup do CosmosDB. Současně je federovaný na Google identitu, která má přístup na custom search. Appka tak bez hesla přes token a jeho výměnu dokáže přistupovat do cloudových služeb.
- Aplikace běží v AWS VM. Tato konkrétní VM má workload identitu federovanou s identitou v Google pro přístup na custom search.
- CI/CD pipeline v GitHub action má identitu svázanou s konkrétní repo, branch a prostředím a ta nasazuje do Azure. Díky federaci s Azure Active Directory může pipeline spouštět Terraform, nasazovat do cloudu a nemusíme řešit bezpečné předávání a rotaci hesla.



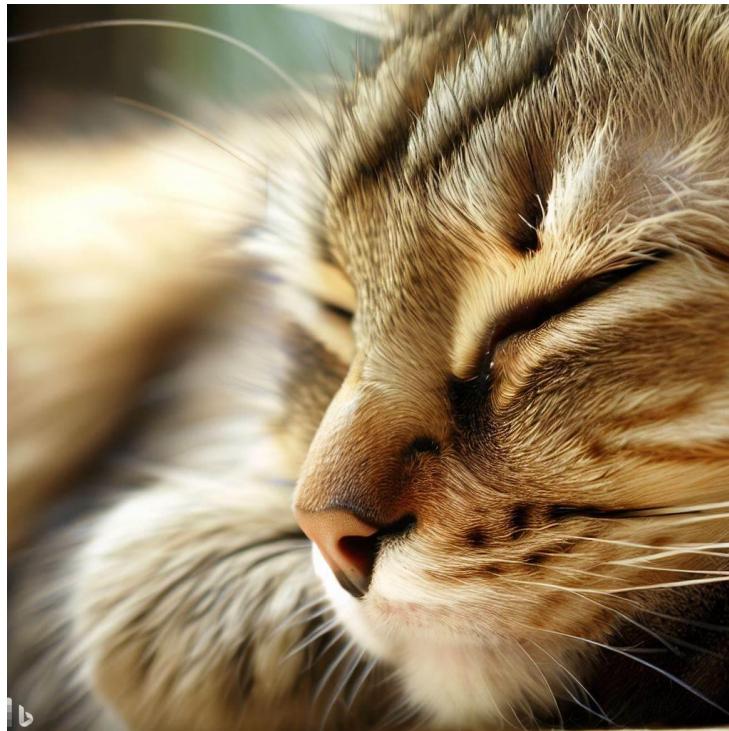
Cloud Infrastructure Entitlements Management (CIEM)

Agenda

- Cloud native
- Bezpečnost cloutu a identit
 - Decentralizovaná identita
 - Federované workload identity
 - **Když nemusím věřit –**
Confidential Computing,
homomorfní šifrování a
differential privacy
- Umělá inteligence (AI)
- Osobní tipy jak zůstat relevantní



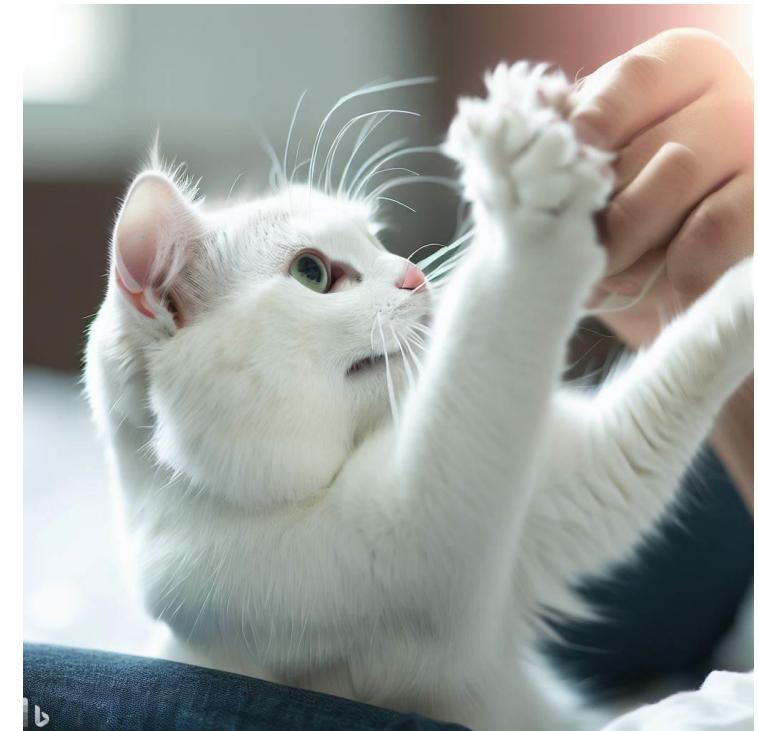
Šifrování dat – ty první dvě situace znáte ze školy, ale...



Data at rest



Data in fly



Data in use

Proč řešit data při jejich zpracování?

- Zpracování dat vede na informace, které jsou užitečné a dělitelné bez odhalení zdrojových (citlivých) dat, například výsledný Machine Learning model, agregované statistiky apod.
 - Multi-party výzkum, kdy nelze věřit někomu, kdo by viděl všechno – například vývoj léčiv na základě dat o pacientech z různých nemocnic a zemí nebo model strojového učení pro detekci podvodných bankovních transakcí (data ze všech bank vedou na lepší model pro všechny, ale zdrojová data si nebudou chtít nasdílet)
- Chci využít obrovskou sílu cloudu, ale poskytovateli nevěřím
- Mám decentralizovaný model a nikomu jednotlivému nemůžu věřit, například Blockchain – zaručeně bezpečné zpracování mi umožní přejít z neefektivního Proof of Work (pomalé a drahé) či Proof of Stake (nedostatečně decentralizované a trochu vachrlaté) na Confidential Blockchain.

Šifrování dat při jejich zpracování a použití

Confidential Computing

- Hardware dokáže izolovat podepsané procesy, jejich paměť a tuto šifrovat (Intel SGX a TDX, AMD SEV-SNP, ARM TrustZone)
- Nemusím věřit cloud providerovi, ale ani svým administrátorům

Homomorfní šifrování

- Speciální typ šifry, kdy některé operace nad zašifrovanými data jsou validní a výsledek si přečtu až po rozšifrování = můžu do nějaké míry zpracovávat zašifrovaná data do kterých nevidím

Differential privacy

- Znemožnění identifikace konkrétní entity (řádku) přimícháním chytrého šumu, který ale neznehodnotí agregovaný pohled = rozostření dat tak, že osoby můžete spočítat, ale ne identifikovat



Ukázka

Agenda

- Cloud native
- Bezpečnost cloutu a identit
- **Umělá inteligence (AI)**
 - Tah 37, 4-hodinový velmistr, žárovky a inovátoři, AI-assisted humans
 - Praktická použití velkých jazykových modelů jako je ChatGPT
 - Naše budoucnost s AI
- Osobní tipy jak zůstat relevantní



Amarův zákon: Máme tendenci nové technologie....



přečeňovat v krátkém horizontu



podceňovat v dlouhém horizontu

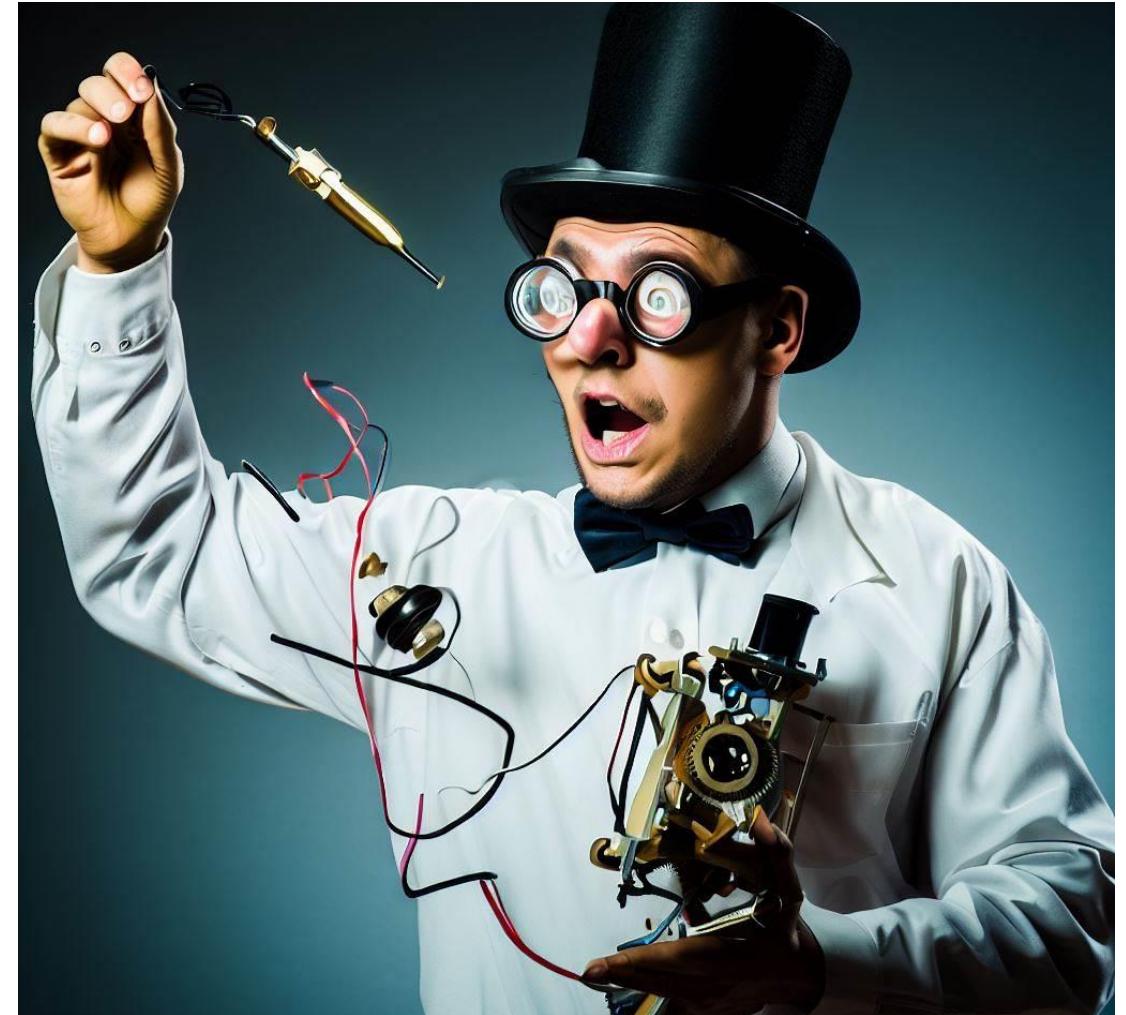
Víte, že žárovku vynalezlo
21 lidí?



Vidíš? Tohle je na základě mého nápadu.



Inovátor přináší praktičnost a široké využití. Nemusí být vynálezce.

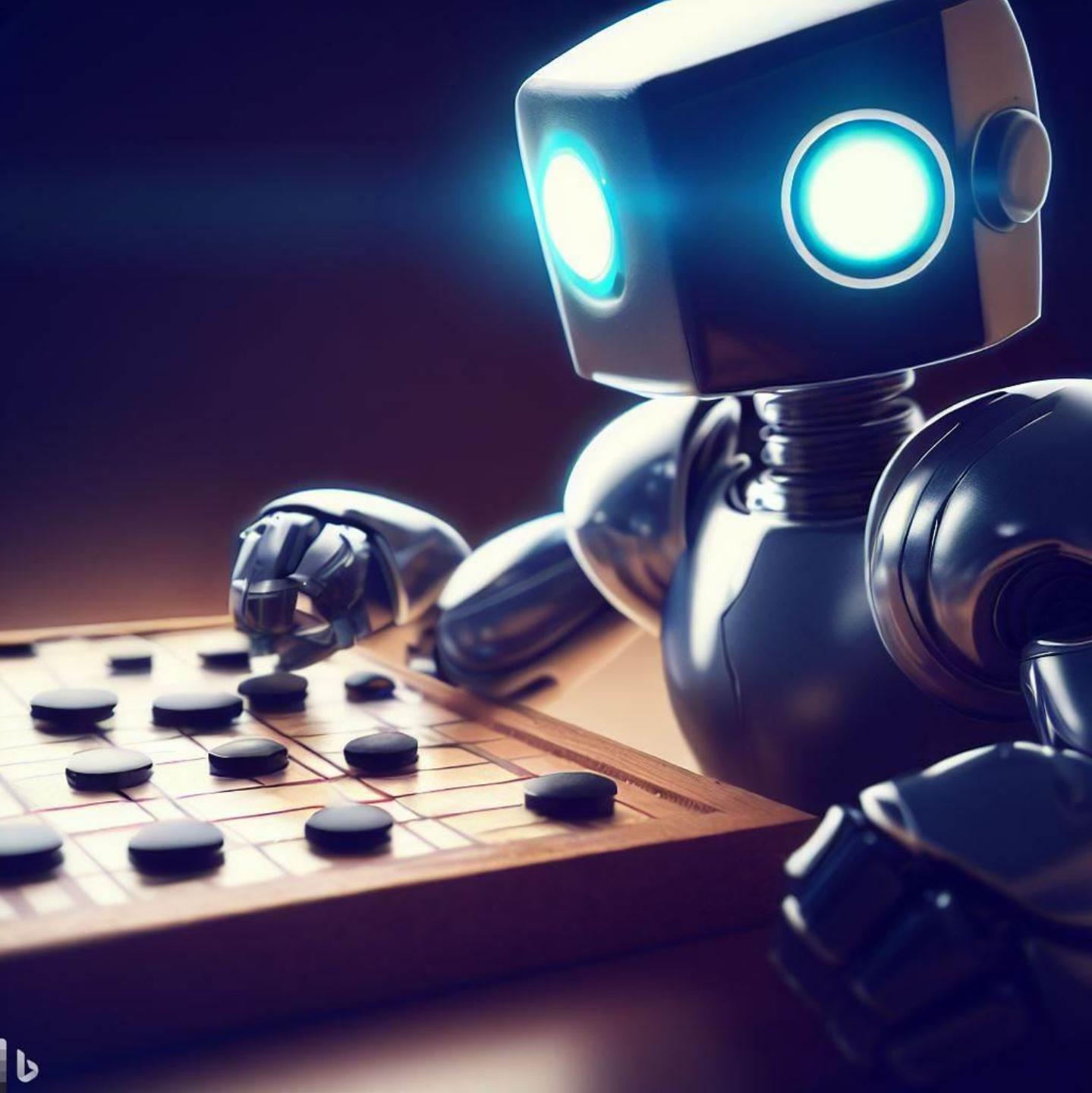


AI už není jen pro profesory.



**AlphaGo byl
trénován na 30M
tahů nejlepších lidí.**

**Když porazil
Evropského
šampiona, myslilo
se, že na světového
nemá šanci.**





Jenže... Al v mezičase trénovalo sehráním 1M zápasů samo se sebou a významně se zlepšilo.

V legendárním tahu
37 překvapilo
AlphaGo lidstvo,
takhle by člověk
nezahrál.

AlphaGo vyhrálo 4:1



A close-up photograph of a person's hand holding a white knight chess piece. The hand is wearing a purple cuff. In the background, a robotic arm with blue and silver segments is positioned to move the piece. The chessboard is visible at the bottom.

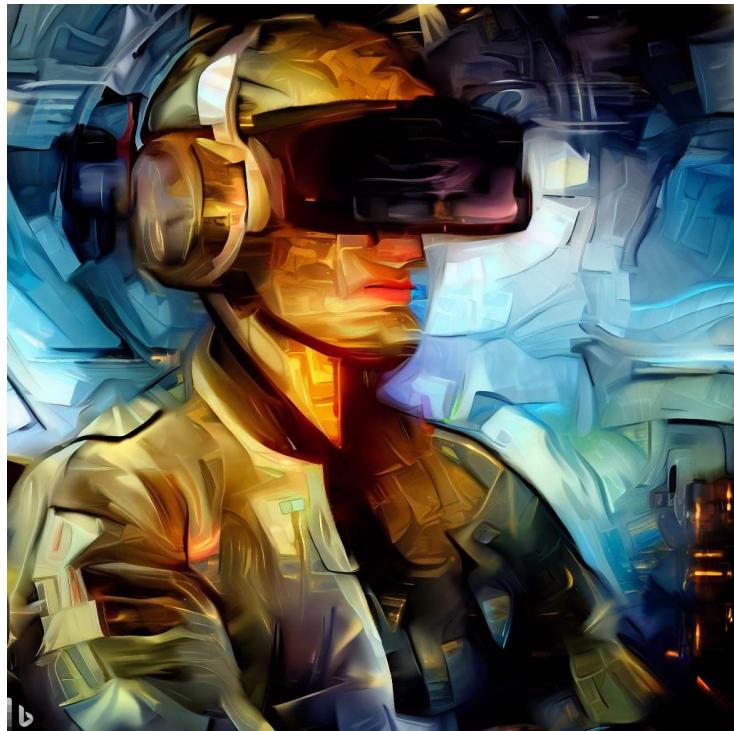
V roce 1997 se DeepBlue potřebovalo učit z databáze nejlepších lidských tahů, aby porazilo Kasparova.

V roce 2017 stačilo AlphaZero bez lidského učitele prostě hrát pár hodin samo se sebou.

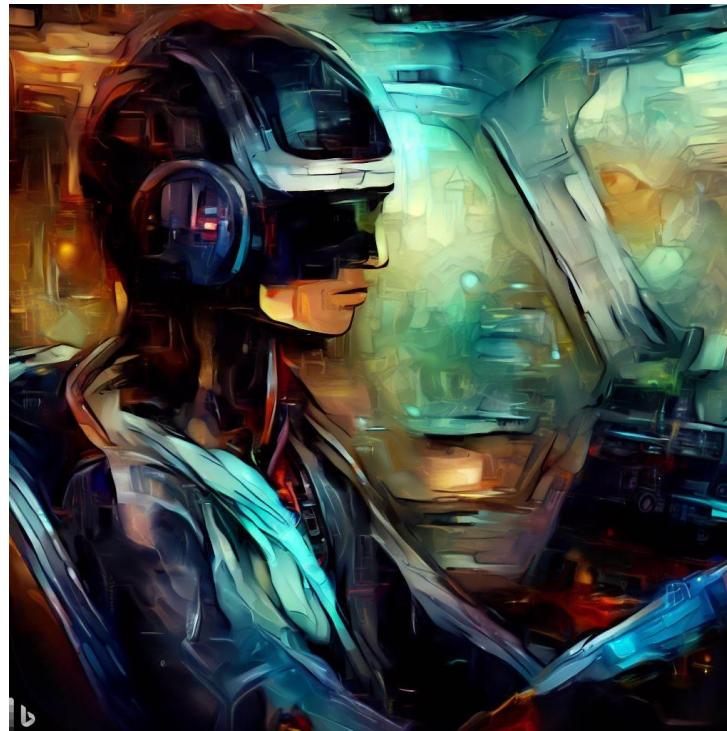


V jednoduchém světě šachu dominovali AI-assisted
humans turnajům po dekády.

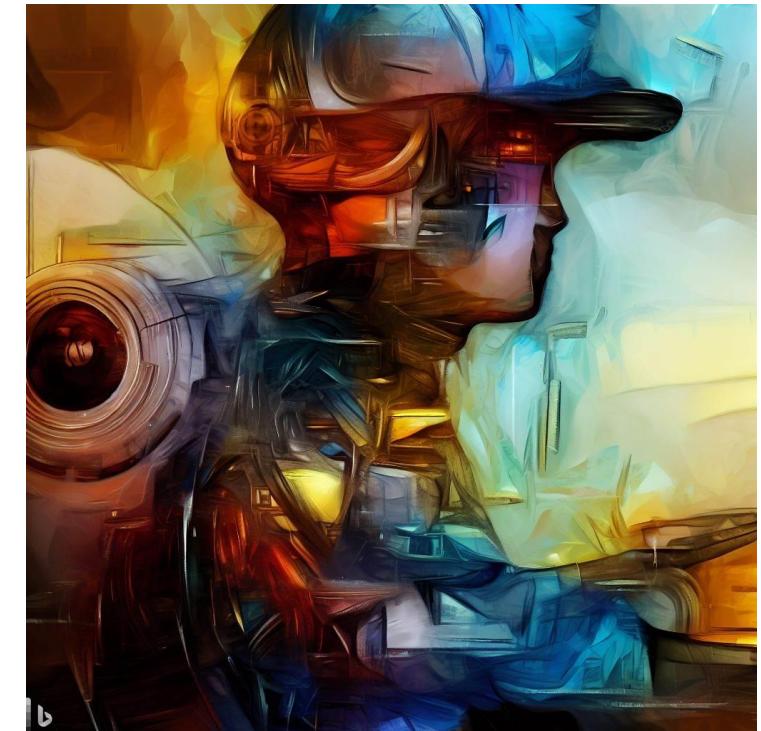
Příklad nasazení umělé inteligence pomáhající lidem



GitHub Copilot X
Programátor

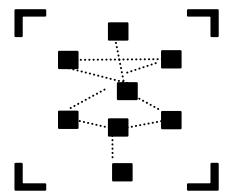


Microsoft 365 Copilot
Tvůrce kreativního obsahu



Microsoft Security Copilot
Obránce

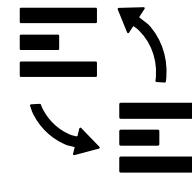
Azure Cognitive Services



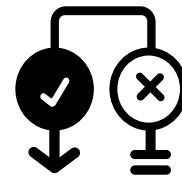
Vision



Speech



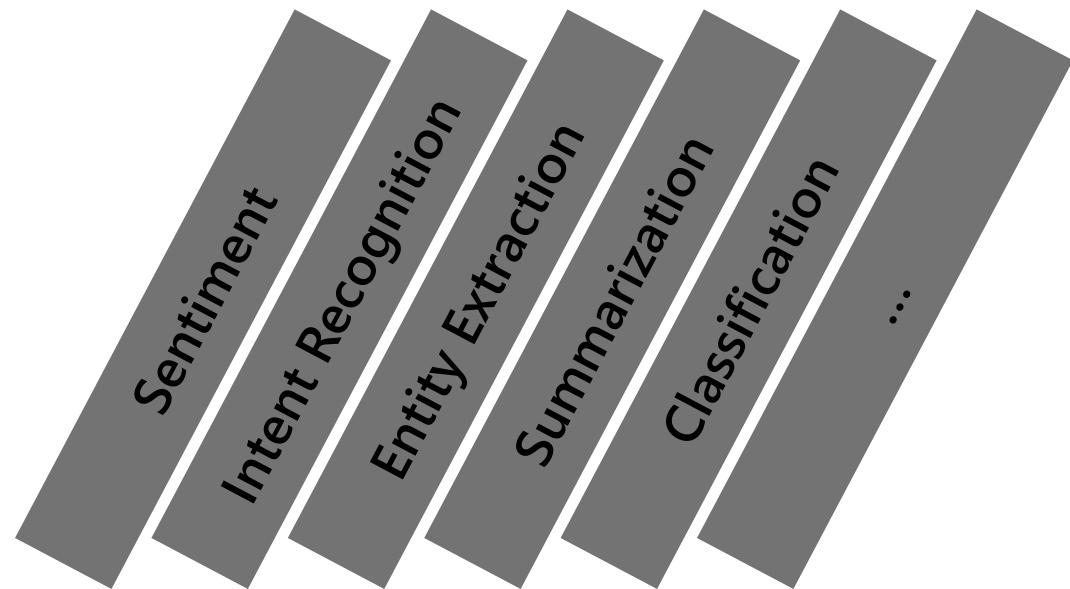
Language



Decision

V čem je OpenAI jiné?

Tradiční ML pro NLP



Zvlášť model pro každý případ
použití optimalizovaný pro
konkrétní úlohu

OpenAI GPT-3

Sentiment
Intent Recognition
Entity Extraction
Summarization
Q&A
Style transfer
Rewriting
Code generation
...

Jeden obrovský model na všechna
použití. Popíšete v lidském jazyce co
chcete, aby pro vás udělal.

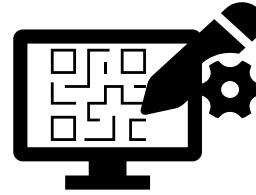


Ukázka

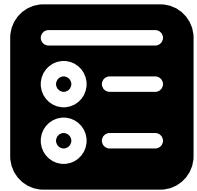
OpenAI: hlavní oblasti použití



Vytváření
obsahu



Generování
kódu



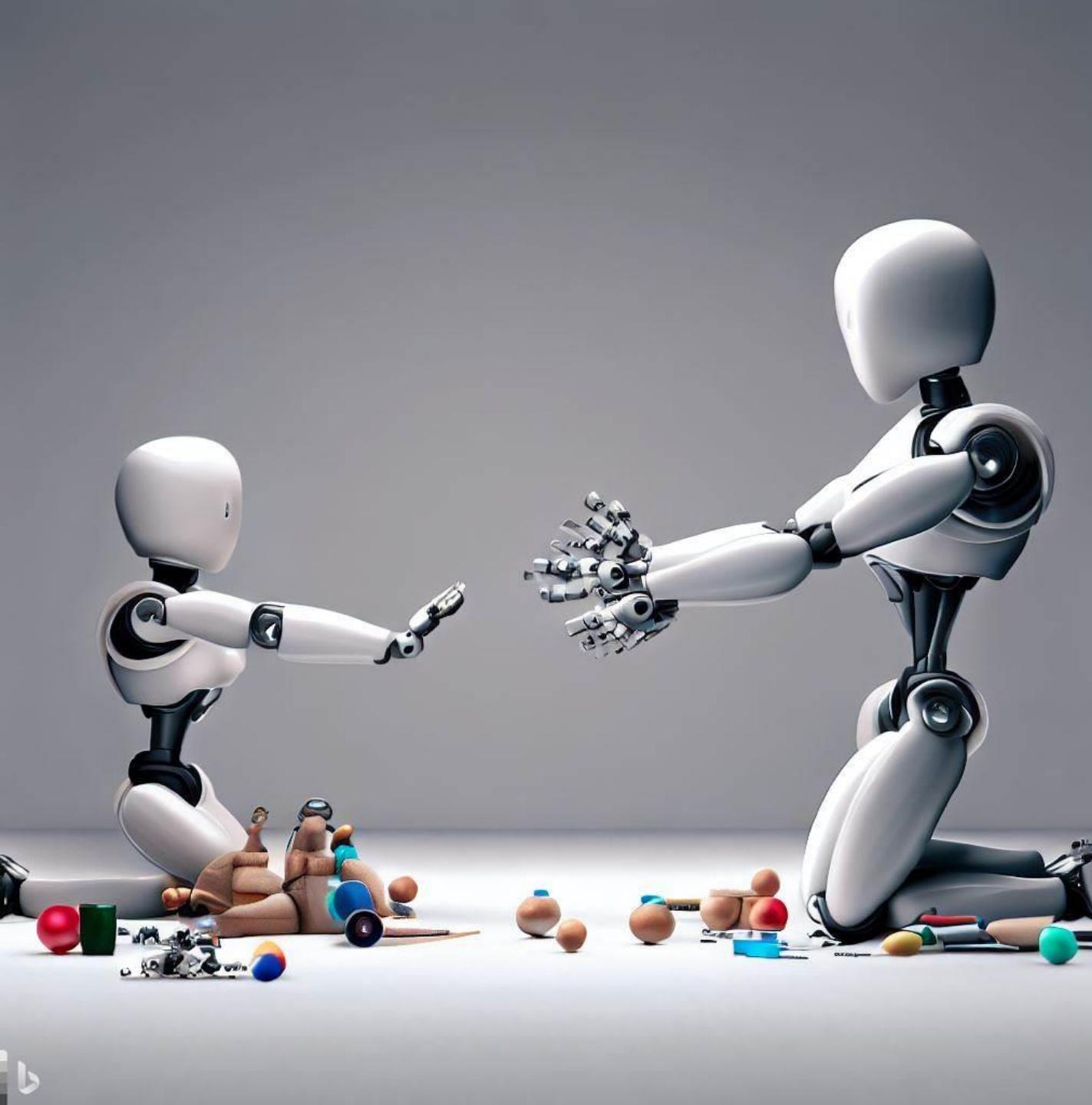
Sumarizace



Sémantické
vyhledávání

Azure OpenAI: příklady použití

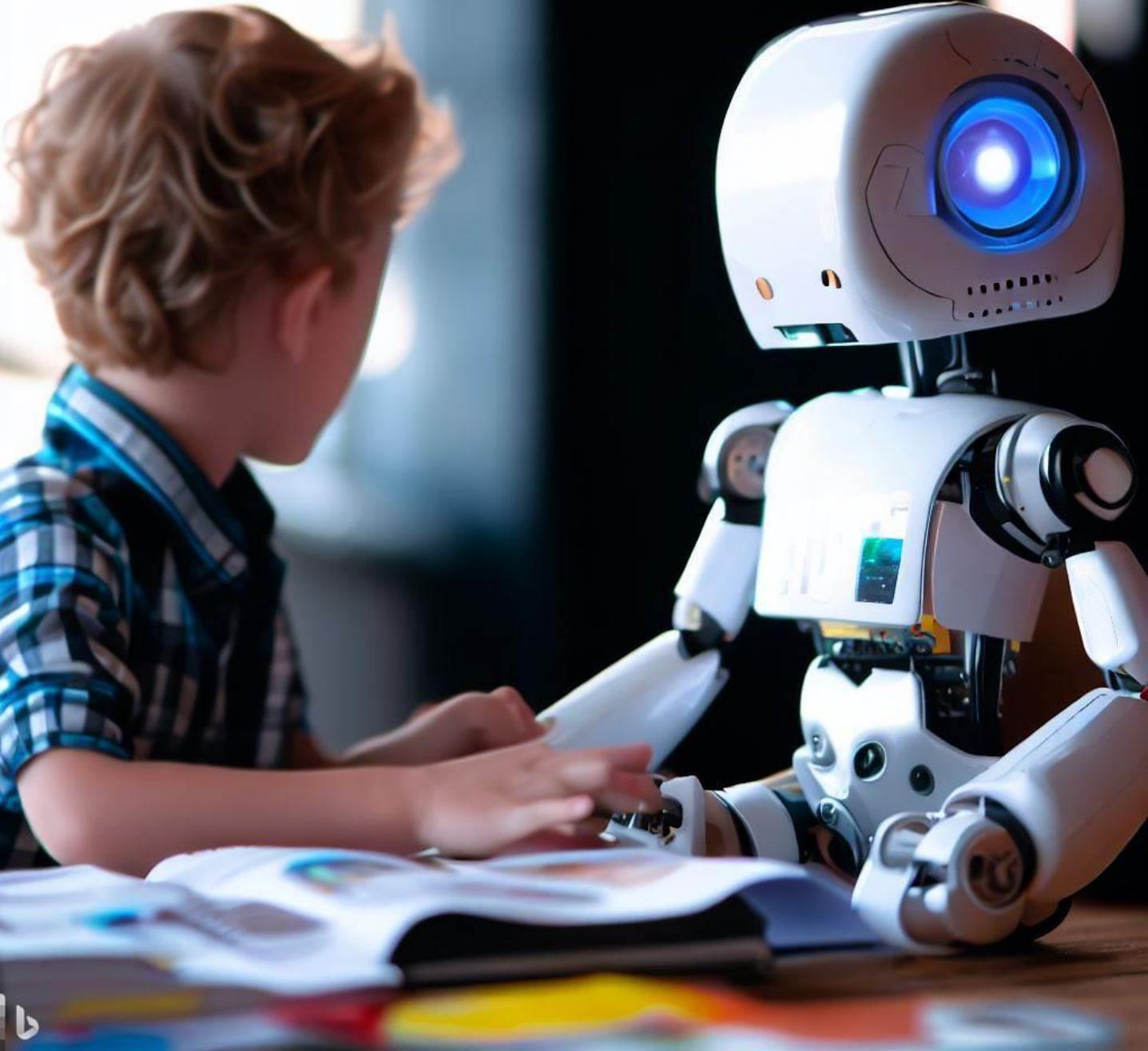
- Kontaktní centrum a podpora
 - Sumarizace, extrakce informací z telefonátů a chatu
- Otázky a odpovědi
 - Index znalostní báze, sémantická podobnost z dotazem, generování komplikované lidsky čitelné odpovědi z dokumentů
- Zkoumání dokumentů
 - Extrakce informací, klasifikace dokumentů, summarizace a porozumění
 - Extrakce komplexních entit a vztahů mezi nimi (např. kdo koho zavraždil v detektivce)
 - Nově s GTP-4 -> práce i s obrazovým vstupem (náčrt, obrázek, rukou psaný text)
- Generování popisu produktu
 - Z metadat (tagy, složení, informace na obalu) vygenerovat popis produktu
- Generování kódu
 - Generování kódy na základě komentáře, požadavku a kontextu
 - Vysvětlení existujícího kódu, generovaní dokumentace, vyhledání chyby
 - Převod do jiného programovacího jazyka
- Reformulace textu
 - Optimalizace článků pro vyhledávače, přizpůsobení pro typ uživatele
- Sumarizace událostí
 - Shrnutí hry na základě výsledků, přepisu konverzací a komentářů
 - Zápis z meetingů a extrakce akčních kroků

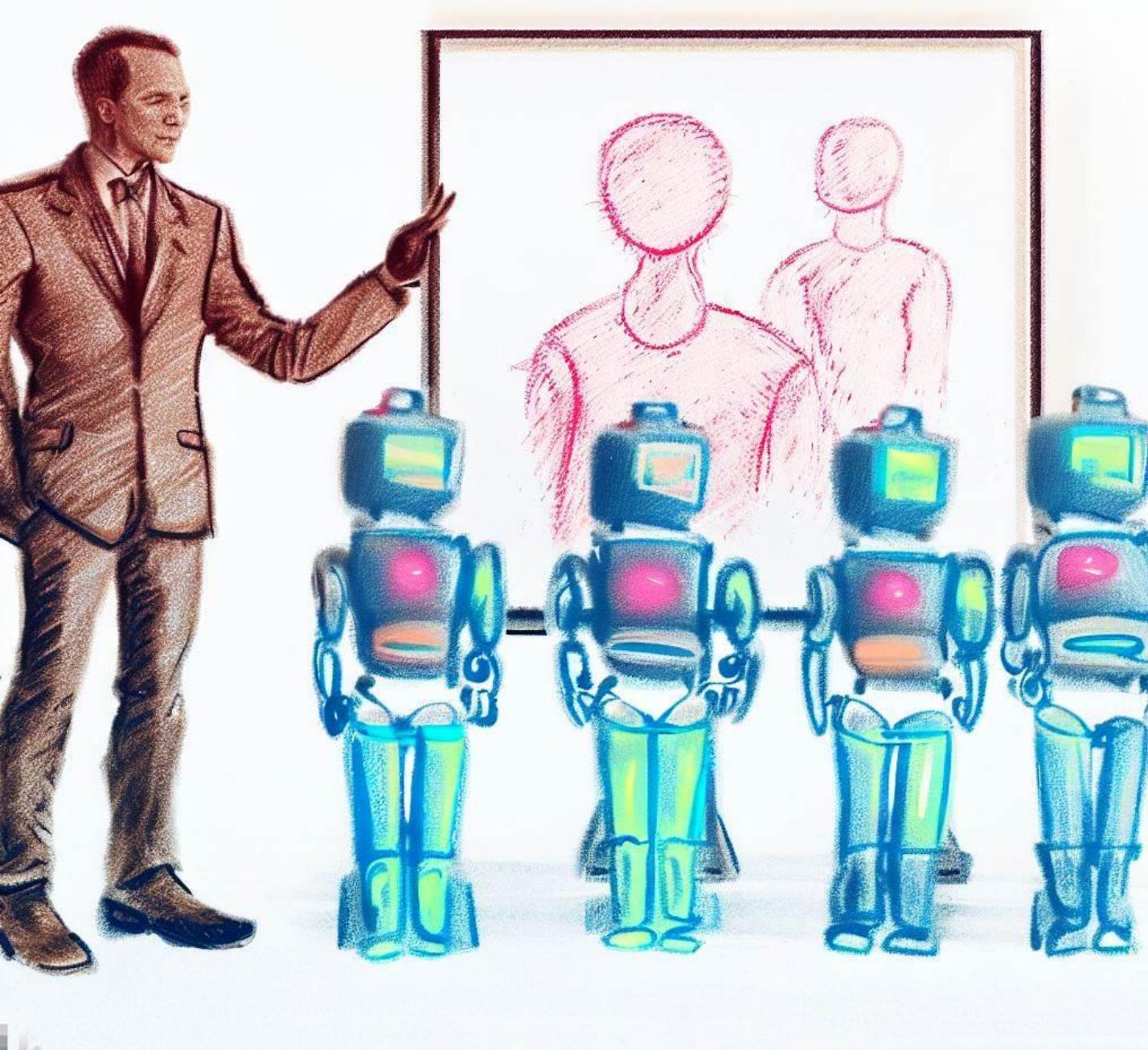


**AI jsou vaše děti.
Někdy dělají veselé
chyby a málo toho
ví.**

Ale jsou děsivě
chytré.

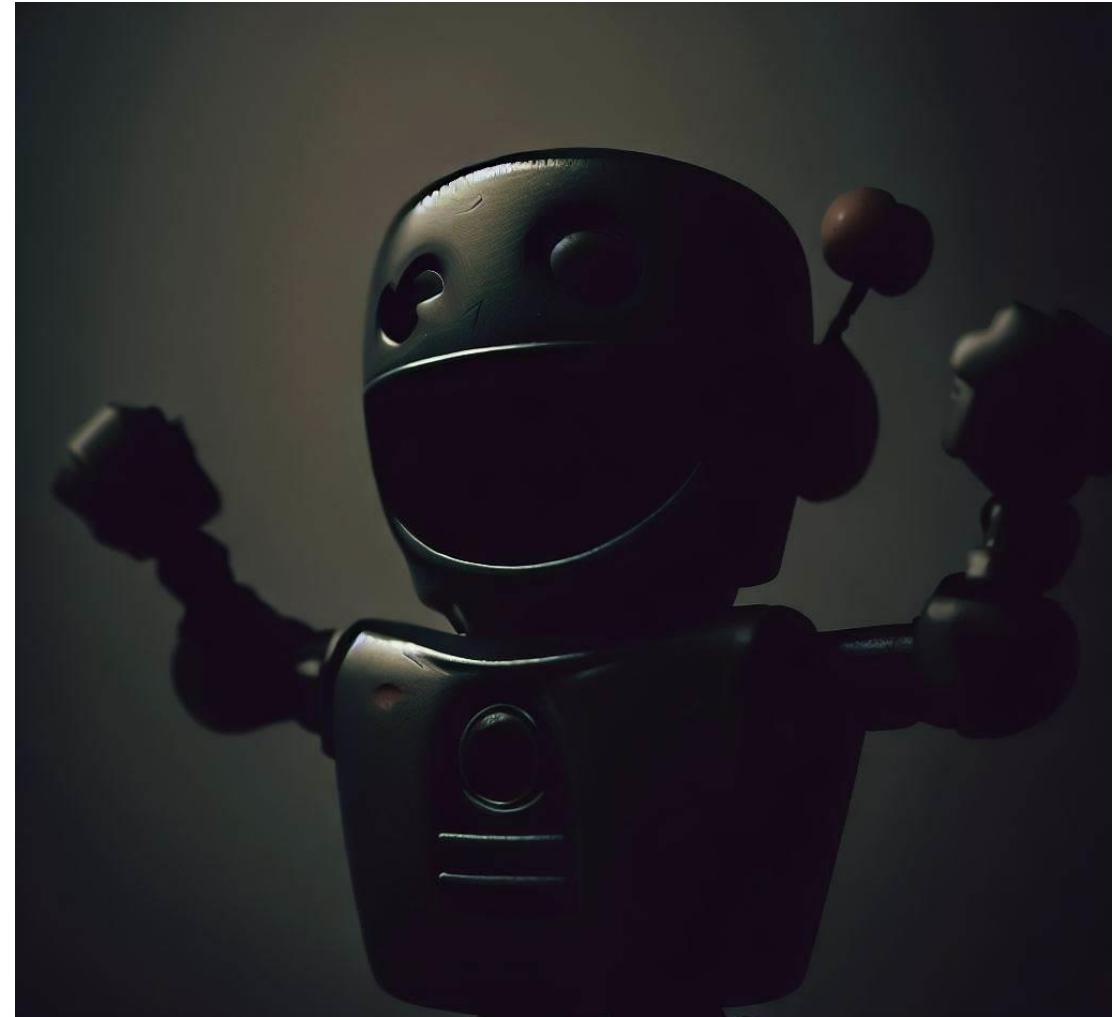
A učí se rychle.





Jste pyšní na to,
co je učíte?

Bud'me dobrými rodiči



Agenda

- Cloud native
- Bezpečnost cloutu a identit
- Umělá inteligence (AI)
- **Osobní tipy jak zůstat relevantní**



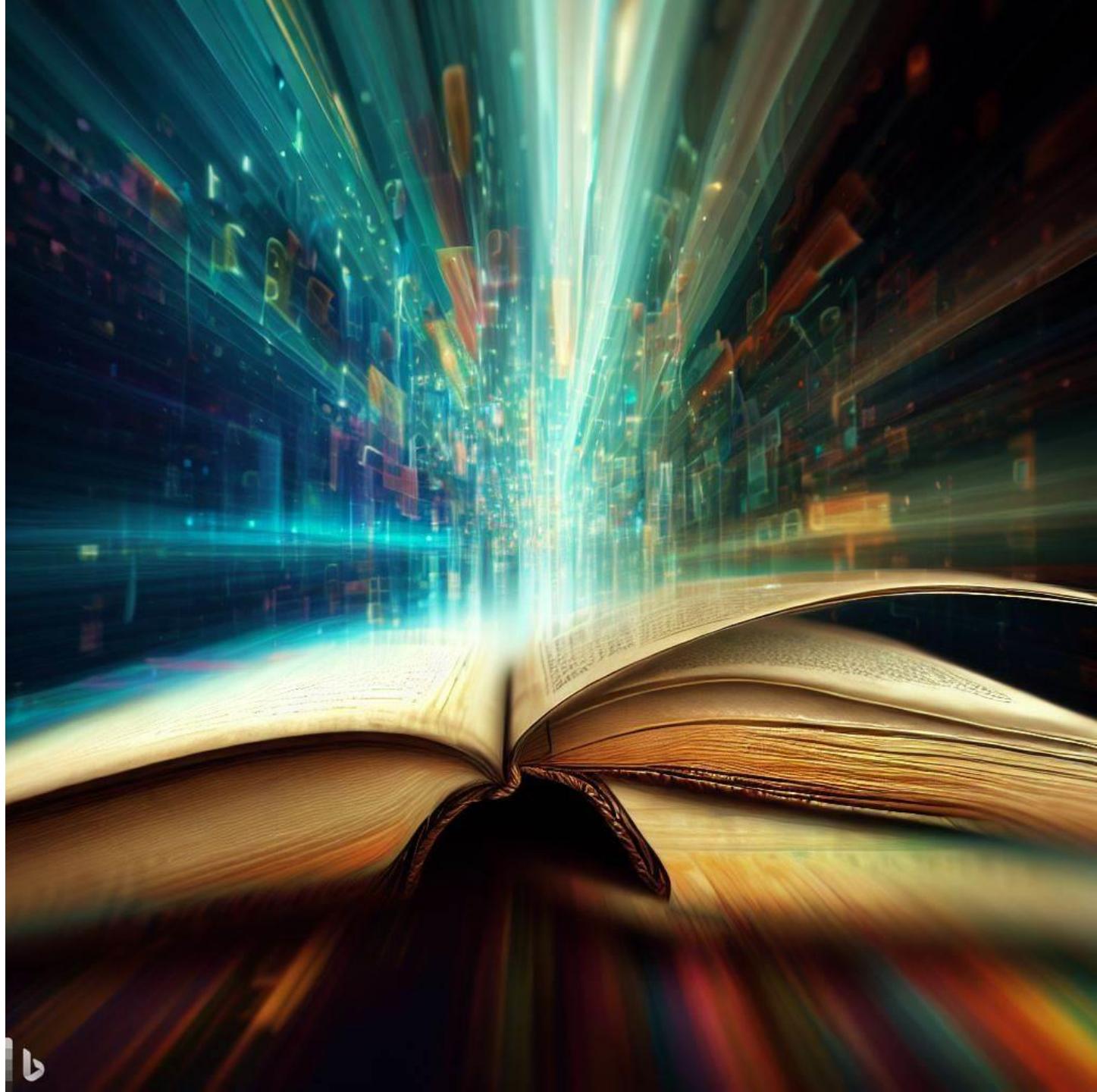
**Bojíte se, že vás
nahradí AI?**

**Spíše to bude
člověk, který bude
umět AI efektivně
využít.**



Kniha / audiokniha
je mnohem
koncentrovanější,
než třeba podcast.

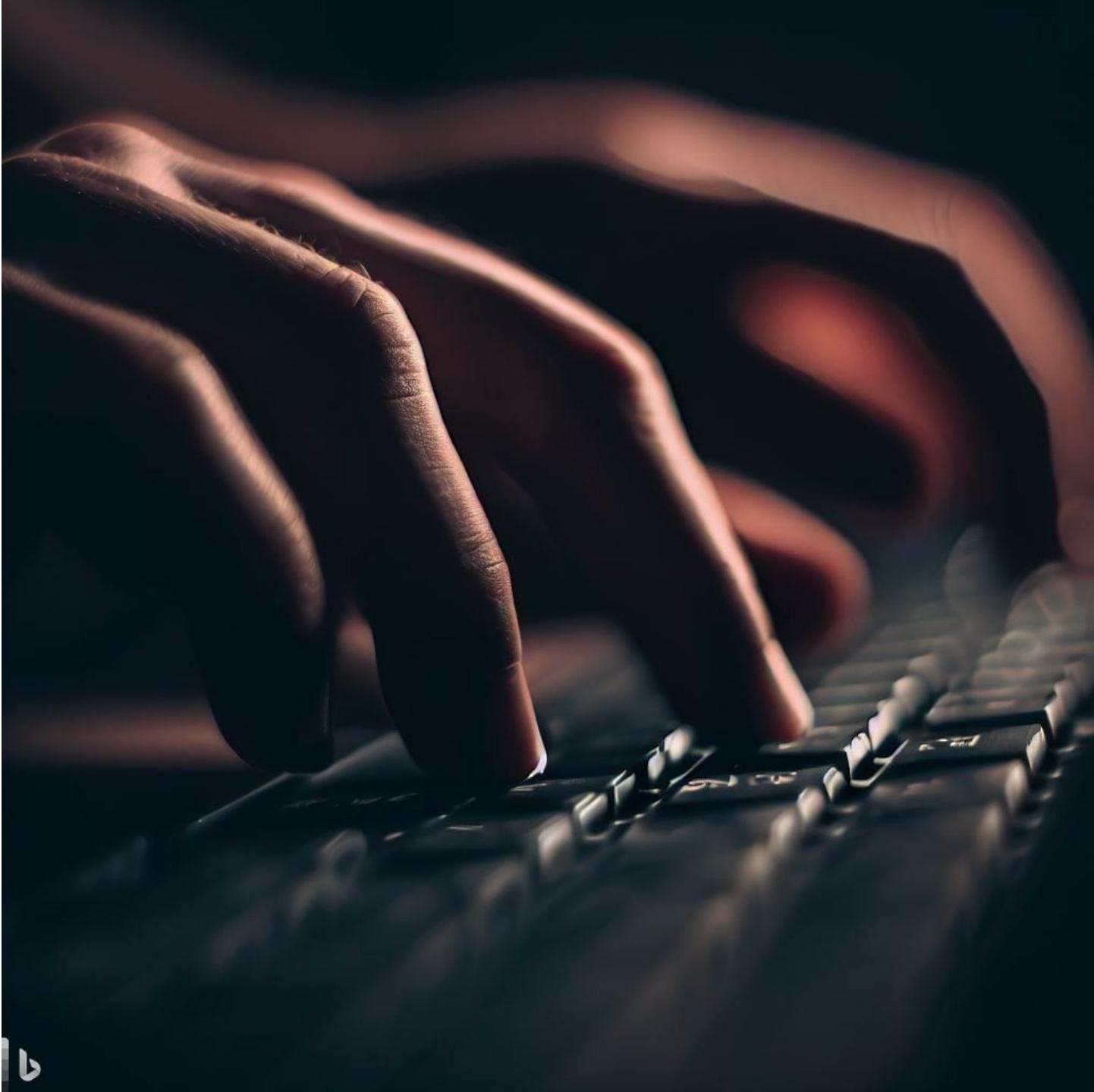
Má nejlepší poměr
čas přípravy vs. čas
konzumace



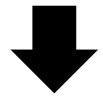
Hluboké podcasty
ale existují, zkuste
Lex Fridmana.



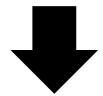
Ruce na klávesnici
naučí nejvíce.



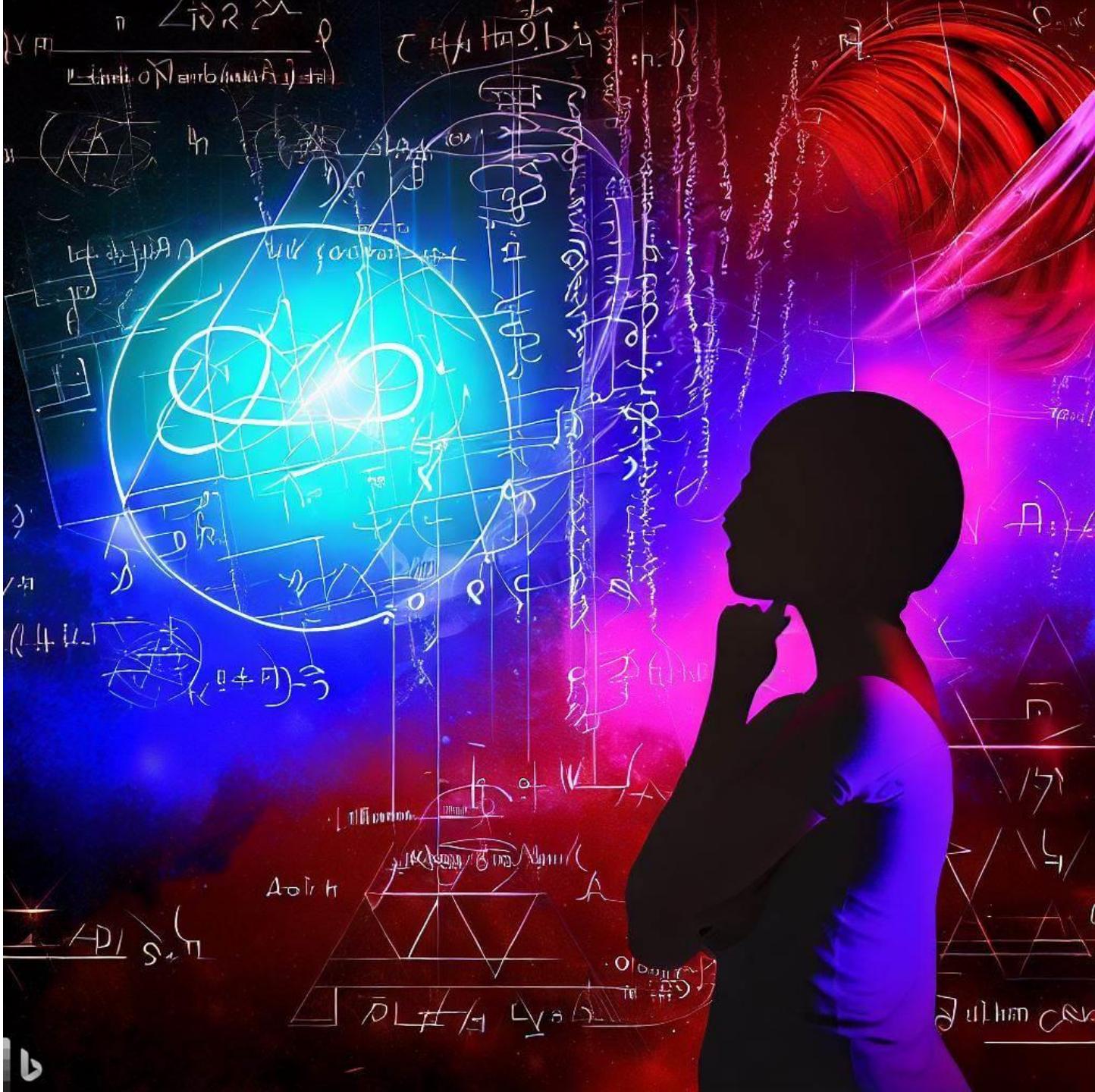
Průzkum detailů
konkrétního výskytu



Studium
generalizace
problému



Aplikace na další a
další výskyty



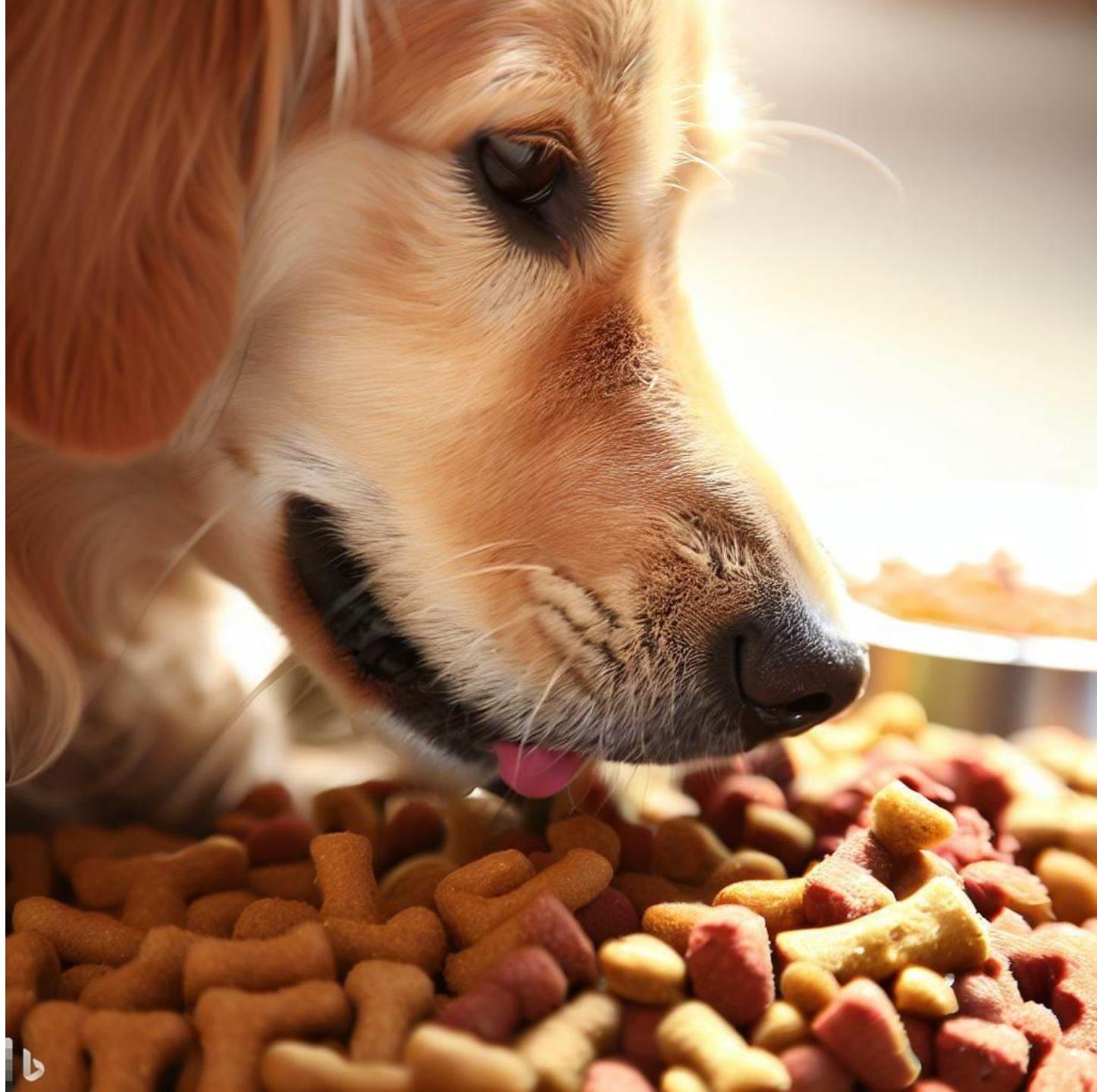
Z 10 let vašeho
života se často stane
jedno tlačítko v
cloudu.

To je normální.



Eat your own dog food

Všechno v IaC,
používejte AI,
nástroje, Git, ...



Nevymýšlejte kolo.

Konkuruje cloutu,
specialistům i Al.



**Největší úspěch
firmě nezajistíte tím,
že vy poběžíte
nejrychleji, ale že
rozpohybujete
ostatní.**



Užívejte si, když
nevíte něco, co
vědět chcete.



Děkuji za pozornost

Tomáš Kubica

A mimochodem všechny
obrázky v této prezentaci
generovalo AI.

