VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

AUTOMATIZACE PROCESŮ PRO ZÁLOHU A OBNOVU KYBERNETICKÉ ARÉNY V ANSIBLE AWX

AUTOMATION OF CYBER ARENA BACKUP AND RECOVERY PROCESSES IN ANSIBLE AWX

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — Bc. Veronika Fišarová

AUTHOR

VEDOUCÍ PRÁCE — Ing. Tomáš Stodůlka

SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Informační bezpečnost**

Ústav telekomunikací

Studentka: Bc. Veronika Fišarová ***ID:*** 211784

Ročník: 2 ***Akademický rok:*** 2022/23

NÁZEV TÉMATU:

Automatizace procesů pro zálohu a obnovu Kybernetické arény v Ansible AWX

POKYNY PRO VYPRACOVÁNÍ:

Hlavním cílem práce je nasadit Ansible AWX a implementovat v něm projekt, který zautomatizuje procedury zálohování a obnovy Kybernetické arény. Analyzujte možnosti nasazení AWX a na základě nejvhodnější volby jej nasaďte na virtuální stroj. Seznamte se s Kybernetickou arénou, která vychází především z cloudové platformy OpenStack. Navrhněte a implementujte kroky zálohující Kybernetickou arénu (včetně struktur vhodných k zálohování) a následně rovněž kroky pro její obnovu z dříve vytvořené zálohy. Obě procedury zautomatizujte a ověřte jejich funkčnost v prostředí simulující Kybernetickou arénu. V rámci implementace obou procedur uvažujte přenositelnost OpenStacku i na jeho novější verze.

DOPORUČENÁ LITERATURA:

[1] STODŮLKA, T.; FUJDIAK, R. Budování Cyber Range platformy s technologií cloud computingu. Brno: Vysoké učení technické v Brně, 2022. 153 s. ISBN: 978-80-214-6064-5.

[2] FREEMAN, J.;KEATING, J. Mastering Ansible. Fourth Edition. Packt Publishing, 2021. ISBN: 978-1801818780.

Termín zadání: 6.2.2023 ***Termín odevzdání:*** 19.5.2023

Vedoucí práce: Ing. Tomáš Stodůlka

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

ABSTRAKT

Diplomová práce se věnuje cyber rangi Kybernetické aréně a cloudové platformě OpenStack, nad kterou je Kybernetická aréna postavena. Teoretická část práce se věnuje třem esenciálním technologiím používaných při stavbě cyber rangů. Těmito technologiemi jsou virtualizace, kontejnerizace a cloud computing. V další části je rozbor zaměřen na platformu OpenStack a její podstatu v Kybernetické aréně. Následuje představení automatizační platformy Ansible AWX, která zaujímá prostředníka mezi Kybernetickou arénou a OpenStackem. Poslední část teorie je věnována problematice záloh a obnov OpenStacku s cílem zkvalitnit životní cyklus Kybernetické arény. V praktické části je popsána tvorba vývojového prostředí, skládající se z OpenStack platformy a platformy AWX tak, aby co nejvíce odpovídalo prostředí s Kybernetickou arénou. V tomto prostředí je následně implementován Ansible projekt, který automatizuje vytváření záloh a následnou obnovu hlavních služeb OpenStacku. Projekt je implementován tak, aby byly procesy přenositelné i na nové verze OpenStacku při zachování stejné metody nasazení.

KLÍČOVÁ SLOVA

Ansible, automatizace, AWX, cyber range, cloud computing, kontejnerizace, Kybernetická aréna, kybernetické hry, OpenStack, virtualizace

ABSTRACT

The master thesis focuses on the cyber range Cyber Arena and the OpenStack cloud platform, on which the Cyber Arena is built. The theoretical part of the work is dedicated to three essential technologies used in the construction of cyber ranges. These technologies are virtualization, containerization, and cloud computing. In the next part, the analysis focuses on the OpenStack platform and its essence in the Cyber Arena. This is followed by an introduction to the Ansible AWX automation platform, which acts as an intermediary between the Cyber Arena and OpenStack. The last part of the theory is dedicated to the issues of backups and recovery of OpenStack with the aim of improving the life cycle of the Cyber Arena. In the practical part, the creation of a development environment is described, consisting of the OpenStack platform and the AWX platform, in such a way as to best match the environment with the Cyber Arena. In this environment, an Ansible project is then implemented, which automates the creation of backups and subsequent recovery of the main OpenStack services. The project is implemented in such a way that the processes are transferable to new versions of OpenStack while maintaining the same deployment method.

KEYWORDS

Ansible, automatization, AWX, cyber range, cloud computing, containerization, Cyber Arena, cyber games, OpenStack, virtualization

FIŠAROVÁ, Veronika. *Automatizace procesů pro zálohu a obnovu Kybernetické arény v Ansible AWX*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2023, 93 s. Diplomová práce. Vedoucí práce: Ing. Tomáš Stodůlka

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Bc. Veronika Fišarová
VUT ID autora:	211784
Typ práce:	Diplomová práce
Akademický rok:	2022/23
Téma závěrečné práce:	Automatizace procesů pro zálohu a obnovu Kybernetické arény v Ansible AWX

Prohlašuji, že svou závěrečnou práci jsem vypracovala samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autorka uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušila autorská práva třetích osob, zejména jsem nezasáhla nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědoma následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autorky*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Ráda bych poděkovala vedoucímu diplomové práce panu Ing. Tomášovi Stodůlkovi za odborné vedení, přívětivé konzultace, trpělivost, podnětné návrhy k práci a pozitivní přístup během celého řešení diplomové práce.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů

Úvod

Informační a komunikační systémy v současnosti zasahují do všech sfér společnosti a postupně splynuly s každodenními potřebami jedince. De facto každý člověk je připojen do veřejné sítě Internet a do této sítě sdílí své informace. Při každém spojení s Internetem jedinec projevuje důvěru v daný systém a spoléhá na dostatečnou implementaci bezpečnostních mechanismů, které mají jeho poskytnuté informace ochránit od neoprávněného přístupu. Pojmem informace je myšlena jakákoliv vazba k tomuto jedinci – osobní údaje, poloha, bankovní účet... Těchto informací nacházejících se v Internetu je v současnosti obrovské množství. Samozřejmě se nejedná pouze o informace o lidských jedincích, ale i informace o neživých systémech, které je nutné zabezpečit.

Zejména v posledních třech letech se projevilo, že omezení funkčnosti ICT systémů v mnoha odvětvích vyústí v kritické situace, které mohou přímo ohrožovat životy. Viz útok na české nemocnice v roce 2020 v době pandemie COVID-19 [1]. Právě v kritickém období se útoky na systémy, které jsou zásadní pro fungování lokálních nebo národních organizací, značně zvyšují neboť útočníci využívají tato období ke zvýšení úspěšnosti svého útoku, ať už se jedná o získání finančních prostředků nebo vyvolání nedůvěry v tyto systémy.

Válka na Ukrajině ještě více než pandemie COVID-19 potvrdila, že ICT systémy mají zásadní vliv na průběh konfliktu. Pomocí ICT systémů může docházet k e-špionáži, propagandě, ekonomické manipulaci nebo „pouze“ k omezení samotné komunikace a následkem těchto narušení může dojít k obratu v aktuálním konfliktu. Národní obrana jednotlivých států si více uvědomuje podstatu ochrany ICT systémů a důležitých entit, které ICT systémy potřebují ke svému fungování, a proto je zajištění a monitorování kyberbezpečnosti státu často v režii armády a organizací, které mají obdobné pravomoci.

Podle slov Dr. Kennetha Geerse[1] je pravděpodobné, že možné budoucí konflikty budou testovat více zabezpečení kyberprostoru než sílu a zbrojní nadvládu [2]. Tato předpověď má za důsledek větší financování zabezpečení ICT systémů a také větší důraz na přípravu expertů starající se o bezpečnost ICT systémů a monitorování pro co nejvčasnější odhalení nadcházejícího útoku. Za účelem nejlepší připravenosti jsou budovány tzv. cyber range, jejichž účelem je vytvořit realistické prostředí, které slouží k simulaci útoků na ICT sítě s cílem vyzkoušet co nejvíce možných scénářů útoků a jejich úspěšné mitigace. Cyber range schopné vytvořit realistické prostředí se staly základem pro bezpečnostní cvičení, které se často ode-

[1] Dr. Kenneth Geers je kyberbezpečnostní expert NATO, který publikoval mnoho literatury týkající se novodobých válek v kyberprostoru a aktivně se zabývá školením expertů v kyberbezpečnosti.

hrávají na mezinárodní úrovni a často mají podobu herních scénářů. Příkladem těchto cvičení je například Locked Shields, které je realizováno v Estonském cyber rangi, kterého se za rok 2022 zúčastnilo 2000 expertů z 32 států [3]. V České republice je každým rokem organizováno bezpečnostní cvičení Cyber Czech, které je realizováno v cyber rangi Kybernetický polygon [4].

Diplomová práce se zabývá technologickými aspekty cyber range a technologiemi využívanými pro jeho budování a správu. Konkrétně je práce zaměřená na cyber range Kybernetická aréna, která byla vybudována na Ústavu telekomunikací Vysokého učení technického v Brně.

V první kapitole práce je čtenář seznámen se třemi základními technologiemi, které se používají k budování cyber rangů.

V druhé kapitole je čtenář seznámen s Kybernetickou arénou, jsou rozebrány cíle Kybernetické arény a způsoby splnění těchto cílů. Zároveň jsou představeny obdobné cyber range a trend jejich vývoje.

Třetí kapitola je věnována cloudové platformě OpenStack, která je stavebním kamenem Kybernetické arény a zajišťuje virtualizaci a orchestraci kyberbezpečnostních her. Rozebrány jsou podrobněji nejzásadnější služby OpenStacku, metody nasazení a odlišnosti těchto metod.

Ve čtvrté kapitole je představena platforma Ansible AWX, která slouží jako prostředník mezi Kybernetickou arénou a OpenStackem a automatizuje procesy zabývající se zejména tvorbou herních scénářů a automatizace životního cyklu Kybernetické arény.

Pátá kapitola se zabývá problematikou obnovy a zálohy částí OpenStacku. Podstatou této kapitoly je přiblížit čtenáři možnosti zálohy a obnovy takových částí OpenStacku, které jsou pro Kybernetickou arénu stěžejní. Cílem je navrhnout takovou implementaci záloh a obnov, která poskytne jak vyšší přenositelnost Kybernetické arény, tak možnost obnovení určitých částí OpenStacku, u kterých při úpravě OpenStacku hrozí riziko znehodnocení funkčnosti. Analyzovány jsou tři hlavní přístupy k implementaci procesů zálohy a obnovy. Šestá kapitola popisuje vytvoření vývojového prostředí s OpenStackem, AWX a následně vlastní implementaci Ansible projektu zálohy a obnovy. Celý postup implementace je vysvětlen, včetně podstatných konfiguračních částí a testování funkčnosti celé implementace. Praktická část je doplněna o příručku nasazení AWX a jeho následného ovládání.

1 Cyber range platformy

Vytvoření CR (cyber range) platformy, která je schopna účastníkům poskytnout dostatečnou simulaci reálného prostředí, je komplexní proces, který vyžaduje hlubokou znalost mnoha technologií, nástrojů a jejich konfigurací. Pro škálovatelnost, rozšiřitelnost a udržitelnost CR je doporučeno institucí NIST[1] (Národní institut standardů a technologie[2]) vyvíjet CR nad virtuální infrastrukturou. Myšleno tak, že místo fyzických zařízení, které jsou součástí herního scénáře (uživatelské zařízení, router, firewall atd.), jsou vytvořena virtuální zařízení, která práci těch fyzických simulují. Tímto je dosaženo, že CR může být dynamicky přizpůsobován v závislosti na scénáři a dedikovanému účelu – vzdělávání, testování nebo výzkum. Ocenitelnou výhodou tohoto přístupu je možnost vytváření zálohy virtuální infrastruktury a jejího nastavení. Tudíž je velmi jednoduché CR měnit podle jeho účelu a je ušetřen čas, který by byl spotřebován pro úpravu fyzických zařízení [5].

1.1 Obecná architektura cyber range

Podle cíle, který má CR naplnit, se mohou měnit jeho poskytované funkce, použité nástroje a celková architektura. V posledních dvou dekádách bylo CR věnováno velké množství výzkumu věnující se implementaci různých technologií, které by co nejvíce umožňovaly splnění výše doporučených vlastností CR.

Na obrázku 1.1 je zobrazena základní architektura CR podle NIST, ze které je doporučeno vycházet [6, 7]. Z obrázku je patrné, že architektura je rozdělena do tří vrstev - vrstva hardwarová, orchestrační vrstva a vrstva uživatelského rozhraní.

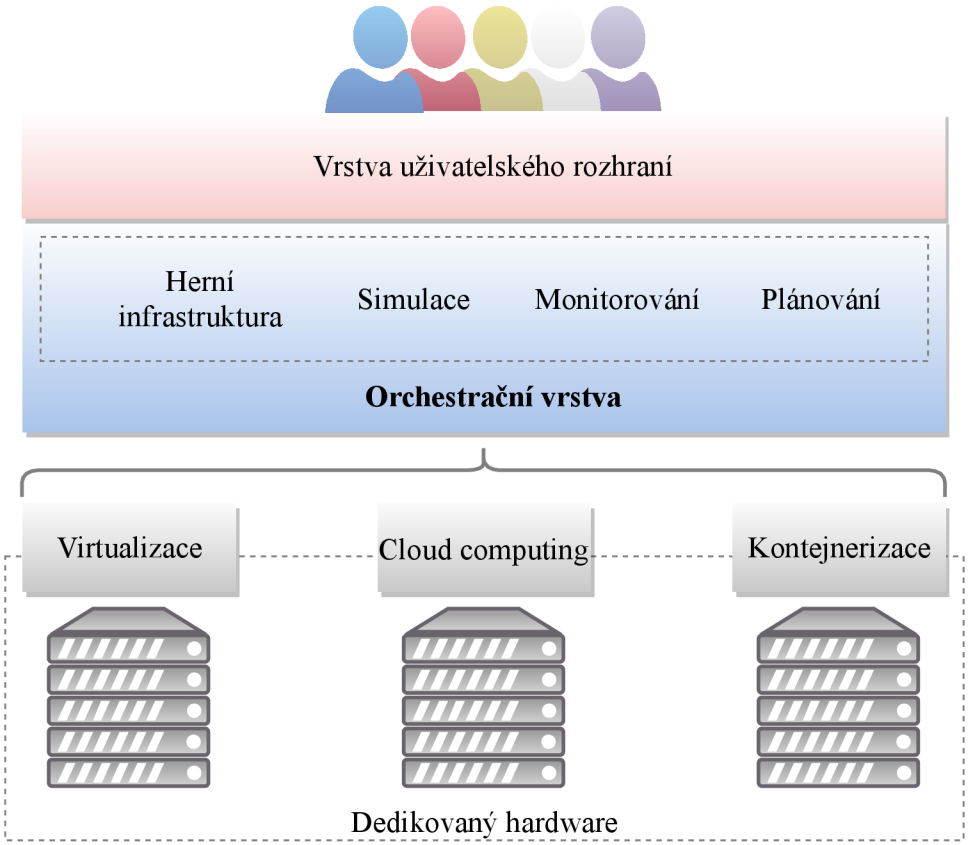
Hardwarová vrstva reprezentuje skutečná fyzická zařízení, která jsou součástí CR. Každý CR je postaven nad množinou síťových zařízení, serverů, úložišť a bezpečnostních prvků[3]. Účelem hardwarové vrstvy je poskytnutí výpočetních a paměťových prostředků k operacím, kterými má CR disponovat.

CR nad daným hardwarem vykonává mnoho operací pro zpravidla mnoho uživatelů, o distribuci výpočetních a paměťových prostředků pro tyto operace se stará **orchestrační vrstva**. Tato vrstva je stěžejní pro schopnosti a spolehlivost CR a její specifikace se liší již od použité technologie, která hardwarové prostředky poskytuje k využití vyšším vrstvám.

[1] Konkrétněji oddělení NICE (National Initiative for Cybersecurity Education), které pod NIST spadá.

[2] Anglicky *National Institute of Standards and Technology.*

[3] Bezpečnostními prvky jsou myšleny např. firewally, honey poty, monitorovací zařízení - kamery a zařízení pro kontrolu vstupu. Zahrnutí bezpečnostních prvků závisí od fyzické lokace hardwaru a také účelu CR. CR nacházející se v prostorách škol bude mít jiné podmínky zabezpečení než CR sloužící národní bezpečnosti apod.



Obr. 1.1: Ideová architektura cyber range podle NIST

Orchestrační vrstva také definuje schopnosti, respektive funkce CR, které se liší podle samotného účelu CR. Je to tedy například správa herní infrastruktury ve smyslu vytváření sítí, spouštění virtuálních stanic, automatizovaná konfigurace zařízení apod. Další funkcí je simulace, tímto je myšlena například simulace internetových služeb, simulace útoku (například generování záplavy zpráv), generování běžného síťového provozu apod. Monitorování průběhu scénářů nebo hráčského pokroku je další základní funkcí CR. Obdobně plánování, které souvisí se spouštěním scénářů a následně operací souvisejících s konkrétním scénářem.

Vrstva uživatelského rozhraní zprostředkovává interakci uživatelů s cyber rangem. Funkce uživatelského rozhraní přímo souvisí s funkcemi CR. V kontextu CR jsou uživatelé přistupující k CR rozděleni do týmů označených barvou. Toto rozdělení se liší podle typu hry, kterou účastníci hrají. Nejčastější typy her jsou červený tým proti modrému a *capture the flag* (CTF). Více informací o kybernetických hrách a týmech jsou dostupné v literatuře [8].

Následující podkapitoly popisují technologie, které jsou stěžejní pro práci orchestrační vrstvy a definují možnosti funkcí celého CR. Těmito technologiemi jsou virtualizace, kontejnerizace a cloud computing (viz obrázek 1.1).

1.1.1 Virtualizace

Virtualizace umožňuje vytvářet simulované prostředí z jednoho nebo více fyzických zdrojů tak, že je možné dynamicky distribuovat výpočetní a paměťové prostředky pro potřebu tohoto prostředí. Těchto prostředí může být na jednom fyzickém zdroji teoreticky nekonečné množství a každé takové prostředí je zcela izolované od ostatních. Prostředí s touto charakteristikou je nazýváno virtuální stroj[4] (VS). Pro současné informační systémy a komunikační infrastruktury se s rostoucí výpočetní a paměťovou náročností stala virtualizace jejich esenciální součástí pro zachování dlouhodobé udržitelnosti, jak z hlediska ekonomického, tak z pohledu správy těchto systémů. Správně nastavený virtuální stroj potom dosahuje takové úrovně, že jeho chování může být nerozeznatelné od fyzického prostředí [9]. Výhody virtualizace jsou shrnuty do těchto bodů:

- redukce provozních nákladů,
- optimalizace distribuce fyzických zdrojů,
- jednodušší správa systémů,
- lepší přístupnost pro testování a vývoj,
- efektivnější záloha a obnova systému,
- bezpečnost.

Virtualizace je rozdělena podle toho, na jaké úrovni architektury systému (počítače, serveru) je implementována. S tímto rozdělením také souvisí typ hypervizoru, který virtualizaci, respektive virtuální stroje, spravuje.

V případě, že je virtualizace implementována přímo na hardwaru počítače, jedná se o **nativní**[5] virtualizaci a tento typ je spravován hypervizorem typu 1. Představitelé tohoto typu hypervizoru jsou například VMware ESXi, Microsoft Hyper-V a Red Hat KVM (Kernel-Based Virtual Machine) [10, 11, 12].

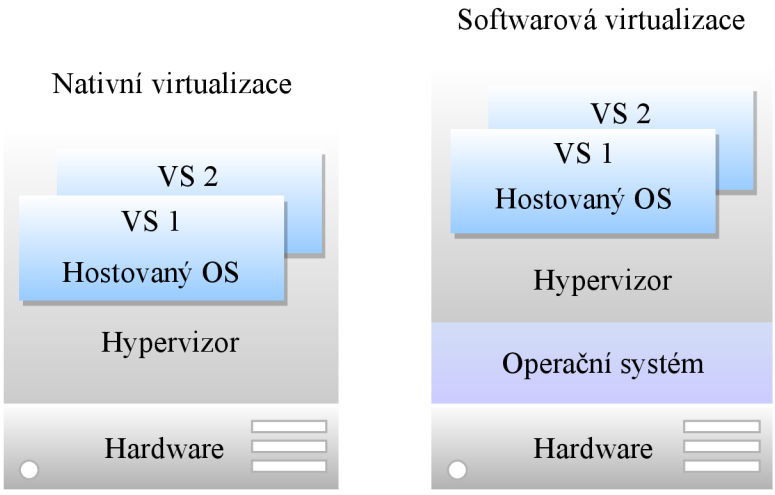
Pokud je virtualizace implementována v prostředí operačního systému (OS), jedná se o **softwarovou virtualizaci** a hypervizor operuje až nad vrstvou operačního systému, nemá tedy přímé spojení s hardwarem počítače. Z tohoto důvodu je tento typ virtualizace pomalejší, protože veškeré žádosti o poskytnutí hardwaru musí být prvně obslouženo hostitelským OS. V tomto případě virtuální stroje spravuje hypervizor typu 2[6]. Představitelé tohoto typu hypervizoru jsou například VMware Player a Oracle VirtualBox [13, 14].

Umístění obou typů hypervizorů v rámci architektury počítače je zobrazeno na obrázku 1.2 [15].

[4] Anglicky *virtual machine*.

[5] Anglicky *native* nebo *bare metal* virtualizace.

[6] Hostovaný hypervizor, anglicky *hosted hypervisor*.



Obr. 1.2: Schéma rozdílných typů virtualizace

1.1.2 Kontejnerizace

Kontejnerizace, stejně jako virtualizace, umožňuje vytvářet izolované pracovní prostředí, ale oproti virtualizaci nedisponuje vlastním OS, ale používá jádro hostitelského OS. Kontejnerizace spočívá v zabalení konkrétního softwaru nebo aplikace, s nezbytnými komponentami, kterými jsou knihovny, závislosti, proměnné atd. do izolovaného prostředí, tzv. kontejneru. Kontejnerizovaný software může být následně spouštěn bez ovlivnění hostitelského systému. Tato technologie je výhodná zejména v systémech, které jsou citlivé na změny systémových nastavení, nebo v systémech, ve kterých jsou pro běh konkrétního softwaru potřebné rozdílné konfigurace, než které používá hostitelský OS. Z toho také vyplývá jednoduchá přenositelnost kontejnerů mezi hostitelskými systémy, ovšem pouze v rámci kompatibility s architekturou jiného OS [16, 17]. Výhody kontejnerizace jsou shrnuty do těchto bodů:

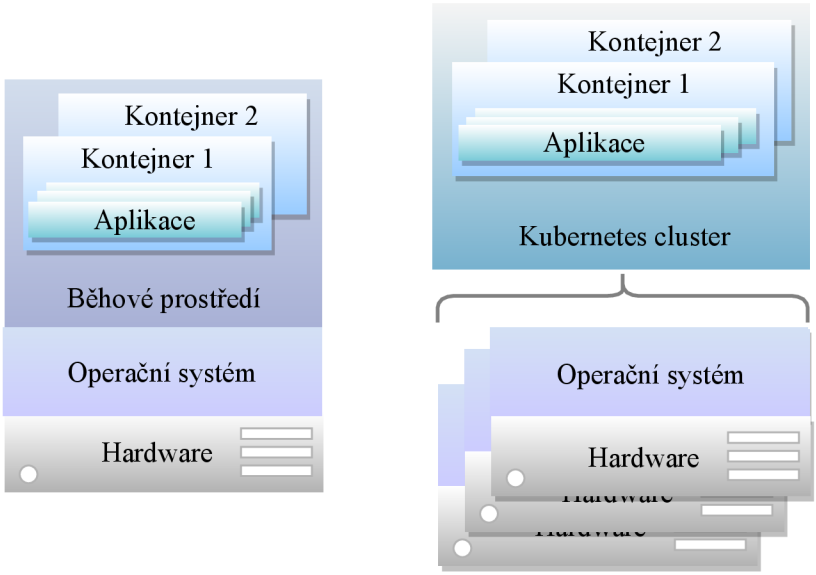
- izolace chyb kontejnerizované aplikace
- nižší výpočetní náklady než u virtualizace,
- přenositelnost mezi OS,
- menší režie než u virtualizace,
- bezpečnost.

Kontejnery poskytují určitou míru bezpečnosti, jelikož je jejich běh oddělen od hostitelského OS a nastavení prostředí kontejneru neovlivňuje nastavení hostitelského OS. Virtualizace ale poskytuje mnohem větší izolaci, jelikož umožňuje vytvoření tzv. *sandboxů*[7] [18].

Správu kontejnerů zajišťuje software[8], který kontejnerům poskytuje tzv. běhové

[7] Sandbox je kontrolované prostředí, které silně omezuje přístup do hostitelského systému.
[8] Anglicky *container management tool*.

prostředí[9]. Nejpříkladnějším zástupcem softwaru pro správu kontejnerů je Docker, kterým byly inspirovány další nástroje, například Podman a LXD, který je navržený speciálně pro systémy Linux [19, 20, 21]. Pro kontejnery, které běží jako součást clusterů je vhodné použít software, který celý cluster umožňuje spravovat z jediného zařízení. Příkladem takového nástroje je Kubernetes, který ke správě využívá Docker kontejnery [22]. Na obrázku 1.3 je znázorněno schéma kontejnerizace v rámci hostitelského počítače.



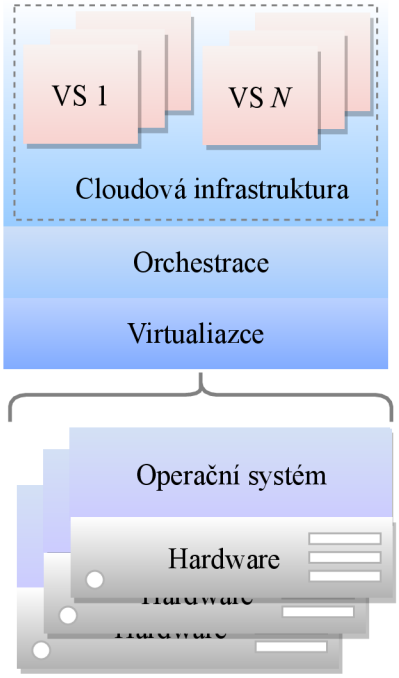
Obr. 1.3: Schéma kontejnerizace a Kubernetes cluteru

1.1.3 Cloud computing

Přesto, že je virtualizace klíčovou technologií pro tvorbu realistických cyber rangů, jejím nedostatkem je, že nedokáže přerozdělit nevyužité výpočetní a paměťové zdroje na celé infrastruktuře, která je z clusterů složena. Cloud computing z virtualizace vychází, avšak abstrahuje fyzické zdroje napříč celou infrastrukturou a zároveň tyto zdroje spravuje. Cloud poskytuje komplexní správu výpočetních zdrojů, obsluhu VS, vytváření sítí, řízení přístupu a další s ohledem na poskytované služby specifického cloudu. Veškeré služby a procesy poskytované a řízené cloudem pro zajištění daného cíle se nazývají orchestrace[10]. Architektura cloud computingu je znázorněna na obrázku 1.4.

[9] Anglicky *container runtime*.

[10] Jinými slovy orchestrace je množina úkolů a procesů, které vedou k vytyčenému cíli, přičemž cesty, které k cíli povedou „rozhoduje“ cloud. Automatizace je zpravidla plnění úkolů bez zásahu uživatele, přičemž tyto kroky vedoucí ke splnění uživatel přesně definoval [23].



Obr. 1.4: Obecná architektura cloud computingu

Nejnižší vrstvou jsou fyzická zařízení, jejichž zdroje jsou předány virtualizační vrstvě, která poskytuje abstrakci hardwarových prostředků tak, že je možné s nimi pracovat jako celek. Cloudová infrastruktura je množina nástrojů, která poskytuje uživateli rozhraní pro použití těchto hardwarových prostředků. Ty jsou využity libovolným způsobem - tvorba VS (virtuální stroj), virtuálních sítí nebo kontejnerů. Orchestrace je implementována v každé vrstvě cloudu. Jedná se o nejkritičtější část cloudu, jelikož se stará o jeho veškerou správu[11] [26].

Využití cloud computingu poskytuje CR platformám škálovatelnost a flexibilitu, což omezuje tvorbu scénářů pouze na představivost správců a použitelnosti technologií vyšších vrstev. K zapojení cloudu do architektury CR platformy je možné přistoupit dvěma odlišnými způsoby. Pokud je cloud computing poskytován od veřejného poskytovatele, jedná se o **veřejný cloud**. V takovém případě jsou služby, výpočetní a paměťové prostředky zpoplatněny, tento princip je zpravidla nazýván *pay as you go*. Nezanedbatelnou výhodou veřejného cloudu je možnost okamžité tvorby samotného CR a celkově odpadá nutnost správy a údržba cloudu, která může být zejména zpočátku časově náročná a komplikovaná, aby cloud poskytoval spolehlivé služby. Další výhodou může být úleva od vydání nemalých finančních prostředků pro hardware cloudu, nicméně toto může být velmi individuální a je nutné porovnat dlouhodobé výdaje za pronájem cloudu od poskytovatele. Příkladem poskytovatelů je AWS (Amazon Web Services), Microsoft Azure nebo Google Cloud Platform

[11] Anglicky *cloud management*.

[24, 27, 28, 29].

Opačným přístupem k využívání cloud computingu je tzv. **privátní cloud**. V tomto případě si uživatel cloud implementuje do své infrastruktury na vlastní hardware a správu cloudu zajišťuje vlastními prostředky. V případě privátního cloudu je největší výhodou naprostá kontrola nad daty, se kterými cloud pracuje. Uživatel si je vědom, kde se data fyzicky nacházejí a kdo k nim má přístup. V tomto případě není nutné se na zajištění ochrany a integrity dat spoléhat na poskytovatele. Nevýhodou privátního cloudu je ztráta flexibility, respektive pokud je nutné přidat další výpočetní uzel, nebo pokud nastane porucha na uzlu cloudu, je nutný fyzický přístup k hardwaru, což může vyústit v dočasné omezení poskytovaných služeb. Správce privátního cloudu by měl zajistit patřičné zálohy dat a záložní uzly, aby v případě poruchy nedošlo zejména ke ztrátě dat uživatelů a případně ztrátu dat, které jsou podstatné pro správné fungování cloudu. Příkladem privátních cloudů je například IBM Cloud, který nabízí jak veřejný, tak privátní cloud v různých cloudových modelech[12], OpenNebula, OpenStack a OpenShift[13] [24, 30, 31, 32]. OpenStack je open-source cloudové řešení a je detailněji rozebrán v kapitole 3.

[12]Cloudové modely - *infrastructure as a service*, *software as a service*, *platform as a service*; více o modelech v literatuře [25].

[13]OpenShift je založen na technologii Kubernetes.

2 Kybernetická aréna

V této kapitole je představena CR platforma Kybernetická aréna, její architektura, funkce a stěžejní technologie, které jsou podrobněji rozebrány v následujících podkapitolách.

Kybernetická aréna je projekt, který vznikal na Vysokém učení technickém v Brně v letech 2019–2022. Projekt spadá do oblasti výzkumu a vývoje řešení v programu Bezpečnostní výzkum České republiky, jehož poskytovatelem je Ministerstvo vnitra [33].

Účelem vzniku Kybernetické arény je edukace, výzkum a testování. Samotným vývojem CR jsou získány zkušenosti, které nejsou dostupné z teoretické roviny. Výsledkem projektu je obsáhlá literatura [34], která tyto nabyté zkušenosti sdílí. Implementovaný CR umožní vývoj nového softwaru nebo jeho testování v zabezpečeném prostředí. CR bude také zaveden do studijních programů k získání praktických zkušeností v informační bezpečnosti (síťová bezpečnost, penetrační testování apod.).

Veškeré podrobné informace o Kybernetické aréně a jejím budování jsou k dispozici v literatuře [34].

2.1 Požadavky Kybernetické arény

Jak bylo uvedeno výše, účelem vzniku Kybernetické arény je edukace, výzkum a testování. Aby byla Kybernetická aréna schopna naplnit tyto cíle, musí umožňovat:

- virtualizaci infrastruktury,
- virtualizaci koncových zařízení typu SCADA/ICS[1],
- vytváření sandboxů, respektive virtuálních strojů nebo kontejnerů s podporou sandboxingu,
- oddělení her pro studenty od výzkumných projektů, s podporou dynamického přidělování výpočetních zdrojů [34].

Z těchto požadavků vyplývá nutnost použití virtualizace. Projekt Kybernetická aréna předpokládá různorodé a dlouhodobé využití, a tedy i měnící se spotřebu hardwarových prostředků. Budování Arény na veřejném cloudu by postupně vedlo ke zvyšování nákladů za jeho pronájem [36]. Z tohoto důvodů plyne, že nejlepším řešením je Arénu budovat s využitím privátního cloud computingu.

[1] Řídící systémy ICS (Industrial Control Systems) je označení pro procesy zajišťující správu průmyslových služeb. Největší skupinou těchto systémů je tzv. SCADA (Supervisory Control and Data Acquisition), což je označení pro software sloužící k monitorování a sběr dat průmyslových zařízení a systémů [35].

2.2 Výběr cloudové platformy

Výběr cloudové platformy je stěžejní krok, podle kterého se odvíjí další vývoj CR. Cloudové platformy se liší v mnoha ohledech a z pohledu vývoje Arény je nejdůležitější analyzovat zejména složitost nasazení cloudu a jeho následnou správu. Dále zhodnotit minimální požadavky pro nasazení na dedikovaný hardware, poskytované služby, možnosti virtualizace, kontejnerizace, tvorby virtuální síťové infrastruktury a dostupnou dokumentaci.

Analýza byla prováděna na třech cloudových platformách - **Kubernetes**[2], **OpenShift** a **OpenStack**. Všechny tři platformy nabízí open-source řešení, a tedy je lze nasadit jako privátní cloud na vlastní infrastrukturu.

Výhodou **Kubernetes** je jednoduchost nasazení a správy, na rozdíl od dalších dvou platforem. Kubernetes se neskládá z velkého množství komponent, což značně ulehčuje řešení problémů. Nevýhodou Kubernetes je, že sám od sebe nepodporuje plnohodnotnou virtualizaci (tvorbu sandboxů) a bylo by nutné doinstalovávat externí nástroj[3]. Další nevýhodou je nemožnost vytvořit směrovače a firewally uvnitř uzlů, ty je možné vytvořit pouze na vstupu a výstupu do uzlu[4].

Základem **OpenShift** platformy je Kubernetes. OpenShift používá sofistikovanější metody správy kontejnerů, a tím je zlepšena bezpečnost (například kontejner nemůže být spuštěn s *root* právy). OpenShift také umožňuje autentizaci a autorizaci přístupu ke kontejnerům a správu přes uživatelské rozhraní. Pro vytváření síťové infrastruktury platí stejná pravidla jako pro Kubernetes - není možné vytvářet směrovače a jiné síťové prvky. Směrovače, stejně jako firewally, je možné nastavit pouze u vstupu do uzlu. Za nevýhodu může být považováno nutnost použití RHEL (Red Hat Enterprise Linux) distribuce Linuxu[5]. Virtualizace je podporována s doinstalování nástroje KubeVirt, nebo nasazením OpenShiftu nad OpenStackem.

OpenStack je rozsáhlá platforma, která nabízí velké množství služeb na základě nasazených komponent. Komponenta Keystone umožňuje autentizaci, autorizaci a nastavení práv pro přístup a správu projektů na cloudu, čímž může být docíleno požadovaného rozdělení na edukaci, výzkum a testování. Vytvoření plnohodnotné síťové infrastruktury umožňuje komponenta Neutron. Plnohodnotná virtualizace je umožněna komponentou Nova a kontejnerizace doinstalováním služeb

[2]Kubernetes patří mezi software pro správu kontejnerů, a tedy nepatří mezi cloudové platformy jako takové, které umožňují plnohodnotnou virtualizaci (viz podkapitola 1.1.2). Kubernetes umožňuje správu kontejnerů na úrovni clusterů a podporuje škálovatelnost, z toho důvodu byl do výběru zařazen.

[3]Například KubeVirt nebo Kind [37].

[4]V Kubernetes znamená pojem uzel (anglicky *nodes*) výpočetní jednotku, ve které běží jeden a více kontejnerů [22].

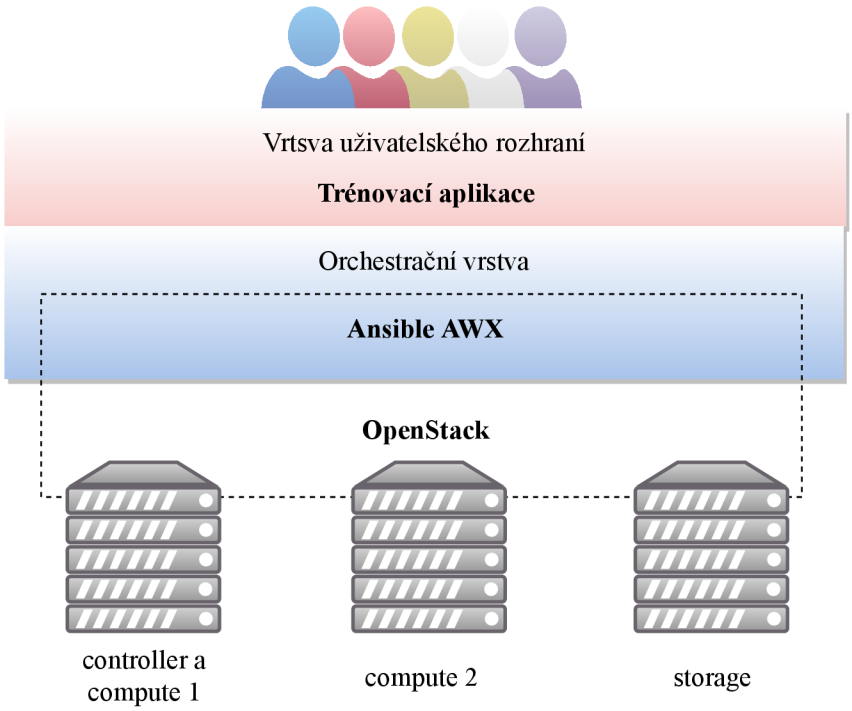
[5]Linuxové distribuce založené na RHELu - CentOS, Fedora, Rocky Linux...

Magnum a Zun. Podpora sandboxů je umožněna doinstalováním Kata kontejnerů[6]. OpenStack je možné ovládat i přes uživatelské rozhraní, které je přívětivé, intuitivní a nabízí grafické znázornění sítí a VS. Nevýhodou OpenStacku je jeho obsáhlá dokumentace, která se liší od způsobu jeho nasazení. Zároveň metoda nasazení OpenStacku určuje možnosti dodatečné instalace dalších komponent a celkové správy cloudu. Je tedy velmi důležité struktuře OpenStacku porozumět a zvolit vhodnou metodu nasazení.

Z porovnání vyplývá, že Kubernetes a OpenShift slouží zejména ke správě kontejnerů v cloudu a virtualizaci podporují až doinstalováním dodatečných nástrojů. Nemožnost vytvářet libovolně vlastní síťovou infrastrukturu je pro budování CR značný nedostatek. Pro účely CR a splnění vytyčených cílů je nejvhodnější použít platformu OpenStack.

2.3 Architektura Kybernetické arény

Obdobně jako architektura obecného CR podle NIST, viz obrázek 1.1, se architektura Arény skládá z vrstvy hardwarové, orchestrační a vrstvy uživatelského rozhraní. Schéma vrstev Kybernetické arény je zobrazeno na obrázku 2.1 [34].



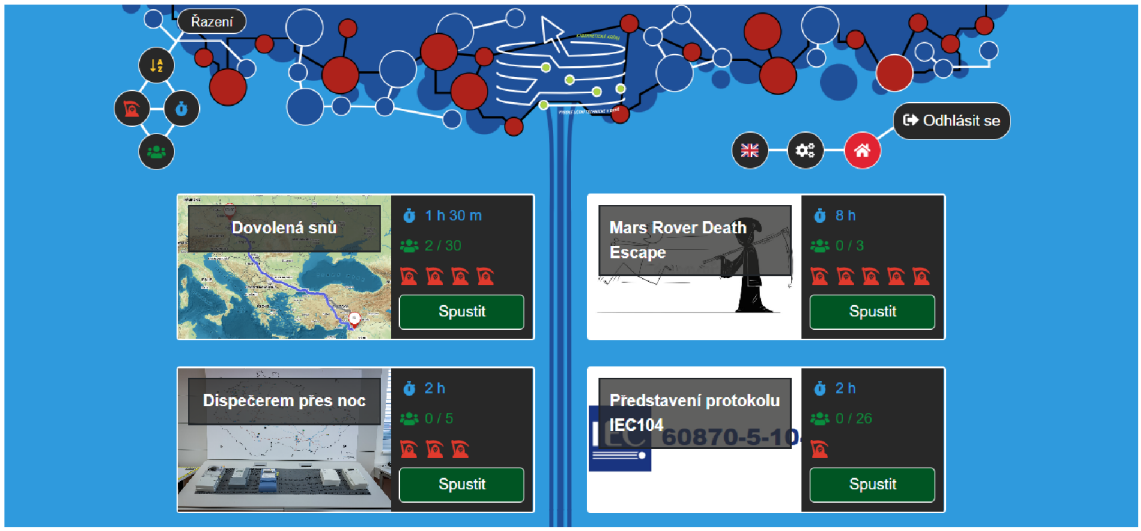
Obr. 2.1: Schéma vrstev Kybernetické arény

[6]Kata kontejnery na OpenStacku je vcelku nové řešení (uvedení v roce 2018) a kombinuje jednoduchost ve vytvoření kontejnerů a izolaci, kterou nabízí virtualizace, více v literatuře [38].

Hardwarová vrstva se skládá ze tří fyzických serverů, z toho jsou dva určené jako výpočetní – *compute*, a jeden jako úložiště – *storage*. Součástí *compute* serveru je i řídící jednotka OpenStacku, tzv. *controller*. Každý server má přiřazené funkce, pro které čerpá ze svých výpočetních, respektive paměťových prostředků. Toto rozdělení funkcí se definuje při nasazení OpenStacku.

Orchestrační vrstvu tvoří samotný OpenStack (podrobněji v kapitole 3) a Ansible AWX (podrobněji v kapitole 4). OpenStack se stará o spolehlivý běh celé Arény. Zajišťuje veškerou správu cloudu, management sítí, běh virtuálních strojů, správu virtuálních sítí a mnohé další. Ansible AWX zajišťuje již samotné spouštění a běh herních scénářů, jedná se tedy o prostředníka mezi uživatelským rozhraním a cloudovou orchestrací.

Vrstvu uživatelského rozhraní tvoří Trénovací aplikace, která umožňuje ovládání a správu Arény pro administrátory a zároveň je to i prostředí pro hraní kybernetických her účastníků Arény. Trénovací aplikace je dostupná přes webové rozhraní. Omezení přístupu k funkcím Arény je kontrolováno přístupovými právy, přičemž přihlašování do Arény je možné se stejnými údaji jako do VUT intranetu. Ukázka uživatelského rozhraní Arény, konkrétně panelu pro administrátory s přehledem her, je zobrazena na obrázku 2.2.



Obr. 2.2: Ukázka administrativního panelu Kybernetické arény, převzato z [34]

2.4 Další zástupci cyber range platforem

V posledních dekádách se výzkum CR velmi rozšířil, byly vypracovány mnohé studie, doporučení a porovnání, pro neustálé posouvání jejich vývoje. CR jsou značně používány v národní obraně a tyto informace jsou často, alespoň dočasně, utajeny. Studie tedy zejména čerpají informace z veřejně dostupných CR určených hlavně pro edukační a testovací účely.

Jedna z prvních srovnávacích studií je *A Survey of Cyber Ranges and Testbeds*, která vznikla pro australské ministerstvo obrany (literatura [39]). Studie je z roku 2013, tudíž uvedené CR nebudou z pohledu současného vývoje CR relevantní (zejména z hlediska použitých technologií), nicméně se jedná o ojedinělý dokument, který popisuje zejména vývoj CR účelně pro národní obranu z minulého desetiletí, které již nejsou klasifikovány.

Obsáhlá studie, která analyzuje až 100 neklasifikovaných CR platforem představených do roku 2019 [40] byla provedena pod záštitou norské univerzity (literatura [41]). Studie je zaměřena na zastoupení odlišných technologií ve vývoji CR, využívaného hardwaru, podporovaných scénářů, metody definicí scénářů, způsoby monitorování her, bodovací systémy a mnohé další. Nejedná se tedy o detailní rozbory konkrétních CR. Výsledkem studie jsou číselné údaje analyzovaných specifikací konkrétních CR, z těchto údajů je možné sledovat trendy ve vývoji CR.

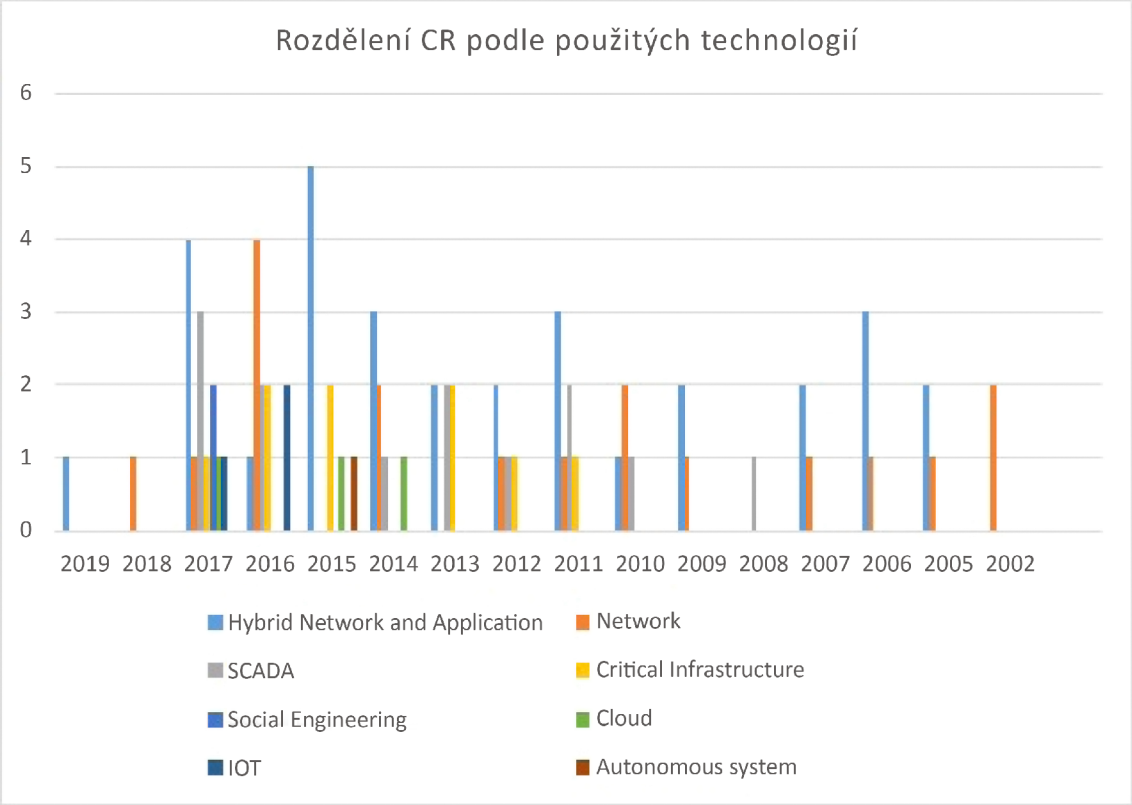
Jeden ze závěrů studie je viditelně se zvyšující počet vývoje CR použitím hybridního systému, viz obrázek 2.3. Podle [41] se jedná o kombinaci emulace, simulace a reálných zařízení pro nejvíce realistické scénáře CR. Kybernetická aréna spadá do klasifikace hybridního systému, jelikož emulací jsou vytvářeny virtuální infrastruktury, simulací je umožněno vytvořit realistické podmínky scénáře[7], reálná zařízení jsou použita zejména ve scénářích se systémy SCADA/ICS.

Právě cloudové platformy, jako například Arénou využívaný OpenStack, splňují podmínky hybridního systému. Důležité je uvědomění, že při vývoji CR neexistuje standardizace[8], vývojáři CR se tedy spoléhají na vlastní průzkumy a dostupné studie. Z výše uvedeného vyplývá, že hybridní systémy, respektive tedy cloud computing, jsou pro vývoj CR obecně nejvhodnější, jelikož trend vývoje CR přirozeně směřoval ke cloud computingu.

Za konkrétní příklad CR, jehož jádrem je cloud computing, je open-source platforma Kybernetický polygon (KYPO), která je vyvíjena na Masarykově univerzitě od roku 2013. Jako open-source byla zveřejněná v roce 2019. KYPO platforma ve spolupráci s NÚKIB (Národní úřad pro kybernetickou a informační bezpeč-

[7]Simulační scénáře v době psaní práce nejsou součástí Arény, nicméně díky použití cloud computingu je možné do budoucna tyto scénáře jednoduše do Arény zakomponovat.

[8]Pouze již zmíněné doporučení NIST, které je ovšem velmi čerstvé z pohledu vývoje CR.



Obr. 2.3: Trendy použitých technologií ve vývoji CR, převzato a upraveno z [41]

nost) a Ministerstvem vnitra ČR hostí mezinárodní bezpečnostní cvičení Cyber Czech, které je soustředěno na ochranu integrovaného záchranného systému. Jedná se tedy o technicky i organizačně unikátní prostředí v České republice, které aktuálně slouží pro edukační a výzkumné účely studentům a pracovníkům Masarykovy univerzity [42, 4].

3 OpenStack

V této kapitole je podrobněji představena platforma OpenStack, která je jádrem Kybernetické arény. Popsána je základní architektura a služby OpenStacku, metody nasazení do vlastí infrastruktury a souvislost s Arénou.

OpenStack je open-source cloudová platforma, která po nasazení spravuje velké množství výpočetních, paměťových a síťových prostředků, a tím umožňuje vytvořit privátní, veřejný i hybridní cloud. První verze OpenStacku vyšla v roce 2010 a do současnosti se udržuje vydávání nových verzí v půlročních cyklech. Poslední (v pořadí 26.) vydaná verze je Zed[1].

Původně byl OpenStack vyvíjen jako společný projekt organizací NASA (National Aeronautics and Space Administration) a Rackspace hosting. Od začátku bylo cílem vytvořit open-source platformu splňující potřeby veřejného i privátního cloudu bez ohledu na velikost prostředků s možností škálovatelnosti. V dnešní době je OpenStack podporován více než 500 členy v rámci fondu OpenInfra [44]. V posledních pěti verzích je největším přispěvatelem společnost Red Hat s průměrným podílem 45 % commitů [45].

3.1 Architektura a služby OpenStacku

Celý OpenStack je tvořen patřičně propojenými komponentami, které zpřístupňují OpenStacku své služby. První verze OpenStacku obsahovaly jednotky komponent poskytující pouze správu výpočetních prostředků a datové úložiště, nyní OpenStack obsahuje více než 40 komponent, respektive služeb (s každým novým vydáním se počet mírně mění) [46]. OpenStack jako celek spoléhá na:

- **virtualizaci**, která umožňuje abstrahovat hardwarové prostředky,
- **operační systém**, který zpracovává příkazy zadané OpenStack skripty.

Pro funkčnost jsou OpenStack, virtualizace a OS na sobě vzájemně závislé a musí být bezpodmínečně kompatibilní. Z tohoto důvodu je OpenStack postaven pouze na linuxových systémech.

Každá komponenta vychází z konkrétního projektu, do kterého přispívají vývojáři v rámci společnosti nebo nezávisle. OpenStack se v době psaní práce skládá celkem z pěti hlavních komponent, které jsou nezbytné k základní funkci cloudu (instalují se vždy), konkrétně jimi jsou Nova, Neutron, Keystone, Glance a Placement [47]. Dalšími doporučenými komponentami jsou Swift a Cinder.

Komponenta **Nova** zpracovává žádosti uživatelů cloudu a je zodpovědná za přidělení výpočetních prostředků na žádost. Nova je tvořena množinou bežících démonů

[1] Vývojáři OpenStacku vydávají verze v abecedním pořadí. Další verze ve vývoji je Antelope, viz [43].

v OS a výpočetní prostředky poskytuje z množiny virtualizovaných prostředků fyzického serveru. Uživatel má tyto prostředky pod kontrolou a je schopný je dynamicky navyšovat a ubírat. Tato komponenta je zásadní pro poskytování téměř všech služeb OpenStacku. Pro svou funkčnost je Nova závislá na komponentách Keystone, Glance, Neutron, Placement.

Neutron je komponenta zodpovědná za veškeré síťové služby. Poskytuje službu NaaS (síť jako služba[2]) a implementuje síťové API (Application Programming Interface) OpenStacku. Neutron se stará o konektivitu mezi jednotlivými službami cloudu, zároveň umožňuje uživatelům vytvářet virtuální sítě a prvky, včetně konfigurace IP adres, síťování, směrování, pravidel firewallu a další. Neutron je závislý na komponentě Keystone.

Keystone poskytuje služby identity. Autentizuje a autorizuje všechny akce služeb OpenStacku. Přes Keystone je možné uživatele a projekty sdružovat do skupin, spravovat práva uživatelů a projektů. Keystone podporuje autentizaci pomocí hesla a jména, tokenů a klíčů, zároveň je možné propojit autentizaci s třetími stranami.

Glance poskytuje úložiště pro obrazy instancí[3] a metadat k nim připojených. Glance spravuje a kontroluje metadata obrazů instancí, přičemž jejich alokaci poskytuje komponenta Nova.

Komponenta **Swift** poskytuje API pro ukládání a získávání uživatelských dat. Data ukládá ve formě objektů, tím je možné uložit data libovolného formátu. Zajišťuje automatické zálohování, archivaci, mazání apod. S komponentou Keystone umožňuje autentizaci a autorizaci přístupu k datům.

Cinder je blokové úložiště, které službě Nova poskytuje vytvoření svazků[4]. Svazky poskytují instancím trvalé úložiště. Uživatel svazky sám definuje podle potřeb specifických instancí.

Placement mapuje využívání výpočetních zdrojů cloudu a tyto informace poskytuje dalším službám.

Celý OpenStack je možné spravovat přes CLI (Command Line Interface), dodatečně lze komponentou **Horizon** poskytovat správu cloudu přes webové rozhraní. Uživatel je schopný přes Horizon spravovat většinu služeb OpenStacku. Přístup je umožněn všem přidaným uživatelům OpenStacku a Horizon následně nabízí jen ty služby, ke kterým mají uživatelé přidělena práva.

Užitečná komponenta zjednodušující správu a konfiguraci cloudu je **Heat**. Heat poskytuje orchestraci aplikací nebo infrastruktury pomocí orchestračních šablon ve formátu YAML (YAML Ain't Markup Language). Uživatel je pomocí Heat slu-

[2]Anglicky *networking as a service*.

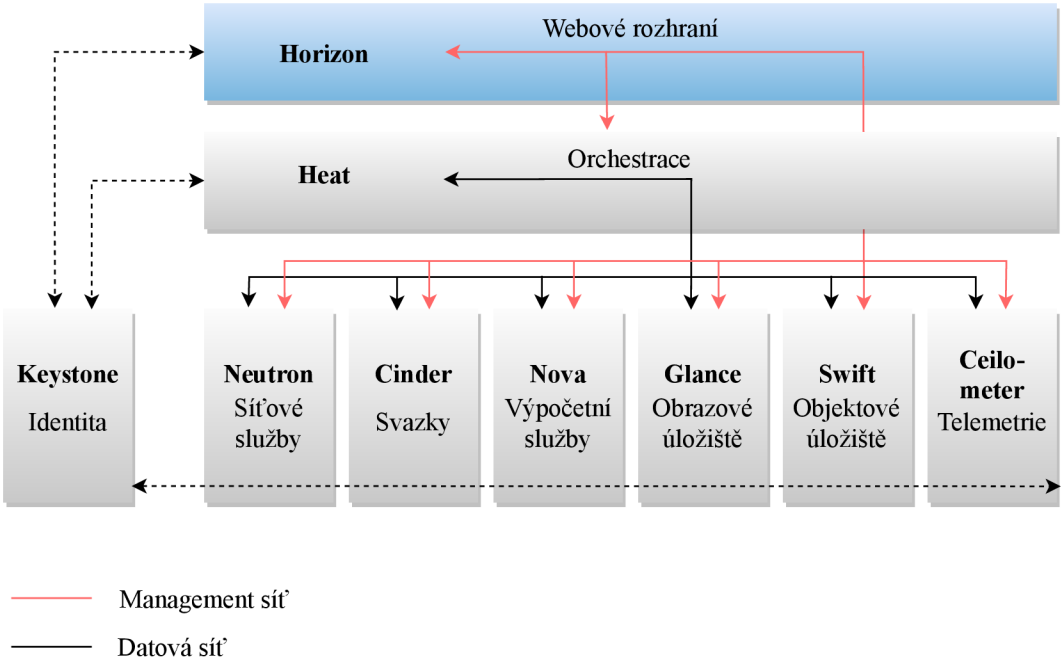
[3]Pojem instance znamená v OpenStacku virtuální stroj nebo kontejner. Každý VS nebo kontejner je *instancí* obrazu Glance komponenty.

[4]Anglicky *volumes*.

žeb schopný automatizovat vytváření sítí, virtuálních strojů, kontejnerů atd. Další komponenty OpenStacku a služby, které poskytují, jsou dostupné v literatuře [46].

Důležité je zmínit, že všechny služby OpenStacku pro svou funkčnost potřebují vzájemnou časovou synchronizaci. Bez časové synchronizace služby nemohou fungovat, a ani není možné cloud nasadit. Pokud je k dispozici přístup do Internetu, časovou synchronizaci zajišťuje tzv. chrony daemon pomocí protokolu NTP, v opačném případě musí být k dispozici lokální časový server.

Architektura základních komponent OpenStacku je zobrazena na obrázku 3.1. Na obrázku se nachází i komponenta Ceilometer, která mapuje vytížení cloudu a jednotlivých služeb [48].



Obr. 3.1: Architektura hlavních služeb OpenStacku [48]

3.2 Metody nasazení

Nasazení OpenStacku do vlastní infrastruktury je velice komplexní záležitost a vyžaduje jeho hluboké znalosti. Existují nástroje, které nasazení OpenStacku usnadňují pomocí automatizace instalace a konfigurace komponent. Při použití těchto nástrojů je důležité, aby si uživatel uvědomil, k jakému účelu bude OpenStack používán, a které komponenty budou v cloudu zahrnuty. Použití těchto nástrojů nabízí kompromis mezi usnadněním nasazení a přizpůsobením individuálnímu nasazení. Některé nástroje částečně omezují konfiguraci komponent nebo ztěžují případně jejich dodatečnou úpravu nebo doinstalaci [56].

Instalační nástroje se liší v přístupu k nasazení OpenStacku a podle toho se také liší hardwarové a softwarové požadavky a uživatelská náročnost nasazení. Nejvýraznějším rozdílem v nasazení je mód instalace, tím je buď:

- mód ***all-in-one***, ve kterém jsou veškeré komponenty nasazovány na jeden uzel,
- mód ***multinode***, OpenStack je instalován na více uzlů, přičemž uzly jsou rozděleny na řídící a výpočetní[5].

Teoreticky je možné nasadit libovolný počet řídících a výpočetních uzlů, limitace vychází pouze z dostupných hardwarových prostředků. Mód *all-in-one* je vhodný pouze pro vývojové a testovací účely pro ověření konceptu [49].

Nástroj nasazení ovlivňuje operační systém, na který je OpenStack nasazen. Toto ovlivnění vychází z technologií, který daný nástroj používá, a ty nemusí být plně kompatibilní se všemi distribucemi.

Hardwarové požadavky souvisí zejména s módem instalace a jsou velmi individuální podle budoucího účelu OpenStacku. Hardwarové prostředky nutné pro instalaci, ale nesouvisí s konečnými prostředky, které má cloud k dispozici, jelikož OpenStack umožňuje dynamické přidávání a ubírání uzlů.

V základě je pro OpenStack a nástrojům nasazení dostupná velmi obsáhlá literatura a případná podpora je nabízená většinou pouze komunitou. Existují poskytovatelé, kteří nabízejí služby podpory při nasazení, následné správy a opravu chyb v rámci používání OpenStacku. Tito poskytovatelé nabízejí podporu zpravidla u starších verzí OpenStacku, například 2 verze zpět. Tohoto přístupu je užitečné využít v případě použití OpenStacku v produkčním prostředí, kde je kritická jeho správná a souvislá funkčnost.

Konfiguraci některých komponent není možné dodatečně upravit v již nasazeném OpenStacku. V takovém případě je k tomu přistupováno tak, abstraktně řečeno, že je daná komponenta izolována od nasazeného OpenStacku a místo ní je nasazena nová, upravená komponenta. Možnosti této dodatečné úpravy jsou zejména dány použitým nástrojem a uživatel se musí obrátit vždy na informace poskytnuté konkrétním instalačním nástrojem[6].

Neméně důležitým faktorem je náročnost použití nástroje nasazení. Náročnost je posuzována podle automatizace instalace, nutnosti znalostí dodatečných nástrojů a podle důsledků vzniklých lidskou chybou. Nicméně se stále jedná o subjektivní posouzení.

[5] Anglicky *controller* a *compute*.

[6] Například komponenty, které jsou nasazovány v podobě kontejnerů (OpenStack Ansible) tuto dodatečnou úpravu umožňuje. Oproti tomu nástroj Packstack, který automatizuje nasazení pomocí Puppet modulů tuto dodatečnou úpravu neumožňuje [52, 53].

Tabulka 3.1 shrnuje výše uvedené parametry odlišných nástrojů nasazení. Veškeré nástroje sloužící k nasazení OpenStacku je možné nalézt v literatuře [56], zejména je důležité si uvědomit, že nasazení určitých komponent nemusí být podporováno vybraným nástrojem, vždy je nutné vycházet z konkrétní dokumentace[7]. Tabulka vychází vždy z nejnovější podporované verze OpenStacku daného nástroje[8]. Podrobnější informace o jednotlivých možnostech nasazení jsou shrnuty v literatuře [50].

Vhodné je upřesnit, že nástroj TripleO[9] využívá k nasazení OpenStacku samotný OpenStack, který nasazení řídí a po nasazení spravuje jednotlivé komponenty, provádí aktualizace apod. Z tohoto důvodu je v tabulce uvedena jako jedna z hlavních technologií nasazení právě OpenStack.

Tab. 3.1: Shrnutí nástrojů pro nasazení OpenStacku [51, 52, 53, 54, 55]

Nástroje					
	DevStack	**RDO Packstack**	**OpenStack Ansible**	**Kolla-Ansible**	**TripleO**
Nejnovější verze	Yoga	Yoga	Yoga	Yoga	Zed
Mód instalace	Multinode	All–in–one	Multinode	Multinode	Multinode
Operační systém	Ubuntu Fedora RHEL CentOS Stream	RHEL CentOS Stream	Debian Ubuntu CentOS Stream Rocky Linux	CentOS Ubuntu Debian Rocky Linux	RHEL CentOS Stream
Minimální hardwarové požadavky	8 GB RAM 30 GB disk	16 GB RAM 40 GB disk	2 NIC 16 GB RAM 150 GB disk	2 NIC 8 GB RAM 40 GB disk	2 NIC 8 Dual core CPU 12 GB RAM 60 GB disk
Hlavní technologie nasazení	Shell	Puppet	Ansible	Ansible	Ansible OpenStack
Podpora vendora	Není	Red Hat	Není	Není	Red Hat
Dodatečné úpravy	Nelze	Nelze	Ano	Ano	Ano
Náročnost	Základní	Základní	Zvýšená	Zvýšená	Těžká

3.3 OpenStack v souvislosti s Kybernetickou arénou

OpenStack v rámci Kybernetické arény slouží, kromě virtualizace hardwarových zdrojů, k:

- přidělování hardwarových zdrojů (komponenta Nova),
- virtualizaci infrastruktury (Nova a Zun),
- síťování (Neutron a Kuryr),
- orchestraci a vytváření herních scénářů (Heat).

[7] Rozcestí dostupných nástrojů dostupné v literatuře [56].
[8] V čase psaní práce je nejnovější verze Zed.
[9] V dokumentaci je uváděn i název „OpenStack on OpenStack“.

Kromě komponent popsaných v podkapitole 3.1 Aréna využívá služby komponent Zun a Kuryr. Zun poskytuje služby spojené se správou kontejnerů s implementací běhového prostředí Kata. Komponenta Kuryr poskytuje připojení kontejneru k síťové topologii spravovanou Neutronem, tedy i připojení do internetu.

V podkapitole 3.1 byla popsána komponenta Heat, která poskytuje služby orchestrace a automatizace pomocí vygenerovaných šablon. Aréna využívá služby Heat pro vytváření herních scénářů v OpenStacku. Komponenta Heat scénáře vytváří pomocí šablony v jazyce YAML, ve kterém se nachází veškeré specifikace nutné pro správné vytvoření scénáře, tedy:

- názvy virtuálních stanic,
- diskové obrazy stanic,
- definici virtuálních stanic (počet vCPU, RAM atd.)[10],
- sítě, do které stanice patří,
- definice síťové infrastruktury,
- vytvoření uživatelů stanic a další [34].

Architektura Kybernetické arény v OpenStacku

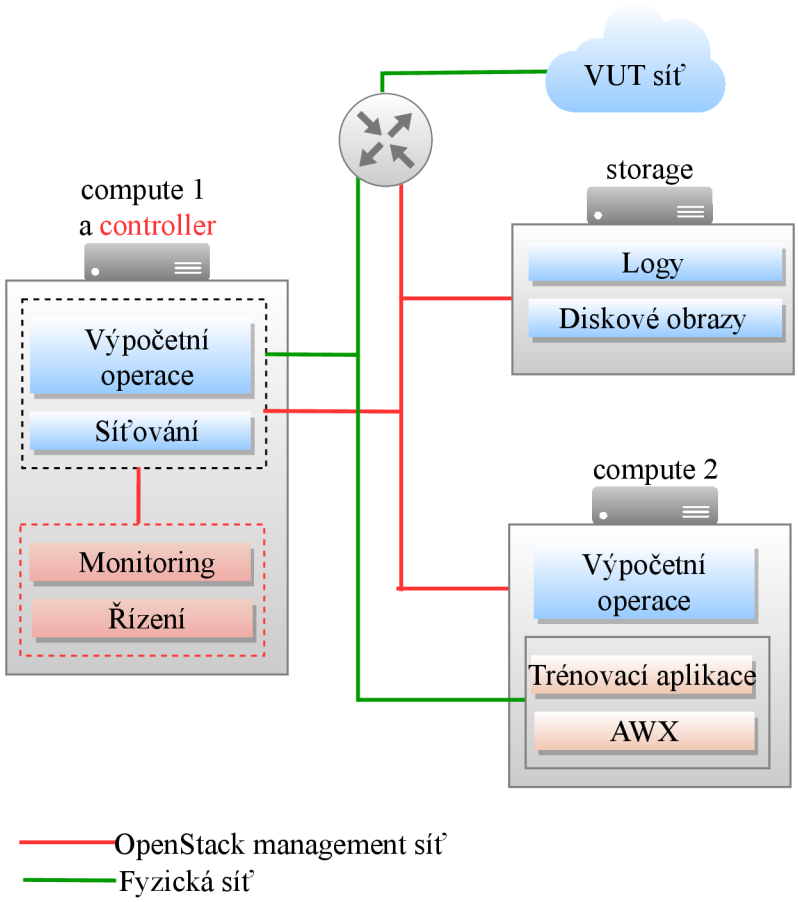
OpenStack spravuje celkem 3 fyzické servery, jejichž logické rozdělení je jeden *controller*, dva *compute* uzly a jeden *storage* uzel, je tedy nasazen v módu *multinode*. Na všech serverech je nainstalován RHEL 8 OS. OpenStack byl nasazen pomocí nástroje Kolla-Ansible ve verzi Victoria. Schéma Kybernetické arény v rámci OpenStacku je zobrazena na obrázku 3.2.

Při nasazení OpenStacku jsou serverům přiřazeny role, které vykonávají. *Controller* spravuje celý OpenStack, nacházejí se zde služby identity, orchestrace, síťování, správa přes CLI a webové rozhraní. Zároveň *controller* monitoruje využití cloudu a zpracovává logy.

Compute uzly virtualizují herní scénáře a síťovou infrastrukturu VS. *Compute 1* disponuje kromě výpočetní rolí také rolí síťového uzlu. Zpracovává tedy veškeré virtuální L3 síťování. Na uzlu *compute 2* se nachází služba Cinder, která slouží v Aréně pro kritické aplikace, které vyžadují vysokou rychlost čtení a zápisu. Část serveru s uzlem *compute 2* je dedikována pro Trénovací aplikaci a Ansible AWX server, které běží nad OpenStackem jako VS.

Storage uzel slouží jako úložiště pro zpracované logy *controlleru*, diskové obrazy VS a pro budoucí zálohy Arény [34].

[10] V OpenStacku jsou tyto zdroje nazývány *flavors*.



Obr. 3.2: Schéma Kybernetické arény v rámci OpenStack cloudu [34]

Na obrázku jsou zobrazeny dva typy sítí – síť dedikovaná pouze pro management OpenStacku a fyzické VLAN (Virtual Local Area Network) sítě. Všechny fyzické VLAN sítě jsou namapovány do OpenStacku a lze k nim připojovat virtuální infrastruktury kyberbezpečnostních her. Zvlástním typem je síť `public`, která slouží především pro komunikaci do Internetu. Ostatní VLAN sítě umožňují propojovat vybrané typy scénářů s fyzickým hardwarem, který je potřebný ke speciálním hrám.

4 Ansible a Ansible AWX

Ansible je po OpenStacku další stěžejní technologií pro fungování Kybernetické arény. Jedná se o open-source nástroj společnosti Red Hat pro automatizaci procesů a zjednodušení správy lokálních i vzdálených uzlů, který využívá Python skripty, tzv. moduly.

Ansible AWX (Ansible Web eXecutable) je webové rozhraní pro usnadnění správy Ansible, vzdálených hostů a zobrazení statistiky. Jedná se o open-source verzi nástroje Ansible Controller[1] od společnosti Red Hat [57]. Základním znakem Ansible je, že je tzv. *agentless*, to znamená, že k tomu, aby byl na zařízení používán není na uzlech nutné předem nainstalovat určitý software nebo nástroj, kromě SSH (Secure Shell) serveru a Python interpreteru[2].

4.1 Architektura Ansible

V Ansible jsou rozlišovány dva typy uzlů[3]:

- **řídící uzel**, na kterém běží Ansible, a ze kterého jsou odesílány instrukce k vykonání,
- **spravovaný uzel** je ten, který instrukce vykonává, těchto uzlů může být libovolný počet a jsou definovány v inventáři[4].

Instrukce vedoucí spravovaný uzel do požadovaného stavu uživatel definuje v YAML souborech syntaxí, kterou Ansible podporuje. Tyto soubory se nazývají **playbooky**. Jedná se tedy o sled instrukcí, kde každá instrukce je kontrolována zda byla dokončena. Pokud vykonávaná instrukce není dokončena úspěšně, vykonávání playbooku je ihned ukončeno a řídícímu uzlu je navrácena chybová hodnota s dodatečnými informacemi. Instrukce je vhodné sjednocovat do **úkolů**[5], aby bylo při vykonávání zřetelné, ve které části se playbook právě nachází, respektive, které instrukce byly úspěšně vykonány.

Vhodným přístupem je použití tzv. **rolí**[6] – role sjednocuje použité Ansible artefakty (úkoly, šablony a další) a umožňuje jejich opakované volání napříč různými úkoly. Není tedy nutné instrukce v novém úkolu specifikovat znovu, což snižuje redundanci kódu [58, 59]. Struktura Ansible je zobrazena na obrázku 4.1.

[1] Dříve Ansible Tower, jehož podpora skončila v listopadu 2022.

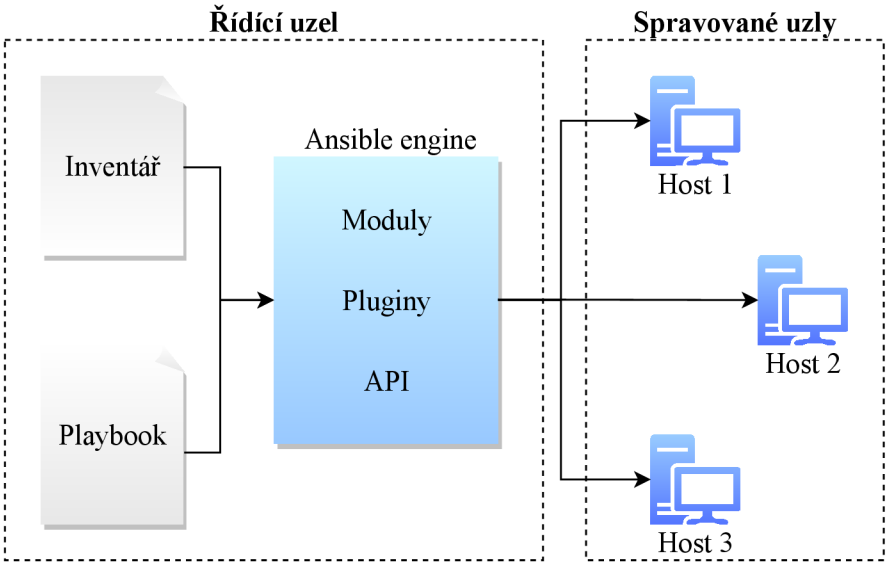
[2] Například Puppet, nástroj rovněž pro automatizaci procesů, funguje na principu klient–server, je tedy nutné, aby se na koncových uzlech nacházel Puppet agent.

[3] Anglicky *controller* a *managed nodes*.

[4] Anglicky *inventory file*.

[5] Anglicky *tasks*.

[6] Anglicky *roles*.



Obr. 4.1: Architektura a struktura Ansible

Zpracování playbooku je uskutečněno Ansible enginem. Každý playbook využívá **modul** nebo moduly. Jedná se o samostatné skripty (nejčastěji v jazyce Python), které zajišťují vykonání požadovaných úkolů v Ansible. Uživatel si moduly může nadefinovat sám nebo využít již vytvořené moduly a ty pouze importovat. Nyní existuje mnoho modulů, které jsou vytvářeny za účelem správy různých systémů, například správy webového serveru, VMware strojů i OpenStacku. Seznam všech modulů je v literatuře [58].

Pluginy v sobě uchovávají informace o již vykonaných úkolech, zejména se jedná o zjištěné informace na spravovaných uzlech v podobě cache a logů. Cache potom slouží k vyhnutí se zbytečnému a náročnému sbírání faktů[7] o uzlech při vykonání jednotlivého úkolu.

Pomocí **Ansible API** lze zvýšit způsoby spojení ke spravovaným uzlům a možnosti jejich ovládání. Například k uzlům nacházejících se na cloudech, ke kterým není přístup z veřejné sítě přes SSH spojení povoleno [60].

4.2 Ansible AWX v souvislosti s Kybernetickou arénou

AWX je postaven nad Ansible a oproti němu obsahuje funkcionality navíc. Nezanedbatelnou součástí AWX je možnost spouštět playbooky pomocí REST (Representational State Transfer) rozhraní, toto umožňuje Ansible integrovat s dalšími

[7] Anglicky *gathering facts*, tento úkol je spouštěn implicitně při spuštění playbooku, nicméně může být přeskočen.

aplikacemi pomocí jednoduchých webových volání. Dalším důležitým doplňkem je plánování běhů playbooků, jejich grafická vizualizace v reálném čase a logování historicky spuštěních playbooků.

AWX tvoří prostředníka mezi OpenStackem a Trénovací aplikací, jak bylo krátce popsáno v podkapitole 2.3. V AWX je implementováno 5 playbooků, které OpenStacku zasílají požadavky na vykonání konkrétního úkonu, a to:

- zapnutí hry v Aréně,
- vypnutí hry v Aréně,
- spuštění hry hráčem,
- vypnutí hry hráčem,
- generace konzole do VS hráčem nebo administrátorem.

Administrátor nebo hráč interaguje s Arénou pomocí Trénovací aplikace (webové rozhraní). Podle akce, která je v Trénovací aplikaci vykonána, je v AWX spuštěn příslušný playbook, který následně předává instrukce OpenStacku a ten, například, vytvoří VS pro daný hrací scénář. V tomto případě probíhá i generování Heat šablon.

Podle aktuální architektury Arény podle obrázku 3.2 je AWX součástí OpenStacku, respektive běží nad ním jako VS. Do budoucna ovšem má být tato architektura změněna a AWX má být nasazen mimo OpenStack, aby v případě nesprávné funkčnosti jedné z technologií se zamezilo ovlivnění funkčnosti té druhé.

4.3 Metody instalace AWX

Obecně není AWX doporučován do produkčního prostředí, jedná se o open-source nástroj, a tedy k němu náleží pouze podpora komunity a zároveň často vychází nové vydání s opravami chyb. V případě Arény se nejedná o prostředí, ve kterém by neustálý běh AWX byl kritický – prostředí je neustále ve vývoji a v případném výskytu chyby je ihned spravována vývojovým týmem Arény.

AWX lze instalovat pouze jako kontejnery s použitím Dockeru, OpenShift nebo Kubernetes. Podle použité technologie se liší způsob instalace AWX. Je doporučováno (podle AWX dokumentace [61]) instalovat AWX do:

- **Dockeru** pro vývojové a testovací účely,
- **Kubernetes** pro stabilnější prostředí s rozsáhlejší infrastrukturou[8],
- **OpenShiftu** v případě využití cloud computingu založeného na kontejnerech[9].

[8]Instalace je možná i s použitím tzv. *minikube* kontejneru, což je odlehčený lokální Kubernetes kontejner [63].

[9]OpenShift je také zatím jediná platforma, která je využita v případě použití privátní Ansible Automation platformy od společnosti Red Hat [62].

Do verze 18 je možné AWX instalovat do všech uvedených kontejnerových managerů. Metody instalace se potom odlišují podle použitého manageru a jeho nastavení a podle samotného nastavení AWX. Pro použití verze vyšší než 17 musí být po instalaci AWX udělána aktualizace AWX na verzi vyšší [61].

Od verze AWX 18[10] je doporučováno instalovat AWX pomocí nástroje `awx-operator` [64]. Nástroj `awx-operator` zjednodušuje instalaci AWX, jelikož je vytvořen pouze pro použití s Kubernetes[11]. Tímto je uživateli usnadněna instalace, protože nemusí přizpůsobovat instalaci AWX konkrétnímu kontejnerovému manageru a může se soustředit zejména na samotné nastavení AWX.

Zvolená metoda instalace nemá v závěru vliv na to, které funkce AWX bude poskytovat[12]. V tabulce 4.1 jsou uvedeny metody instalace AWX rozdělené podle kontejnerového manageru.

Tab. 4.1: Metody instalace AWX [19, 22, 32, 61, 63, 64, 65]

	Manuální			awx-operator		
Kontejner manager	**Docker**	**Kubernetes**	**OpenShift**	**Kubernetes**	**Minikube**	**K3S**
Podporované verze AWX	<18	Všechny verze	<18	>18	>18	>18
Minimální hardwarové požadavky běhu AWX	2 CPU 4 GB RAM 20 GB disk			4 CPU 6 GB RAM 20 GB disk		
Minimální hardwarové požadavky manageru	2 CPU 4 GB RAM	3 CPU 6 GB RAM	3 CPU 6 GB RAM	3 CPU 6 GB RAM	2 CPU 2 GB RAM	1 CPU 1 GB RAM
Nástroje	Ansible 2,8+; GNU Make; Git; Python 3,6+; Docker; Docker Python modul					
Možnost upgrade	Ano					

4.3.1 Hlavní odlišnosti AWX verzí

Existuje několik klíčových rozdílů mezi verzemi AWX 17, kterou aktuálně používá Aréna a aktuáln verzí 21. Hlavním přídavkem v AWX 21 je funkce rozdělování úloh, která umožňuje rozdělit velké úlohy na menší kousky a spouštět je souběžně na několika strojích. Další důležitý přídavek je možnost vytvářet vlastní typy přihlašovacích údajů, které lze použít k ukládání a správě jakýchkoli tajných dat potřebných pro automatizované pracovní postupy. Správa inventáře byla také vylepšena v AWX 21,

[10] V čase psaní práce je nejnovější verzí verze 21. Nové verze vychází v dvou až čtyřměsíčních intervalech.

[11] Jelikož je OpenShift založen na Kubernetes kontejnerech, pro `awx-operator` je taktéž vhodným kandidátem, ovšem správa OpenShiftu a instalace AWX je podstatně složitější a vhodná pouze pro náročnější produkční účely.

[12] Například u OpenStacku je to opačně, kde zvolená metoda nasazení ovlivňuje budoucí poskytované služby.

což usnadňuje správu hostitelů a skupin a umožňuje flexibilní filtrování dat inventáře. AWX 21 dále obsahuje vylepšené hashovací algoritmy a podporu dvoufaktorové autentizace. Celkově tyto nové funkce dělají z AWX 21 oproti jeho předchůdci robustnější a bezpečnější nástroj pro správu automatizace. Verze 21 rovněž používá novou verzi ansible-runneru, konkrétně verzi 2.10, která zahrnuje podstatné změny[13].

4.3.2 Kubernetes distribuce

V současnosti existuje více Kubernetes distribucí, a tedy i AWX je možné nainstalovat pomocí `awx-operator` do více distribucí. Kompletní seznam certifikovaných distribucí Kubernetes je dostupný v literatuře [67]. Tabulka 4.1 zobrazuje tři distribuce – původní Kubernetes[14], Minikube a K3S. Z této tabulky vyplývá odlišnost pouze v minimálních výpočetních a paměťových požadavcích, ovšem tyto distribuce se odlišují i ve svých funkcích, případech použití a modelech nasazení.

K8S je standardní, plně vybavená distribuce Kubernetes, která může běžet na jakékoli infrastruktuře. Jedná se o složitý systém, který vyžaduje složitější konfiguraci a následnou správu. Obvykle se používá pro rozsáhlé produkční nasazení [22].

Minikube je distribuce navržená tak, aby běžela pouze na jednom uzlu a zároveň podporovala většinu funkcí Kubernetes. Minikube je velmi snadné a rychlé spustit a je určen pouze pro lokální testovací účely, jelikož nabízí omezenou síťovou správu a implicitně neumožňuje přístup ke službám uvnitř Minikube clusteru. Pro přístup k těmto službám je nutné použít reverzní proxy, tzv. *ingress controller*, která směřuje provoz od externích klientů k příslušným službám v clusteru definovaných podle nastavitelných *ingress* pravidel [63].

K3S je tzv. odlehčená distribuce navržená pro běh na zařízeních s omezenými výpočetními prostředky, zejména na IoT a tzv. edge computing zařízeních. Hlavní odlišností od předchozích distribucí je možnost instalace pouze z jednoho binárního souboru. Na rozdíl od Minikube je K3S určen do produkčního prostředí a podporuje více uzlovou infrastrukturu. K vnitřním aplikací je možné přistoupit pomocí protokolu TLS a K3S implicitně podporuje *ingress controller* a vyvažovač zátěže[15] [65].

[13] Jedna ze zásadních změn je způsob, jakým Ansible pracuje s kolekcemi a moduly, konkrétní změny jsou dostupné v Ansible dokumentaci pod literaturou [66]

[14] V literatuře jinak nazývaný *vanilla* Kubernetes nebo zkratkou K8S.

[15] Anglicky *service load balancer*.

5 Podstata zálohy a obnovy Kybernetické arény

Kybernetická aréna a CR obecně jsou zpravidla dynamické systémy, ve kterých je nutné pro splnění vytyčených cílů upravovat poskytované služby. V případě Arény se jedná o vytvoření simulačního prostředí pro edukaci, výzkum a testování. Z toho vyplývá, že každé vytvořené prostředí bude odlišné v závislosti na momentovém naplnění cíle Arény.

V kapitolách věnujících se OpenStacku a Aréně bylo uvedeno, jakou roli OpenStack hraje pro neustále měnící se prostředí Arény. Podle konkrétního scénáře her je pomocí AWX OpenStacku předána šablona pro vytvoření specifického virtuálního prostředí – sítě, síťové prvky, stanice, uživatele, databáze a další, a také jsou veškeré elementy virtuálního prostředí přesně podle šablony nakonfigurovány. Tento celý proces je reverzibilní a může být libovolně znovu spuštěn.

Díky vrstvě OpenStacku v architektuře Arény se může zdát, že celý systém je snadný na ovládání a správu. OpenStack je ovšem složitá platforma a na změny velmi citlivá, proto úprava aplikovaná v samotném OpenStacku může zapříčinit nežádané chování OpenStacku a následně i Arény. Veškeré úpravy v již nasazeném OpenStacku je třeba provádět opatrně a počítat s možností nenávratných změn. Pokud takový případ nastane, je vhodné se vrátit do zálohy vytvořené v bodě požadované funkčnosti. V případě potřeby vytvoření zálohy se nabízejí dvě možnosti:

- zálohovat systém Arény jako kompletní celek,
- zálohovat logicky související části Arény.

První volba zahrnuje potřebu dostatečně velkého externího úložiště a celý proces zálohy a obnovy systému je časově náročný. Ovšem z pohledu komplexnosti není záloha a obnova složitá a je možné využít externí nástroje podle typu OS, na kterém OpenStack s Arénou běží, například Relax and Recover od společnosti Red Hat [71]. Takto vytvořené zálohy musí být obnoveny na stejný hardware, z čehož plyne redukce schopnosti přenést celý systém na jiný typ hardwaru.

Druhá volba znamená rozdělení systému na menší, zpravidla logicky související části a tvořit zálohy z těchto menších částí. Tento způsob je značně komplikovanější a způsob řešení vyplývá z interní funkčnosti samotného OpenStacku. Jak bylo uvedeno v kapitole 3, OpenStack nabízí své služby pomocí nakonfigurovaných komponent a zároveň je většina poskytovaných služeb závislá na službách poskytovaných jinými komponentami. Pro implementaci tvorby záloh z dílčích částí je nutné jít v souladu s tím, jak tyto služby přesně fungují. Obecně je vhodné navrhnout proces zálohy tak, aby bylo docíleno úspěšné obnovy do žádaného stavu a zároveň nebyla negativně ovlivněna funkčnost jiných služeb. Z tohoto způsobu také plyne vý-

hoda přenést Arénu na odlišný OpenStack a spustit automatizovaný proces obnovy. Tímto, alespoň částečně, odpadne nutnost ruční konfigurace nového OpenStacku pro potřeby Arény.

Pro návrh struktur zálohy a obnovy je nejvhodnější vycházet z již logického rozdělení OpenStacku, tedy struktury navrhnout tak, aby následovala rozdělení jednotlivých komponent – Keystone, Neutron, Cinder, Swift a další. Z každé komponenty lze potom vybrat části služeb, jejichž data jsou vhodná a užitečná k zálohování. Implementace konkrétní zálohy poté bude postupovat podle závislostí jednotlivých služeb. Je nutné přesně vymezit, co záloha musí obsahovat, aby bylo možné její úspěšné a funkční obnovení. Pro splnění tohoto bodu je vhodné vycházet z logické architektury OpenStack služeb, dostupné v literatuře [68]. Návrh struktury[1] pro vytváření záloh je zobrazeno na obrázku 5.1. Ze schématu vyplývá, že každá OpenStack služba se skládá z mnoha částí, tedy i souborů a každý chybějící soubor může vést k poškození záloh. Nutno zdůraznit, že schéma je pouze náčrt, co každá služba zahrnuje, ve skutečnosti je součástí každé služby mnohem více. Proto je klíčové zajistit kompletní zálohování všech částí služeb, jak je naznačeno v samotném náčrtu, aby byla zajištěna integrita a dostupnost kritických dat a služeb.

Důležité je upřesnit, že účelem zálohování OpenStacku není vytvořit zálohy zmíněných částí, které se týkají herních scénářů. Veškeré údaje, které OpenStack potřebuje k úspěšnému vytvoření herní infrastruktury se nacházejí v Heat šablonách a není tedy potřeba je zálohovat. Podstata zálohy a obnovy OpenStacku je zajistit, aby kritické části, které ovlivňují práci Arény, bylo možné obnovit v případě, že během úpravy OpenStacku nebo jeho aktualizace dojde k ovlivnění požadované funkčnosti Arény.

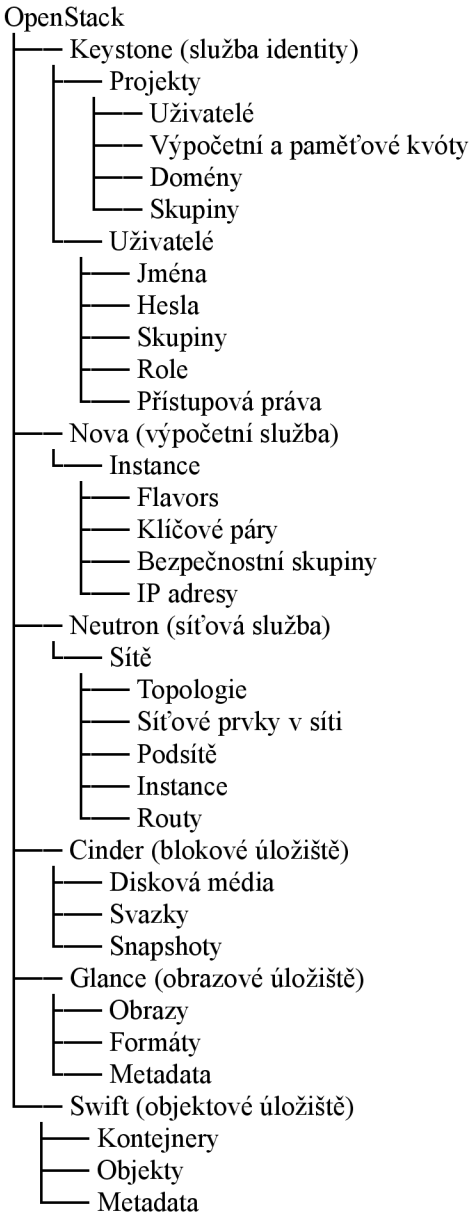
5.1 Přístupy k záloze a obnově částí OpenStacku

V rámci procesu obnovy a zálohy vybraných částí OpenStacku je nutné rozlišit dva odlišné přístupy, které se ale v některých ohledech shodují. Prvním přístupem je tzv. OpenStack migrace, neboli migrace z cloudu na cloud[2]. Tento přístup je vhodný, pokud je nutné přenést většinu dat cloudu na cloud odlišný, a to buď v rámci stejné verze, případě verze odlišné. Proces migrace zahrnuje zkopírování požadovaných dat a konfigurací a zároveň zajištění zachování všech závislostí. Je velmi důležité brát ohled na to, že možnost provedení migrace je velmi ovlivněna metodou nasazení

[1] Navržené schéma je pouze ilustrativní pro získání představy o hlavních službách OpenStacku, nejedná se o kompletní výčet zdrojů určených pro zálohu a obnovu. Konkrétní výčet zdrojů je uveden v podkapitole 5.1.1.

[2] Anglicky *cloud to cloud migration* a *OpenStack migration*.

OpenStacku. Příkladem může být OpenStack nasazený pomocí TripleO, který k nasazení používá Heat šablony, naproti tomu OpenStack nasazený pomocí nástroje Kolla-Ansible, který využívá pro nasazení Docker kontejnery. Takto odlišně nasazené OpenStack cloudy budou mít rozdílné konfigurační soubory, odlišnou adresářovou strukturu, případně rozdílné komponenty a další. Tyto odlišnosti by značně ovlivnily postup migrace, z čehož vyplývá, že neexistuje jednotný přístup ke cloudové migraci. Pro cloudovou migraci bylo vyvinuto již několik nástrojů, přičemž značná část z nich je proprietárních a umožňuje migrovat data i v rámci dvou různých clou-



Obr. 5.1: Struktura služeb zálohy a obnovy částí OpenStacku

dových platforem. Nativním OpenStack nástrojem pro migraci, zálohu a obnovu je projekt Freezer, který umožňuje pořízení snapshotů různých částí cloudu, zároveň také podporuje inkrementální zálohy. Freezer ovšem nepodporuje všechny OpenStack verze. Z verzí, které jsou ještě udržovány podporuje verze Xena, Yoga a Zed. Verze Victoria, která je aktuálně používána Arénou, podporována není [72].

Druhý způsob spočívá v sestavení procesu, který postupně získá data vybrané části OpenStacku, a to v souladu s tím, že bude zajištěna celková integrita těchto dat. Sestavený proces musí zajišťovat sběr všech dat, která jsou nutná pro vytvoření plnohodnotného celku pro zálohu. Pokud by tato podmínka nebyla dodržena, obnova by byla pravděpodobně neúspěšná, případně neúplná. Pro proces zálohy se obecně nabízejí tři různé možnosti přístupu:

- získávání potřebných dat pomocí OpenStack CLI,
- využití Ansible OpenStack kolekcí,
- získávání dat z MariaDB[3] a jejich zpracování přes Ansible.

OpenStack CLI je nativní nástroj pro celkovou správu OpenStack cloudu. Pomocí CLI je možné získat údaje o cloudové konfiguraci, uživatelských datech a je možné připojit ke všem nainstalovaným komponentám. Použití pouze OpenStack CLI pro účel zálohy a obnovy má dvě zásadní limitace. První z nich je ta, že v závislosti na zdrojích některých dat, které by bylo nutné pro zálohu shromáždit, nemusí OpenStack CLI poskytovat kompletní údaje nebo důležité vlastnosti dat, respektive metadat. K získání takových údajů a vlastností by mohlo být vyžadováno dodatečné volání API. Příkladem takové skutečnosti je použití příkazu `nova list` pro výpis všech instancí a následně použití příkazu `nova show <jméno-instance>` pro výpis metadat konkrétní instance. Nicméně v takto získaném výpisu metadat chybí například údaje o přiřazených svazcích k dané instanci nebo připojená síťová rozhraní. Pro získání těchto údajů by bylo nutné dodatečně použít příkazy `cinder show`, pro získání přiřazených svazků, a `neutron port-show`, pro získání připojených síťových rozhraní. Z tohoto příkladu vyplývá, že použití CLI může být velmi časově náročně a komplexní. S tímto souvisí druhá limitace použití OpenStack CLI. V závislosti na velikosti nasazení OpenStacku tento proces není škálovatelný a to i v případě použití automatizace pro získání potřebných dat. V závěru je důležité zmínit, že použití dodatečných pluginů může celkový proces značně ztížit[4] z důvodu nutnosti použití specializovaného CLI pro konkrétní plugin.

Ansible nabízí kolekci `openstack.cloud` (kolekce je dostupná v literatuře [76]), která umožňuje získávat informace a spravovat určité služby OpenStacku pomocí již implementovaného Python modulu. Použití automatizace značně zjednoduší sběr

[3]Nebo jiné databáze, kterou konkrétní OpenStack používá. V tomto případě je přímo uvedena MariaDB, jelikož je tato databáze nainstalována pro OpenStack, který využívá Kybernetická aréna.

[4]Příkladem může být Neutron plugin LBaaS (Load Balancing as a Service) [73].

potřebných dat pro provedení zálohy. Podobně jako v první možnosti, existují potenciální omezení, která si je důležité uvědomit, a které mohou značně ovlivnit průběh a výsledek zálohy. První omezení souvisí s nekompletními Ansible kolekcemi pro všechny komponenty, které obsahují podstatná data pro provedení zálohy. Například Ansible nenabízí žádnou kolekci, která by umožňovala přístup k orchestračním souborům komponenty Heat[5], pokud by měla probíhat záloha uživatelů a dat s nimi souvisejících, tedy i Heat souborů, nebude možné k nim přistoupit s použitím pouze Ansible kolekcí. Řešením by mohlo být použití vlastního Python modulu, který by interagoval s Heat API, ovšem složitost takového modulu není možné dopředu určit a takové řešení by bylo vyžadováno u více oblastí. I přesto, že `openstack.cloud` kolekce nabízí omezené možnosti pro získávání dat pro účely zálohy, tato kolekce umožňuje správu cloudu a jeho procesů bez nutnosti použití samotného OpenStack CLI, což značně zjednoduší správu cloudu během samotných procesů zálohy a obnovy.

Třetí možností je získávat potřebné údaje z databáze MariaDB, kterou OpenStack používá jako primární databázový backend pro ukládání a správu informací o konfiguraci a stavu různých služeb. To zahrnuje informace o uživatelích, projektech, instancích, obrazech, svazcích a dalších zdrojích. MariaDB je komunitou spravovaná verze databáze MySQL a nabízí tedy i stejné služby. Do těchto služeb patří i export obsahu databáze pomocí `mysqldump` příkazu, kterým je vytvořen soubor obsahující SQL příkazy pro opětovné vytvoření schématu databáze s původními daty. Tyto soubory mohou sloužit jako záloha pro následnou obnovu databáze s vyexportovanými daty. Je možné také upravovat data v databázi přímo, pokud je známo místo uložení těchto dat, nicméně přímá úprava databáze je riskantní a měla by být prováděna s opatrností, protože jakékoli provedené změny mohou ovlivnit stabilitu a spolehlivost OpenStacku. Při vytváření zálohy databáze je zásadní ošetřit situace, ve kterých by mohla vzniknout nekonzistentní data a v případě obnovy dat do databáze zajistit, aby nedošlo k souběhu[6]. V těchto případech mohou být výsledkem poškozená data. Pro ošetření těchto případů je možné provádět zálohu a obnovu v čase údržby OpenStacku, ve kterém žádné služby nebudou do databáze přistupovat, případně před vytvářením zálohy databázi uzamknout příkazem `FLUSH TABLES WITH READ LOCK`, pomocí kterého je přístup to tabulek databáze uzamknut, a tedy je zajištěna konzistence dat [74].

Při sestavování procesu zálohy a obnovy je velmi důležité kontrolovat integritu zálohovaných dat, a to bez ohledu na zvolený přístup. Toto může být docíleno vytvořením otisku zálohovaných dat a následnou kontrolu tohoto otisku zda nedošlo k nevyžádané změně při uložení zálohy nebo při obnově těchto dat.

[5] Anglicky *heat orchestration files*.

[6] Anglicky *race condition*.

Z výše uvedených možností vyplývá, že pro sestavení automatizovaného procesu, který vytváří zálohu a provede následnou obnovu vybraných částí OpenStacku, je nutné zkombinovat OpenStack CLI, Ansible kolekce a přístup do MariaDB databáze. Je to z toho důvodu, že samostatně ani jedna z možností plnohodnotně nepodporuje vytvoření automatizovaného procesu zálohy a obnovy.

5.1.1 Databáze MariaDB

V předchozím textu byla popsána důležitost MariaDB v rámci procesu zálohy a obnovy Arény, ovšem přístup ke čtení a zápisu dat v MariaDB je v celém procesu nenahraditelný. V příkladové situaci, kdy je nutné provést zálohu všech uživatelů OpenStacku bez použití přístupu k MariaDB by bylo možné využít Ansible modul `openstack.cloud.identity_user_info`. Pro obnovu uživatelů ze zálohy by bylo možné využít Ansible modul `openstack.cloud.identity_user`. Tento přístup má ale dvě omezení, pomocí těchto modulů nelze získat heslo uživatele, ani jeho otisk. Druhé omezení je to, že OpenStack nenabízí možnost vytvoření uživatele se specifickým UUID, to je vždy náhodně generováno nové. Právě stejné UUID uživatele je ale využíváno v dalších službách OpenStacku, je tedy zásadní pro funkčnost dalších služeb, aby UUID bylo stejné. UUID nelze změnit ani pomocí OpenStack CLI. Řešením tohoto problému je přímá úprava tabulek v MariaDB databázích.

Výpis 5.1: Výpis MariaDB databázích v kontextu Arény

```
MariaDB [(none)]> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| cinder             |
| glance             |
| heat               |
| keystone           |
| neutron            |
| nova               |
| nova_api           |
| nova_cell0         |
| placement          |
+--------------------+
```

MariaDB obsahuje několik databází, ke kterým OpenStack služby přistupují, konkrétně ve verzi Arény databáze se jedná o databáze Cinder, Glance, Heat, Keystone, Neutron, Nova, Nova API, Nova Cell a Placement, které slouží k poskytování potřebných dat stejně jmenovitým službám OpenStacku. Výpis těchto databází je zobrazen ve výpisu 5.1 Tyto databáze poté obsahují tabulky, ve kterých se nachází již jednotlivé informace o OpenStack zdrojích. Například databáze Keystone obsahuje tabulky `access_role`, `access_rule`, `user`, `user_option` a mnoho dalších.

Pro představu databáze Keystone obsahuje 49 tabulek a databáze Neutron, která je nejobsáhlejší, obsahuje 181 tabulek. Ve výpisu 5.2 se nachází výpis položek obsažených v tabulce `user` v databázi Keystone.

Výpis 5.2: Výpis položek z tabulky `user` v databázi Keystone

```
MariaDB [keystone]> DESCRIBE user;
+--------------------+-------------+------+-----+---------+-------+
| Field              | Type        | Null | Key | Default | Extra |
+--------------------+-------------+------+-----+---------+-------+
| id                 | varchar(64) | NO   | PRI | NULL    |       |
| extra              | text        | YES  |     | NULL    |       |
| enabled            | tinyint(1)  | YES  |     | NULL    |       |
| default_project_id | varchar(64) | YES  | MUL | NULL    |       |
| created_at         | datetime    | YES  |     | NULL    |       |
| last_active_at     | date        | YES  |     | NULL    |       |
| domain_id          | varchar(64) | NO   | MUL | NULL    |       |
+--------------------+-------------+------+-----+---------+-------+
```

Z předchozího textu vyplývá, že databáze obsahují velké množství informací potřebných pro jednotlivé OpenStack služby i OpenStack jako celek. Z tohoto je důležité si uvědomit, že pro zajištění kompletní a úspěšné zálohy a obnovy by bylo nutné veškeré informace obsažené v databázových tabulkách získat a uložit, a to vzhledem k jejich robustnosti by bylo komplikované, což by i zvýšilo pravděpodobnost výskytu chyb. Z toho plyne, že nejvíce spolehlivý způsob je zálohovat databáze jako celek se všemi obsaženými tabulkami a informacemi v nich pomocí výše popsaného příkazu `mysqldump`.

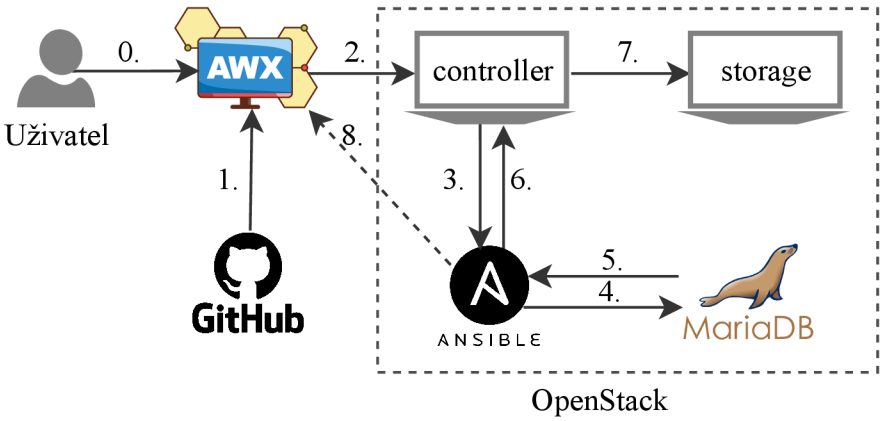
MariaDB obsahuje také databázi *information schema*, která obsahuje metadata o databázi samotné, tedy například o struktuře databáze, tabulkách a uživatelských oprávněních aj. Další databází je *mysql*, jedná se o implicitní systémovou databázi pro MariaDB, která obsahuje data o systémových nastaveních. Databáze *performance schema* obsahuje informace o provozu samotné databáze, udržuje data o frontách, zámcích, vláknech aj. Tyto databáze není nutné do zálohy zahrnout.

5.1.2 Návrh procesů zálohy a obnovy s využitím MariaDB

Před návrhem samotného postupu procesů zálohy a obnovy je podstatné si uvědomit, které prerekvizity je nutné splnit:

- AWX musí mít přístup k OpenStack stanicím a náležitá práva k vykonávání akcí na nich,
- AWX musí mít poskytnuty náležité playbooky, podle kterých bude vykonávat konkrétní akce zálohy a obnovy nad OpenStack stanicemi,
- Přístup do databáze MariaDB s náležitými právy pro provádění akcí nad jednotlivými databázemi.

Za předpokladu splnění těchto prerekvizit je vytvořen diagram procesů zobrazený na obrázku 5.2.



Obr. 5.2: Diagram procesů zálohy a obnovy MariaDB

Uživatel přistupuje k AWX pomocí webového rozhraní, pomocí kterého probíhá veškerá správa přístupu k OpenStack stanicím, správa aktualizace Ansible projektů a jejich spouštění (proces 0). Na Github úložišti se nachází aktuální verze Ansible projektu, AWX k tomuto repozitáři přistupuje a lokálně zdrojový kód uloží. Uživatel může spustit aktualizace manuálně, případně nastavit interval, po kterém se lokálně uložený projekt aktualizuje (proces 1). Po spuštění playbooku pro zálohu, respektive obnovu, se AWX připojí přes SSH do stanice *controller*, na kterém se nachází Docker kontejner s MariaDB databází (proces 2). Na stanici *controller* se nachází Ansible engine, který vykonává akce definované v konkrétním playbooku (proces 3). Ansible přistupuje do Docker kontejneru s databází MariaDB a získá potřebné údaje pro vytvoření záloh, případně vykoná akce související s obnovou databází jednotlivých OpenStack služeb (proces 4). Výsledek procesu 4, včetně návratového kódu a výstupních souborů, je odeslán zpět do Ansible ke zpracování (proces 5). Ansible poté uloží výstupní soubory do předem definovaného adresáře, svazku nebo externího serveru (proces 6 a 7). Současně Ansible odesílá výstup vykonaných akcí do prostředí AWX (proces 8). Proces zálohy i obnovy sdílejí diagram kromě procesu 6 a 7, které mají opačnou orientaci ve smyslu získání souborů zálohy z místa jejich uložení.

6 Praktické řešení

Tato kapitola obsahuje postup přípravy experimentálního pracoviště, ve kterém jsou následně implementovány konkrétní postupy zálohování. Cílem implementace záloh a obnov vybraných částí OpenStacku je především zjednodušení správy Arény a zajištění její přenositelnosti. Z tohoto důvodu pracovní prostředí vychází z prostředí samotné Kybernetické arény.

Příprava pracovního prostředí se skládá z nasazení OpenStacku a Ansible AWX, který proces zálohy a obnovy zautomatizuje. Pro zprovoznění pracovního prostředí je poskytnut OpenStack, který se nachází v laboratoři na Ústavu komunikací a zároveň je nad tímto OpenStackem postavena Kybernetická aréna. Přístup k OpenStacku je zajištěn pomocí VPN (Virtual Private Network). Pro odlišení jednotlivých OpenStack platforem v následujícím textu je hlavní OpenStack, na kterém je postavena Aréna, nazván **majoritní OpenStack** a OpenStack, který bude nasazen pro potřebu implementace záloh je nazván **minoritní OpenStack**. Nasazení minoritnímu OpenStacku je věnována následující podkapitola.

6.1 Nasazení platformy OpenStack

Pro zajištění kompatibility záloh implementovaných v minoritním OpenStacku s majoritním OpenStackem musí pracovní prostředí odpovídat *skutečnému* prostředí s Arénou. Proto je minoritní OpenStack nasazen na jeden kontrolní uzel `ctrl01`, dva výpočetní uzly `comp01` a `comp02` a jeden úložný uzel `stor01`. Tabulka 6.1 obsahuje podrobnosti o minoritním OpenStacku.

Tab. 6.1: Údaje o virtuálních strojích pro nasazení minoritního OpenStacku

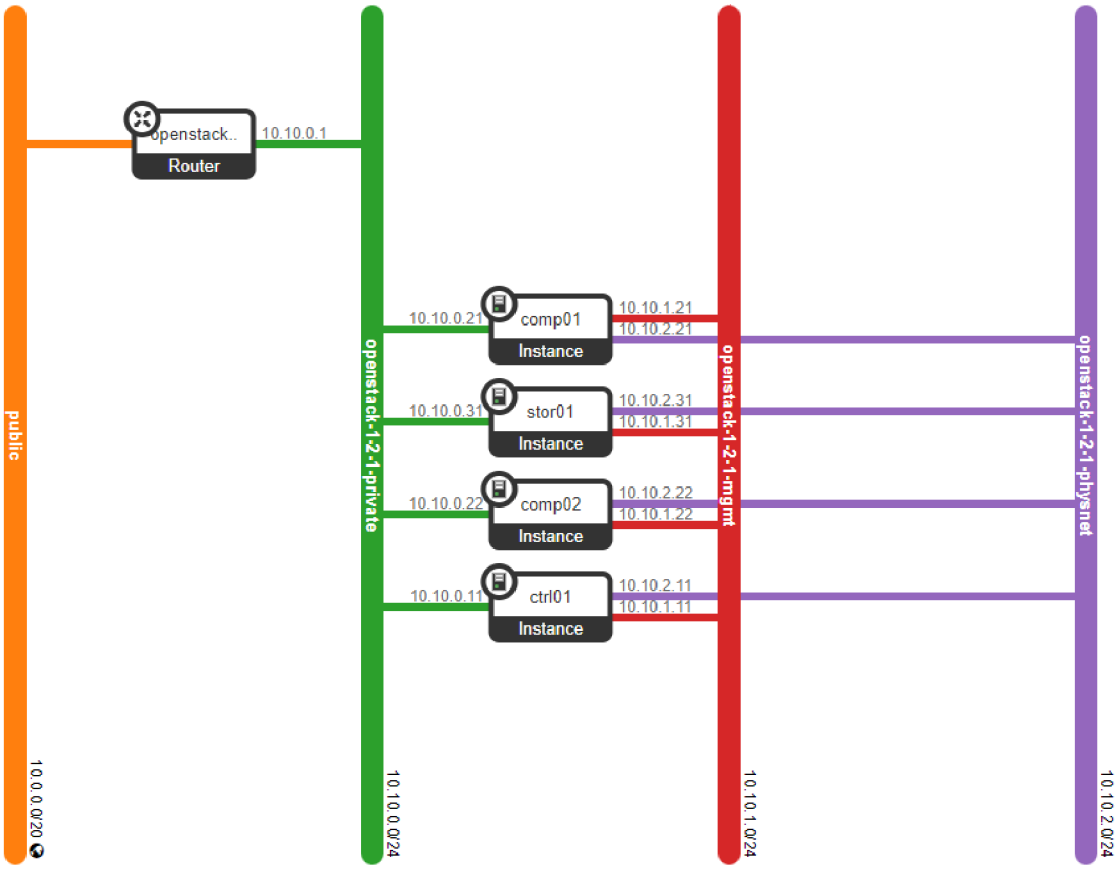
Způsob nasazení	Kolla-Ansible			
Verze OpenStacku	Victoria			
Komponenty	Nova, Neutron, Keystone, Cinder, Glance, Heat, Horizon, Placement			
Název uzlu	ctrl01	comp01	comp02	stor01
Plovoucí IP adresa	10.0.6.11	10.0.6.12	10.0.6.13	10.0.6.14
Privátní IP adresa	10.10.0.11	10.10.0.21	10.10.0.22	10.10.0.31
Operační systém	CentOS 8 Stream			
Počet vCPU	12	8	8	2
RAM	16 GB	6 GB	6 GB	2 GB
Disk	200 GB	200 GB	200 GB	200 GB

Společně s uzly `ctrl01`, `comp01`, `comp02` a `stor01` jsou v majoritním OpenStacku připraveny virtuální sítě:

- `openstack-1-2-1-private` (dále uváděno jako síť `private`),

- `openstack-1-2-1-physnet` (dále uváděno jako síť `physnet`),
- `openstack-1-2-1-mgmt` (dále uváděno jako síť `mgmt`).

Síť `private` slouží pro připojení OpenStack uzlů k internetu a propůjčuje plovoucí IP adresy[1]. Síť `physnet` simuluje fyzickou síť, která spojuje jednotlivé uzly, zároveň se jedná o síť pro datovou komunikaci mezi uzly OpenStacku. Síť `mgmt` slouží pro interní komunikaci mezi OpenStack komponentami. Síťová topologie se nachází na obrázku 6.1[2].



Obr. 6.1: Síťová topologie minoritního OpenStacku

Nasazení minoritního OpenStacku následuje doporučení pro nasazení, které je dostupné v literatuře [75]. Kolla-Ansible při nasazení postupuje podle konfiguračního souboru `globals.yml`, který před nasazením musí být upraven tak, aby nasazený OpenStack vyhovoval specifickým potřebám a danému prostředí. Podle tohoto souboru se při nasazení spouštějí konkrétní playbooky, které zajišťují instalaci a konfiguraci jednotlivých komponent. Upravené části souboru `globals.yml` se nachází ve výpisu 6.1.

[1] Anglicky *floating IP addresses*, tyto adresy slouží pro přístup k instancím z externích sítí. Fungují na principu NAT 1:1 (Network Address Translation).

[2] Tato topologie je vygenerována v prostředí Horizon majoritního OpenStacku.

Výpis 6.1: Upravené části souboru `globals.yml`

```
kolla_base_distro: "centos"
kolla_install_type: "source"
openstack_release: "victoria"
kolla_internal_vip_address: "10.10.1.11"
neutron_plugin_agent: "linuxbridge"
enable_openstack_core: "yes"
enable_haproxy: "no"
enable_chrony: "yes"
enable_cinder: "yes"
enable_cinder_backup: "no"
enable_cinder_backend_lvm: "yes"
enable_nova_ssh: "yes"
network_interface: "eth1"
neutron_external_interface: "eth2"
```

Kromě editace souboru `globals.yml` je nutné poskytnout Ansible přístup do uzlů `ctrl01`, `comp01`, `comp02` a `stor01`, toho je docíleno úpravou souboru `multinode`, který slouží jako inventář. V inventáři je také možné nastavit instalaci komponent na konkrétní uzly. V souboru jsou definovány dvě odlišnosti, kterými minoritní OpenStack bude disponovat oproti majoritnímu OpenStacku:

- uzel `comp01` není součástí stanice s uzlem `ctrl01`, v topologii minoritního OpenStacku se jedná o dva samostatné VS,
- komponenta Glance je součástí uzlu `stor01`, v majoritním OpenStacku je komponenta Glance instalována na *controller*.

Tyto změny byly provedeny z toho důvodu, že v budoucnu je plánováno Arénu přenést na takto nasazený OpenStack [34]. Po úspěšném nasazení jsou vygenerovány přístupové údaje do prostředí Horizon a přístupový soubor k OpenStack CLI. Upravené části souboru `multinode` jsou zobrazeny ve výpisu 6.2.

Výpis 6.2: Upravené části souboru `multinode`

```
[control]
ctrl01 ansible_ssh_private_key_file=/home/centos/openstack.pem
   ansible_ssh_user=centos ansible_user=centos

[network]
comp01 ansible_ssh_private_key_file=/home/centos/openstack.pem
   ansible_ssh_user=centos ansible_user=centos

[compute]
comp01 ansible_ssh_private_key_file=/home/centos/openstack.pem
   ansible_ssh_user=centos ansible_user=centos
comp02 ansible_ssh_private_key_file=/home/centos/openstack.pem
   ansible_ssh_user=centos ansible_user=centos

[monitoring]
ctrl01 ansible_ssh_private_key_file=/home/centos/openstack.pem
   ansible_ssh_user=centos ansible_user=centos

[storage]
comp02 ansible_ssh_private_key_file=/home/centos/openstack.pem
   ansible_ssh_user=centos ansible_user=centos

[deployment]
localhost ansible_connection=local become=true

[glance-server]
stor01 ansible_ssh_private_key_file=/home/centos/openstack.pem
   ansible_ssh_user=centos ansible_user=centos

[glance:children]
glance-server
```

6.2 Nasazení Ansible AWX

Ansible AWX je nasazen do VS, který je součástí majoritního OpenStacku. Jako operační systém VS je zvolen Rocky Linux 8, který patří do linuxové distribuce založené na RHEL. Na OS RHEL běží OpenStack a AWX Kybernetické arény.

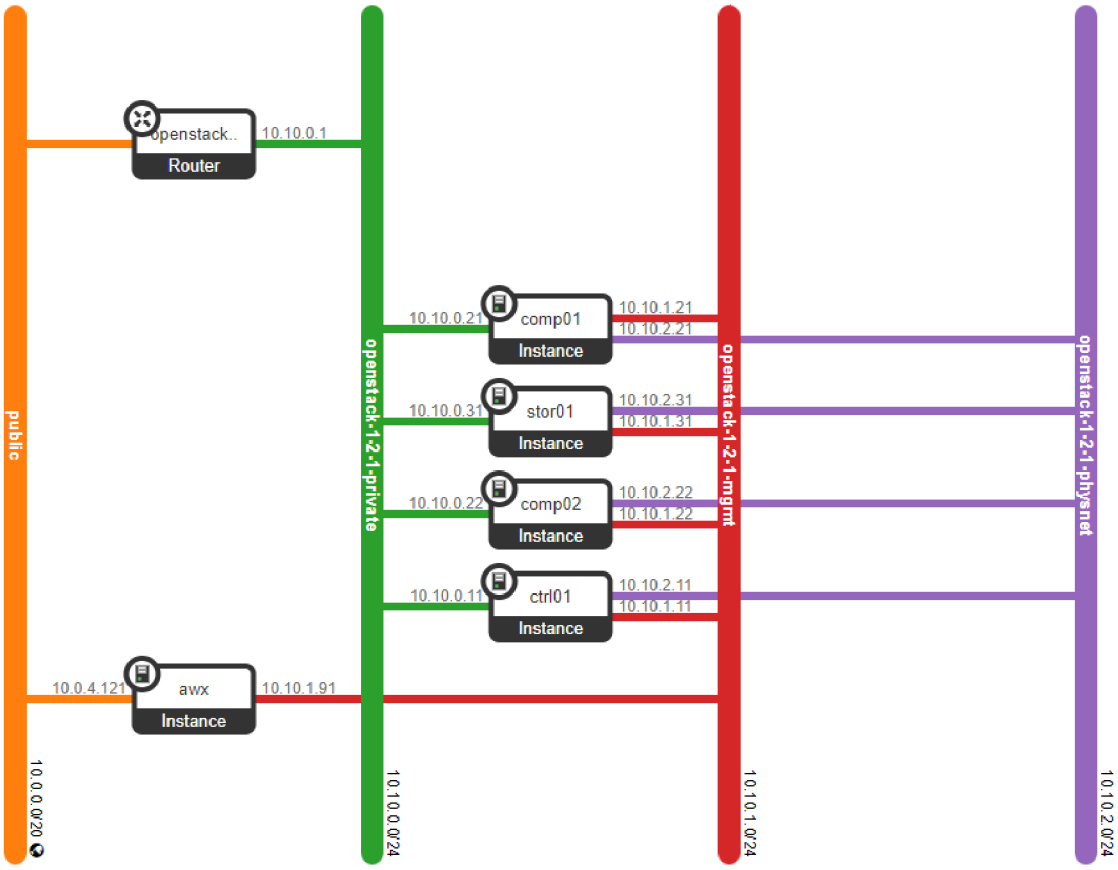
Metoda instalace následuje doporučení AWX projektu a je zvolena metoda instalace pomocí `awx-operator` se zabezpečeným přístupem HTTPS (Hypertext Transfer Protocol Secure). Jako kontejnerový manager byl zvolen K3S, a to z těch důvodů, že potřebuje menší výpočetní prostředky, je vhodný do produkčního prostředí a implicitně podporuje přístup k vnitřním službám z externí sítě pomocí TLS spojení, viz podkapitola 4.3.2.

Údaje o VS, do kterého je AWX nasazen, jsou uvedeny v tabulce 6.2. Přístup do VS je zajištěn SSH spojením pomocí klíčového páru generovaného RSA (Rivest Shamir Adleman) algoritmem. Podrobný postup instalace a ovládání AWX je obsažen v příručce, která je připojena v příloze A této práce.

Tab. 6.2: Údaje o virtuálním stroji pro instalaci AWX

Název instance v majoritním OpenStacku	awx
Operační systém	Rocky Linux 8.6
Počet vCPU	6
RAM	8 GB
Disk	40 GB
IP adresa rozhraní `eth0`	10.0.4.121
IP adresa rozhraní `eth1`	10.10.1.91
Sudo uživatel	rocky

VS disponuje dvěma rozhraními, `eth0` patřící do sítě `public`, která zajišťuje spojení s veřejnou sítí, a rozhraní `eth1`, které patří do sítě `mgmt`. Rozhraní `eth1` slouží k ustanovení spojení mezi OpenStack cloudem a AWX, přes které je směrována veškerá správa minoritního OpenStacku. Topologie minoritního OpenStacku a Ansible AWX je zobrazena na obrázku 6.2.



Obr. 6.2: Síťová topologie minoritního OpenStacku a Ansible AWX

6.2.1 AWX a komunikace s OpenStackem

Z obrázku 6.2 je zřetelné, že stanice `awx` je přímo spojena se sítí `mgmt`, a tedy AWX má přímý přístup k OpenStack stanicím. Pro implementaci procesů obnovy a zálohy je nutné, aby AWX byl autorizován k vykonání akcí nad OpenStack stanicemi `ctrl01`, `comp01`, `comp02` a `stor01`. Pro tento účel je stanice `awx` se stanicí `ctrl01` autentizována, v AWX prostředí jsou nastaveny údaje pro SSH spojení, které AWX uchovává v šifrované podobě. Jelikož AWX vykonává akce vždy přes stanici `ctrl01`, není nutné AWX zadávat údaje i pro zbylé tři stanice. Z důvodu bezpečnosti byl vytvořen na stanici `ctrl01` nový uživatel `ansible`, na kterého se AWX připojuje při autentizaci. Tento uživatel patří do `sudo` skupiny a je tedy schopný spouštět příkazy s administrátorskými právy, nejedná se ale o administrátora stanice `ctrl01`.

Přístup k OpenStack CLI

Aby mohl AWX vykonávat úkoly nad minoritním OpenStackem, je nutné AWX poskytnout oprávnění nejen pro přístup do stanice `ctrl01`, ale i ke službám OpenStacku. Ansible nabízí kolekci `openstack.cloud` (kolekce je dostupná v literatuře [76]), která umožňuje získávat informace a spravovat určité služby OpenStacku pomocí již implementovaného Python modulu. Aby mohl AWX kolekci použít, je nutné splnit tyto prerekvizity:

- Ansible verze 2.8 a vyšší,
- Python verze 3.6,
- balíček `openstacksdk` ve verzi vyšší než 0.36 a menší než 0.99 a nainstalovaný na spravovaném uzlu[3],
- autentizační údaje ke službám OpenStacku.

Pro splnění těchto prerekvizit jsou na uzlu `ctrl01` nainstalovány výše uvedené balíčky, viz následující výpis 6.3.

Výpis 6.3: Instalace prerekvizit pro použití Ansible OpenStack kolekce

```
$ sudo dnf install python36
$ sudo pip3 install ansible
$ sudo pip3 install "openstacksdk>=1.0"
$ sudo ansible-galaxy collection install openstack.cloud
```

Posledním uvedeným příkazem je nainstalována samotná `openstack.cloud` kolekce. Dále je nutné této kolekci poskytnout autentizační údaje pro přístup k OpenStack službám. Zpravidla je možné se připojit k libovolnému množství OpenStack

[3]Respektive na uzlu, který tento modul spouští. Důležité je dodat, že i přesto, že dokumentace přesně stanovuje rozmezí vyhovujících verzí, při použití tohoto balíčku (spuštění Ansible playbooku) je hlášena chyba o nekompatibilní verzi openstacksdk. Z tohoto důvodu bylo odkloněno od `openstack.cloud` dokumentace a byla nainstalována verze `openstacksdk` 1.0.

cloudů s tím, že ke každému jsou poskytnuté Keystone autentizační údaje. Toho je docíleno vytvořením souboru `clouds.yaml`, který `openstack.cloud` kolekce vyhledává při svém spuštění. Nově vytvořený soubor `clouds.yaml` je zobrazen ve výpisu 6.4.

Výpis 6.4: Obsah autentizačního Keystone souboru `clouds.yaml`

```
---
clouds:
  openstack-1-2-1: #Libovolné označení cloudu
    identity_api_version: 3
    auth:
      auth_url: http://10.10.1.11:35357/v3
      password: <heslo>
      project_name: admin
      username: admin
      user_domain_name: Default
      project_domain_name: Default
    regions:
    - name: RegionOne
```

Údaje v souboru `clouds.yaml` jsou získány ze souboru `admin-openrc.sh`, který je vygenerován po úspěšně nasazeném OpenStacku pomocí nástroje Kolla-Ansible.

6.2.2 Přístup k databázi MariaDB

OpenStack databáze MariaDB se nachází na stanici `ctrl01` v Docker kontejneru. Pro provádění akcí s MariaDB je nutné mít zajištěn plnohodnotný přístup do tohoto kontejneru a do MySQL databáze, která v něm běží. Při nasazení OpenStacku Kolla-Ansible vytváří přístupové údaje s admin právy do MySQL databáze. Jelikož ale není vhodné, aby AWX prováděl akce přes admin uživatele, je pro účel zálohy a obnovy vytvořen uživatel `ansible`, který bude mít práva pro přístup do MySQL tabulek, jejich vytváření, mazání, upravování, zobrazování, zapisování, zamykání, vytváření `dump` souborů a jejich obnovování. Následující výpis 6.5 obsahuje příkazy pro přístup do MariaDB kontejneru a vytvoření nového MySQL uživatele `ansible` s výše uvedenými právy[4].

[4]Pro provádění jakýchkoliv akcí v OpenStacku je nejprve nutné autentizovat uživatele pomocí příkazu `source` a uvedení autentizačního souboru `admin-openrc.sh`.

Výpis 6.5: Přístup do MariaDB kontejneru a vytvoření nového MySQL uživatele

```
$ sudo docker exec -ti mariadb /bin/bash
$ mysql -u root -p
 CREATE USER 'ansible'@'%' IDENTIFIED BY '<heslo>';
 GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, INDEX, ALTER,
    CREATE TEMPORARY TABLES, LOCK TABLES, EXECUTE, CREATE VIEW,
    SHOW VIEW, EVENT, TRIGGER, REFERENCES, FILE ON *.* TO 'ansible'
    @'%' WITH GRANT OPTION;
 FLUSH PRIVILEGES;
```

Pro úspěšné vykonání akcí musí být AWX schopný přistoupit do Docker kontejneru a přihlásit se na MySQL uživatele `ansible`. Přístup do Docker kontejnerů a vykonávání akcí v něm umožňuje Ansible kolekce `community.docker`, která poskytuje moduly pro základní správu a získávání informací o Docker kontejnerech. Tato kolekce pro svou funkčnost potřebuje nainstalované balíčky:

- Docker API verze 1.25 a vyšší,
- balíček `requests` [77].

Kolekce `community.docker` obsahuje moduly pro získání informací o příslušném kontejneru a jeho stavu a spouštění příkazů v něm, konkrétně se jedná o moduly `docker_container_info` a `docker_container_exec`. Nevýhoda `docker_container_exec` modulu je ta, že obsahuje pouze funkci `command`, nikoliv `shell`, tudíž není možné tento modul použít ke spouštění příkazů, které obsahují speciální znaky `<`, `|`, `&` apod., což je v rámci použití MySQL příkazů značně omezující. V implementaci je proto toto omezení vyřešeno použitím `ansible.builtin.shell` modulu a příkazu `docker exec`, který v MariaDB kontejneru spustí požadovaný příkaz.

Pro provádění akcí v MySQL databázi v MariaDB kontejneru je nutné přihlásit se na konkrétního uživatele s potřebnými právy, k tomuto účelu byl vytvořen zmíněný uživatel `ansible` s příslušným heslem. Playbooky, které budou zálohu a obnovu automatizovat, musí mít tyto údaje dostupné, ale z důvodu bezpečnosti je nežádoucí tyto informace v playboocích uchovávat přímo. Pro tyto účely existuje nástroj Ansible-vault, který heslo k MariaDB zašifruje a při spuštění playbooků je nutné zadat heslo k dešifrování šifrovacího klíče, který Ansible-vault uchovává. Pro použití šifrovaného hesla k MySQL databázi v MariaDB kontejneru je nutné šifru vygenerovat s příslušným heslem ke klíči, viz následující příkaz 6.6.

Výpis 6.6: Generování zašifrovaného hesla pomocí Ansible vault

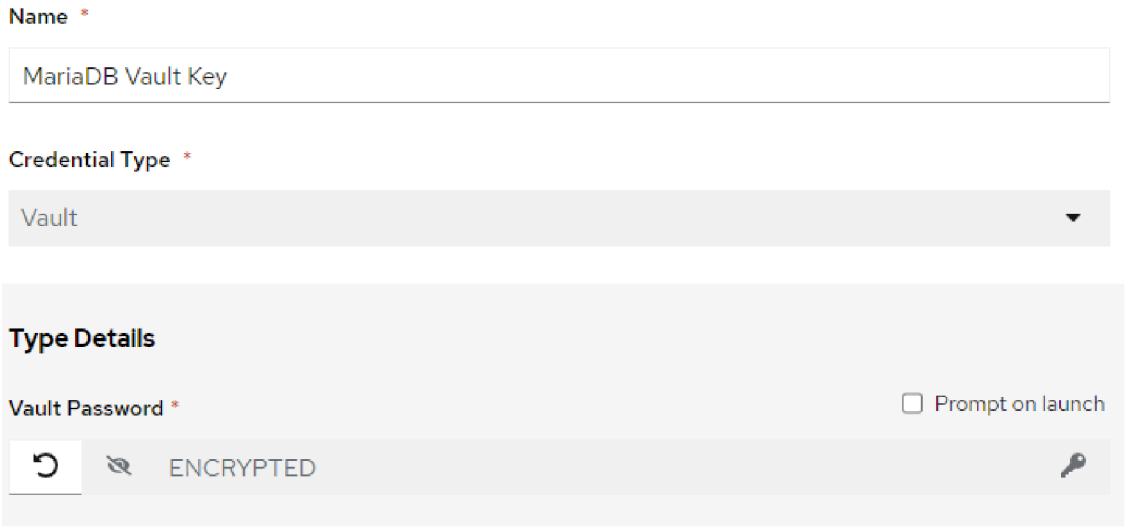
```
$ ansible-vault encrypt-string <heslo> --ask-vault-pass
password: <heslo-k-sifrovacimu-klici>
```

Po spuštění tohoto příkazu je uživatel vyzván k zadání hesla pro šifrování a dešifrování zašifrovaného hesla, výsledek je samotná šifra a údaje o použitém algoritmu pro Ansible, viz příkaz 6.7. Ansible-vault je nativní funkcionalita Ansible a není nutné ho dodatečně instalovat.

Výpis 6.7: Ukázka zašifrovaného MariaDB hesla pomocí Ansible vault v Ansible projektu

```
mariadb_password: !vault |
$ ANSIBLE_VAULT;1.1;AES256;
373966336239393233636165363132353865643539643636396330613932663734303
366164303264336537396532663161633630393363613432356565362650a326432
336463343232643061343434396535663036623762396365396536343932333335343
633732336366386139653366633865376264313365356237363330a37653662656534
666165386366336536316333613634396363939376466353265
```

AWX podporuje Ansible-vault funkcionalitu uložením hesla k dešifrování šifrovacího klíče v záložce *credentials* v AWX prostředí. Pokud je heslo ztraceno, neexistuje žádný způsob, jak ho získat zpět a musí být vygenerována nová šifra s novým heslem ke klíči. Na obrázku 6.3 je zobrazeno zašifrované heslo ke klíči, který dešifruje heslo k MySQL databázi uživatele `ansible`.

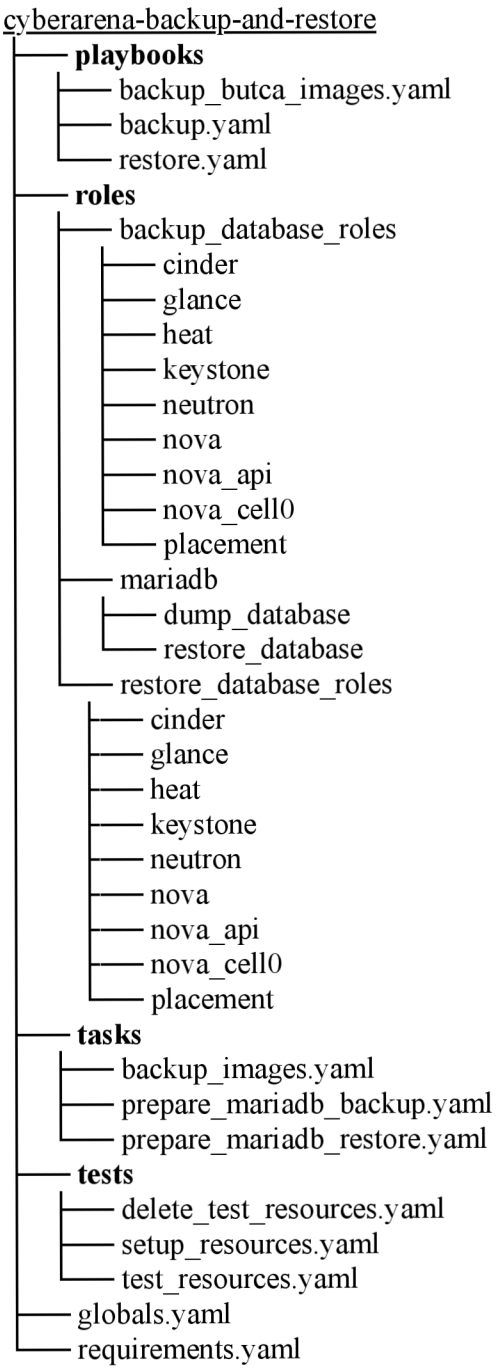


Obr. 6.3: Ansible-vault údaj v prostředí AWX

6.3 Implementace procesů zálohy a obnovy

V této části je rozebrána implementace jednotlivých kroků, které tvoří procesy zálohy a obnovy Arény. Při implementaci je důležité si uvědomit, že celý Ansible projekt by měl být snadno laditelný a snadno pochopitelný pro budoucí obsluhu, replikaci a úpravy. Toho je docíleno rozvržením projektu do jednotlivých úkolů (*tasks*) a rolí (*roles*). Na obrázku 6.4 je zobrazena struktura Ansible projektu, ve kterém jsou implementovány jednotlivé kroky zálohy obnovy. Ansible projekt MariaDB-backup-and-restore se nachází na GitHub úložišti, dostupné v referenci [78].

Struktura projektu je rozdělena do pěti hlavních částí:



Obr. 6.4: Kompletní struktura Ansible projektu

- globální parametry *globals.yaml*,
- *playbooks*,
- *roles*,
- *tasks*,
- *tests*.

Na nejvyšší úrovni projektu se nacházejí soubory `globals.yaml`, ve kterých se na-

cházejí globální proměnné používané v projektu. Tyto proměnné mají nejmenší prioritu a jejich hodnoty jsou použity pouze v případě, že nejsou přepsány hodnotami lokálními a `extra-vars`. Konkrétně se v souboru `globals.yaml` nachází zašifrované heslo k databázi MariaBD, cesta k adresáři na stanici `ctrl01` a dostupné databáze v MariaDB.

V adresáři *playbooks* se nacházejí hlavní playbooky `backup.yaml`, `restore.yaml` a `backup_butca_images.yaml`. Poslední jmenovaný playbook zálohuje obrazy operačních systémů obsahující scénáře kyberbezpečnostních her. MariaDB tyto obrazy neobsahuje, tudíž je nutné je získat přímo spojením se službou Glance. Tyto hlavní playbooky jsou spouštěny z prostředí AWX. K tomu, aby je AWX byl schopen rozeznat, je nutné v playbooku specifikovat parametr `hosts`, který musí mít shodnou hodnotu jako hodnota zadaná v prostředí AWX. V tomto případě hodnota parametru `hosts` je `controller`. IP adresa je ke stanici *controller* zadána v prostředí AWX.

6.3.1 Proces zálohy

Proces zálohy se celkem skládá ze tří částí:

1. Načtení hlavního playbooku a parametrů.
2. Kontrola prerekvizit procesu zálohy.
3. Vykonání akcí procesu zálohy pro zadané databáze a vytvoření zálohy na externím serveru.

Proces zálohy obsahuje pouze jeden parametr, který je definován jako `extra-vars` před spuštěním hlavního playbooku. Uživatel pomocí `extra-vars` určí, které databáze budou zálohovány, konkrétní databáze jsou zadány jako položky seznamu parametru `selected_databases`. V případě, že mají být zálohovány všechny databáze, lze všechny databáze označit hodnotou *all*. Ukázka naplnění parametru `selected_databases` zobrazuje výpis 6.8.

Výpis 6.8: Naplnění parameru `selected_databases`.

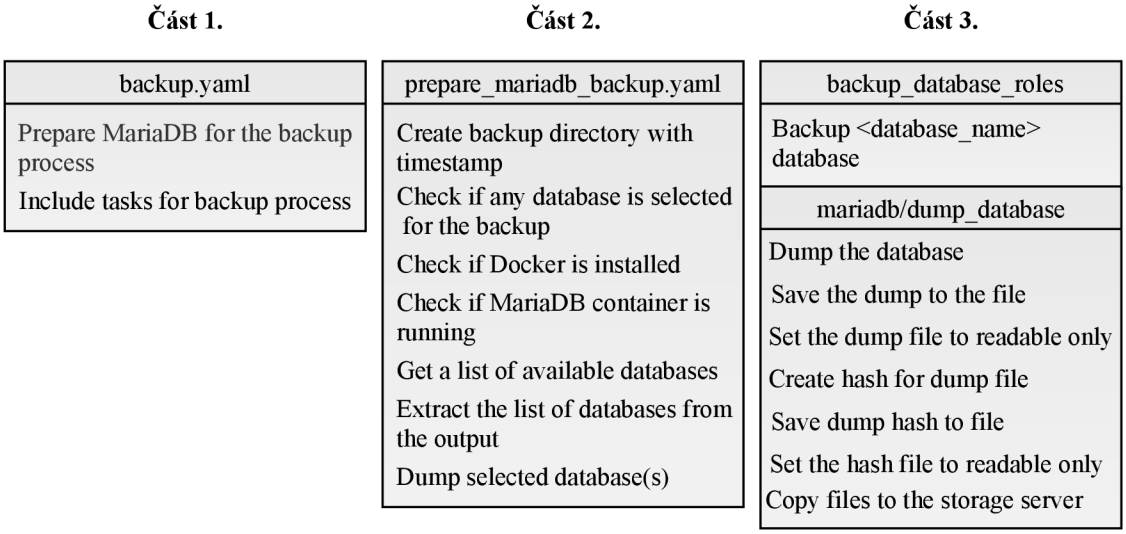
```
---
selected_databases:
  - cinder
  - keystone
  - neutron
  - nova-api
```

Z hlavního playbooku `backup.yaml` je dále zavolán playbook `prepare_mariadb_backup.yaml`, ve kterém jsou definovány akce související s kontrolou prerekvizit, zda proces záloh požadovaných databází může být spuštěn a zároveň je vytvořen adresář. Konkrétně se jedná o akce spojené s:

- tvorbou adresáře s aktuální časovou stopou,

- ověřením, zda alespoň jedna databáze je zvolena k záloze,
- ověřením, zda MariaDB kontejner je ve stavu *running*,
- ověřením, zda databáze vybrané k záloze skutečně existují v MariaDB.

V případě pozitivního výsledku části 2. se z playbooku `prepare_mariadb_backup.yaml` volají role jednotlivých databází, které uživatel zadal k zálohování. Z playbooku `prepare_mariadb_backup.yaml` jsou tedy volány požadované role nacházející se v `backup_database_roles/`. Z těchto konkrétních rolí je volána jednotná role `mariadb/dump_database/`, která již specifické akce zálohy provádí. Postup zálohování všech databází je shodný, důvod rozdělení do jednotlivých rolí je ten, že v budoucnu mohou být definovány dodatečné akce specifické pro konkrétní databázi. S použitím těchto rolí není nutné v takovém případě zasahovat do již vytvořené struktury projektu. Celý proces zálohy a volání jednotlivých playbooků s konkrétními podstatnými akcemi znázorňuje diagram na obrázku 6.5.



Obr. 6.5: Diagram procesu zálohy

Výsledkem procesu zálohy jsou tzv. *dump* soubory a hash obsahu každého souboru zálohy. Hash je získán z důvodu kontroly před procesem obnovy, zda je obsah *dump* souboru stále stejný. Soubory zálohy se nacházejí ve složce `backup_path/`, která je konkrétně nastavena v souboru `globals.yaml` na `/home/ansible/openstack_backups/`. Při spuštění procesu zálohy se vytvoří nový adresář, která je pojmenována podle aktuální časové stopy, tímto jsou jednotlivé zálohy odlišeny. Obsah tohoto adresáře po úspěšně provedeném procesu zálohy všech dostupných databází je zobrazen ve výpisu 6.9.

Výpis 6.9: Obsah adresáře `/home/ansible/openstack_backups/` po úspěšně provedeném procesu zálohy.

```
[ansible@ctrl01 2023-04-29_17:31:16]$ ls -l
-r--r--r--. 1 root root  47312 Apr 29 17:30 cinder_dump.sql
-r--r--r--. 1 root root     64 Apr 29 17:31 cinder_dump.sql.hash
-r--r--r--. 1 root root 105924 Apr 29 17:31 glance_dump.sql
-r--r--r--. 1 root root     64 Apr 29 17:31 glance_dump.sql.hash
-r--r--r--. 1 root root  25351 Apr 29 17:31 heat_dump.sql
-r--r--r--. 1 root root     64 Apr 29 17:31 heat_dump.sql.hash
-r--r--r--. 1 root root  57799 Apr 29 17:31 keystone_dump.sql
-r--r--r--. 1 root root     64 Apr 29 17:31 keystone_dump.sql.hash
-r--r--r--. 1 root root 209733 Apr 29 17:31 neutron_dump.sql
-r--r--r--. 1 root root     64 Apr 29 17:31 neutron_dump.sql.hash
-r--r--r--. 1 root root  57048 Apr 29 17:32 nova_api_dump.sql
-r--r--r--. 1 root root     64 Apr 29 17:32 nova_api_dump.sql.hash
-r--r--r--. 1 root root 130379 Apr 29 17:32 nova_cell0_dump.sql
-r--r--r--. 1 root root     64 Apr 29 17:32 nova_cell0_dump.sql.hash
-r--r--r--. 1 root root 173609 Apr 29 17:32 nova_dump.sql
-r--r--r--. 1 root root     64 Apr 29 17:32 nova_dump.sql.hash
-r--r--r--. 1 root root  35820 Apr 29 17:32 placement_dump.sql
-r--r--r--. 1 root root     64 Apr 29 17:32 placement_dump.sql.hash
```

6.3.2 Proces obnovy

Obdobně jako proces zálohy se proces obnovy skládá ze tří částí:

1. Načtení hlavního playbooku a parametrů.
2. Kontrola prerekvizit před spuštěním obnovy.
3. Obnova vybraných databází.

Pro spuštění procesu obnovy je nutné kromě zadání databází pro obnovy také zadat údaj o časové stopě, která byla vytvořena při procesu zálohy. Obdobně jako při spouštění hlavního playbooku pro zálohu jsou tyto parametry zadány jako `extra_vars` v prostředí AWX, konkrétní příklad je uveden v následujícím výpisu 6.10.

Výpis 6.10: Naplnění parametrů `selected_backups` a `timestamp`.

```
---
selected_backups:
  - all
timestamp:
  - 2023-04-29_17:31:16
```

Z hlavního playbooku `restore.yaml` je volán playbook `prepare_mariadb_restore.yam`, který obsahuje akce sloužící k ověření, zda je možné samotnou obnovu uskutečnit. Konkrétně se jedná o:

- ověření, zda je alespoň jedna databáze zvolena k obnově a zda se jedná o validní databázi,
- ověření, zda adresář se zadaným `timestamp` existuje, a zda se v něm nacházejí zvolené databáze,
- ověření, zda MariaDB kontejner je dostupný,

- vytvoření záloh aktuálních databázích.

V případě, že ověření neselže, jsou volány role `restore_database_roles/` a následně `mariadb/restore_database/`. Stejně jako v případě procesu zálohy, je proces obnovy pro všechny databáze stejný. V případě změn v budoucnu jsou tyto role připraveny k úpravě bez nutnosti zasahovat do již vytvořené struktury projektu.

Před obnovou jednotlivých databází je získána záloha z aktuálního stavu databáze, a to pro případ, že původní zálohy jsou poškozené, nebo by proces obnovy nečekaným způsobem selhal. Tyto zálohy jsou uloženy do adresáře s hodnotou zadaného `timestamp`, do nového adresáře `backup_before_restore`. Před tím než je možné nahrát obsah zálohovaných souborů do nových databází je nutné aktuální databáze smazat (v MySQL terminologii se jedná o tzv. *drop*). Následně vytvořit databázi s novým názvem a obsah do databází nahrát ze zálohovaných souborů. Samotná obnova ze zálohy je vykonána příkazem ve výpisu 6.11.

Výpis 6.11: Ansible *task* pro obnovu konkrétní databáze.

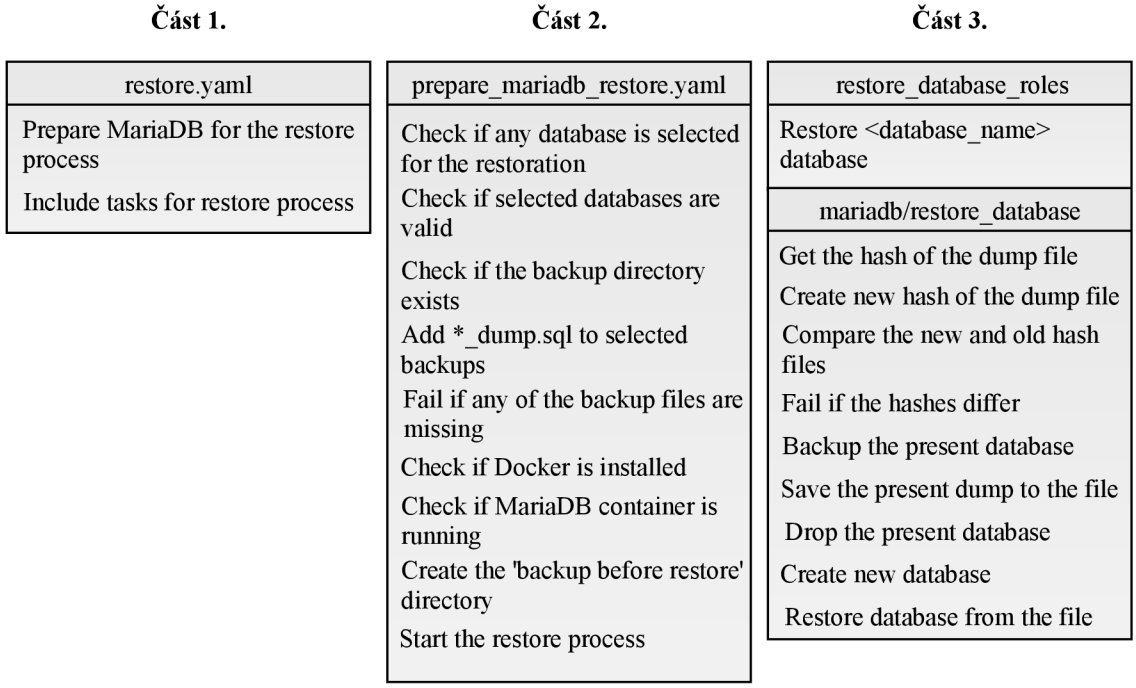
```
- name: Restore database from the file
  shell: "sudo docker exec -i mariadb mysql -u ansible -p{{
      mariadb_password }} {{ database_name }} < {{ backup_dir }}/{{
      database }}"
```

Pro provedení této akce byl použit Ansible modul `ansible.builtin.shell`. Toto řešení není přívětivé, jelikož při selhání není možné získat přesný výpis s důvodem selhání. Jak bylo ale uvedeno v textu dříve, `community.docker` kolekce neobsahuje modul `shell`, který by podporoval znak přesměrování vstupu.

Na obrázku 6.6 se nachází podstatné akce implementované v Ansible projektu, které jsou spjaty s procesem obnovy.

6.3.3 Záloha Glance obrazů

Záloha Glance obrazů se skládá ze dvou playbooků `backup_butca_images.yaml`, který je spouštěn z prostředí AWX, a playbooku `backup_images.yaml`, který obsahuje jednotlivé akce pro získání obrazů ze služby Glance. Kolekce `openstack.cloud` neobsahuje žádný modul, který by umožňoval uložit obrazy dostupné v Glance. Z toho důvodu je použito OpenStack CLI, k jeho použití je nutné se autentizovat. Toho je docíleno načtením autentizačního souboru `admin-openrc.sh`. Z důvodu, že Ansible pro každý *task* vytváří nové shell prostředí, do kterého se autentizace nepřenáší, je nutné autentizaci a uložení Glance obrazů vykonat v jedné shell relaci, tudíž v jednom Ansible *task*. Implementace v Ansible je zobrazena ve výpisu 6.12.



Obr. 6.6: Diagram procesu obnovy

Výpis 6.12: Implementace autentizace a uložení Glance obrazů v Ansible.

```
- name: Download Glance image
  shell: "source ~/admin-openrc.sh; openstack image save --file {{
      backup_path }}/glance_images/{{ ansible_date_time.date }}_{{
      ansible_date_time.time }}/{{ item.name }}.{{ item.disk_format }}
      {{ item.name }}"
  with_items: "{{ glance_images.images }}"
```

V této implementaci je pomocí smyčky `with_items` iterováno přes získané názvy jednotlivých obrazů v Glance. Výsledkem jsou zálohované obrazy dostupné v adresáři s konkrétním `timestamp` obdobně jako v případě procesu zálohy.

6.4 Ověření funkčnosti procesů zálohy a obnovy

Při implementaci obsáhlého projektu je důležité průběžně ověřovat jeho funkčnost, respektive zda výsledek odpovídá očekávání. Ověření funkčnosti procesů zálohy a obnovy je provedeno vytvořením OpenStack struktur. Konkrétní kroky, které jsou vykonány pro ověření funkčnosti projektu, jsou:

- vytvoření OpenStack struktur,
- vytvoření zálohy,
- smazání struktur,
- obnovení zálohy,
- automatické a manuální testování,

- testování existence a funkčnosti původních struktur.

Aby bylo možné tyto kroky vykonávat v případě provedení úprav v projektu opakovaně, bylo implementováno automatizované vytvoření OpenStack struktur, jejich smazání a kontrola, zda se v OpenStacku nacházejí pro provedení procesu obnovy. Tyto tři části byly implementovány v playboocích:

- `setup_resources.yaml`
- `delete_resources.yaml`
- `test_resources.yaml`

První playbook vytváří v OpenStacku tyto konkrétní struktury – projekty, uživatelé, přístupová práva, sítě, podsítě, směrovače, bezpečnostní skupiny a jejich pravidla, klíčový pár, *flavors* a instance. Druhý zmíněný playbook tyto vytvořené struktury smaže a třetí playbook kontroluje existenci původně vytvořených struktur. Tyto tři playbooky používají kolekci `openstack.cloud`, která umožňuje jednoduše vytvořit požadované struktury a získat o nich informace. Přestože tato kolekce urychluje a zjednodušuje správu struktur, kolekce neumožňuje otestovat funkčnost struktur, pouze jejich existenci a dodatečné podrobnosti jako název, ID, stav apod. I když jsou implementovány automatizované testy, které ověřují existenci a některé aspekty funkčnosti OpenStack struktur, tyto testy nejsou plnohodnotnou zárukou správného a kompletního obnovení OpenStack cloudu. Nejedná se tedy o plnohodnotné testování funkčnosti a automatizované testování je nutné doplnit o manuální ověření funkčnosti těchto struktur.

Automatizované testování je vhodné spustit při provedení úprav v Ansible projektu, a tím je umožněno případné zásadní chyby odhalit ihned. V případě provedení zásadních změn v projektu je nutné toto automatizované testování doplnit o manuální ověření funkčnosti struktur. Následující tabulka 6.3 shrnuje testy, které byly provedeny pro ověření funkčnosti procesů zálohy a obnovy.

Z tabulky vyplývá, že uvedené testy proběhly úspěšně. OpenStack je ovšem velmi obsáhlý a i v případě zálohy pouze klíčových OpenStack služeb, zálohované soubory obsahují rozsáhlé množství dat a komplexní konfigurace, a proto je důležité provést obnovu a ověřit její správnost a funkčnost v reálném produkčním prostředí.

Tab. 6.3: Shrnutí provedených testů

OpenStack služba	**Akce**	**Manuální/ Automatizované**	**Výsledek**
Keystone	Existence projektů	A	✓
Keystone	Existence uživatelů	A	✓
Keystone	Kontrola uživatelských údajů	M	✓
Keystone	Autentizace uživatelů	M	✓
Glance	Existence totožných údajů o obrazech	M	✓
Neutron	Existence sítí a podsítí	A	✓
Neutron	Existence směrovačů	A	✓
Neutron	Funkčnost směrování	M	✓
Neutron	Existence bezpečnostních skupin	A	✓
Nova	Existence klíčů	A	✓
Nova	Funkčnost klíčů (autentizace s instancí)	M	✓
Nova	Existence flavors	A	✓
Nova	Existence instance	A	✓
Nova	Vstup do instance z `ctrl01`	M	✓
Neutron a Nova	Propojení instance s virtuální sítí	M	✓
Neutron a Nova	Propojení instance s fyzickou sítí	M	✓
Neutron a Nova	Komunikace instance v rámci vnitřní sítě	M	✓
Neutron a Nova	SSH do instance z `comp01`	M	✓

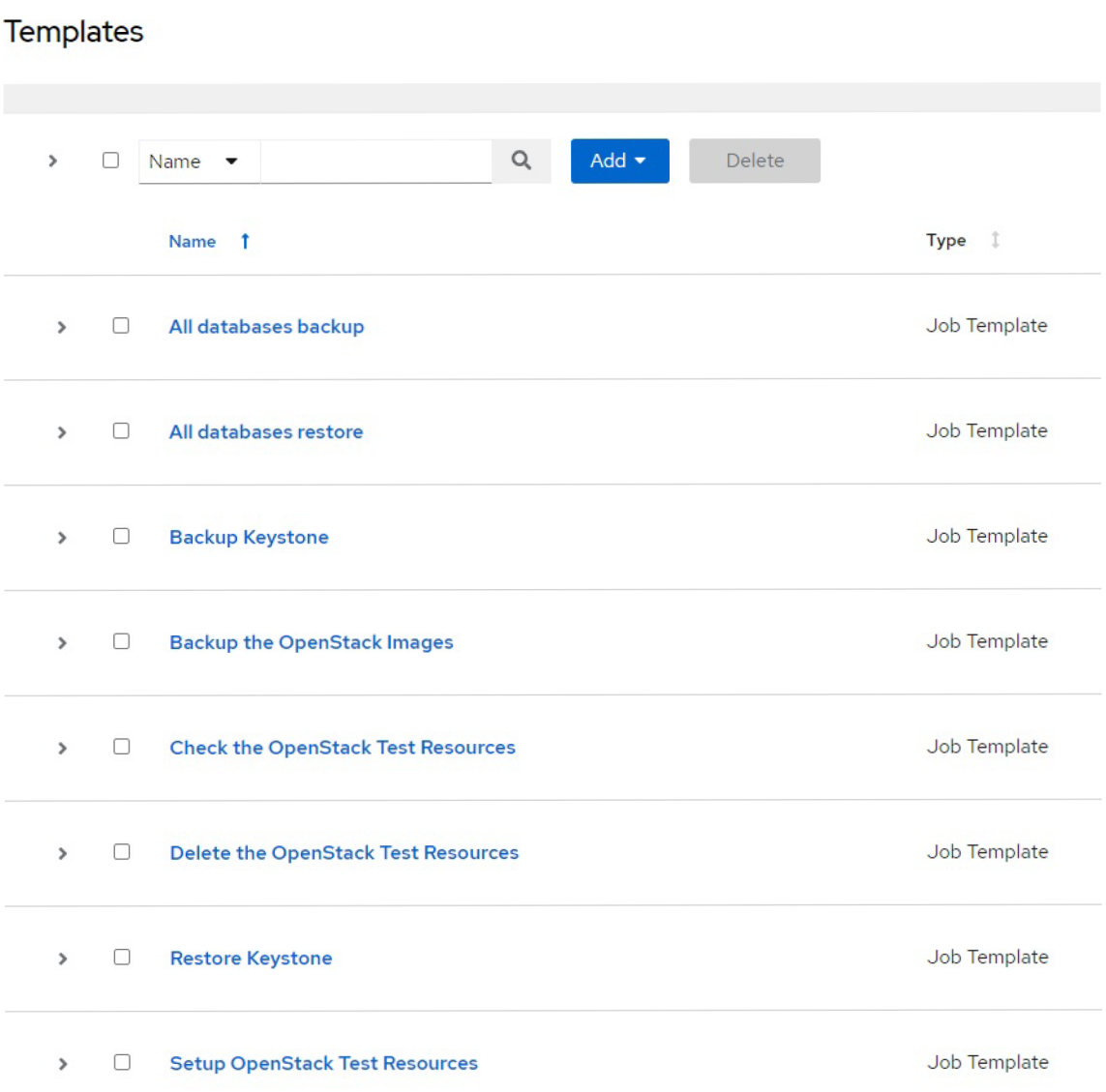
6.5 Ansible projekt v prostředí AWX

Pro spouštění implementovaného Ansible projektu v prostředí AWX je nutné patřičně vytvořit organizaci, přidělit přístup k Git repozitáři s projektem, vytvořit AWX inventář s dostupnými údaji ke stanicím `ctrl01` a `stor01` a vytvořit příslušné šablony pro spuštění jednotlivých procesů. Konkrétní postup pro splnění těchto bodů je uveden v příloze A. Po splnění těchto bodů je možné spustit potřebné procesy v prostředí AWX a sledovat průběh procesu v reálném čase. Konkrétně byly vytvořeny šablony pro:

- zálohování všech databází,
- zálohování databáze Keystone,
- obnovu všech databází,

- obnovu databáze Keystone,
- zálohování Glance obrazů,
- vytvoření OpenStack struktur,
- smazání vytvořených OpenStack struktur,
- ověření dostupnosti obnovených OpenStack struktur.

Tyto šablony se nacházejí na obrázku 6.7. Následně je možné vytvořit libovolné šablony s předdefinovanými `extra-vars` a ty jednoduše spouštět. Případně nastavit u jednotlivých šablon interval spouštění pro pravidelné zálohování.



Obr. 6.7: Ukázka vytvořených šablon pro zálohu, obnovu a testování funkčnosti v AWX

6.6 Kompatibilita projektu mezi odlišnými OpenStack cloudy

Vytvořený Ansible projekt byl implementovaný přímo pro OpenStack, který používá Aréna. Přenositelnosti projektu na jiné verze OpenStacku, případně na OpenStack, který byl nasazen odlišnou metodou než Kolla-Ansible, je možná, ale je nutné si uvědomit následující fakta:

- MariaDB je MySQL databáze a používá MySQL syntaxi,
- MariaDB obsahuje databáze majoritních služeb OpenStacku,
- Kolla-Ansible nasazuje služby OpenStacku v Docker kontejnerech.

Z toho důvodu, že MariaDB používá SQL syntaxi, je možné příkazy spojené se zálohou a obnovou použít i na jiné databáze založené na SQL jazyce. Jedná se zejména o příkazy `mysql dump` a `mysql database_name < dump_file_name`. Důležité je zkontrolovat privilegia uživatele provádějící zálohu a obnovu, jelikož se v jednotlivých SQL databázích mohou lišit. Zároveň je nutné vytvořit uživatele `ansible` v nové databázi MariaDB se stejným heslem, případně vygenerovat novou šifru.

Projekt zálohuje hlavní služby OpenStacku, které se nacházejí ve všech udržovaných verzích a zpravidla nejsou zásadně měněny. Nicméně databáze OpenStack služeb dostupné v MariaDB se mohou lišit v závislosti na konkrétním nastavení cloudu. Například nasazení OpenStacku s více instancemi služby Nova, které jsou nainstalovány na fyzicky oddělených uzlech[5], může vést ke změně počtu databází `nova_cell` v MariaDB. Tyto rozdíly nemají významný dopad na implementovaný Ansible projekt a zálohovatelné databáze je vždy možné upravit pomocí parametru `mariadb_databases` v konfiguračním souboru `globals.yaml`.

Specifické pro metodu nasazení Kolla-Ansible je instalace OpenStack služeb do Docker kontejnerů, včetně služby MariaDB. Implementovaný Ansible projekt je postavený na přístupu do Docker kontejneru s MariaDB. V případě použití jiného typu kontejneru, by bylo nutné patřičně upravit přístup k tomuto kontejneru. Toho by bylo docíleno použitím odlišné Ansible kolekce, které ale může používat odlišné moduly a tedy není možné určit výši kompatibility původní implementace. Příkladem může být nasazení TripleO, které instaluje služby do Podman kontejnerů. V rámci přenositelnosti projektu na nový OpenStack je způsob nasazení největší překážkou a pravděpodobně by bylo nutné implementaci od základu změnit. Množství potřebných úprav a výši kompatibility již naimplementovaných Ansible playbooků není možné předem určit.

V případě aktualizace OpenStacku na novou verzi a zachování metody nasazení Kolla-Ansible, poslední vydaná verze Antelope stále používá Docker kontejnery

[5] Z důvodu škálovatelnosti při větším zatížení cloudu.

pro OpenStack služby, včetně MariaDB [54]. Aktuální implementace Ansible projektu je kompatibilní s novými verzemi OpenStacku a je velmi nepravděpodobné, že by se zásadně musel upravit.

Závěr

V diplomové práci byl vysvětlen pojem cyber range a byly představeny možnosti přístupu ke stavbě cyber rangů. Tři nejdůležitější technologie, které jsou esenciální k vybudování realistického prostředí cyber range jsou virtualizace, kontejnerizace a cloud computing. Tyto technologie byly rozebrány v první kapitole. V následující kapitole byl představen cyber range Kybernetická aréna, který byl vybudován na Ústavu telekomunikací. Kybernetická aréna je postavena s použitím cloud computingu, který se čím dál tím více stává stěžejní technologií pro vznik kvalitních cyber rangů. Kybernetická aréna využívá cloudovou platformu OpenStack, která byla představena v kapitole 3. OpenStack je stavebním kamenem Kybernetické arény a pomocí služeb, které nabízí, zajišťuje Aréně veškerou její funkčnost. Pro automatizaci procesů, které slouží zejména pro vytváření herních scénářů, slouží Ansible AWX, který funguje jako prostředník mezi Arénou a OpenStackem.

Pro zkvalitnění životního cyklu Arény byl navržen a implementován Ansible projekt, který vytváří zálohy důležitých konfiguračních a uživatelských dat z hlavních OpenStack služeb Cinder, Glance, Heat, Keystone, Neutron, Nova a Placement. V rámci této implementace bylo vytvořeno vývojové prostředí s OpenStack cloudem a AWX simulující prostředí, ve kterém se nachází Aréna. Při vytváření tohoto prostředí byla provedena analýza možností nasazení AWX, které v budoucnu nahradí aktuální verzi AWX nacházející se v Kybernetické aréně. V rámci instalace nové verze AWX byla vytvořena příručka dostupná v příloze A, která popisuje jednotlivé kroky potřebné k jejímu nasazení do produkčního prostředí a ovládání. Jelikož nejnovější verze AWX podporují instalaci pouze do Kubernetes clusteru, příručka zahrnuje i instalaci jeho odlehčené verze.

V kapitole 5 byla provedena analýza hlavních přístupů vedoucích k implementaci procesů zálohy a obnovy. Byly rozebrány možnosti použití OpenStack CLI, Ansible `openstack.cloud` kolekce a dalších, a použití MariaDB pro získání důležitých dat pro vytvoření plnohodnotných záloh. Z této analýzy vyplynulo, že pro splnění podmínek pro získání obnovitelných záloh je nejlepší volbou potřebná data získávat z databáze MariaDB. Na tomto základě byl vytvořen diagram procesů, který znázorňuje postup vytváření záloh a jejich následnou obnovu. Pro automatizování procesů zálohy a obnovy byly jednotlivé kroky implementovány v jazyce Ansible s použitím vhodně vybraných kolekcí. Celý Ansible projekt byl implementován tak, aby bylo možné vytvářet šablony v prostředí AWX a následně je pouze spouštět.

Praktická část diplomové práce rozebírá tvorbu vývojového prostředí, tedy nasazení platformy OpenStack, prostředí AWX a jejich vzájemnou konfiguraci a způsob komunikace mezi nimi. Vytvořené prostředí simuluje zjednodušenou achritekturu Kybernetické arény. Druhá část praktické části detailně rozebírá podstatné kroky

a vlastní způsob implementace Ansible projektu pro zálohu a obnovu. Celá implementace se skládá ze tří hlavních částí – implementace procesu zálohy MariaDB, implementace procesu obnovy ze zálohovaných souborů a implementace procesu zálohy Glance obrazů Kybernetické arény. Všechny procesy byly řádně otestovány, zejména funkčnost jednotlivých služeb po provedení kompletního procesu obnovy. Během testování nebyla zjištěna žádná nestabilita nebo nefunkčnost OpenStack cloudu. Důležité je ovšem zmínit, že cloud je velmi robustní a plné otestování funkčnosti by bylo možné až v rámci produkčního běhu cloudu. Jelikož je předpokládáno, že implementace může být upravována v závislosti na měnícím se prostředí Arény, byly vytvořeny automatizované funkční testy, které vytvářejí OpenStack struktury a následně ověřují jejich existenci po procesu obnovy. Tyto testy jsou součástí Ansible projektu a značně usnadňují odhalení případných chyb vzniklých při budoucí úpravě implementace.

Funkčnost implementovaného Ansible projektu byla ověřena za použití kombinace automatizovaných a manuálních testů s důrazem na použité hlavní služby OpenStacku. V tabulce 6.3 je toto testování shrnuto a v rámci manuálního a automatizovaného testování nebyly nalezeny indície, které by nasvědčovaly chybě, či nefunkčnosti v implementaci Ansible projektu. Ovšem není možné z tohoto testování vyhodnotit absolutní funkčnost procesů zálohy a obnovy, a to z toho důvodu, že OpenStack je velmi obsáhlý, i v případě pouze klíčových komponent. Zálohované soubory obsahují rozsáhlé množství dat a komplexní konfigurace, a proto je důležité provést obnovu a ověřit její správnost v reálném produkčním prostředí. A až po kontinuálním běhu OpenStack cloudu, který byl částečně obnoven ze získaných záloh je možné získat přesný závěr o funkčnosti všech implementovaných procesů.

Na závěr praktické části byla rozebrána kompatibilita projektu v souvislosti s jinými metodami nasazení a nových verzí OpenStack cloudu. V případě použití metody nasazení Kolla-Ansible a použití novější verze než stávající Victoria, není předpokládáno, že aktuální Ansible projekt by byl nekompatibilní. Kolla-Ansible zachovává způsob instalace a uchování dat k jednotlivým OpenStack službám, kterých se procesy zálohy a obnovy týkají. V případě použití odlišné metody nasazení cloudu je naopak nepravděpodobné, že by aktuální implementace byla spustitelná a funkční v novém nasazení. Každá metoda nasazení používá odlišný způsob instalace jednotlivých služeb, což je v rámci implementace zásadní. Dopředu není možné určit, zda by bylo možné Ansible projekt více či méně upravit, aby byl kompatibilním s novým OpenStackem.

Výsledky diplomové práce byly úspěšně prezentovány na studentské konferenci EEICT 2023 s článkem *Modernizing Cyber Arena's AutomationComponent: Designing Backups and Recovery Process on OpenStack.*

Literatura

[1] Národní úřad pro kybernetickou a informační bezpečnost. *ZPRÁVA O STAVU KYBERNETICKÉ BEZPEČNOSTI ČESKÉ REPUBLIKY ZA ROK 2020* [online]. 22. 10. 2022 [cit. 28. 11. 2022]. Dostupné z: <https://www.nukib.cz/download/publikace/zpravy_o_stavu/Zprava_o_stavu_KB_2020.pdf>.

[2] ANTONIUK Daryna. *Q&A: Kenneth Geers on the cyber war between Ukraine and Russia* [online]. 22. 10. 2022 [cit. 28. 11. 2022]. Dostupné z: <https://therecord.media/qa-kenneth-geers-on-the-cyber-war-between-ukraine-and-russia/>.

[3] The NATO Cooperative Cyber Defence Centre of Excellence. *Locked Shields* [online]. 2022 [cit. 28. 11. 2022]. Dostupné z: <https://ccdcoe.org/exercises/locked-shields/>.

[4] CSIRT - MU. *Cyber Czech Security Exercise* [online]. ©2022 [cit. 12. 10. 2022]. Dostupné z: <https://csirt.muni.cz/projects/cyber-czech>.

[5] National Initiative for Cybersecurity Education. *The Cyber Range: A Guide* [online]. 2020 [cit. 5. 10. 2022]. Dostupné z: <https://www.nist.gov/system/files/documents/2020/06/25/The%20Cyber%20Range%20-%20A%20Guide%20%28NIST-NICE%29%20%28Draft%29%20-%20062420_1315.pdf>.

[6] UKWANDU Elochukwu, AMINE Mohamed FARAH BEN, HINDY Hanan, BROSSET David, KAVALLIEROS Dimitris, ATKINSON Robert, TACHTATZIS Christos, BURES Miroslav, ANDONOVIC Ivan, BELLEKENS Xavier. *A Review of Cyber-Ranges and Test-Beds: Current and Future Trends* [online]. 2020 [cit. 3. 10. 2022]. Dostupné z: <https://arxiv.org/abs/2010.06850>.

[7] European Cyber Security Organization. *Understanding Cyber Ranges: From Hype to Reality* [online]. 2020 [cit. 3. 10. 2022]. Dostupné z: <https://www.cyberranges.com/whitepaper-download-understanding-cyber-ranges-from-hype-to-reality/>.

[8] VYKOPAL Jan, VIZVARY Martin, OSLEJSEK Radek, CELEDA Pavel, TOVARNAK Daniel. *Lessons Learned From Complex Hands-on Defence Exercises in a Cyber Range* [online]. ©2017 [cit. 6. 10. 2022]. Dostupné z:

<https://is.muni.cz/publication/1391675/2017-FIE-lessons-learned-exercises-cyber-range-paper.pdf>.

[9] Red Hat, Inc. *Understanding virtualization* [online]. ©2018 [cit. 29. 9. 2022]. Dostupné z: <https://www.redhat.com/en/topics/virtualizations>.

[10] WMware, Inc. *VMware ESXi* [online]. ©2022 [cit. 30. 9. 2022]. Dostupné z: <https://www.vmware.com/cz/products/esxi-and-esx.html>.

[11] Microsoft. *Introduction to Hyper-V on Windows 10* [online]. ©2022 [cit. 30. 9. 2022]. Dostupné z: <https://learn.microsoft.com/en-us/virtualization/hyper-v-on-windows/about/>.

[12] RedHat, Inc. *What is KVM?* [online]. ©2022 [cit. 30. 9. 2022]. Dostupné z: <https://www.redhat.com/en/topics/virtualization/what-is-KVM>.

[13] WMware, Inc. *VMware Workstation Player* [online]. ©2022 [cit. 30. 9. 2022]. Dostupné z: <https://www.vmware.com/cz/products/workstation-player.html>.

[14] Oracle. *VirtualBox* [online]. ©2022 [cit. 30. 9. 2022]. Dostupné z: <https://www.virtualbox.org/>.

[15] RAGHUNATHAN Anand, JHA N.K. *Secure Virtual Machine Execution under an Untrusted Management OS* [online]. 2010 [cit. 29. 9. 2022]. Dostupné z: <https://www.researchgate.net/publication/224170016_Secure_Virtual_Machine_Execution_under_an_Untrusted_Management_OS>.

[16] Microsoft. *What is a container?* [online]. ©2022 [cit. 30. 9. 2022]. Dostupné z: <https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-a-container/#overview>.

[17] Red Hat, Inc. *What is containerization?* [online]. ©2021 [cit. 30. 9. 2022]. Dostupné z: <https://www.redhat.com/en/topics/cloud-native-apps/what-is-containerization>.

[18] Microsoft. *Windows Sandbox* [online]. ©2022 [cit. 30. 9. 2022]. Dostupné z: <https://learn.microsoft.com/en-us/windows/security/threat-protection/windows-sandbox/windows-sandbox-overview>.

[19] Docker, Inc. *Use containers to Build, Share and Run your applications* [online]. ©2022 [cit. 1. 10. 2022]. Dostupné z: <https://www.docker.com/resources/what-container/>.

[20] Podman. *What is Podman?* [online]. ©2019 [cit. 1. 10. 2022]. Dostupné z: <https://docs.podman.io/en/latest/index.html>.

[21] LXD contributors. *LXD Documentation* [online]. ©2022 [cit. 1. 10. 2022]. Dostupné z: <https://linuxcontainers.org/lxd/docs/master/>.

[22] The Kubernetes Authors. *Kubernetes Documentation* [online]. ©2022 [cit. 1. 10. 2022]. Dostupné z: <https://kubernetes.io/docs/concepts/overview/>.

[23] GURSIMRAN Singh. *Orchestration vs Automation - Understanding the Difference* [online]. 2020 [cit. 2. 10. 2022]. Dostupné z: <https://www.xenonstack.com/insights/orchestration-vs-automationw/>.

[24] Red Hat, Inc. *Understanding cloud computing* [online]. ©2022 [cit. 5. 10. 2022]. Dostupné z: <https://www.redhat.com/en/topics/cloud>.

[25] Amazon Web Services, Inc. *Types of Cloud Computing* [online]. ©2022 [cit. 5. 10. 2022]. Dostupné z: <https://aws.amazon.com/types-of-cloud-computing/>.

[26] MUSSE Hodan M., ALAMRO Lama A. *Cloud Computing: Architecture and Operating System* [online]. 2016 [cit. 7. 10. 2022]. Dostupné z: <https://ieeexplore.ieee.org/document/7976640>.

[27] Amazon Web Services, Inc. *About AWS* [online]. ©2022 [cit. 5. 10. 2022]. Dostupné z: <https://aws.amazon.com/about-aws/>.

[28] Microsoft. *What is Azure?* [online]. ©2022 [cit. 5. 10. 2022]. Dostupné z: <https://azure.microsoft.com/en-us/>.

[29] Google Cloud. *Google Cloud product documentation* [online]. ©2022 [cit. 5. 10. 2022]. Dostupné z: <https://cloud.google.com/docs>.

[30] OpenStack. *OpenStack, Software* [online]. 2022 [cit. 5. 10. 2022]. Dostupné z: <https://www.openstack.org/software/>.

[31] OpenNebula. *OpenNebula Overview* [online]. 2022 [cit. 5. 10. 2022]. Dostupné z: <https://docs.opennebula.io/6.4/overview/opennebula_concepts/opennebula_overview.html>.

[32] Red Hat, Inc. *Red Hat OpenShift* [online]. ©2022 [cit. 5. 10. 2022]. Dostupné z: <https://www.redhat.com/en/technologies/cloud-computing/openshift>.

[33] Technologická agentura ČR. *Kybernetická aréna pro výzkum, testování a edukaci v oblasti kyberbezpečnosti* [online]. 2019 [cit. 8. 10. 2022]. Dostupné z: <https://starfos.tacr.cz/cs/project/VI20192022132#project-main>.

[34] STODŮLKA Tomáš, FUJDIAK Radek. *Budování Cyber Range platformy s technologií cloud computingu* Brno: Vysoké učení technické v Brně, 2022. 153 s. ISBN: 978-80-214-6064-5.

[35] ENISA. *ICS SCADA* [online]. 2022 [cit. 8. 10. 2022]. Dostupné z: <https://www.enisa.europa.eu/topics/critical-information-infrastructures-and-services/scada>.

[36] STODŮLKA Tomáš, MARTINÁSEK Zdeněk, FUJDIAK Radek. *Cloudové platformy a jejich srovnání: Kubernetes, OpenStack a OpenShift (Cloud platforms and their comparison: Kubernetes, OpenStack and OpenShift)* [online]. 2022 [cit. 9. 12. 2022]. Dostupné z: <http://www.elektrorevue.cz/cz/clanky/komunikacni-technologie/0/cloudove-platformy-a-jejich-srovnani--kubernetes--openstack-a-openshift--cloud-platforms-and-their-comparison--kubernetes--openstack-and-openshift-/>.

[37] THIRY Daniel. *Kubernetes Sandboxes – Easy Development in a Realistic Environment* [online]. 2020 [cit. 8. 10. 2022]. Dostupné z: <https://loft.sh/blog/kubernetes-sandboxes-easy-development-in-a-realistic-environment/>.

[38] Katacontainers. *Kata Containers, The speed of containers, the security of VMs* [online]. 2017 [cit. 9. 10. 2022]. Dostupné z: <https://katacontainers.io/collateral/kata-containers-1pager.pdf>.

[39] DAVIS Jon, MAGRATH Shane. *A Survey of Cyber Ranges and Testbeds* [online]. 2013 [cit. 12. 10. 2022]. Dostupné z: <https://apps.dtic.mil/dtic/tr/fulltext/u2/a594524.pdf>.

[40] CHOULIARAS Nestoras. KITTES George, KANTZAVELOU Ioanna, MAGLARAS Leandros, PANTZIOU Grammati, FERRAG Mohamed Amine. *Cyber Ranges and TestBeds for Education, Training, and Research* [online]. 2021 [cit. 12. 10. 2022]. Dostupné z: <https://www.mdpi.com/2076-3417/11/4/1809/htm>.

[41] YAMIN Muhammad Mudassar, KATT Basel, GKIOULOS Vasileios. *Cyber ranges and security testbeds: Scenarios, functions, tools and architecture* [online]. 2020 [cit. 12. 10. 2022]. Dostupné z: <https://www.sciencedirect.com/science/article/pii/S0167404819301804#!>.

[42] ČELEDA Pavel, ČEGAN Jakub, VYKOPAL Jan, TOVARŇÁK Daniel. *KYPO – A Platform for Cyber Defence Exercises* [online]. 2018 [cit. 12. 10. 2022]. Dostupné z: <https://is.muni.cz/publication/1319597/en/KYPO-A-Platform-for-Cyber-Defence-Exercises/Celeda-Cegan-Vykopal-Tovarnak>.

[43] OpenStack. *Release Series* [online]. 2022 [cit. 14. 10. 2022]. Dostupné z: <https://releases.openstack.org/>.

[44] OpenInfra Foundation. *Organizations supporting the OpenInfra Foundation* [online]. 2022 [cit. 14. 10. 2022]. Dostupné z: <https://openinfra.dev/members/>.

[45] Stackalytics. *OpenStack Foundation, reviews by company* [online]. [cit. 14. 10. 2022]. Dostupné z: <https://www.stackalytics.io/?release=zed>.

[46] OpenStack. *OpenStack Components* [online]. 2022 [cit. 16. 10. 2022]. Dostupné z: <https://www.openstack.org/software/project-navigator/openstack-components/#openstack-services>.

[47] OpenStack. *Install OpenStack services* [online]. 2022 [cit. 16. 10. 2022]. Dostupné z: <https://docs.openstack.org/install-guide/openstack-services.html>.

[48] Red Hat, Inc. *Red Hat OpenStack Platform - Components* [online]. 2022 [cit. 16. 10. 2022]. Dostupné z: <https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/9/html/architecture_guide/components>.

[49] Red Hat, Inc. *All-in-one OpenStack installation* [online]. 2022 [cit. 16. 10. 2022]. Dostupné z: <https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/14/html/quick_start_guide/all-in-one-openstack-installation>.

[50] FIŠAROVÁ Veronika. *Bezpečnostní cvičení na platformě OpenStack* [online]. Brno: Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií. Vedoucí práce Ing. Tomáš Lieskovan. 2021 [cit. 18. 10. 2022]. Dostupné z: <https://dspace.vutbr.cz/handle/11012/198046>.

[51] OpenStack. *DevStack* [online]. ©2022 [cit. 18. 10. 2022]. Dostupné z: <https://docs.openstack.org/devstack/latest/>.

[52] RDO Project. *Packstack: Create a proof of concept cloud* [online]. ©2022 [cit. 18. 10. 2022]. Dostupné z: <https://www.rdoproject.org/install/packstack/>.

[53] OpenStack. *OpenStack-Ansible Deployment Guide* [online]. 2022 [cit. 18. 10. 2022]. Dostupné z: <https://docs.openstack.org/project-deploy-guide/openstack-ansible/yoga/>.

[54] OpenStack. *Kolla Ansible's documentation* [online]. 2022 [cit. 18. 10. 2022]. Dostupné z: <https://docs.openstack.org/kolla-ansible/latest/?_ga=2.242712642.1904085995.1665571331-2078193485.1657181866>.

[55] OpenStack. *TripleO documentation* [online]. 2022 [cit. 18. 10. 2022]. Dostupné z: <https://docs.openstack.org/tripleo-docs/latest/>.

[56] OpenStack. *Deployment tools* [online]. 2022 [cit. 18. 10. 2022]. Dostupné z: <https://www.openstack.org/software/project-navigator/deployment-tools>.

[57] Red Hat, Inc. *Automation controller* [online]. ©2022 [cit. 10. 12. 2022]. Dostupné z:
<https://www.ansible.com/products/controller?extIdCarryOver=true&sc_cid=701f2000001OH7YAAW>.

[58] Red Hat, Inc. *Ansible Documentation* [online]. ©2022 [cit. 20. 10. 2022]. Dostupné z:
<https://docs.ansible.com/ansible/latest/index.html>.

[59] Opensource. *What is Ansible?* [online]. 2022 [cit. 20. 10. 2022]. Dostupné z:
<https://opensource.com/resources/what-ansible>.

[60] Chiradeep BASUMALLICK. *What Is Ansible? Uses, Working, Architecture, Features* [online]. 15. 7. 2022 [cit. 20. 10. 2022]. Dostupné z:
<https://www.spiceworks.com/tech/devops/articles/what-is-ansible/>.

[61] Ansible. *AWX Project* [online]. 2022 [cit. 26. 10. 2022]. Dostupné z:
<https://github.com/ansible/awx/tree/devel>.

[62] Red Hat Ansible. *Automation controller* [online]. ©2022 [cit. 26. 10. 2022]. Dostupné z:
<https://www.ansible.com/products/controller>.

[63] The Kubernetes Authors. *Minikube* [online]. ©2022 [cit. 26. 10. 2022]. Dostupné z:
<https://minikube.sigs.k8s.io/docs/start/>.

[64] Ansible. *Ansible AWX operator* [online]. 2022 [cit. 26. 10. 2022]. Dostupné z:
<https://github.com/ansible/awx-operator>.

[65] K3s Project Authors. *K3s - Lightweight Kubernetes* [online]. ©2022 [cit. 30. 10. 2022]. Dostupné z:
<https://docs.k3s.io/>.

[66] Ansible. *Ansible documentation* [online]. ©2023 [cit. 4. 4. 2023]. Dostupné z:
<https://docs.ansible.com/ansible/latest/index.html>.

[67] Kubernetes. *Kubernetes Partners* [online]. ©2023 [cit. 12. 2. 2023]. Dostupné z:
<https://kubernetes.io/partners/#conformance>.

[68] OpenStack. *Logical architecture* [online]. ©2022 [cit. 19. 11. 2022]. Dostupné z:
<https://docs.openstack.org/install-guide/get-started-logical-architecture.html>.

[69] Traefik. *Traefik documentation* [online]. ©2022 [cit. 31. 10. 2022]. Dostupné z: <https://github.com/kurokobo/awx-on-k3s/tree/0.29.0>.

[70] Red Hat, Inc. *The Tower User Interface* [online]. ©2022 [cit. 31. 10. 2022]. Dostupné z: <https://docs.ansible.com/ansible-tower/latest/html/userguide/main_menu.htmls>.

[71] Red Hat, Inc. *ReaR: Backup and recover your Linux server with confidence* [online]. ©2022 [cit. 19. 11. 2022]. Dostupné z: <https://www.redhat.com/sysadmin/rear-backup-and-recover>.

[72] OpenStack. *OpenStack Freezer's documentation* [online]. ©2023 [cit. 19. 2. 2023]. Dostupné z: <https://docs.openstack.org/freezer/latest/>.

[73] OpenStack. *OpenStack Neutron's documentation* [online]. ©2023 [cit. 19. 2. 2023]. Dostupné z: <https://docs.openstack.org/neutron/victoria/index.html>.

[74] MariaDB. *MariaDB Documentation* [online]. ©2023 [cit. 19. 2. 2023]. Dostupné z: <https://mariadb.com/docs/>.

[75] OpenStack. *Kolla Ansible's Victoria documentation* [online]. 2022 [cit. 19. 11. 2022]. Dostupné z: <https://docs.openstack.org/kolla-ansible/victoria/?_ga=2.242712642.1904085995.1665571331-2078193485.1657181866>.

[76] Ansible. *Ansible Openstack.Cloud Documentation* [online]. 2022 [cit. 26. 11. 2022]. Dostupné z: <https://docs.ansible.com/ansible/latest/collections/openstack/cloud/index.html>.

[77] Ansible. *Ansible Community.Docker Documentation* [online]. 2022 [cit. 7. 4. 2023]. Dostupné z: <https://docs.ansible.com/ansible/latest/collections/community/docker/index.html>.

[78] Deydra71 (Veronika Fisarova). *cyberarena-backup-and-restore* [online]. GitHub. 2023 [cit. 4. 5. 2023]. Dostupné z: <https://github.com/Deydra71/cyberarena-backup-and-restore>.

Seznam symbolů a zkratek

AWX	Ansible Web eXecutable
AWS	Amazon Web Services
CLI	Command Line Interface
CPU	Central Processing Unit
CR	Cyber Range
CTF	Capture the Flag
DNF	Dandified Yum
GUI	Graphical User Interface
HTTPS	Hypertext Transfer Protocol Secure
IP	Internet Protocol
KVM	Kernel-Based Virtual Machine
KYPO	Kybernetický polygon
K8S	Kubernetes
LBaaS	Load Balancing as a Service
NASA	National Aeronautics and Space Administration
NICE	National Initiative for Cybersecurity Education
NÚKIB	Národní úřad pro kybernetickou a informační bezpečnost
OS	Operační systém
RAM	Random Access Memory
REST	Representational State Transfer
RHEL	Red Hat Enterprise Linux
RSA	Rivest Shamir Adleman
SSH	Secure Shell
TLS	Transport Layer Security

URL	Uniform Resource Locator
vCPU	Virtuální CPU
VPN	Virtual Private Network
VLAN	Virtual Local Area Network
VS	Virtuální stroj
YAML	YAML Ain't Markup Language

A Příručka instalace a ovládání Ansible AWX

Cílem první části příručky je provést uživatele veškerými kroky, které vedou k úspěšné instalaci AWX do Kubernetes pomocí nástroje `awx-operator`. Druhá část se věnuje ovládání AWX.

Příručka používá následující značení:

Blok s CLI příkazy

Označení bloku s obsahem souboru, přičemž provedené změny nebo důležité výrazy jsou zvýrazněny červeně

Označení bloku s výpisem z konzole

Označení bloku s poznámkou

A.1 Terminologie a základní ovládání Kubernetes

Běh AWX je spravován kontejnerovým managerem Kubernetes, pro lepší orientaci při instalaci AWX je v následujícím textu vysvětlena základní terminologie Kubernetes:

- **Cluster** – sada uzlů, na kterých běží kontejnerizované aplikace, skládá se z řídicí roviny a pracovních uzlů (tzv. *workers)*.
- **Řídící rovina** (anglicky *control plane*) – část clusteru, která spravuje celkový stav daného clusteru, součástí řídící roviny je Kubernetes API server, plánovač (*kube-scheduler)* a etcd[1].
- **Uzel** (anglicky *node*) – fyzický nebo virtuální počítač, který je součástí clusteru. Může to být fyzický stroj, virtuální stroj nebo cloudová instance.
- **Pod** – nejmenší spravovaná jednotka, pod obsahuje jeden nebo více kontejnerů a představuje jednu instanci běžícího procesu v clusteru.
- **Kontejner** – samostatný a spustitelný softwarový balík, který obsahuje vše potřebné ke spuštění aplikace, včetně kódu, runtime, systémových nástrojů, knihoven a nastavení.
- **Služba** – abstrakce, která definuje logickou sadu podů a politiku, pomocí které se k nim přistupuje.

[1]Etcd je distribuované, konzistentní úložiště typu klíč–hodnota pro sdílenou konfiguraci, zajišťování služeb a koordinaci plánovače.

- **Nasazení** – Spravuje aktualizaci množin podů a jejich škálovatelnost v rámci předkonfigurovaných nastavení.
- **ReplicaSet** – zdroj, který zajišťuje, že v daný moment běží požadovaný počet replik podů. Pokud nějaký pod selže, ReplicaSet zajistí vytvoření nového podu.
- **Svazek** – adresář, ke kterému přistupují kontejnery v podu, poskytuje sdílení dat mezi kontejnery a umožňuje kontejnerům ukládat data, která přetrvávají i po odstranění kontejnerů [22].
- **Jmenný prostor** (anglicky *namespace*) – virtuální cluster pro rozdělení prostředků Kubernetes clusteru do logicky souvisejících skupin.

Kubernetes podporuje správu různých běhových prostředí kontejnerů, jako například Docker, QEMU, Hyper-V, Podman a další. Při interakci s clusterem pomocí CLI je komunikace zprostředkována Kubernetes API serverem, který poskytuje konzistentní rozhraní pro správu clusteru bez ohledu na základní běhové prostředí kontejneru. Kubernetes CLI je tedy neměnné, bez ohledu na použití různých běhových prostředí. Následující tabulka A.1 obsahuje užitečné CLI příkazy pro základní správu Kubernetes clusteru.

Tab. A.1: Základní příkazy pro správu Kubernetes clusteru

CLI příkaz	**Popis**
kubectl get	Výpis nejdůležitějších informací o požadovaném zdroji, např. uzel, pod, namespace apod.
kubectl create	Vytvoření prostředků clusteru – uzel, pod, jmeného prostoru a další.
kubectl run	Spuštění specifikovaného obrazu v clusteru.
kubectl apply	Aplikace změn v nastavení clusteru specifikovaných v YAML souboru.
kubectl delete	Smazání prostředků clusteru – uzel, pod, jmeného prostoru a další.
kubectl exec	Spuštění příkazu v kontejneru v rámci specifikovaného podu.
kubectl logs	Zobrazení logu specifikovaného podu.
kubectl describe	Zobrazení informací o zadaném prostředku
kubectl –help	Vypsání všech příkazů ke správě clusteru

A.2 Instalace Ansible AWX

Tato příručka následuje doporučený postup instalace dostupný v literaturách [64] a [65]. Instalace AWX je rozdělena do těchto bodů, které jsou popsány v následujících částech příručky:

- příprava stanice,
- instalace kontejnerového manageru K3S,
- instalace AWX pomocí `awx-operator`.

Instalace K3S

Nejdříve je nutné nainstalovat potřebné systémové balíčky:

```
$ sudo dnf update -y
$ sudo dnf install -y git make curl
```

K3S je instalován pomocí skriptu dostupného na stránkách projektu K3S[2]. Následující příkaz stáhne instalační skript pro K3S a spustí jej pomocí roury:

```
$ curl -sfL https://get.k3s.io | INSTALL_K3S_EXEC="server \
--node-allocatable='cpu=4000m,memory=6Gi'" sh -
```

Takto vytvořenému clusteru jsou alokovány výpočetní prostředky 4 vCPU a 6 GB RAM, což jsou prostředky doporučené pro instalaci AWX pomocí `awx-operator`. Po dokončení skriptu jsou ve složce `/etc/rancher/k3s/` vytvořeny konfigurační soubory K3S. K těmto souborům mají implicitně přístup pouze uživatelé s právy sudo.

Tímto je K3S cluster vytvořen a je možné k němu přistupovat pomocí Kubernetes CLI. Základní informace o clusteru jsou vypsány příkazy:

```
$ kubectl cluster-info
$ kubectl config view
```

Instalace AWX

Prvním krokem k instalaci AWX je nutné nainstalovat balíček Kustomize, který umožňuje nastavovat Kubernetes cluster pomocí specifikací napsaných v YAML souborech:

```
$ curl -s "https://raw.githubusercontent.com/kubernetes-sigs/
   kustomize/master/hack/install_kustomize.sh"
$ sh ./kustomize build .
```

V pracovní složce je vytvořen soubor `kustomization.yaml` s následujícím obsahem:

[2]Skript je možné zobrazit zde: `https://get.k3s.io/`.

```
---
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization
resources:
  - github.com/ansible/awx-operator/config/default?ref=\1.1.4

images:
  - name: quay.io/ansible/awx-operator
    newTag: 1.1.4

namespace: awx
```

V tomto souboru je pod klíčem `resources` uveden zdroj, podle kterého se cluster nakonfiguruje před samotnou instalací AWX. Důležité je použít odkaz na požadovanou verzi `awx-operator`. Klíč `images` obsahuje odkaz na obraz kontejneru, který bude použit pro instalaci AWX, verze uvedená pod položkou `newTag` musí odpovídat verzi na GitHubu. Klíč `namespace` obsahuje název vlastního clusteru, tento název je libovolný, zde je zvolen jmenný prostor `awx`. V případě, že uživatel chce nainstalovat více AWX instancí, musí být odlišeny jmenným prostorem. Nyní je nutné jmenný prostor vytvořit, to je provedeno pomocí příkazu:

```
$ kubectl create namespace awx
```

Po vytvoření jmenného prostoru je možné nasadit `awx-operator` do K3S clusteru:

```
$ kustomize build . | kubectl apply -f -
```

Výsledkem je připravený pod `awx-operator`, jeho úspěšný start lze ověřit ve výpisu:

```
$ kubectl get pods -n awx
NAME                                  READY   STATUS    RESTARTS   AGE
awx-operator-controller-manager       2/2     Running   0          56s
```

Dalším krokem v instalaci AWX je vytvoření *self-signed* certifikátu, který umožní HTTPS spojení do webového prostředí AWS:

```
$ AWX_HOST="awx.arena.com"
$ openssl req -x509 -nodes -days 3650 -newkey rsa:2048 -out tls
    .crt -keyout .key -subj "/CN=${AWX_HOST}/O=${AWX_HOST}" -
    addext "subjectAltName = DNS:\${AWX_HOST}"
```

Následně jsou vytvořeny a editovány soubory, které specifikují již samotné AWX prostředí, k tomu je využit opět nástroj Kustomize. V pracovním adresáři je vytvořen nový soubor `awx.yaml` s následujícím obsahem:

```
---
apiVersion: awx.ansible.com/v1beta1
kind: AWX
metadata:
  name: awx
spec:
  admin_user: admin
  ingress_type: ingress
  ingress_tls_secret: awx-secret-tls
  hostname: awx.arena.com

  postgres_configuration_secret: awx-postgres-configuration
  postgres_storage_class: awx-postgres-volume
  postgres_storage_requirements:
    requests:
      storage: 8Gi

  projects_persistence: true
  no_log: false
```

Následující tabulka A.2 obsahuje popis použitých specifikací v souboru `awx.yaml`. Důležité je zmínit, že některé hodnoty jsou získávány až ze specifikací v souboru `pv.yaml`, který je uveden dále v příručce.

Tab. A.2: Použité specifikace v souboru `awx.yaml`

Proměnná	**Hodnota**	**Popis**
admin_user	admin	Jmnéno uživatele s administrátorskými právy.
ingress_type	ingress	Vystavení služby AWX externím sítím. Zvolený způsob směrování v Kubernetes.
ingress_tls_secret	awx-secret-tls	Název objektu, který obsahuje privátní klíč a certifikát pro ustanovení TLS spojení.
hostname	awx.arena.com	Zvolené DNS jméno.
postgres_configuration_secret	awx-postgres-configuration	Název objektu, který obsahuje konfiguraci databáze PostrgeSQL.
postgres_storage_class	awx-postgres-volume	Úložiště pro ukládání dat v PostgreSQL.
postgres_storage_requirements	8Gi	Potřebná velikost úložiště.
projects_persistence	true	Úložiště s Ansible projekty je zachováno i po odinstalaci AWX.
no_log	false	Citlivé informace v Ansible výpisu nejsou cenzurovány.

Následně je vytvořen soubor `pv.yaml`, který obsahuje vlastní specifikace pro trvalé úložiště PosteSQL databáze `awx-postgres-13-volume` a úložiště Ansible projektů `awx-projects-volume` obsah souboru je následující:

```
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: awx-postgres-13-volume
spec:
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  capacity:
    storage: 8Gi
  storageClassName: awx-postgres-volume
  hostPath:
    path: /data/postgres-13

---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: awx-projects-volume
spec:
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  capacity:
    storage: 2Gi
  storageClassName: awx-projects-volume
  hostPath:
    path: /data/projects
```

Následující tabulka A.3 obsahuje popis použitých specifikací v souboru `pv.yaml`, duplicitní specifikace nejsou v druhé části tabulky zahrnuty.

Tab. A.3: Použité specifikace v souboru `pv.yaml`

Proměnná	**Hodnota**	**Popis**
PostgreSQL databáze awx-postgres-13-volume		
accessModes	ReadWriteOnce	Svazek je připojen v režimu čtení a zápisu.
persistentVolumeReclaimPolicy	Retain	Svazek je nadále uchován, i když již není aktivně používán.
capacity	8Gi	Dedikovaná velikost úložiště.
storageClassName	awx-postgres-volume	Název úložiště databáze PostgreSQL.
hostPath	/data/postgres-13	Lokální cesta kde je úložiště připojeno.
Úložiště Ansible projektů awx-projects-volume		
storage	2Gi	Dedikovaná velikost úložiště.
storageClassName	awx-projects-volume	Název úložiště Ansible projektů.
hostPath	/data/projects	Lokální cesta kde je úložiště připojeno.

Posledním krokem pro úspěšné vytvoření svazků je nutné vytvořit samotné adresáře na příslušné cestě v souborovém systému:

```
$ sudo mkdir /data/postgres-13
$ sudo mkdir /data/projects
```

V pracovní složce je vytvořen nový soubor `kustomization.yaml`, pomocí kterého bude do Kubernetes nasazen AWX s patřičnými vlastnostmi. Celý soubor `kustomization.yaml` přímo před nasazením AWX má následující obsah:

```
---
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization
namespace: awx

generatorOptions:
  disableNameSuffixHash: true

secretGenerator:
  - name: awx-secret-tls
    type: kubernetes.io/tls
    files:
      - tls.crt
      - tls.key

  - name: awx-postgres-configuration
    type: Opaque
    literals:
      - host=awx-postgres-13
      - port=5432
      - database=awx
      - username=awx
      - password=<zvolene-heslo>
      - type=managed

  - name: awx-admin-password
    type: Opaque
    literals:
      - password=<zvolene-heslo>

resources:
  - github.com/ansible/awx-operator/config/default?ref=1.1.4
  - pv.yaml
  - awx.yaml
```

Samotná instalace AWX je spuštěna příkazem:

```
$ kustomize build . | kubectl apply -f -
```

Nasazení může trvat několik minut, průběh je možné sledovat pomocí příkazu:

```
$ kubectl -n awx logs -f deployments/awx-operator- \
controller-manager -c awx-manager
```

Po dokončení instalace je vypsán výsledek běhu Ansible playbooků:

```
Ansible Task Status Event StdOut
(awx.ansible.com/v1beta1, Kind=AWX, awx/awx)

PLAY RECAP
    *********************************************************************

localhost: ok=74         changed=0         unreachable=0         failed=0
           skipped=61    rescued=0         ignored=1
```

Po úspěšně dokončeném nasazení je možné vypsat veškeré nasazené zdroje příkazem:

```
$ kubectl get all,ingress -n awx
```

V tomto případě výsledek příkazu obsahuje následující výpis[3]:

```
NAME                              READY   STATUS    RESTARTS
pod/awx-operator-
controller-manager                2/2     Running   0
pod/awx-postgres-13               1/1     Running   0
pod/awx                           4/4     Running   0

NAME                              TYPE         CLUSTER-IP    EXTERNAL-IP  PORT(S)
service/awx-operator-
controller-manager-
metrics-service                   ClusterIP    10.43.41.64   <none>       8443/TCP
service/awx-postgres-13           ClusterIP    None          <none>       5432/TCP
service/awx-service               ClusterIP    10.43.209.162 <none>       80/TCP

NAME                              READY   UP-TO-DATE   AVAILABLE
deployment.apps/awx-operator-
controller-manager                1/1     1            1
deployment.apps/awx               1/1     1            1

NAME                             DESIRED   CURRENT   READY
replicaset.apps/awx-operator
-controller-manager              1         1         1
replicaset.apps/awx              1         1         1

NAME                                    READY
statefulset.apps/awx-postgres-13        1/1

NAME                             CLASS     HOSTS           ADDRESS      PORTS
ingress.networking.k8s.io/
awx-ingress                      traefik   awx.arena.com   10.0.4.121   80, 443
```

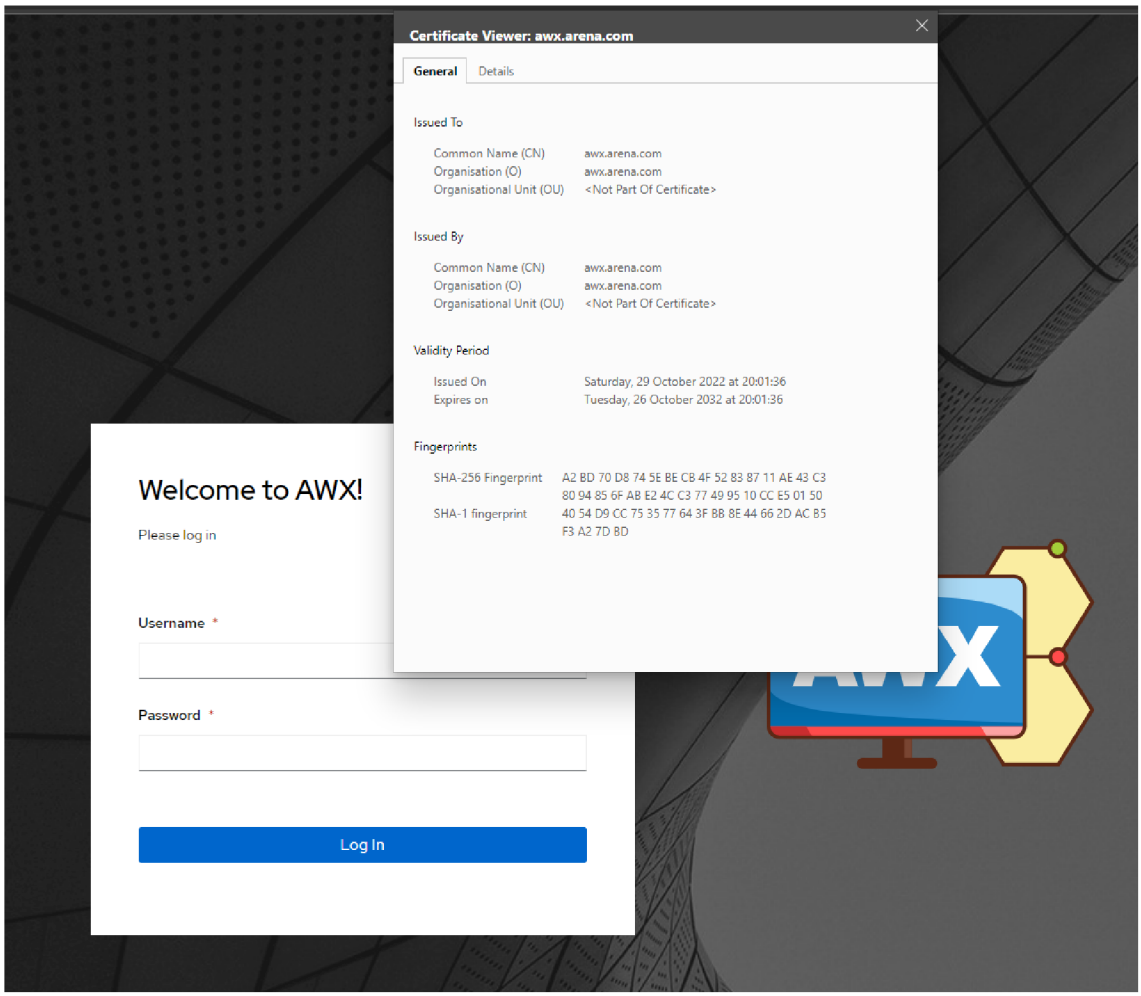
Výše uvedený výpis obsahuje informace o všech Kubernetes zdrojích – *pod*, *service*, *deployment*, *replicaset* a *statefulset* v jmenném prostoru `awx` v jednom Kubernetes clusteru. Následující tabulka A.4 zobrazuje popis všech zdrojů v Kubernetes clusteru.

[3]Výpis je zkrácen o vygenerovanou číselnou část názvů a stáří nasazených zdrojů z důvodu omezeného místa pro výpis.

Tab. A.4: Vysvětlení výpisu všech zdrojů v Kubernetes clusteru

Pod	
Název	Popis
awx-operator-controller-manager	Správce AWX služeb v clusteru.
awx-postgres-13	Správce databázového serveru PostgreSQL.
awx	Webový server, zajišťuje přístup do GUI a správu kompletního AWX prostředí.
Služba (service)	
awx-operator-controller-manager-metrics-service	Monitorování metriky nasazení a zpřístupňování těchto údajů dalším zdrojům nasazeného AWX. IP adresa slouží k vnitřní komunikaci mezi zdroji clusteru.
awx-postgres-13	Zajištění síťového spojení s awx-postgres-13 statefulset.
awx-service	Zajištění síťového spojení s podem awx.
Nasazení (deployment)	
awx-operator-controller-manager	Opatření, že pod s awx-operator bude neustále běžet a vyřizovat žádosti.
awx	Specifikuje a zajišťuje požadovaný stav podu awx.
ReplicaSet	
awx-operator-controller-manager	Tato replika zajišťuje, že běží požadovaný počet podů awx-operator-controller-manager.
awx	Tato replika zajišťuje, že běží požadovaný počet podů awx.
StatefulSet	
awx-postgres-13	Správa nasazení databáze PostgreSQL
Ingress	
awx-ingress	Ingress zdroj spravuje přístup ke službám AWX z externích sítí. Vystavuje služby AWX do externích sítí. V tomto případě je Ingress controller Traefik a služby AWX jsou dostupné pomocí hostname awx.arena.com.

AWX server je dostupný pouze pomocí definovaného `hostname` v souboru `awx.yaml`, zde `awx.arena.com`. Je tedy nutné přidat záznam do lokálního DNS serveru, případně upravit soubor `hosts` na klientovi přistupujícího do AWX. Toto omezení je z toho důvodu, že přístup do služeb AWX spavuje Kubernetes pomocí ingress směrování. Na základě specifikovaného `hostname` je žádost přesměrována k vyřízení konkrétnímu podu, respektive službě. V Kubernetes je možné spravovat chování směrování podle nastavitelných ingress pravidel. Tyto pravidla musí být specifikována v příslušném souboru a předána Kubernetes k jejich aplikaci. Traefik je zvolen z toho důvodu, že je obecně doporučovaný jako Ingress controller pro Kubernetes pro jednoduchost jeho použití a integraci s Kubernetes, více o Traefiku v literatuře [69] a v kapitole *Services, Load Balancing, and Networking* v literatuře [22]. Přihlašovací stránka AWX s *self-signed* certifikátem je zobrazena na obrázku A.1



Obr. A.1: Úvodní webová stránky AWX

A.3 Ovládání Ansible AWX

Nejvyšší logická úroveň uživatelského rozhraní AWX je rozdělena do pěti hlavních sekcí:

- **Views**, ve které jsou zobrazeny statusy spuštěných úloh[4], plánování úloh, souhrn úspěšně a neúspěšně dokončených úloh a další.
- **Resources**, která slouží k definici jednotlivých úloh a dalších parametrů, které jsou nezbytné k úspěšnému provedení úlohy. Jedná se zejména o vytvoření projektu, inventáře, souvisejících playbooků a specifikaci přístupu do spravovaných uzlů.
- **Access**, ve které je možné vytvořit uživatele, skupiny a spravovat jejich přístup do AWX a k jednotlivým projektům.
- **Administration**, která umožňuje nastavit automatické spouštění úloh, zís-

[4] AWX používá anglický výraz *jobs*.

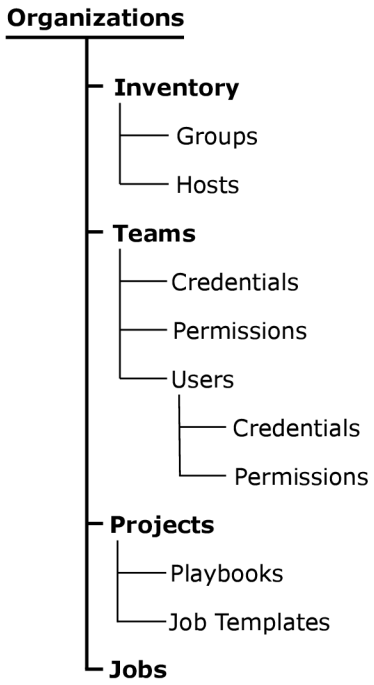
kání informací o instancích a uzlech a zobrazení topologie, se kterou AWX pracuje.

- **Settings**, která slouží k nastavení samotného AWX, tedy například nastavení logování, metod autentizace, spojení s aplikacemi třetích stran, nastavení uživatelského rozhraní a další.

Uživatelské rozhraní může být z počátku nepřehledné a nemusí být ihned zřetelné, jaké kroky má uživatel provést, aby dosáhl úspěšného spuštění úloh, proto je v následující části uveden postup pro úspěšné spuštění úlohy. Pro rozsáhlejší popis uživatelského rozhraní AWX je dostupná literatura [70].

Spuštění ukázkové úlohy

K úspěšnému provedení úlohy na spravovaném uzlu AWX postupuje obdobně jako Ansible při spouštění úloh pomocí CLI. Podrobnosti o konkrétní úloze jsou postupně zadávány do hierarchické struktury, podle které AWX postupuje při spuštění dané úlohy na spravovaném uzlu. Schéma této hierarchie je zobrazena na obrázku A.2[5].



Obr. A.2: Schéma hierarchie AWX

Kroky, které je nutné vykonat pro úspěšné spuštění úlohy na spravovaném uzlu jsou následující:

[5]Názvy v obrázku jsou účelně ponechány v angličtině, aby byly eliminovány nesrovnalosti z překladu.

1. vytvoření organizace,
2. vytvoření projektu a vybrání zdroje, ze kterého budou čerpány zdrojové soubory pro běh úlohy (např. Git, Subversion aj.),
3. vytvoření inventáře,
4. definice přístupu ke spravovanému uzlu,
5. vytvoření *template*, který označuje množinu všech výše nastavených definicí potřebných k provedení dané úlohy,
6. spuštění úlohy.

Vytvoření organizace

Postup pro vytvoření organizace je následující:

1. v sekci *Access* zvolit *Organization*,
2. zvolit *Add*,
3. doplnit požadované informace - libovolný název, popis a volitelně další,
4. doplněné údaje uložit.

Tímto je vytvořena nejvyšší úroveň hierarchie, se kterou jsou logicky spojeny další přidávané údaje.

Vytvoření projektu

Pro vytvoření projektu je nutné poskytnout zdroj, kde jsou uloženy playbooky ve formátu YAML. K tomuto účelu byl vytvořen jednoduchý playbook, který na spravovaném uzlu nainstaluje webový server Apache[6]. Spravovaný uzel, na který je instalován Apache server je vytvořen v OpenStacku s IP adresou `10.0.3.95` a OS Ubuntu Minimal.

Postup pro vytvoření projektu je následující:

1. v sekci *Resources* zvolit *Projects*,
2. zvolit *Add*,
3. doplnit požadované informace - libovolný název, popis a zejména zvolit příslušnou organizaci vytvořenou výše,
4. vybrat požadovaný *Source control*, v tomto případě Git,
5. zadat URL adresu k danému repozitáři,
6. doplněné údaje uložit.

V tomto kroku je důležité si uvědomit, zda je daný repozitář na externím úložišti dostupný. V uvedeném příkladě je Git repozitář veřejný a je možné k němu přistoupit přes HTTPS odkaz. Pokud je repozitář soukromý, nebo je přístup k němu omezený heslem, případně tokenem, je možné k němu i tak z AWX přistoupit, ale je navíc

[6]Repozitář je dostupný na této adrese: `https://github.com/Deydra71/awx-test-job`.

nutné specifikovat přístupové údaje k tomuto repozitáři v sekci *Resources*, v záložce *Credentials*.

Vytvoření inventáře

Inventář obsahuje seznam uzlů, v rámci kterých AWX vykonává danou úlohu. V případě exekuce úlohy pomocí CLI je tento seznam uveden v samostatném souboru zpravidla pojmenovaném *inventory* nebo *hosts*. V rámci AWX jsou spravované uzly specifikovány v sekci *Resources*. Celý postup je následující:

1. v sekci *Resources* zvolit záložku *Inventories*,
2. zvolit *Add*,
3. doplnit požadované informace - libovolný název, popis a zejména zvolit příslušnou organizaci,
4. doplněné údaje uložit,
5. v sekci *Resources* zvolit *Hosts*,
6. zvolit *Add*,
7. doplnit IP adresu spravovaného uzlu,
8. doplněný údaj uložit[7].

Definice přístupu ke spravovanému uzlu

Ansible se ke spravovanému uzlu připojuje pomocí SSH spojení, aby bylo SSH spojení možné je nutné AWX poskytnout potřebné údaje. V tomto případě je vygenerován klíčový pár, pomocí kterého je navázáno spojení do VS v OpenStacku. Pro přidání privátního klíče do AWX je postup následující:

1. v sekci *Resources* zvolit záložku *Credentials*,
2. zvolit *Add*,
3. doplnit požadované informace - libovolný název, popis a zejména zvolit příslušnou organizaci,
4. jako *Credential Type* je zvoleno *Machine*,
5. do kolonky *Username* je dopsáno `ubuntu`,
6. do pole *SSH private key* je importován privátní klíč,
7. doplněné údaje uložit.

Obecně lze pro přístup do spravovaných uzlů místo SSH klíče použít uživatelské jméno a heslo.

[7] Je možné spravovaný uzel specifikovat libovolným názvem a jeho IP adresu doplnit jako proměnnou v příslušném prostoru pod názvem uzlu (YAML nebo JSON formát).

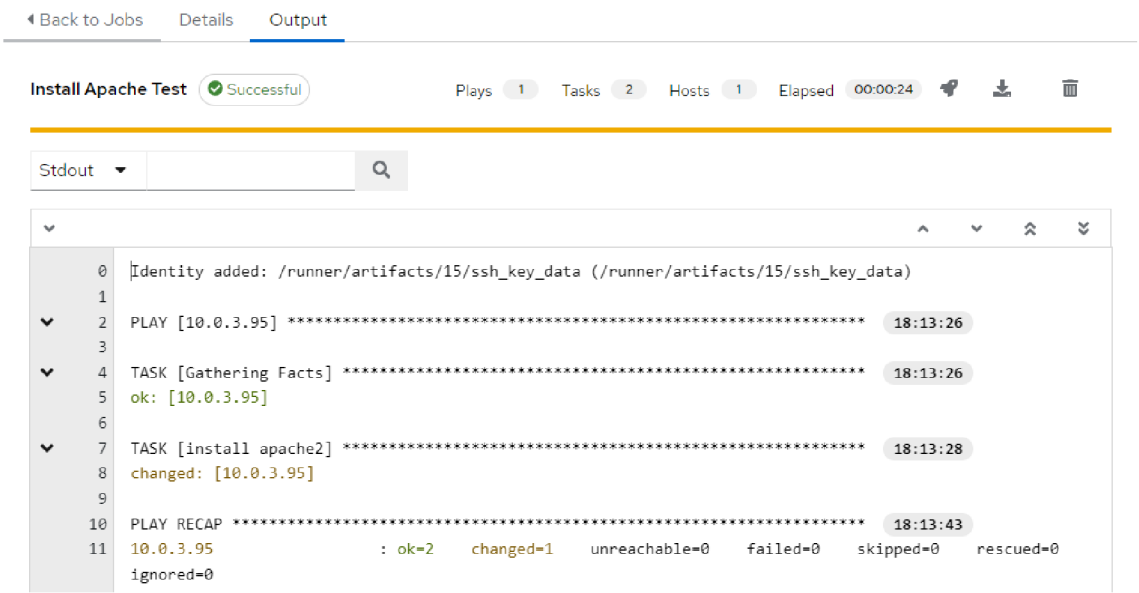
Vytvoření *Template* a spuštění úlohy

Pomocí *templates* jsou všechny zadané informace o úloze sjednoceny a pomocí tohoto *template* je spuštěn daný *job*. Postup vytvoření *template* je následující:

1. v sekci *Resources* zvolit záložku *Templates*,
2. zvolit *Add* a vybrat *Add job template*,
3. doplnit požadované informace - libovolný název, popis a zejména zvolit příslušnou organizaci,
4. jako *Job Type* zvolit *Run*,
5. v poli *Inventory* vyhledat příslušný dříve vytvořený inventář a v poli *Project* vytvořený projekt,
6. v poli *Playbook* zvolit hlavní playbook z vytvořeného Ansible projektu,
7. doplněné údaje uložit.

V případě použití přístupu přes uživatelské jméno a heslo je nutné zaškrtnout volbu Prompt on launch v poli *Credentials*.

V této chvíli je vše připraveno pro spuštění úlohy. Úloha se spustí pomocí tlačítka *Launch* u vytvořeného *template*. Na obrázku A.3 je na zobrazen běh playbooku v prostředí AWX.



Obr. A.3: Úspěšně dokončená úloha v prostředí AWX