Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.92
DOI: https://doi.org/10.24867/21BE28Acimovic

СИСТЕМИ ЗА УПРАВЉАЊЕ СОФТВЕРСКИМ КОНТЕЈНЕРИМА

SYSTEMS FOR SOFTWARE CONTAINERS MANAGEMENT

Радош Аћимовић, *Факултет техничких наука, Нови Сад*

Област - ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај - *У овом раду истражени су системи за управљање софтверским контејнерима: историјат њиховог развоја, поређење разних алата уз навођење њихових предности и мана и начина на који се могу заједно користити.*

Кључне речи: *контејнери, оркестрација, Докер, Кубернетес, AWS*

Abstract - *This work explores systems for software containers management: history of their development, comparison of various tools - explaining their advantages and disadvantages as well as ways in which they can be used together.*

Keywords: *containers, orchestration, Docker, Kubernetes, AWS*

1. УВОД

Софтверски контејнери [1] представљају виртуелно окружење за извршавање програма и процеса. Унутар контејнера је могуће покретати програме, од микросервиса до великих монолитних апликација. Контејнер садржи све непходне извршне фајлове, библиотеке и конфигурационе фајлове. За разлику од традиционалних виртуелних машина, контејнери не садрже у себи читав оперативни систем, већ користе кернел оперативног система машине на којој се налазе. Због тога они користе мање ресурса него виртуелне машине.

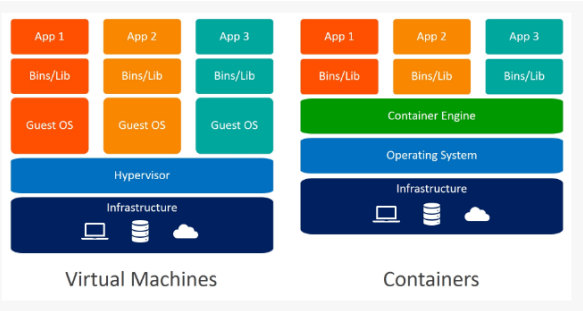
2. ВИРТУЕЛИЗАЦИЈА

Виртуелизација [2] је процес којим се рачунарски ресурси деле на независне логичке целине. Циљ је да се повећа искоришћеност физичких ресурса, да би се на тај начин повећала ефикасност и смањили трошкови. Постоји неколико врста виртуелизационих технологија. Једна од најчешћих је хардверска виртуелизација. Она креира логичку апстракцију хардверске платформе у облику виртуелних машина, контејнера и сличних технологија.

Виртуелне машине представљају виртуелну репрезентацију физичког рачунара и имају сопствени оперативни систем. Виртуелизација омогућава креирање више виртуелних машина на једној физичкој машини. Свака виртуелна машина има сопствени оперативни систем и апликације.

Контејнерске технологије користе карактеристике оперативног система физичке машине, да би направиле изоловано окружење. Контејнер представља посебан процес који је изолован од остатка система. Њему се ограничава приступ другим процесима, фајл систему; ограничава се количина процесорске моћи, оперативне меморије, диска итд. Добијено окружење из перспективе корисника изгледа као посебан одвојен оперативни систем. Зато контејнеризација представља подврсту хардверске виртуелизације која се назива виртуелизација на нивоу оперативног система. Предности контејнерских технологија су мања комплексност, брже перформансе и то што заузимају мање простора.

На слици 1. налази се поређење виртуелних машина са контејнерима.



Слика 1. - Поређење виртуелних машина са контејнерима [3]

Осим хардверске вритуелизације, постоји још неколико врста: виртуелизација десктоп окружења, виртуелизација апликација, мрежна виртуелизација, меморијска виртуелизација, виртуелизација података и виртуелизација складишног простора (*storage*-а).

3. ИСТОРИЈАТ КОНТЕЈНЕРСКИХ ТЕХНОЛОГИЈА

Контејнерске технологије нису нови концепт. Оне су стекле широку употребу након појаве Докера, али већ дуги низ година постоје алати који нуде неки облик контејнеризације [4].

Chroot је алат који је направљен 1979. године за *Unix V7* оперативни систем. Овај алат користи системски позив који мења коренски фолдер, тако да произвољан фолдер може да постане коренски. Процес који је покренут у *chroot* окружењу, као и сви процеси који настану од њега, немају приступ ничему што је ван тог окружења. Због тога се то окружење често назива *jail* (срп. затвор).

FreeBSD jail се појавио 2000. године на *FreeBSD* оперативном систему. Осим трансформације именског простора фајл-система *FreeBSD jail* такође омогућава да виртуелно окружење тј. *jail* има

НАПОМЕНА:
Овај рад проистекао је из мастер рада чији ментор је био др Игор Дејановић, ред. проф.

сопствене IP адресе. Процес покренут унутар окружења не може да приступи спољашњем окружењу нити може да утиче на спољашње процесе. Поврх тога, могуће је управљање корисницима на нивоу *jail*-а.

Након тога, појавили су се и алати као што су *Linux-VServer*, *Solaris* зоне и *Open VZ*, али један од битнијих помака се догађа 2006. године када компанија *Google* развија технологију ***Process containers***. Циљ је да се омогући ограничење и изолација употребе системских ресурса у оквиру Линукс оперативног система. Годину дана касније је назив ове технологије промењен у „контролне групе“. Механизам контролних група користи већина модерних контејнерских алата, као што су *LXC* и Докер.

Још једна значајна година у развоју софтверских контејнера је 2013. година када се појавио **Докер.** Докер је понудио читав екосистем алата за управљање контејнерима и знатно поједноставио рад са контејнерима. Са појавом Докера креће масовна употреба контејнера.

4. ОСНОВНА ТЕРМИНОЛОГИЈА

Пре дискусије о конкретним алатима потребно је појаснити и усагласити терминологију [5].

Контејнерска слика у логичком смислу представља скуп података, метаподатака и конфигурација на основу којих се креира контејнер, а у физичком смислу представља обичну структуру фолдера и фајлова. Слика најчешће нема монолитну структуру, него се састоји од више **слојева.**

Репозиторијум је колекција сродних слика. То могу бити верзије слика, а могу бити и варијанте са различитим пропратним садржајем.

Container host је машина на којој се налази и извршава контејнер. Оперативни систем на тој машини се такође тако назива.

Container engine је алат који има API на кога корисници шаљу захтеве тј. команде. Команде се најчешће шаљу преко командне линије. Циљ је омогућити довољан ниво апстракције тако да вршење операција над контејнерима буде што једноставније. *Engine* у себи садржи и компоненту нижег нивоа која се назива ***container runtime.*** *Engine* велики број операција делегира тој компоненти, док он служи као „омотач“ који поједностављује рад са контејнерима.

Оркестрација контејнера представља технике обједињеног управљања са великим бројем контејнера.

5. КОНТРОЛНЕ ГРУПЕ И ОСТАЛИ МЕХАНИЗМИ КЕРНЕЛА

Постоји неколико механизама из Линуксовог кернела који омогућавају имплементацију контејнера.

Именски простори [6] су механизам кернела који омогућава да неки процес има сопствени поглед на систем: друге процесе, кориснике, фајл-систем, итд. На овај начин се добија изолација ресурса. Овај механизам омогућава да процеси у једном контејнеру не могу да утичу на процесе из другог контејнера. Постоји више типова именских простора, а сваки процес је у по једном од простора сваког типа. На пример, PID именски простор изолује идентификаторе процеса, мрежни именски простор омогућава прављење мрежа изолованих од остатка система итд.

Контролне групе [7] су механизам кернела који омогућава ограничавање и мерење употребе неког ресурса. Није пожељно да један контејнер заузме читаву меморију, да заузме све CPU циклусе, заузме све уређаје, и тако даље. Због тога су потребне контролне групе. Контролне групе су организоване хијерархијски у облику стабла. Обично постоји више хијерархија за различите врсте ресурса и оне су независне једна од друге. Процес се налази у по једном чвору од сваке од тих хијерархија. Примери ресурса који могу бити ограничени контролним групама су: количина оперативне меморије, CPU циклуси итд.

Union File System (UFS) [8] нуди апстракцију над више фајл-система и омогућује да се они виде као један јединствен фајл-систем. Другим речима, могуће је имати фајлове и фолдере на различитим местима, а да они уз употребу UFS-а, са аспекта корисника буду на једном јединственом месту. Када су контејнерске технологије у питању, он омогућава слојевиту структуру контејнерске слике. Постоји више имплемнтација UFS-а, а једна од најпопуларнијих је *OverlayFS.*

Осим наведених механизама, кернел нуди и неке додатне механизме који служе као безбедносне контроле [9]. Пример таквог механизма су *seccomp* профили који се састоје из листа у којима се експлицитно наводи који системски позиви су дозвољени. Осим њих, користе се и механизми као што су *capabilities, SELinux* и *AppArmor.*

6. СТАНДАРДИ

Током развоја софтверских контејнера појавило се неколико стандарда који омогућавају интероперабилност разних класа алата за контејнере. Два најважније је креирала фондација *Open Containers initiative (OCI)* [10]. То су спецификације *image-spec* која дефинише стандардни формат контејнерске слике и *runtime-spec* која дефинише конфигурацију, извршно окружење и животни циклус контејнера.

7. CONTAINER RUNTIME

Container runtime [11] је алат преко кога се извршавају операције над контејнерима. *Container runtime*-ови се обично не користе самостално, већ се употребљавају као компоненте других контејнерских технологија. Представљају „срце“ сваког система за рад са контејнерима. Могу их је поделити на *runtime*-ове нижег нивоа, који раде директно са кернелом (покретање, заустављање контејнера итд.), и *runtime*-ове вишег нивоа, који омогућавају и основне операције са сликама, подешавања мреже итд. Најпознатији *runtime*-ови нижег нивоа су: *runc, railcar, crun, rkt, gVisor, Nabla* и *Kata Containers.* Најпознатији *runtime*-ови више нивоа су *containerd* и *cri-o.*

8. CONTAINER ENGINE

Container engine [5] представља класу алата којима се управља контејнерима. Ови алати су направљени тако да их лако могу користити и људи и програми. Раде

на високом нивоу апстракције и велики део посла делегирају *container runtime*-овима високог и ниског нивоа.

Докер нуди читаву палета алата за рад са контејнерима [12]. Између осталог, ту је и *Docker Engine* - најпопуларнији *container engine*. Он се састоји из сервера (*Docker Daemon*) и API-ја. Команде на *daemon* који управља објектима се шаљу преко *docker client* алата за командну линију. Објекат је заједнички назив за све ентитете којима рукује Докер - слике, контејнери, мреже, *volume*-и, *plugin*-и, итд. Неке од операција над сликама су: преузимање са репозиторијума, креирање нове слике, приказ информација, брисање итд. Операције над контејенерима су: креирање, покретање, заустављање, брисање, листање информација итд. Мреже и *volume*-и се могу креирати, листати, брисати и сл. Најкомплекснија операција је креирање слике. За креирање слике се користи *Dockerfile*. То је фајл који се састоји из низа инструкција преко којих се прави дефиниција слике. На пример: инструкција FROM дефинише слику на основу које се прави нова слика, инструкција COPY копира фајлове или фолдере у слику итд.

Buildah [13] је алат чији је фокус прављење слика: "од нуле", на основу постојеће слике, помоћу *Dockerfile*-а и на основу постојећег контејнера. Осим тога, подржава и остале операције са сликама.

Skopeo [14] је алат чији је фокус рад са сликама и репозиторијумима. Најчешће се користи за конвертовање слика из једног формата у други, као и за преузимање информација о сликама без преузимања читавих слика.

Podman [15] је алат чији је фокус управљање контејнерима и *pod*-овима (група контејнера), а може и да врши операције са сликама. *Podman* управља читавим животним циклусом контејнера, од креирање до уништавања. Такође, може да управља мрежним подешавањима и подешавањима *storage*-а.

При избору *container engine*-а, постоје две опције. Прва је употреба Докера, а друга је употреба *Podman*-а заједно са *Buildah*. Боља опција је Докер, јер има већи број *feature*-а, додатне алате, а такође, има и већу заједницу корисника. *Skopeo* се може користити као помоћни алат у оба случаја.

9. КОНТЕЈНЕРСКИ РЕГИСТРИ

Контејнерски регистри [16] представљају скуп репозиторијума на којима се налазе слике. Они омогућавају складиштење слика, као и дељење тих слика са другим корисницима и системима. Осим тога, многи регистри омогућавају безбедносне контроле као што су аутентификација и ауторизација корисника и програма, скенирање слика итд. Регистри могу бити приватни и јавни. Осим тога, могу се поделити на *third-party* регистре и *self-hosted* регистре. Неки од најпознатијих регистара су: *Docker Hub, Docker Registry, Amazon ECR, Google Artifact Registry, JFrog Container Registry* итд.

10. ОРКЕСТРАЦИЈА КОНТЕЈНЕРА

Оркестрација контејнера [5] представља аутоматизовано управљање контејнерима. Алати који врше оркестрацију се називају оркестратори контејнера. Операције као што су скалирање контејнера и машина, повезивање великог броја контејнера у мрежу и сл. би биле веома споре и неефикасне без оркестратора.

Docker Compose [17] је Докеров алат који служи за оркестрацију контејнера у оквиру једне *host* машине. Користи се углавном за прављење окружења за развој и тестирање софтвера. Може се користити и на продукцији ако се користи само једна *host* машина. Окружење се дефинише декларативно у YAML фајлу, који се најчешће назива *docker-compose.yml*. Све што може да се уради ручно преко Докерових команди и параметара - прављење контејнера, мрежа, дефинисање ресурса, безбедносних подешавања и тако даље - то исто може да се уради и преко овог алата, али на много бржи и једноставнији начин.

Podman Compose [18] је алат из истог пројекта као *Podman*. За креирање окружења користи фајлове формата као за *Docker Compose*. Команде су такође веома сличне. Битна разлика је у томе што се контејнери које дефинише *docker compose* фајл стављају у *pod*.

Swarm [19] је Докеров алат за оркестрацију контејнера на више *host* машина. Свака машина у том кластеру се назива чвор. Постоје два типа чворова. Радници су чворови на којима се покрећу контејнери. Менаџери управљају кластером - оркестрирају кластер тако што додељују послове радницима, управљају конфигурацијом итд. Основни објекат са којим ради *Swarm* назива се сервис. Он дефинише која слика треба да се користи за покретање контејнера, које портове треба отворити, на коју мрежу ставити контејнер, дефинише ограничења ресурса и тако даље. Веома често је потребно имати више инстанци (реплика) истог контејнера. У том случају ће сервис да има више задатака. Задаци представљају најмање јединице управљања у *Swarm*-у. Задаци се распоређују по расположивим радницима и могу имати максимално један контејнер.

Кубернетес је алат направљен од стране компаније *Google*. Кубернетес кластер [20] се састоји од контролне равни и чворова. Контролна раван обједињује управљачке компоненте кластера. Чворови су машине на којима се покрећу контејнери. Контролна раван се састоји од неколико независних компонената. Свака компонента је специјализована за одређени посао и налази се у контејнеру. Компоненте могу бити на једној машини, или на више њих. Једна компонента може да има више инстанци. Најважнији објекти у Кубернетесу су [21]:

- *Pod* је основна јединица управљања у Кубернетес кластеру. Може да садржи један или више контејнера.
- Сервис је објекат коме се додељују IP адресе и DNS имена. Захтеве који буду послати на те адресе и портове он прослеђује *pod*-овима, тачније, врши расподелу оптерећења и прослеђује саобраћај неком од *pod*-ова наведеним у дефиницији сервиса.
- *Replica set* је објекат који дефинише број реплика *pod*-ова и предузима акције да број *pod*-ова увек буде једнак том броју. То омогућава једноставно скалирање.

- *Deployment* је објекат који омогућава ажурирање верзије апликације без *downtime*-а. *Deployment* се користи искључиво за *stateless* апликације.
- *Stateful set* је објекат који управља *pod*-овима у којима се покрећу *stateful* апликације. Иако сви *pod*-ови у *stateful set*-у настају из истог дефинисаног шаблона, сваки од њих има свој посебан идентитет и стање.

Amazon ECS [22] је алат за оркестрацију контејнера који је директно интегрисан у инфраструктуру *AWS cloud*-а. Састоји се од контролне равни и равни података. ECS је сервис којим у потпуности управља AWS, тако да је контролна раван у потпуности „брига“ AWS-а. Раван података у којој се покрећу контејнери се дефинише при креирању кластера. Инфраструктура за раван података се обезбеђује из три врсте „извора“: *Fargate* (омогућава да ECS аутоматски добије инфраструктуру за покретање контејнера), EC2 инстанци и спољних сервера који су ван *AWS Cloud*-а. Основна јединица управљања у ECS-у је задатак. Може да има један или више контејнера. Прво је потребно креирати дефиницију задатка. Дефиниција задатка је независна од кластера. У неком одређеном кластеру је затим потребно креирати задатак или сервис. Сервис може да има један или више реплика неког задатка који се креира на основу дефиниције задатка.
Највећи број функционалности нуди Кубернетес. Истовремено, он је најкомплекснији алат. Будући да има велику заједницу корисника, постоји велики број готових решења за ствари опште намене, као што су CI/CD, мониторинг итд.

11. ЗАКЉУЧАК

Предмет истраживања овог мастер рада су били системи за управљање софтверским контејнерима. Први циљ овог рада је био да се да преглед свих релевантних технологија у овој области - њихових предности и мана, када које треба користити и како се међусобно допуњују. Други циљ овог рада је био да се опише како поједини алати функционишу интерно.
Употреба софтверских контејнера је и даље у успону и све више организација их користи. Постојећи алати се унапређују, а истовремено се појављују и многи нови алати.

12. ЛИТЕРАТУРА

[1] https://www.netapp.com/devops-solutions/what-are-containers/, последњи приступ: октобар 2022.
[2] https://kompjuteras.com/pojam-virtuelizacije-i-osnovni-termini/, последњи приступ: октобар 2022.
[3] https://www.weave.works/blog/a-practical-guide-to-choosing-between-docker-containers-and-vms, последњи приступ: октобар 2022.
[4] https://blog.aquasec.com/a-brief-history-of-containers-from-1970s-chroot-to-docker-2016, последњи приступ: октобар 2022.
[5] https://developers.redhat.com/blog/2018/02/22/container-terminology-practical-introduction, последњи приступ: октобар 2022.
[6] Mastering Linux Kernel Development, Raghu Bharadwaj, 2017, стр. 99-106
[7] https://man7.org/linux/man-pages/man7/cgroups.7.html, последњи приступ: октобар 2022.
[8] https://medium.com/@knoldus/unionfs-a-file-system-of-a-container-2136cd11a779, последњи приступ: октобар 2022.
[9] https://www.suse.com/c/rancher_blog/introduction-to-container-security/, последњи приступ: октобар 2022.
[10] https://opencontainers.org/about/overview/, последњи приступ: октобар 2022.
[11] https://www.capitalone.com/tech/cloud/container-runtime/, последњи приступ: октобар 2022.
[12] https://www.whizlabs.com/blog/docker-architecture-in-detail/, последњи приступ: октобар 2022.
[13] https://github.com/containers/buildah/blob/main/docs/tutorials/01-intro.md, последњи приступ: октобар 2022.
[14] https://github.com/containers/skopeo, последњи приступ: октобар 2022.
[15] https://github.com/containers/podman, последњи приступ: октобар 2022.
[16] https://www.redhat.com/en/topics/cloud-native-apps/what-is-a-container-registry, последњи приступ: октобар 2022.
[17] https://gabrieltanner.org/blog/docker-compose/, последњи приступ: октобар 2022.
[18] https://phoenixnap.com/kb/podman-compose, последњи приступ: септембар 2022.
[19] https://docs.docker.com/engine/swarm/key-concepts/, последњи приступ: октобар 2022.
[20] https://kubernetes.io/docs/concepts/overview/components/, последњи приступ: октобар 2022.
[21] https://medium.com/devops-mojo/kubernetes-objects-resources-overview-introduction-understanding-kubernetes-objects-24d7b47bb018, последњи приступ: октобар 2022.
[22] https://docs.aws.amazon.com/AmazonECS/latest/developerguide/Welcome.html, последњи приступ: октобар 2022.

Кратка биографија:

Радош Аћимовић рођен је 12. априла 1995. године у Суботици. На Факултет техничких наука у Новом Саду се уписао 2014. године - смер: Софтверско инжењерство и информационе технологије. Дипломирао је 2018. године и исте те године уписао мастер студије на истом факултету - смер: Софтверско инжењерство и информационе технологије, усмерење Електронско пословање. Положио је све испите предвиђене планом и програмом.
Контакт: radosacimovic@gmail.com