Učinkovita in prilagojena podpora poslovnim procesom s pomočjo odprtokodnih rešitev

Miroslav Beranič

Bintegra d.o.o., Maribor, Slovenija
miroslav.beranic@bintegra.com

Sinopsis V prispevku bomo predstavili celostno infrastrukturno in arhitekturno rešitev za podjetja za povečanje konkurenčne prednosti in oplemenitenje dodane vrednosti obstoječim poslovnim procesom ter celovito pot razvoja sprememb v poslovnih procesih s pomočjo rešitev Red Hat OpenShift, Liferay DXP in Quarkus. Red Hat OpenShift je odprtokodna rešitev, ki nudi poenoteno delovanje in upravljanje izbranih aplikacij, zajema celovit nabor aplikacij in procesov za avtomatizirano delo z izbranimi rešitvami ter nadzor nad izvajanjem in delovanjem. Liferay DXP je odprtokodna rešitev, ki omogoča izgradnjo uporabnikom prijazno in prilagojeno spletno rešitev. Razvijalci lahko pri razvoju sledijo in sodelujejo z odločevalci poslovnih procesov, ki se ne razumejo v poglobljene razvojne procese. Quarkus je ogrodje, ki omogoča optimizirano izvajanje Java aplikacij ter odpira vrata v mikrostoritveno arhitekturo rešitve.

S pomočjo prikazanih rešitev je podjetjem omogočeno hitro prilagajanje pravilom in ažurna komunikacija s strankami na vseh kanalih, na katerih je podjetje prisotno, bodisi javna spletna stran, samopostrežni terminali, email ali socialni mediji.

Ključne besede:
poslovni procesi
optimizacija
Red Hat OpenShift
Liferay DXP
Quarkus

Univerzitetna založba Univerze v Mariboru
ISBN 978-961-286-639-6
DOI https://doi.org/10.18690/um.feri.10.2022.14

1 Uvod

Informacijska podpora v podjetjih se razlikuje glede na področje v katerim podjetje deluje, starostjo podjetja in tudi ciljnim bazenom končnih kupcev oz. strank. V tem kontekstu, bi lahko v grobem podjetja delili v tista, ki informacijske rešitev proizvajajo oz. jih nadgrajujejo in nudijo ostalim, podjetja ki uporabljajo IT kot jederno gonilno silo in prilagodijo poslovne procese uporabljenim IT rešitvam ter tistim, ki IT uporabljajo občasno oz. za delno avtomatizacijo mesečnih in letnih zahtev zakonodaje.

Razvoj IT rešitev ter sama vpeljava – namestitev in vzdrževanje se je skozi čas vršil bolj ali manj enako – po enakem postopku. Razvoj se je izvajal v IT podjetjih. Namestitev so izvajali informatiki s strani podjetja, ki je IT rešitev razvilo ali drugo specializirano IT podjetje, ki je imelo znanje namestiti programsko opremo na strojno opremo ali interni IT strokovnjaki. Strojna oprema je bila ali v lasti končnega podjetja ali ponudnika IT rešitev.

Vse te aktivnosti so imele cikel – od razvoja do vzpostavitve in nato vzdrževanja, ki je zajemalo tudi nadgradnje. Pogosto vidimo, tudi pri lokalnih ponudnikih bančnih storitev, obvestilo o ne-delovanju sistema zaradi nadgradnje. Takšne posodobitve se izvajajo v času, ko ni velikega števila uporabnikov, to so predvsem nočni časi.

Podjetje, ki posluje globalno pa časovnega okna, ko ni veliko uporabnikov nima. Takšno podjetje, si ne mora privoščiti časovnega okna, v katerem bi izvajalo posodobitve. V realnosti, se to dogaja in se izvaja na tak način, da se je potrebno odločiti, v katerem časovnem območju se bo to izvajalo – da bo izpad in vpliv čim manjši. To dejstvo je posledica dosedanjega načina že samega razvoja in predaje v uporabo programske rešitve. Pri takšnem načinu, določena ciljna skupina uporabnik, v času posodobitve ne mora uporabljati ali pa je uporaba rešitve omejena.

Predlagane in opisane rešitve so tako že dlje časa v uporabi s strani podjetij, ki imajo globalne trge in so bila primorana rešitve najti med prvimi. Skozi številne iteracije in različne industrije se je oblikoval konzorcij rešitev in predlaganih načinov razvoja, ki lahko pomagajo tudi manjšim podjetjem, ki nimajo primerljivih problemov, a hkrati se je zgodilo, da so uporabniki postali razvajeni z uporabo IT rešitev na način – da preprosto delujejo, ko jih potrebujejo, za namene, ki jih potrebujejo in niso vpleteni v »trenutno ne poslujemo, ker posodabljamo« programsko opremo. Primer tega so številne rešitve podjetji kot so Microsoft, Google, Amazon in Facebook.

V nadaljevanju bom poskušal opisati predlagano rešitev, ki temelji na odprtokodni in javno dostopni rešitvi imenovani Kubernetes. V prispevku se bom osredotočil na eno od številnih specializiranih rešitev, ki temelji na rešitvi Kubernetes. Predstavil bom rešitev podjetja Red Hat, imenovano OpenShift. V povezavi z rešitvijo OpenShift bo predstavljen tudi programski paket podjetja Liferay Inc imenovan Liferay DXP. Za dodajanje lastnih vmesnikov ter integracijo z zunanjimi viri bo predstavljeno odprtokodno ogrodje Quarkus.

2 Predstavitev predlaganih IT Rešitev

2.1. Liferay DXP

Liferay DXP je programska rešitev, ki omogoča podjetju, da izpostavi svojo digitalno prisotnost na spletu. Hkrati pa poenoti svoje notranje poslovne procese in jih digitalizira. Liferay DXP vsebuje gradnike, s katerimi je mogoče:

- izdelati javno spletno stran
- vprašalnike in zbiranje odziva uporabnikov,
- gradnja poslovnih samopostrežnih terminalov, ki pomagajo razbremeniti podporo
- združevanje komunikacijskih kanalov na družbenih omrežjih
- promocija produktov in rešitev

- objava poslovnih novic
- izdelava spletne trgovine povezane z ostalimi notranjimi podpornimi sistemi
- dostop poslovnim partnerjem do dokumentacije in podpori v poprodajnih kanalih

Liferay DXP ima vgrajeno podporo za delovanje na globalnih trgih, kjer je mogoče za vsako interesno ali geografsko področje prilagoditi vsebino in dostope.

2.2. OpenShift

Red Hat OpenShift je t.i. distribucija odprtokodnega projekta Kubernetes. Kot večina ostalih IT rešitev, ki jih nudi podjetje Red Hat je tudi OpenShift komercialni produkt drugače odprtokodnega t.i »upstream« projekta imenovanega OKD. Danes imam nadet Red Hat klobuk.

Red Hat OpenShift je izvajalno (ang. »runtime«) okolje za IT rešitve. Lahko si ga predstavljamo kot virtualno strojno opremo oz. operacijski sistem, ki omogoča in nudi prostor, kamor se namesti aplikacija, ki je nato dostopna preko enega od možnih spletnih dostopov, to je najpogosteje »običajen« spletni dostop – spletne aplikacije.

Red Hat OpenShift rešuje številne oz. vse izzive glede izvajanja in vzdrževanja t.i. mikrostoritvenih rešitev, predstavlja celostno vitualizirano infrastrukturno okolje.

2.3. Quarkus

Quarkus je projekt oz. ogrodje, ki omogoča optimiziranje razvojne poti projekta in izboljšanje metrik v času izvajanja programov. Primarno je namenjen za Java projekte, ki zajemajo tudi ogrodja in programske jezike, ki temeljijo na programskem jeziku Java – kot so Kotlin.

Quarkus je orientiran in primarno namenjen za razvoj t.i. mikrostoritev.

3 Virtualizacija infrastrukture

Virtualizacija infrastrukture omogoča hitrejše prilagajanje na spremembe v potrebah po procesorski zmogljivosti, dostopnosti za programerje in uporabnike ter preprostejšo in celovitejšo vzdrževanje za vzdrževalce.

Red Hat OpenShift Container Platform je temelj za razvoj in zagon kontejneriziranih aplikacij. Zasnovan je na način, da omogoča, da se začne z manjšim osnovnim naborom procesorskih virov in raste z rastjo potreb po dodatnih zmogljivostih. Omogoča upravljanje in zagon od nekaj kontejnerjev to nekaj tisoč in nudenje storitev miljonom uporabnikov.

OpenShift temelji na Kubernetes, ki ga nadgradi z dodatnimi orodji in izvedbenimi procesi, ki standardizirajo in poenotijo tok od analize, razvoja, testiranja, predaje v produkcijsko okolje in nadaljno vzdrževanje in upravljanje.

3.1. Kubernetes

Kubernetes je orhestracijsko orodje in ogrodje, ki omogoča upravljanje z večjim številom kontejneriziranih aplikacij in določitev odvisnosti med njimi. Orhestracija, ki jo opravlja, nudi avtomatsko posodabljanje verzij aplikacij in prilagoditev velikosti glede na potrebe ter nudi vpogled v delovanje vsake posamezne aplikacije. Samo prenos aplikacije v kontejnerizirano obliko ima za razvijalca in vzdrževalca veliko več dodatnega dela, ki ga je mogoče avtomatizirati in to avtomatizirano vlogo opravlja Kubernetes.

Zraven orhestracije samih kontejnerjev, pa Kubernetes nudi tudi upravljanje s podatkovnimi viri – upravlja z zahtevami po procesorski moči, potrebnim delovnim spominom in velikostjo dolgotrajne hrambe podatkov. Vse to je opisano v opisni obliki, Kubernetes se nato sam odloči (glede na svoje vedenje postavitve infrastrukture), kje, kako dolgo in na kak način bo izvedel posamezen kontejner. Analogno kot kontejnerji v pristanišču, Kubernetes, premika kontejnerje na način, da optimizira podatkovne in procesne tokove.

3.2. Hramba podatkov

Kontejnerji sami ne hranijo podatkov, ob zagonu nimajo vedenja o predhodnih zagonih, pomeni – vsak zagon je prvi zagon. Kontejnerji so t.i. »immutable«, pomeni, da nimajo spreminjajočega stanja. To pomeni, da si ničesar ne zapomnijo, pomeni da nimajo vedenja predhodnih zagonov. Na prvi pogled čudno delovanje, je jedro uspeha in ena glavnih lastnosti preboja tehnologije v ospredje in v širšo uporabo.

To vedno ni želeno stanje, recimo podatkovna baza – PostgreSQL – si želimo kontejnerizirati storitev »SQL podatnovne baze«, a hkrati želimo tudi obdržati podatke. Kontejnerje je potrebno povezati v zunanji sistem preko t.i. »volume« - izmenljivih podatkovnih vmesnikih. Podatkovni vmesniki omogočajo, da se pri opisu posameznega kontejnerja določi, katero mesto (direktorij) znotraj kontejnerja je povezan na katero mesto izven kontejnerja in na tak način omogoči, da kontejner ob ponovnem zagonu ne izgubi podatkov.

Pri tem je potrebno seveda paziti, na kak način se to izvede, da se podatki med kontejnerji ne prepisujejo ali pokvarijo.

3.3. Mrežna povezljivost

Kontejnerji privzeto ne vidijo zunanjega sveta – izven kontejnerja, kar je z varnosti zelo ugodno, iz uporabnega stališča pa ne tako zelo. Podobno kot pri izmenjavi podatkov, je potrebno za vsak kontejner določiti preko katerega porta se bo izvedla komunikacija. Določi se port na strani kontejnerja in na strani infrastrukure – kar omogoča, da zunanji sistemi in kontejnerji med seboj lahko komunicirajo.

3.4. Namestitev in posodobitev

Virtualizacija infrastrukure omogoča, da se neglede na jederno infrastrukturo, proti uporabniku uporabljena infrastruktura obnaša enako. OpenShift omogoča namestitev na lokalno fizično strojno opremo, na lokalno virtualizirano okolje ali strežniško okolje v oblaku ali pa vse hkrati – kot hibridno distribuirano okolje, ki je hkrati tudi priporočljiv način namestitve.

Virtualizacija omogoča, da se kompleksnost, ki jo številčni kontejnerji zahtevajo poenostavi s preprostim uporabniškim vmesnikom, ki je prilagojen za vlogo posameznika in daje popolno osredotočenost na določeno nalogo, ki jo dana vloga opravlja; tako je recimo uporabniški vmesnik za vlogo razvojnika drugačna od uporabniškega vmesnika za administratorja sistema.

Posodobitev virtualiziranega okolja omogoča hitre spremembe na vseh nivojih in ker so virtualizirani vsi sestavni deli infrastrukture, je mogoče ustvarjanje virtualnih omrežij in kompleksnih varnostnih pravil enostavno in avtomatizirano, hkrati pa se tudi samodejno prilagaja na spremembe v omrežju.

3.5. Razširljivost in odpornost

Virtualizirana infrastruktura omogoča, da se prilagaja potrebam bolje in hitreje, kot običajna fizična infrastruktura, predvsem iz stališča avtomatizacije nalog in »programskega« preklapljanja drugače fizičnih povezav – mrežni kabli, trdi diski, celotni strežniki, vse to je programska koda in virtualizirana za hiter prenos med podatkovnimi skladišči.

4 Postavitev okolja

Začeli bomo s postavitvijo okolja, ki bo predstavljajo osrednjo IT infrastrukturo. V članku bo predstavljen prosto dostopen način za vzpostavitev OpenShift okolja. Pripravljena instanca je brez večjih omejitev funkcionalnosti in je v polnem obsegu; omejena ja na vrhnjo domeno ».testing«, kar med drugimi omejitvami onemogoča, da se uporablja v produkcijskem okolju, prav tako niso omogočeni nekateri vtičniki, ki omogočajo izvajanje v produkcijskem obsegu.

Red Hat ponuja tudi portal, ki omogoča kreiranje lastne OpenShift instance in se nahaja na naslovu [1]. Trenutna verzija Red Hat OpenShift-a, ki je dostopna preko portala je 4.10, kar je praktično zadnja verzija. Omogoča vse, kar si bomo pogledali danes, z razliko – da se izvaja v oddaljenem računalniškem oblaku z omejenimi viri – CPU in RAM, kar terja daljši čas izvajanja operacij.

Dokumentacija je na voljo na [2], kjer so za vsako verzijo »Release notes«. Dokumentacija zajema veliko vsebine in je zelo uporabna.

Pogledali bomo Red Hat OpenShift Local oz. do pred kratkim imenoval CodeReady Containers [3]. Za namestitev zadošča srednje zmogljiv prenosnik računalnik, povprečno z 8 CPU in 16GB RAM ter 60 GB prostora na disku.

Uporabljena Red Hat OpenShift Local distribucija je verzije 4.10.

4.1. Namestitev

Predstavljena namestitev je opisana za operacijski sistem Linux. Celotna navodila, za ostale operacijske sisteme in v celoti so objavljene na [4].

Po namestitvi terminalnega vmesnika – CLI imenovanega crc [5], se preveri verzija:

```
$ crc version
CRC version: 2.6.0+d606e64
OpenShift version: 4.10.22
Podman version: 4.1.0
```

Določitev osnovnih začetnih parametrov in privolitev v zbiranje telemetričnih podatkov.

```
$ crc config set preset openshift
$ crc config set consent-telemetry yes
$ crc config set cpus 16
$ crc config set memory 24576
```

Sledi glavni namestitveni proces, ki se izvede z ukazom:

```
$ crc setup
INFO Using bundle path /home/miroslav/.crc/cache/crc_libvirt_4.10.22_amd64.crcbundle
INFO Checking if running as non-root
INFO Checking if running inside WSL2
INFO Checking if crc-admin-helper executable is cached
INFO Checking for obsolete admin-helper executable
INFO Checking if running on a supported CPU architecture
INFO Checking minimum RAM requirements
INFO Checking if crc executable symlink exists
INFO Checking if Virtualization is enabled
```

```
INFO Checking if KVM is enabled
INFO Checking if libvirt is installed
INFO Checking if user is part of libvirt group
INFO Checking if active user/process is currently part of the libvirt group
INFO Checking if libvirt daemon is running
INFO Checking if a supported libvirt version is installed
INFO Checking if crc-driver-libvirt is installed
INFO Checking crc daemon systemd service
INFO Checking crc daemon systemd socket units
INFO Checking if AppArmor is configured
INFO Checking if systemd-networkd is running
INFO Checking if NetworkManager is installed
INFO Checking if NetworkManager service is running
INFO Checking if dnsmasq configurations file exist for NetworkManager
INFO Checking if the systemd-resolved service is running
INFO Checking if /etc/NetworkManager/dispatcher.d/99-crc.sh exists
INFO Checking if libvirt 'crc' network is available
INFO Checking if libvirt 'crc' network is active
INFO Checking if CRC bundle is extracted in '$HOME/.crc'
INFO Checking if /home/miroslav/.crc/cache/crc_libvirt_4.10.22_amd64.crcbundle exists
INFO Getting bundle for the CRC executable
INFO Downloading crc_libvirt_4.10.22_amd64.crcbundle
3.16 GiB / 3.16 GiB [-------------------------------------------------------------------------------] 100.00% 9.00 MiB p/s
INFO Uncompressing /home/miroslav/.crc/cache/crc_libvirt_4.10.22_amd64.crcbundle
crc.qcow2: 12.62 GiB / 12.62 GiB [-----------------------------------------------------------------] 100.00%
oc: 117.14 MiB / 117.14 MiB [---------------------------------------------------------------------] 100.00%
Your system is correctly setup for using CRC. Use 'crc start' to start the instance
```

Po preveritvi in izpolnitvi vseh predpogojev, se namesti delujoč sistem. Za namestitvijo se lahko požene prva OpenShift Local Cluster, to se izvede preko ukaza:

```
$ crc start
INFO Checking if running as non-root
INFO Checking if running inside WSL2
INFO Checking if crc-admin-helper executable is cached
INFO Checking for obsolete admin-helper executable
INFO Checking if running on a supported CPU architecture
INFO Checking minimum RAM requirements
INFO Checking if crc executable symlink exists
INFO Checking if Virtualization is enabled
INFO Checking if KVM is enabled
INFO Checking if libvirt is installed
INFO Checking if user is part of libvirt group
INFO Checking if active user/process is currently part of the libvirt group
INFO Checking if libvirt daemon is running
INFO Checking if a supported libvirt version is installed
INFO Checking if crc-driver-libvirt is installed
INFO Checking crc daemon systemd socket units
INFO Checking if AppArmor is configured
INFO Checking if systemd-networkd is running
INFO Checking if NetworkManager is installed
INFO Checking if NetworkManager service is running
```

```
INFO Checking if dnsmasq configurations file exist for NetworkManager
INFO Checking if the systemd-resolved service is running
INFO Checking if /etc/NetworkManager/dispatcher.d/99-crc.sh exists
INFO Checking if libvirt 'crc' network is available
INFO Checking if libvirt 'crc' network is active
INFO Loading bundle: crc_libvirt_4.10.22_amd64...
INFO Starting CRC VM for OpenShift 4.10.22...
INFO CRC instance is running with IP 192.168.130.11
INFO CRC VM is running
INFO Check internal and public DNS query...
INFO Check DNS query from host...
WARN Wildcard DNS resolution for apps-crc.testing does not appear to be working
INFO Verifying validity of the kubelet certificates...
INFO Starting OpenShift kubelet service
INFO Waiting for kube-apiserver availability... [takes around 2min]
INFO Waiting for user's pull secret part of instance disk...
INFO Starting OpenShift cluster... [waiting for the cluster to stabilize]
INFO 2 operators are progressing: network, operator-lifecycle-manager-packageserver
INFO 3 operators are progressing: dns, network, operator-lifecycle-manager-packageserver
INFO Operator network is progressing
INFO Operator network is progressing
INFO Operator node-tuning is not yet available
INFO All operators are available. Ensuring stability...
INFO Operators are stable (2/3)...
INFO Operators are stable (3/3)...
INFO Adding crc-admin and crc-developer contexts to kubeconfig...
Started the OpenShift cluster.

The server is accessible via web console at:
https://console-openshift-console.apps-crc.test
Log in as administrator:
Username: kubeadmin
Password: I8Ube-zhGBv-wKzG7-8iDSe

Log in as user:
Username: developer
Password: developer

Use the 'oc' command line interface:
$ eval $(crc oc-env)
$ oc login -u developer https://api.crc.testing:6443
```

Po nekaj minutah, se v terminal izpišejo dostopni podatki za dostop do spletnega vmesnika in privzetih uporabniških računov. Pripravljena sta dva računa developer – za razvijalca in kubeadmin – za administratorja. Primarno se uporablja uporabniški račun za razvijalca.

4.2. Dostop do postavljene infrastrukture

Z nameščanje infrastrukture smo s tem praktično končali. Sledi urejanje uporabniških računov, diskovnih polj, mrežnih komunikacijskih kanalov ter ostala administracijska opravila, ki jih danes ne bomo pogledali.

Do postavljene infrastrukture lahko dostopamo na dva načina:

- Preko terminalne / ukazne lupine z uporabo orodja »oc« ali
- Spletna aplikacija, dostopna na naslovu izpisanem v postopku namestitve.

```
$ oc login -u developer https://api.crc.testing:6443


$ oc login -u kubeadmin -p I8Ube-zhGBv-wKzG7-8iDSe https://api.crc.testing:6443
Login successful.


You have access to 58 projects, the list has been suppressed. You can list all projects
with ' projects'


Using project "default".


$ oc whoami
kube:admin


$ crc console
Opening the OpenShift Web Console in the default browser...
```

4.3. Odstranitev OpenShiftLocal

Po končanem testiranju se lahko Red Hat OpenShift Local v celoti odstrani. To se izvede preko terminala. V terminalu se izvede ukaz za zaustavitev:

```
$ crc stop
```

Počakamo, da se OpenShift zaustavi in nato se izvede ukaz za izbris celotnega OpenShift Local Clustra. To se izvede z ukazom:

```
$ crc delete
```

4.4. Namestitev v produkcijsko okolje

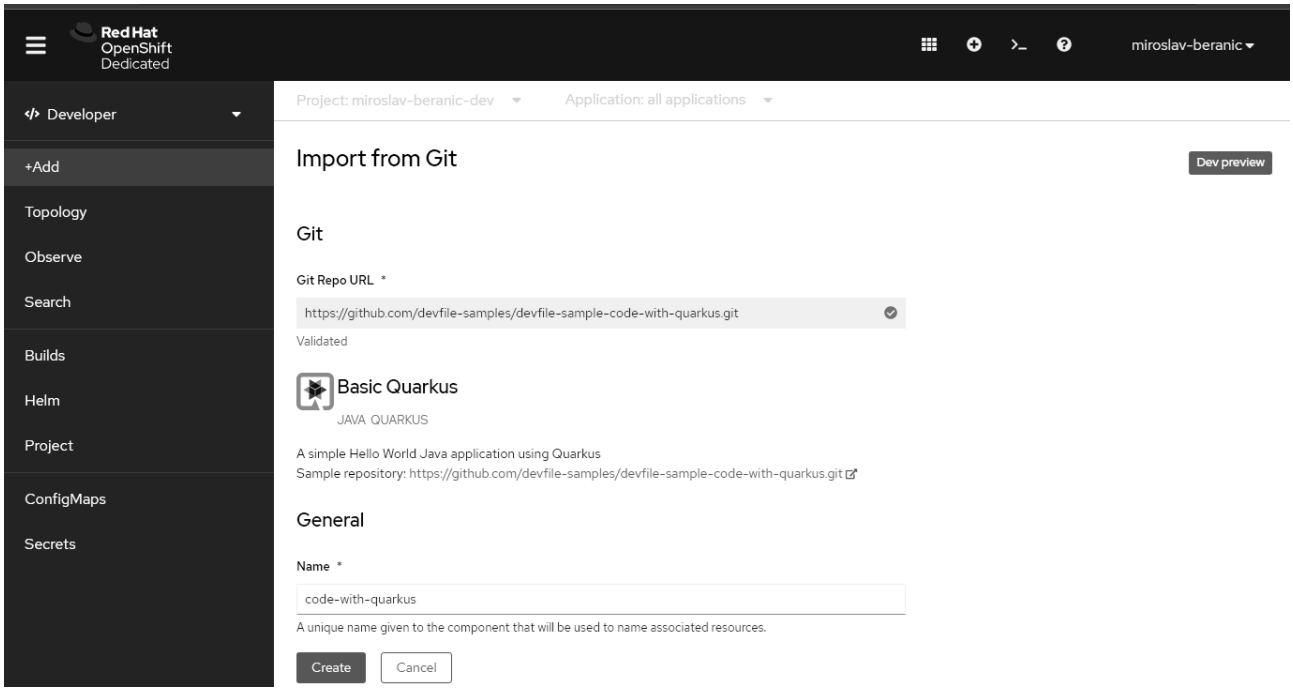
Namestitev v produkcijsko okolje je praviloma analogno prikazanemu, za razliko, da se uporablja večje število fizičnih ali virtualnih strojnih strežnikov. Praviloma se priporoča vsaj tri ločene strežnike za »worker node« in prilagojenim diskovnim poljem, za katerega je priporočljivo, da je lokalno in sestavljeno iz hitrih pogonskih enot.

5 Nameščanje lastnih aplikacijskih rešitev na postavljeni infrastrukturi

Ko imamo enkrat postavljeno infrastrukturo, lahko začnemo nameščati aplikacije. Danes bom pokazal kako se namesti večja IT rešitev in ločena mikrostoritev za uporabo Quarkus ogrodja. Na voljo so tudi številne druge pred-pripravljene rešitve, ki se jih lahko namesti iz vgrajenega kataloga dostopnih aplikacij. Začeli bomo s preprostim in potrebnim »Hello world« primerom.

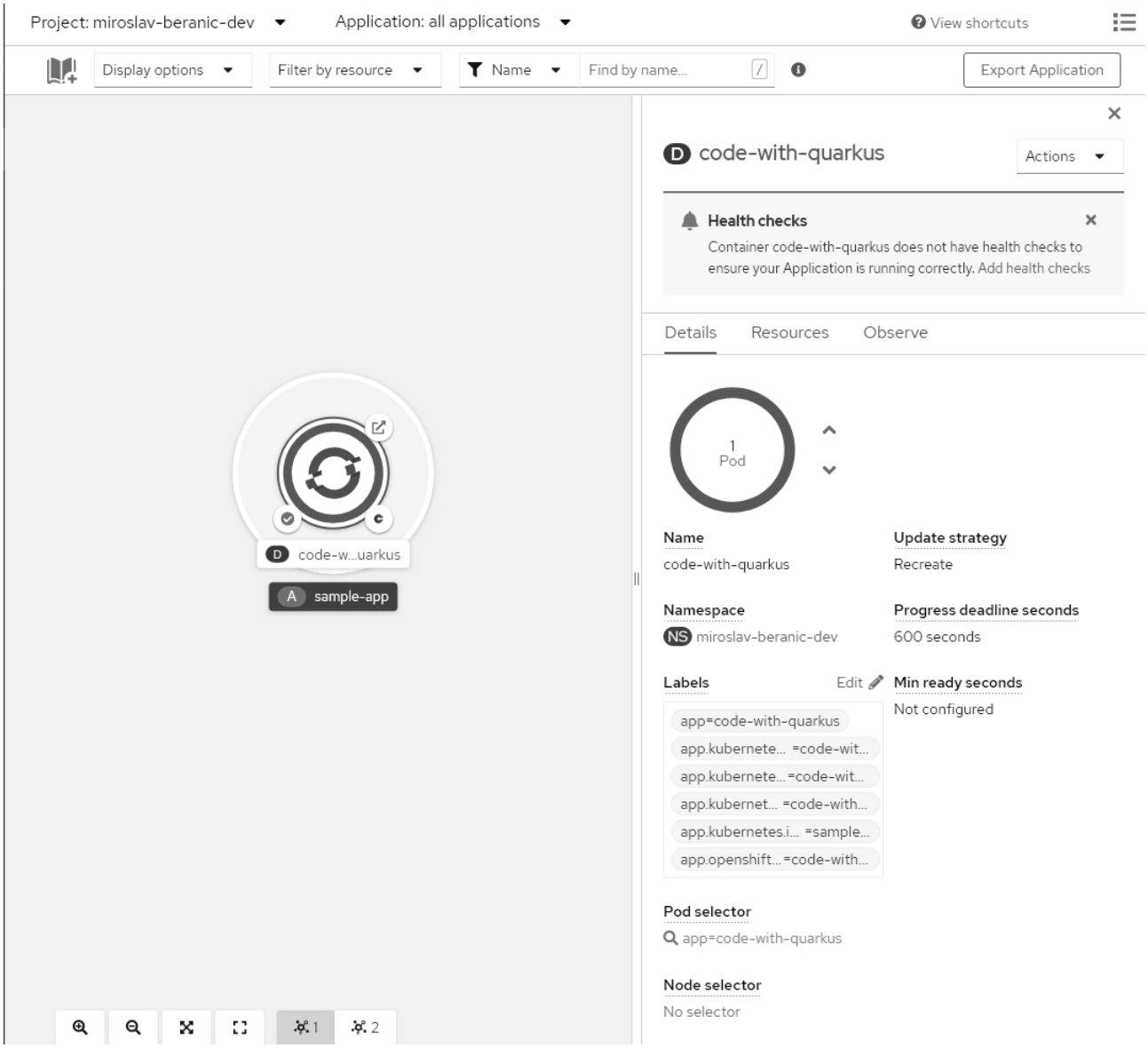
5.1. Hello world

Za začetek je vedno primerno narediti »Hello world« primer, da vidimo osnove, kako sistem deluje. Na desni strani izberemo »+Add«, nato pod »Getting started resources > Create applications using samples « izberemo »Basic Quarkus«. Prikaže se izborno okno, ki ga potrdilo z izborom »Create«.



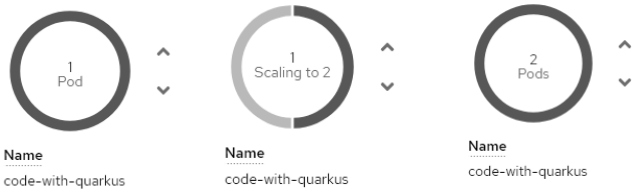
Slika 1: Red Hat dodajanje aplikacije preko Git povezave.
Vir: lasten.

V prikazanem oknu »Topology« se prikaže krogec, ki prikazuje ustvarjeno aplikacijo. Potrebno je počakati, da se kreirajo vsi viri, da se iz izvorne kode ustvari aplikacija in da se požene. Po parih minutah se prikaz posodobi in prikaže sledeče:



Slika 2: Red Hat Topology.
Vir: lasten.

Tukaj je mogoče začeti uporabljati sadeže naše postavljene infrastrukture. Lahko povečamo število instanc s katerimi delamo in s tem povečamo propustnost oz. omogočimo, da je storitev dostopna številčnejšim uporabnikom. Povečamo replikacije (mikro) storitve s tem, da povečamo število Pod-ov. To naredimo enostavno z izborom puščice gor, desno od kroga v katerem je izpisano trenutno aktivnih instanc storitve.



Slika 3: Red Hat OpenShift prehod med številom Pod-ov
Vir: lasten.

5.2. Zagon obstoječega kontejnerja aplikacije

Liferay DXP omogoča zagon aplikacije kot kontejnerizirana aplikacija. To je tudi v splošnem prvi korak, s katerim lahko obstoječe aplikacije prenesemo iz obstoječe infrastrukture in novo OpenShift infrastrukturo. To ni vedno optimalna rešitev, a omogoča prenos obstoječih aplikacij, brez sprememb ali z zelo malo spremembami. Na tak način omogočimo nadaljnji razvoj na aplikacijah in uporabo vseh funkcionalnosti, ki jih nudi nova infrastruktura.

Pri postavitvi Liferay aplikacije, si bomo pomagali z obstoječimi orodji in razširitvami, ki jih že omogoča Liferay za zagon oz. izvajanje v kontejnerski obliki. Za lažjo postavitev v OpenShift pa bomo dodali podporo za Helm chart-e, to so namenske opisne datoteke in infrastruktura, ki omogoča opisati aplikacijske zahteve kot podloga za nameščanje različnih instanc.

```
$ JAVA_HOME=/opt/jdk-8 ./gradlew clean deploy

$ JAVA_HOME=/opt/jdk-8 ./gradlew createDockerFile

$ podman build -t default-route-openshift-image-registry.apps-
crc.testing/ots2022/liferay-portal-7.4-ga35:1.0.0 .

STEP 1/10: FROM liferay/portal:7.4.3.35-ga35

STEP 2/10: ENV LIFERAY_WORKSPACE_ENVIRONMENT=local

--> Using cache 19f7fdcfdae63f14d277c5517ca796efa8c5d1784b77bb6191d52f8fa0de1e0c

--> 19f7fdcfdae

STEP 3/10: COPY --chown=liferay:liferay deploy /mnt/liferay/deploy

--> 361343206a3

STEP 4/10: COPY --chown=liferay:liferay patching /mnt/liferay/patching

--> a0bcfdf0e11

STEP 5/10: COPY --chown=liferay:liferay scripts /mnt/liferay/scripts

--> 632ece4bfd2

STEP 6/10: COPY --chown=liferay:liferay configs /home/liferay/configs

--> c3053bf3ce8

STEP 7/10: COPY --chown=liferay:liferay 100_liferay_image_setup.sh
/usr/local/liferay/scripts/pre-configure/100_liferay_image_setup.sh

--> a166be6a8b7

STEP 8/10: USER root

--> 62dbfc24f3e

STEP 9/10: RUN chgrp -R 0 /opt/liferay && chmod -R g=u /opt/liferay

--> 6c301d81b28

STEP 10/10: RUN chgrp -R 0 /mnt/liferay && chmod -R g=u /mnt/liferay

COMMIT default-route-openshift-image-registry.apps-crc.testing/ots2022/liferay-portal-
7.4-ga35:1.0.0

--> cd720f6c91f

Successfully tagged default-route-openshift-image-registry.apps-
crc.testing/ots2022/liferay-portal-7.4-ga35:1.0.0

cd720f6c91f5548da25992b1d45dccb3847c3e88f7d1d5318c81a3513eb0a093


$ podman push default-route-openshift-image-registry.apps-crc.testing/ots2022/liferay-
portal-7.4-ga35:1.0.0

Getting image source signatures

Copying blob 75e706e0ae82 done
```

```
Copying blob 7c2b733ed8ca done
Copying config cd720f6c91 done
Writing manifest to image destination
Storing signatures
```

Če bi želel vsako storitev nameščati ločeno, bi se to naredilo nas sledeč način:

```
$ oc apply -f database-deployment.yaml -n=ots2022
service/database created
persistentvolumeclaim/database-data created
deployment.apps/database created
```

Helm uporabljamo z namenom, da se naredi predloga, ki hrani vse odvisnosti in zahteve ter tako omogoča na enostaven način namestitev večih med seboj odvisnih storitev. Kot argument se poda datoteka z vrednostmi, ki zapolnijo prostornike v predlogi. To se naredi na sledeč način:

```
$ helm install liferay-chart . -f values.yaml
NAME: liferay-chart
LAST DEPLOYED: Mon Aug  1 18:26:29 2022
NAMESPACE: ots2022
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

Spletna aplikacija je na voljo na spletnem naslovu:
http://liferay-dxp-ots2022.apps-crc.testing

6 Zaključek

V prispevku smo predstavili infrastrukturo IT sistema, ki omogoča hitro in učinkovito razširitev za nove IT projekte ter hkrati možnost uporabe obstoječih IT rešitev na novi infrastrukturi. Predstavljen je Red Hat OpenShift v povezavi z Liferay DXP in mikrostoritvami na osnovi ogrodja Quarkus.

Literatura

[1] https://developers.redhat.com/developer-sandbox, OpenShift sandbox, dostopano 10. 8. 2022.

[2] https://docs.openshift.com/container-platform, Welcome | About | OpenShift Container Platform 4.10, dostopano 10.08.2022

[3] https://developers.redhat.com/products/openshift-local/overview, Red Hat OpenShift Local Overview | Red Hat Developer, dostopano 10.08.2022

[4] https://access.redhat.com/documentation/en-us/red_hat_openshift_local/2.5, Product Documentation for Red Hat OpenShift Local 2.5 | Red Hat Customer Portal, dostopano 10.08.2022

[5] https://docs.openshift.com/container-platform/4.10/cli_reference/openshift_cli/getting-started-cli.html, Getting started with the OpenShift CLI - OpenShift CLI (oc) | CLI tools | OpenShift Container Platform 4.10, dostopano 10.08.2022