



REPUBLIKA SLOVENIJA
MINISTRSTVO ZA DIGITALNO PREOBRAZBO



NAČRT ZA
OKREVANJE
IN ODPORNOST



Financira
Evropska unija
NextGenerationEU

Idejna zasnova in postavitve pilota ekosistema interneta stvari z algoritmičnimi orodji

Projekt za izvedbo

Naziv projekta	Idejna zasnova in postavitve pilota ekosistema interneta stvari z algoritmičnimi orodji
Status dokumenta	Končna verzija
Datum izdelave dokumenta	20. 9. 2023
Datum zadnje spremembe	5. 10. 2023
Verzija dokumenta	1.0
Avtor dokumenta	Marko Bajec, Jernej Cvek, Matjaž Pančur, Gregor Burger, Franc Drobnič, Andrej Kos



Zgodovina dokumenta

<i>Datum</i>	<i>verzija</i>	<i>avtor</i>
20. 9. 2023	0.1	Marko Bajec
5. 10. 2023	1.0	Marko Bajec

Kazalo vsebine

1	Uvod	4
1.1	Namen dokumenta	4
2	Zahteve	4
2.1	Zahteve naročnika	4
2.1.1	Funkcionalnosti platforme	4
2.1.2	Osrednji sestav - Splošne tehnične zahteve pilota	5
3	Predlagana arhitektura	7
3.1	Visokonivojski pogled	7
3.2	Opis gradnikov	9
3.2.1	Chirpstack	9
3.2.2	Context Broker	10
3.2.3	Grafana	11
3.2.4	IoT Agent Manager	12
3.2.5	IoT Agent	13
3.2.6	KeyCloak	14
3.2.7	Kong	15
3.2.8	MongoDB	17
3.2.9	Mosquitto	18
3.2.10	Quantum Leap	19
3.2.11	Redis	20
3.2.12	ThingsBoard	21
3.2.13	TimeScaleDB	23
4	Namestitev	24
4.1	Docker	24
4.1.1	Sistemske zahteve	24
4.1.2	Seznam vseh gradnikov	24
4.1.3	Konfiguracija omrežja	28
4.1.4	Postopek namestitve	29

1 Uvod

1.1 Namen dokumenta

V dokumentu je podan **opis** ter **načrt za postavitev** pilotnega sestava, ki nastaja v okviru projekta »*Idejna zasnova in postavitve pilota ekosistema interneta stvari z algoritmičnimi orodji*« (v nadaljevanju »platforma«). V strukturirani obliki so zbrani podatki o **arhitekturi**, vključno z visokonivojskim pogledom in vsebovanimi **gradniki** pilota. Podan je postopek namestitve Docker vsebnikov s pripadajočimi podatki o **sistemskih zahtevah** in **konfiguraciji omrežja**.

2 Zahteve

2.1 Zahteve naročnika

V tehnični specifikaciji naročila za vzpostavitev platforme je naročnik podal naslednje zahteve:

2.1.1 Funkcionalnosti platforme

Platforma je zasnovana na osnovi gradnikov fundacije FIWARE in omogoča interoperabilnosti med sorodnimi sestavi. Uporabo odprtokodnih digitalnih gradnikov FIWARE in standardov podpira tudi Evropska komisija.

Izpostavljena sta predvsem dva gradnika:

- Standardizirani informacijski model in aplikacijski programski vmesnik NGSI-LD (ali vsaj NGSI-V2).
- CEF digitalni gradnik Context Broker. Uporaba odprtokodnega digitalnega gradnika je brezplačna, zagotovljeno je tudi dolgoročno vzdrževanje ter podpora pri uvajanju s strani Evropske komisije oz. CEF.

Glavne funkcionalnosti platforme, ki jih zasledujemo:

- podpora za prenos podatkov z in v naprave IoT preko različnih standardnih protokolov,
- možnost naročanja na podatke glede na njihov kontekst,
- shranjevanje podatkov časovnih vrst,
- upravljanje z napravami IoT, uporabniki in njihovimi dostopi,
- vizualizacija podatkov v obliki nadzornih plošč,
- varnost na vseh slojih.

2.1.1.1 Funkcionalne zahteve v okviru pilota za uporabnika DSP - Prvi uporabnik

Pilotni sestav mora omogočati spremljanje porabe energije v stavbah, tako agregirane kot tudi po posameznih porabnikih, zaradi zahtevanega energetskega knjigovodstva in spremljanja učinkovitosti rabe energije. Zagotovljeno mora biti:

- zbiranje podatkov o meritvah porabe električne energije,
- zbiranje podatkov o meritvah porabe toplotne energije,
- zbiranje podatkov o meritvah porabe plina,

- zbiranje podatkov o meritvah porabe vode.

2.1.1.2 Funkcionalne zahteve izven okvira pilotnega projekta

Pilotni projekt mora biti zasnovan tako, da bodo v prihodnosti mogoče njegove nadgradnje in razširitve. V nadaljevanju in izven okvira pilotnega projekta se predvideva vzpostavitev energetskega vodenja stavb z namenom energetske-ekonomske optimizacije procesov, ki so v stavbah povezani s porabo energije. Pilotni projekt mora biti zasnovan na način, da bo za potrebe energetskega vodenja zgradb omogočal dvosmerni podatkovni prenos med osrednjim sestavom in končnimi napravami interneta stvari v zgradbi (senzorji, števcji, ventili, aktuatorji itd.). Osrednji sestav mora omogočati povezavo z ločenim programskim gradnikom, ki bo razvit izven okvira projekta in bo udeležan krmilno-regulacijske funkcionalnosti, ki so potrebne za energetske vodenje zgradb.

2.1.1.3 Predvideni uporabniki pilota

Predvidena uporabnika pilota sta:

- MDP / Direktorat za informatiko in
- MJU / Direktorat za stvarno premoženje.

2.1.2 Osrednji sestav - Splošne tehnične zahteve pilota

2.1.2.1 Skladnost s standardi

Z namenom zagotavljanja čim večje splošnosti, standardizacije, interoperabilnosti in odpornosti na izzive, ki jih prinaša prihodnost, je zahtevana skladnost s specifikacijami NGSI-LD (ali vsaj NGSI-v2), t.j. standardiziranim informacijskim modelom in aplikacijskim programskim vmesnikom za objavljane, poizvedovanje in naročanje na informacije o kontekstu, katerega namen je olajšati odprto izmenjavo in souporabo strukturiranih informacij med različnimi zainteresiranimi stranmi.

V okviru pilotnega projekta mora biti na vseh ravneh omogočeno upravljanje celotnega življenjskega cikla informacij o kontekstu, vključno s posodobitvami, poizvedbami, registracijami in naročninami poenoteno in skladno s specifikacijami NGSI-LD (ali vsaj NGSI-v2).

2.1.2.2 Povezljivost s končnimi napravami interneta stvari - prenosni in podatkovni protokoli in smer podatkovnega toka

Osrednji sestav mora omogočati povezovanje s končnimi napravami interneta stvari po naslednjih prenosnih in podatkovnih protokolih:

- HTTP - Ultralight,
- HTTP - JSON,
- MQTT - Ultralight,
- MQTT - JSON,
- LoRaWAN - Cayenne LPP,
- LoRaWAN - CBOR,
- LoRaWAN - možnost določitve dekodirske funkcije za lastniški podatkovni format.

Osrednji sestav mora omogočati razširitve z dodatnimi prenosnimi in podatkovnimi protokoli ter povezovanje končnih naprav interneta stvari (to so viri podatkov (npr. senzorji), ponori podatkov (npr. aktuatorji) in naprav, ki so tako viri kot tudi ponori podatkov (npr. v zgradbah nameščeni koncentratorji)). Zagotavljati mora dvosmeren podatkovni tok.

2.1.2.3 Varnostne zahteve

Osrednji sestav mora biti navzven zavarovan z uporabo mehanizmov avtentikacije in granularne avtorizacije ter z uporabo kriptografskih postopkov pri prenosu podatkov navzven, kar omogoča zavarovanje na vseh ravneh. Tudi komunikacija med osrednjim sestavom in končnimi napravami interneta stvari mora biti, razen v izjemnih primerih, ko je to tehnično neizvedljivo, zavarovana z uporabo mehanizmov avtentikacije in avtorizacije ter z uporabo kriptografskih postopkov pri prenosu podatkov.

2.1.2.4 Skalabilnost

Sestav mora biti zasnovan na način, ki z naraščajočo obremenitvijo sestava omogoča strojno in programsko skalabilnost.

2.1.2.5 Uporabniki

Sestav mora omogočati več organizacijsko, več uporabniško in več najemniško delovanje.

2.1.2.6 Obdelava osebnih podatkov

Sestav ne obdeluje osebnih podatkov.

2.1.2.7 Pravni vidiki

Potreben je pregled in potrditev ustreznosti licenčnih pogojev vseh posameznih gradnikov sestava za uporabo v okolju državne uprave.

2.1.2.8 Finančni vidiki

Potrebna je analiza finančnih učinkov sestava za uporabo v okolju državne uprave.

3 Predlagana arhitektura

3.1 Visokonivojski pogled

V tem poglavju je predstavljen visokonivojski pogled na arhitekturo platforme.

Za opazovanje in upravljanje stvari iz realnega sveta se uporabljajo različne naprave (senzorji in aktuatorji). Naloga prehodov na robu je, da iz množice senzorjev in z njimi povezanih protokolov pridemo do obvladljivega načina prenosa podatkov v platformo. Predvidena je uporaba TCP/IP prehodov in LoRa prehodov z nameščeno ustrezno programsko opremo. V primeru LoRa prehodov se podatki v platformo prenesejo po protokolu UDP do Chirpstack Gateway Bridge-a, od tam naprej pa po MQTT do Mosquitta, na katerega je povezan tudi Orion-LD Context Broker.

Prenos podatkov med preходом na robu in vstopno točko platforme IoT poteka po zaščiteni povezavi; za TCP/IP povezave se uporablja protokol TLS => HTTPS, MQTTS.

Ob vstopu v platformo IoT zahtevke (sporočila) najprej prestreže API prehod Kong, ki poskrbi za njihovo ustrezno preusmerjanje (angl. routing) in omejevanje (angl. rate limiting), po potrebi tudi za beleženje dostopov (angl. access logging). V primeru komunikacije preko protokola HTTP izvede tudi odstranitev zaščitene povezave ("TLS termination") in preverjanje pristnosti.

Kong sporočila glede na aplikacijski omrežni protokol posreduje v zaledni del:

- HTTP sporočila posreduje direktno agentom IoT,
- MQTT sporočila posreduje posredniku sporočil Mosquitto.

Posrednik sporočil MQTT Mosquitto poskrbi za preverjanje pristnosti. Nanj so preko nezaščitene MQTT povezave povezani tudi Orion-LD Context Broker, Chirpstack in agenti IoT.

Agenti IoT so zadolženi za (1) pretvorbo iz različnih formatov sporočil (npr. Ultralight 2.0) v standardni format NGSI in (2) pretvorbo iz različnih aplikacijskih omrežnih protokolov (npr. MQTT) v protokol HTTP.

Orion-LD Context Broker izpostavlja NGSI-LD API, preko katerega se nanj povežejo agenti IoT. Kong na tej točki med drugim poskrbi za nadzor dostopa.

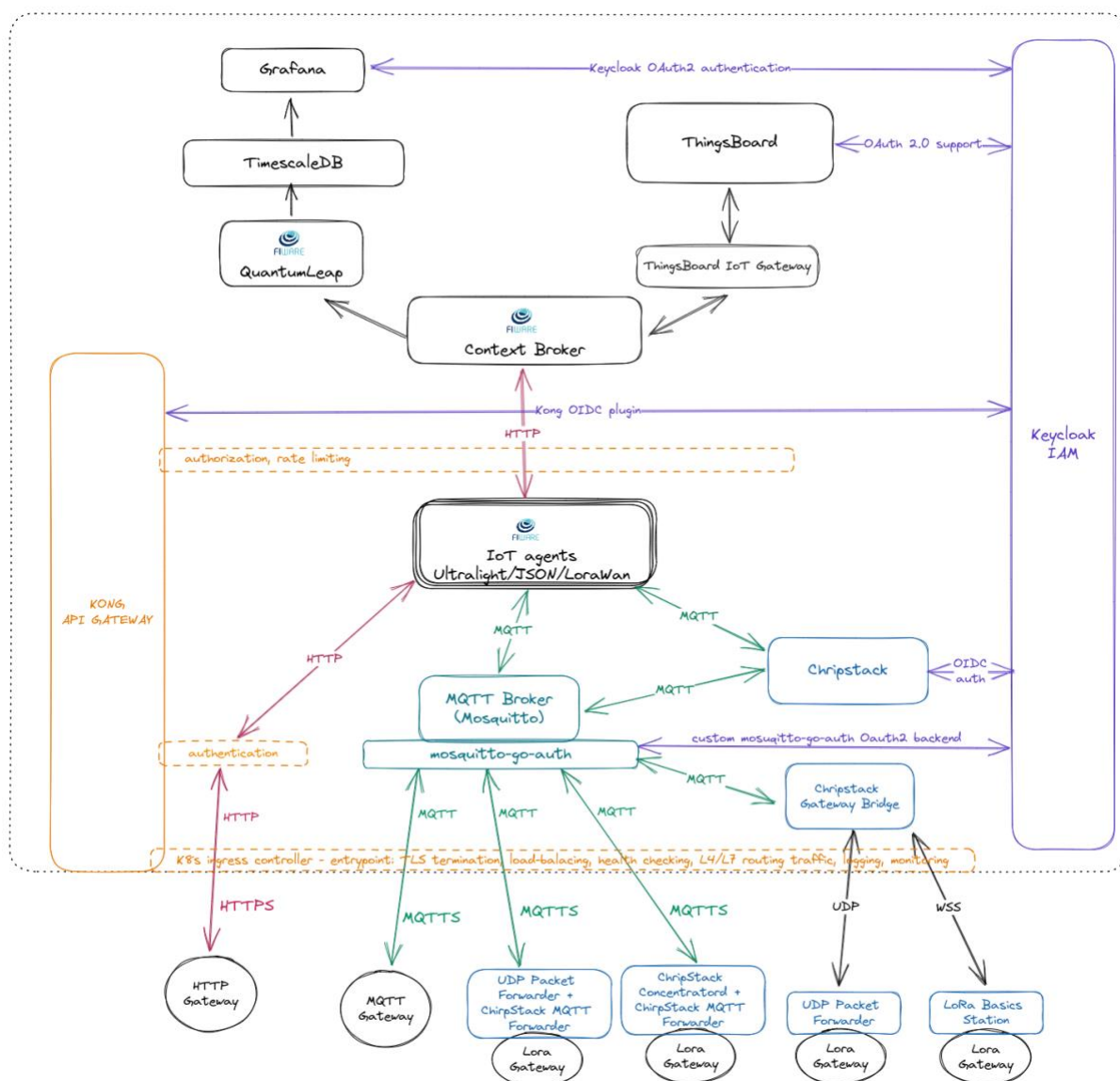
Za shranjevanje in vizualizacijo podatkov sta trenutno predvideni dve možnosti:

- QuantumLeap se naroči na podatke Context Broker-ja in jih shranjuje v podatkovno bazo TimescaleDB, ki je vir podatkov za vizualizacije v Grafani.
- S pomočjo programske opreme ThingsBoard IoT Gateway se vzpostavi povezava med Context Broker-jem in ThingsBoard-om, ki omogoča gradnjo bogatih nadzornih plošč.

Za upravljanje identitet in dostopov je v celotnem skladu predviden Keycloak, na katerega se s pomočjo integracij povežejo ostali gradniki.

Večina gradnikov, ki za svoje delovanje potrebuje shrambo podatkov, uporablja podatkovno bazo PostgreSQL. Podatkovna baza ni nujno del platforme, temveč je lahko zunanja storitev (DBaaS, ipd.).

Na spodnji sliki so prikazani ključni gradniki platforme.



Slika 1: Visokonivojska arhitektura z gradniki

3.2 Opis gradnikov

3.2.1 Chirpstack

Chirpstack je odprtokodni omrežni strežnik, ki služi za vzpostavitev omrežij po protokolu LoRaWAN. Z uporabo spletnega vmesnika je možno upravljati prehode, naprave in najemnike in tudi vzpostaviti povezave s ponudniki oblačnih storitev, podatkovnimi bazami in drugimi storitvami za obdelavo podatkov iz naprav.

Chirpstack je sestavljen iz treh osnovnih gradnikov:

- Network Server
- Application Server
- Gateway Bridge

Dodatne informacije

Gradnik	Github	Dokumentacija	Docker slika	Verzija
Chirpstack Network Server	repo	docs	chirpstack/chirpstack-network-server	3
Chirpstack Application Server	repo	docs	chirpstack/chirpstack-application-server	3
Chirpstack Gateway Bridge	repo	docs	chirpstack/chirpstack-gateway-bridge	3
Postgres DB 9.x (za Chirpstack)		docs	postgres	9.6-alpine

OPOMBA

Zaradi združljivosti s starejšimi storitvami predlagamo uporabo različice v3. Na voljo je že različica v4, vendar bi bilo treba njeno delovanje preveriti v ciljnem okolju.

3.2.2 Context Broker

Context Broker Orion je vzorčna implementacija Context Brokerja za upravljanje s podatki konteksta (Context Data Management). Je eden izmed gradnikov CEF Evropske komisije. Podpira API NGSI-LD v skladu s specifikacijami ETSI in tudi deloma NGSIv2.

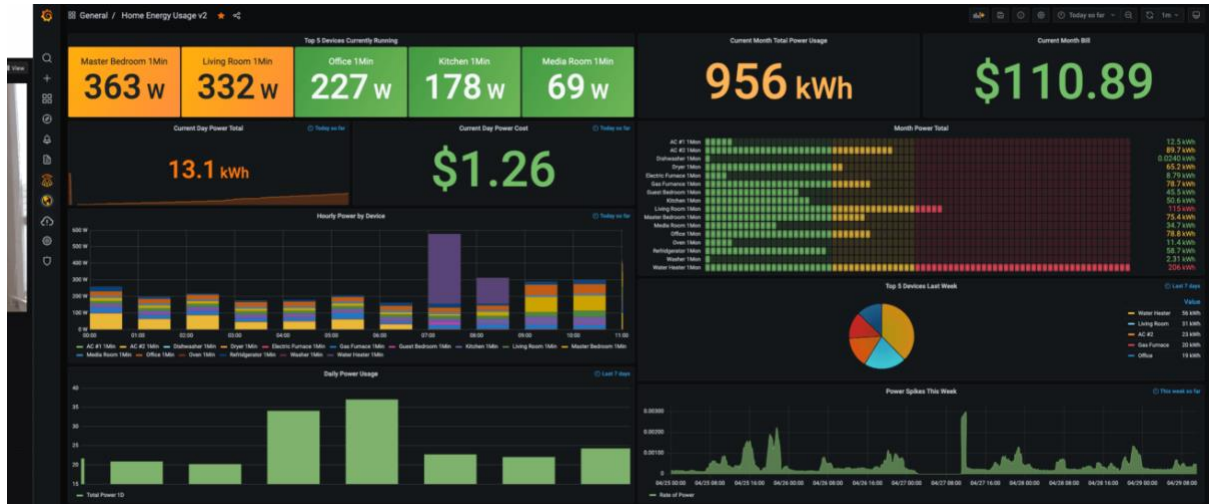
Predlagamo uporabo različice Orion-LD, ki jo ponuja FIWARE. Vsi ostali gradniki bodo ustrezno nastavljeni za uporabo protokola NGSI-LD.

Dodatne informacije

Gradnik	Github	Dokumentacija	Docker slika	Verzija
Orion-LD Context Broker	repo	docs	fiware/orion-lid	1.2.1

3.2.3 Grafana

Grafana je vodilna odprtokodna platforma za vizualizacijo in nadzorne plošče, ki omogoča poizvedovanje, vizualizacijo, opozarjanje in razumevanje podatkov ne glede na to, kje so shranjeni.



Dodatne informacije

Gradnik	Github	Dokumentacija	Docker slika	Verzija
Grafana	repo	docs	grafana/grafana	9.5.2

3.2.4 IoT Agent Manager

Gradnik **IoT Agent Manager** ni obvezen, ponuja zmožnost naslavljanja aktuatorjev po različnih protokolih in posreduje zahteve ustreznemu IoT Agentu glede na protokol.

Dodatne informacije

Gradnik	Github	Dokumentacija	Docker slika	Verzija
IoT Agent Manager	repo		telefonicaiot/iotagent-manager	2.1.0-distroless

3.2.5 IoT Agent

IoT Agenti pretvarjajo sporočila senzorskih naprav, ki jih te pošiljajo po različnih protokolih, v obliko NGSI-LD, ki je primerna za vnos v Context Broker. IoT Agenti omogočajo tudi varnostne ukrepe (avtentikacija in avtorizacija kanala).

Več različnih IoT Agentov lahko deluje hkrati, morajo pa imeti nastavljena različna vrata na severni strani (v smeri proti Context Brokerju).

Vsi IoT Agenti podpirajo delovanje po specifikaciji NGSI-LD, kar dosežemo z nastavitvijo okoljske spremenljivke `IOTA_CB_NGSI_VERSION=ld`.

3.2.5.1 Ultralight

IoT Agent Ultralight podpira protokol Ultralight z uporabo HTTP/REST API-ja, ki zahteva zelo malo pasovne širine za prenos podatkov.

Vzorčna postavitev: <https://github.com/FIWARE/tutorials.IoT-Agent>

3.2.5.2 LoRa

IoT Agent LoRa podpira protokol LoRaWAN in lahko sprejema podatke v oblikah TheThingsNetwork v3 in Chirpstack.

3.2.5.3 OPC-UA

IoT Agent LoRa podpira protokol OPC UA (OPC Unified Architecture), ki je prenosljiva razširitev uveljavljenega protokola OPC.

Dodatne informacije

Gradnik	Github	Dokumentacija	Docker slika	Verzija
IoT Agent LoRa	repo	docs	ioeai/iotagent-lora	1.2.5
IoT Agent Ultralight	repo	docs	quay.io/fiware/iotagent-ul	2.0.0-distroless
IoT Agent OPC-UA	repo	docs	iotagent4fiware/iotagent-opcua	2.1.9
IoT Agent JSON	repo	docs	quay.io/fiware/iotagent-json	2.0.0-distroless

3.2.6 KeyCloak

Keycloak je odprtokodna rešitev za upravljanje identitet in dostopov (IAM), ki temelji na standardnih protokolih in zagotavlja podporo za OpenID Connect, OAuth 2.0 in SAML.

3.2.6.1 Osnovne funkcionalnosti

Keycloak ima naslednje osnovne funkcionalnosti:

- Upraviteljska konzola: Prek upraviteljske konzole lahko skrbniki centralno upravljajo vse vidike strežnika Keycloak.
- Konzola za upravljanje računov: V konzoli za upravljanje računov lahko uporabniki upravljajo svoje račune. Posodobijo lahko profil, spremenijo gesla in nastavijo dvofaktorsko preverjanje pristnosti.
- Storitve avtorizacije: Če avtorizacija na podlagi vlog (RBAC) ne pokriva potreb, ponuja Keycloak tudi storitve avtorizacije s fino zrnatostjo (npr. ABAC). Single-Sign On (SSO)
- Posredovanje identitete in prijava z družabnimi omrežji: Omogočanje prijave z družabnimi omrežji je enostavno dodati prek upraviteljske konzole.
- Federacija uporabnikov: Keycloak ima vgrajeno podporo za povezavo z obstoječimi strežniki LDAP ali Active Directory. Če se uporabniki shranjujejo v drugih shrambah, npr. v relacijski podatkovni zbirki, je možna tudi implementacija lastnega ponudnika.

3.2.6.2 Možnosti integracije

Kong

Na podlagi pregleda seznama [Kong vtičnikov](#) sta za integracijo s Keycloak-om najprimernejša vtičnika [OpenID Connect](#) in [OAuth 2.0 Introspection](#), ki pa sta oba na voljo le v plačljivi različici Kong-a.

Mosquitto

Za preverjanje pristnosti je na voljo vtičnik [Mosquitto Go Auth](#), ki ponuja več različnih možnosti avtentikacije. Za OAuth 2.0 sicer ni na voljo direktne integracije z rešitvami IAM, se pa lahko implementira vtičnik po meri. Za primer se lahko vzame <https://github.com/gewv-tu-dresden/mosquitto-go-auth-oauth2>.

Chirpstack

Chirpstack ponuja integracijo z OpenID ponudniki, med katere se šteje tudi Keycloak.

Grafana

Grafano je moč integrirati s Keycloak-om s pomočjo t.i. Keycloak OAuth2 avtentikacije. Primer je opisan [tukaj](#).

ThingsBoard

Thingsboard ponuja podporo za integracijo na podlagi [protokola OAuth 2.0](#).

Dodatne informacije

Gradnik	Github	Dokumentacija	Docker slika	Verzija
Keycloak	repo	docs	bitnami/keycloak	22.0.1

3.2.7 Kong

Kong ali **Kong API Gateway** je "cloud-native", platformno agnostični, skalabilni prehod API, ki ga odlikujeta visoka zmogljivost in razširljivost s pomočjo vtičnikov.

Med drugim podpira sledeče funkcionalnosti:

- preverjanje pristnosti in nadzor dostopa,
- napredne možnosti usmerjanja (angl. "routing"), uravnoteženja obremenitve (angl. "load balacing"),
- preoblikovanje oblike vsebine zahtevkov in odgovorov na zahteve (angl. "request/response transformations"),
- terminacija TLS,
- podpora za različne protokole na prenosnem (L4) in aplikacijskem (L7) omrežnem sloju.

Kong deluje v sistemu Kubernetes zahvaljujoč uradnemu kontrolniku *Kubernetes Ingress Controller*.

3.2.7.1 Različice Kong-a

Kong se lahko namesti na dva načina:

- (1) v oblaku s storitvijo Kong Konnect in
- (2) lokalno (angl. "on-premises").

Na voljo so naslednje različice Kong-a:

KONG OSS (open source)

Na voljo so zgolj nekatere funkcionalnosti in odprtokodni vtičniki.
Upravljanje je možno zgolj preko API-ja.

Kong FREE

Vse iz Kong OSS.

Dodatno: upravljanje s pomočjo uporabniškega vmesnika upravljalne konzole Kong Manager

Kong Premium

\$250 / storitev, na voljo zgolj za postavitve v oblaku (Kong Konnect)

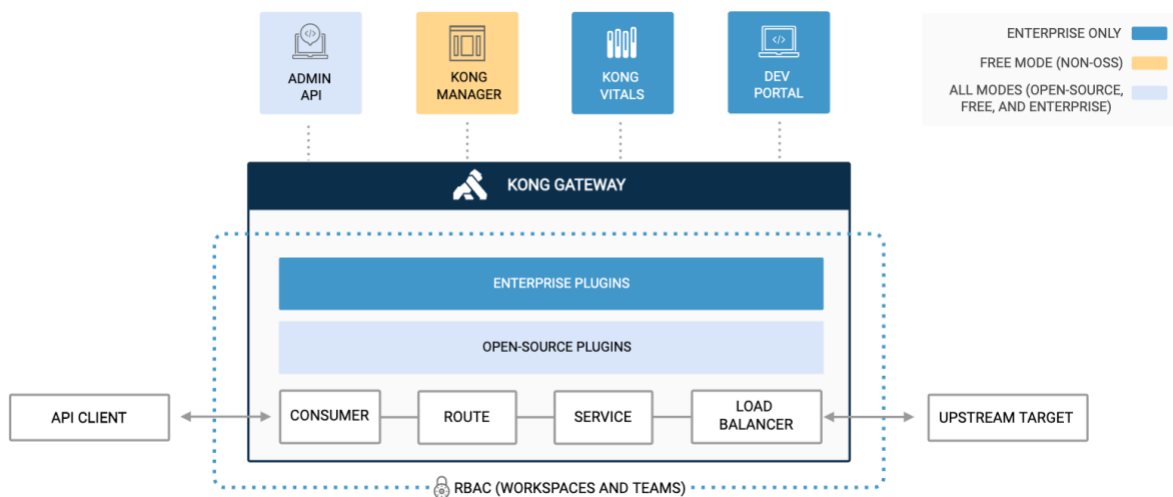
Dodatne funkcionalnosti in plačljivi vtičniki.

Kong Enterprise

Še več dodatnih funkcionalnosti:

- Kong Dev Portal se uporablja za uvajanje novih razvijalcev in ustvarjanje dokumentacije API, ustvarjanje strani po meri, upravljanje različic API-ja in varen dostop razvijalcev.
- Kong Vitals zagotavlja uporabne metrike o stanju in zmogljivosti vozlišč prehoda Kong ter metrike o uporabi API-jev.
- RBAC model nadzora dostopa za uporabnike, ki upravljajo Kong,

- Še več plačljivih vtičnikov.



Dodatne informacije

Gradnik	Github	Dokumentacija	Docker slika	Verzija
Kong Gateway	repo	docs	kong/kong-gateway	3.3.1.0

3.2.8 MongoDB

MongoDB je odprtokodna NoSQL baza, ki shranjuje podatke v obliki JSON.

Uporabljajo jo različni gradniki za shranjevanje dinamičnih nastavitev, sezname registriranih naprav ipd.

Dodatne informacije

Gradnik	Github	Dokumentacija	Docker slika	Verzija
MongoDB	repo	docs	mongo	4.4

3.2.9 Mosquitto

Mosquitto je implementacija MQTT brokerja, ki jo razvija Eclipse Foundation in podpira verzije MQTT 5, 3.1.1 in 3.1.

Uporabljajo ga različni IoT Agenti kot začasno shrambo za podatke naprav (meritve senzorjev ipd.) in druge metapodatke.

Dodatne informacije

Gradnik	Github	Dokumentacija	Docker slika	Verzija
Mosquitto	repo	docs	eclipse-mosquitto	1.6.14

3.2.10 Quantum Leap

QuantumLeap je storitev REST za shranjevanje, poizvedovanje in iskanje prostorsko-časovnih podatkov NGSI v2 in NGSI-LD (eksperimentalna podpora).

Med drugim podpira sledeče funkcionalnosti:

- QuantumLeap pretvori polstrukturirane podatke NGSI v tabelarno obliko in jih shrani v podatkovno zbirko časovnih vrst, pri čemer vsak zapis podatkovne zbirke poveže s časovnim indeksom in, če je podatek prisoten, z lokacijo na Zemlji. Odjemalci lahko nato pridobijo entitete NGSI s filtriranjem nizov entitet prek časovnih razponov in prostorskih operatorjev.
- Funkcionalnost poizvedb, ki je na voljo prek vmesnika REST, je precej osnovna, za bolj zapletene poizvedbe morajo odjemalci običajno neposredno dostopati do podatkovne zbirke.
- Za shranjevanje podpira podatkovne zbirke CrateDB in TimescaleDB.

Dodatne informacije

Gradnik	Github	Dokumentacija	Docker slika	Verzija
QuantumLeap	repo	docs	orchestracities/quantumleap	edge ¹

1

orchestracities/quantumleap@sha256:cfec3ebfba1f091ec541c67102581d0f0f315318924fcb6429030930ccf7119

3.2.11 Redis

Redis je odprtokodna shramba parov ključ - vrednost.

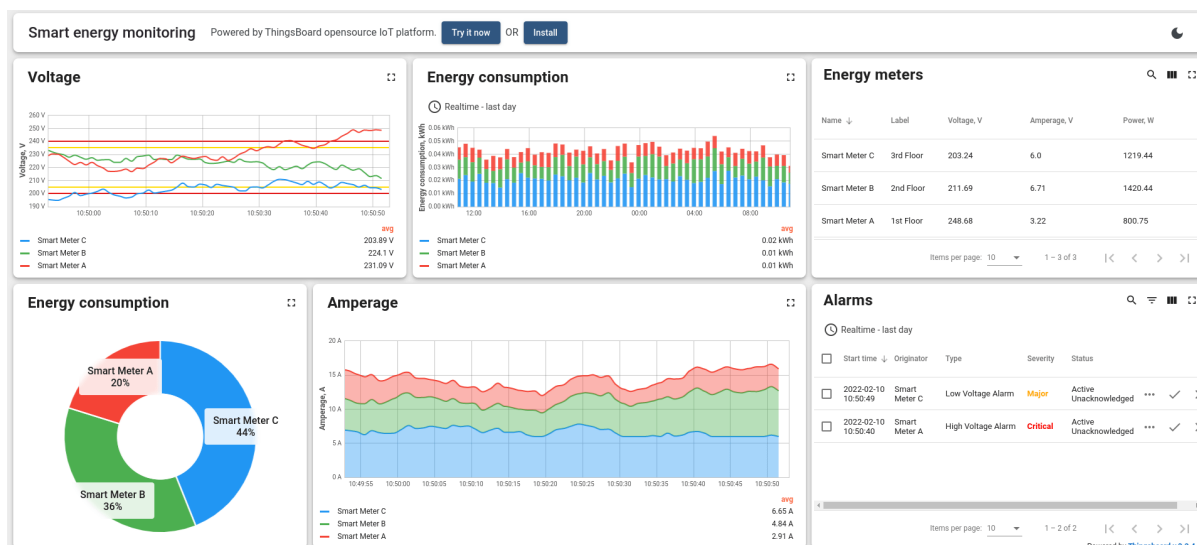
Uporabljen je v več gradnikih kot začasna shramba v pomnilniku, medpomnilnik ali sporočilna vrsta.

Dodatne informacije

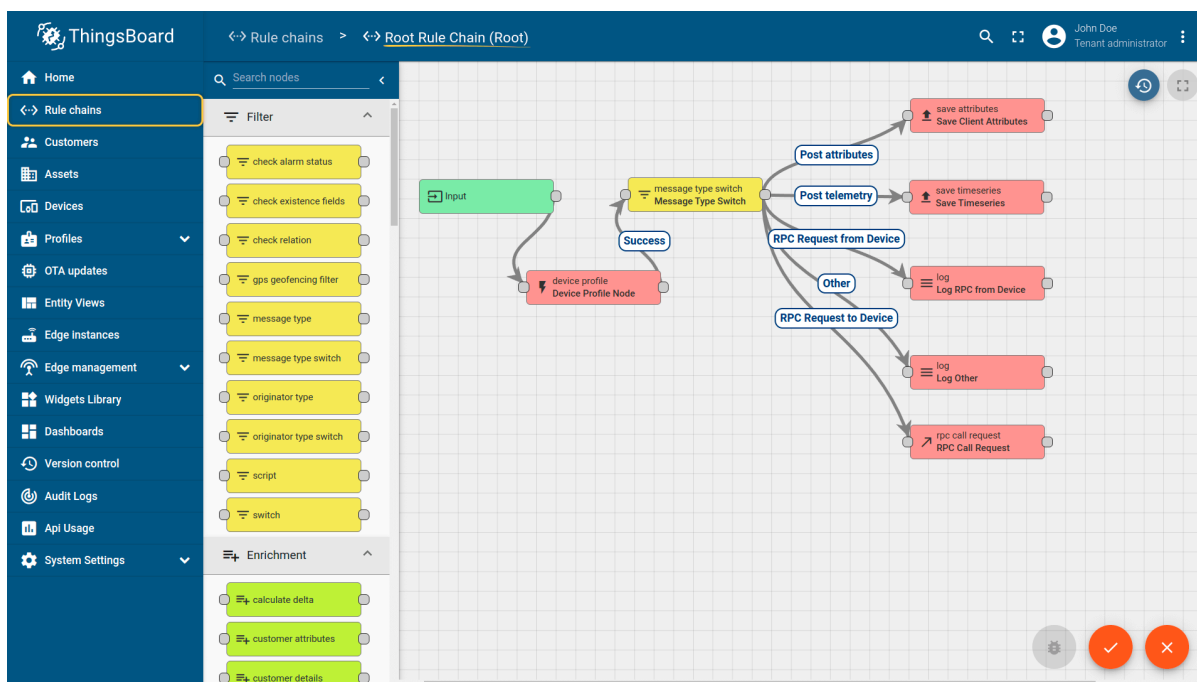
Gradnik	Github	Dokumentacija	Docker slika	Verzija
Redis	repo	docs	bitnami/redis	6.2.13

3.2.12 ThingsBoard

ThingsBoard je odprtokodna platforma IoT za zbiranje, obdelavo in vizualizacijo podatkov ter upravljanje naprav interneta stvari. Več kot 30 prilagodljivih gradnikov omogoča izdelavo bogatih nadzornih plošč po meri končnega uporabnika za večino primerov uporabe interneta stvari.



Z uporabo t.i. "Rule Engine" je možno tudi ustvarjanje zapletenih verig pravil (angl. "rule chain") za obdelavo podatkov ter prilagajanje specifičnim primerom njihove uporabe.



Dodatne informacije

Gradnik	Github	Dokumentacija	Docker slika	Verzija
ThingsBoard	repo	docs	thingsboard/tb-postgres	3.5.1
ThingsBoard IoT Gateway	po meri	docs	thingsboard/tb-gateway	3.3

3.2.13 TimeScaleDB

TimescaleDB je odprtokodna podatkovna zbirka časovnih vrst, zasnovana za hiter vnos, zapletene poizvedbe in preprosto uporabo. Navzven je videti kot PostgreSQL (pravzaprav je pakirana kot razširitev), kar pomeni, da je podedovala zanesljivost, orodja in obsežen ekosistem PostgreSQL-a.

V primerjavi s PostgreSQL ima TimescaleDB za časovne vrste:

- 20-krat večje število vstavitev (konstantno tudi pri milijardah vrstic),
- od 1,2-krat do več kot 14.000-krat hitrejša poizvedba,
- 2000-krat hitrejša brisanje, kar je ključnega pomena za izvajanje politik hrambe podatkov,
- nove časovno usmerjene funkcije, ki še olajšajo manipulacijo časovnih vrst v jeziku SQL.

Dodatne informacije

Gradnik	Github	Dokumentacija	Docker slika	Verzija
TimescaleDB	repo	docs	timescale/timescaledb-ha	pg15-ts2.10

4 Namestitev

4.1 Docker

4.1.1 Sistemske zahteve

Sistemske zahteve so odvisne od številnih parametrov, ki vplivajo na potrebo po računskih virih. Med ključne parametre spadajo:

1. **število in vrsta naprav**, ki bodo priključene oziroma bodo v komunikaciji s platformo,
2. **obseg podatkov**, ki se bodo izmenjevali s platformo,
3. intenzivnost komunikacije (frekvenca pošiljanja/prejemanja podatkov posameznih priključenih IoT naprav),
4. **arhitektura platforme**, to je komponente, ki jo sestavljajo ter odvisnosti med njimi - pomemben vidik, ki bo vplival na sistemske zahteve, je vezan tudi na K8 okolje, ki ga bo zagotovil naročnik,
5. **zahtevane nefunkcionalne zahteve platforme**, kot so na primer razpoložljivost, varnost, zanesljivost, propustnost ipd.

Minimalne sistemske zahteve bomo podali, ko bodo znani naštetih parametri.

4.1.2 Seznam vseh gradnikov

Za boljši pregled podajamo vse gradnike v eni skupni tabeli.

Gradnik	Github	Dokumentacija	Docker slika	Verzija
Chirpstack Network Server	repo	docs	chirpstack/chirpstack-network-server	3
Chirpstack Application Server	repo	docs	chirpstack/chirpstack-application-server	3
Chirpstack Gateway Bridge	repo	docs	chirpstack/chirpstack-gateway-bridge	3
Postgres DB 9.x (za Chirpstack)		docs	postgres	9.6-alpine
Postgres DB 9.x (za Kong)		docs	postgres	9.5
Grafana	repo	docs	grafana/grafana	9.5.2
IoT Agent Manager	repo		telefonicaiot/iotagent-manager	2.1.0-distoreless
IoT Agent LoRa	repo	docs	ioeair/iotagent-lora	1.2.5
IoT Agent Ultralight	repo	docs	quay.io/fiware/iotagent-ul	2.0.0-distoreless

Gradnik	Github	Dokumentacija	Docker slika	Verzija
IoT Agent OPC-UA	repo	docs	iotagent4fiware/iotagent-opcua	2.1.9
IoT Agent JSON	repo	docs	quay.io/fiware/iotagent-json	2.0.0-distoreless
Kong Gateway	repo	docs	kong/kong-gateway	3.3.1.0
Keycloak	repo	docs	bitnami/keycloak	22.0.1
MongoDB	repo	docs	mongo	4.4
Mosquitto	repo	docs	eclipse-mosquitto	1.6.14
Orion-LD Context Broker	repo	docs	fiware/orion-ld	1.2.1
QuantumLeap	repo	docs	orchestracities/quantumleap	edge ¹
Redis	repo	docs	bitnami/redis	6.2.13
ThingsBoard	repo	docs	thingsboard/tb-postgres	3.5.1
ThingsBoard IoT Gateway	po meri	docs	thingsboard/tb-gateway	3.3
TimescaleDB	repo	docs	timescale/timescaledb-ha	pg15- ts2.10

OPOMBA: V končni različici postavitve v Docker okolju se bo uporabila zgolj ena instanca (tj. en Docker vsebnik) podatkovne baze PostgreSQL z najvišjo verzijo, ki je še združljiva z vsemi gradniki. V tej instanci bodo vsebovane podatkovne zbirke vseh gradnikov, ki za svoje delovanje potrebujejo podatkovno bazo.

4.1.2.1 Seznam odvisnosti

Spodnja tabela prikazuje medsebojne odvisnosti gradnikov.

Gradnik	Odvisnosti
Chirpstack Network Server	Redis, Mosquitto, Chirpstack Application Server
Chirpstack Application Server	Postgres, Redis, Mosquitto
Chirpstack Gateway Bridge	Mosquitto
IoT agent LoRa	MongoDB, Orion-LD Context Broker
IoT agent Ultralight	MongoDB, Mosquitto, Orion-LD Context Broker
IoT agent OPC-UA	MongoDB, Orion-LD Context Broker
IoT agent JSON	MongoDB, Mosquitto, Orion-LD Context Broker
Kong Gateway	Postgres
Keycloak	Postgres
Orion-LD Context Broker	MongoDB
QuantumLeap	TimescaleDB, Redis

Gradnik	Ovisnosti
ThingsBoard IoT Gateway	ThingsBoard
ThingsBoard	Postgres (interno)

4.1.2.2 Seznam ranljivosti

V tem podpoglavju je navedeno število programskih ranljivosti za posamične gradnike.

Ranljivosti se lahko pojavijo zaradi programskih napak, nepravilne konfiguracije, slabih varnostnih praks med razvojem ali drugih vzrokov. Programske ranljivosti so napake, pomanjkljivosti ali slabosti v programski opremi, ki omogočajo napadalcem, da jih izkoristijo in pridobijo nepooblaščen dostop, povzročijo izpad sistema, ukradejo podatke ali izvedejo druge škodljive dejavnosti.

Opomba: Za odkrivanje ranljivosti se uporablja odprtokodni varnostni skener [Trivy](#).

Docker slika	Ranljivosti
chirpstack/chirpstack-network-server:3	Total: 32 (UNKNOWN: 0, LOW: 0, MEDIUM: 14, HIGH: 18, CRITICAL: 0)
chirpstack/chirpstack-application-server:3	Total: 32 (UNKNOWN: 0, LOW: 0, MEDIUM: 14, HIGH: 18, CRITICAL: 0)
chirpstack/chirpstack-gateway-bridge:3	Total: 4 (UNKNOWN: 0, LOW: 0, MEDIUM: 0, HIGH: 4, CRITICAL: 0)
postgres:9.6-alpine	Total: 44 (UNKNOWN: 0, LOW: 0, MEDIUM: 13, HIGH: 29, CRITICAL: 2) CVE-2022-37434 CVE-2022-37434
grafana/grafana:9.5.2	Total: 19 (UNKNOWN: 0, LOW: 2, MEDIUM: 10, HIGH: 7, CRITICAL: 0)
ioeairi/iotagent-lora:1.2.5	Total: 77 (UNKNOWN: 1, LOW: 34, MEDIUM: 11, HIGH: 28, CRITICAL: 3) Total: 46 (UNKNOWN: 0, LOW: 2, MEDIUM: 18, HIGH: 21, CRITICAL: 5) CVE-2022-1664 CVE-2019-12900 CVE-2019-8457 CVE-2021-3807 CVE-2021-44906 CVE-2021-44906 CVE-2023-3696 CVE-2021-28918
quay.io/fiware/iotagent-ul:2.0.0-distoreless	Total: 23 (UNKNOWN: 0, LOW: 11, MEDIUM: 8, HIGH: 4, CRITICAL: 0) Total: 9 (UNKNOWN: 0, LOW: 0, MEDIUM: 5, HIGH: 2, CRITICAL: 2) CVE-2023-3696 CVE-2022-0686
iotagent4fiware/iotagent-opcua:2.1.9	Total: 125 (UNKNOWN: 1, LOW: 89, MEDIUM: 10, HIGH: 24, CRITICAL: 1) Total: 4 (UNKNOWN: 0, LOW: 0, MEDIUM: 2, HIGH: 1, CRITICAL: 1) CVE-2019-8457 CVE-2023-3696
quay.io/fiware/iotagent-json:2.0.0-distoreless	Total: 23 (UNKNOWN: 0, LOW: 11, MEDIUM: 8, HIGH: 4, CRITICAL: 0)

Docker slika	Ranljivosti
	Total: 9 (UNKNOWN: 0, LOW: 0, MEDIUM: 5, HIGH: 2, CRITICAL: 2) CVE-2023-3696 CVE-2022-0686
kong/kong-gateway:3.3.1.0	Total: 107 (UNKNOWN: 0, LOW: 50, MEDIUM: 50, HIGH: 7, CRITICAL: 0)
bitnami/keycloak:22.0.1	Total: 128 (UNKNOWN: 0, LOW: 85, MEDIUM: 17, HIGH: 23, CRITICAL: 3) CVE-2023-23914 CVE-2023-23914 CVE-2019-8457
mongo:4.4	Total: 25 (UNKNOWN: 0, LOW: 25, MEDIUM: 0, HIGH: 0, CRITICAL: 0) Total: 4 (UNKNOWN: 0, LOW: 1, MEDIUM: 2, HIGH: 1, CRITICAL: 0)
eclipse-mosquitto:1.6.14	Total: 31 (UNKNOWN: 0, LOW: 0, MEDIUM: 2, HIGH: 25, CRITICAL: 4) CVE-2021-36159 CVE-2021-3711 CVE-2021-3711 CVE-2022-37434
fiware/orion-ld:1.2.1	Total: 381 (UNKNOWN: 0, LOW: 144, MEDIUM: 219, HIGH: 18, CRITICAL: 0)
orchestracities/quantumleap:edge ¹	Total: 74 (UNKNOWN: 0, LOW: 2, MEDIUM: 17, HIGH: 44, CRITICAL: 11) CVE-2021-36159 CVE-2022-22822 CVE-2022-22823 CVE-2022-22824 CVE-2022-23852 CVE-2022-25235 CVE-2022-25236 CVE-2022-25315 CVE-2021-3711 CVE-2021-3711 CVE-2022-37434 CVE-2022-29361
bitnami/redis:6.2.13	Total: 83 (UNKNOWN: 0, LOW: 68, MEDIUM: 4, HIGH: 10, CRITICAL: 1) CVE-2019-8457
thingsboard/tb-postgres:3.5.1	Total: 225 (UNKNOWN: 0, LOW: 151, MEDIUM: 36, HIGH: 37, CRITICAL: 1) Total: 29 (UNKNOWN: 0, LOW: 0, MEDIUM: 11, HIGH: 14, CRITICAL: 4) CVE-2019-8457 CVE-2023-20873 CVE-2023-34034 CVE-2023-20862 CVE-2016-1000027
thingsboard/tb-gateway:3.3	Total: 319 (UNKNOWN: 0, LOW: 227, MEDIUM: 42, HIGH: 43, CRITICAL: 7) CVE-2023-38426 CVE-2023-38427 CVE-2023-38428 CVE-2023-38429 CVE-2023-38430 CVE-2023-38431 CVE-2023-38432
timescale/timescaledb-ha:pg15-ts2.10	Total: 163 (UNKNOWN: 0, LOW: 52, MEDIUM: 111, HIGH: 0, CRITICAL: 0)

OPOMBA: Tekom razvoja se bodo gradniki posodabljali na najnovejše združljive verzije.

4.1.3 Konfiguracija omrežja

To poglavje vsebuje informacije o konfiguraciji omrežja, ki je potrebna za pravilno delovanje platforme.

4.1.3.1 Seznam omrežnih vrat

Gradnik	Vrata vsebnika	Vrata gostitelja
Chripstack Network Server	8000/tcp	
Chripstack Application Server	8080/tcp, 8081/tcp	
Chripstack Gateway Bridge	1700/udp	
Postgres DB 9.x (za Chripstack)	5432/tcp	
Postgres DB 9.x (za Kong)	5432/tcp	
Grafana	3000/tcp	
IoT agent LoRa	4041/tcp, 4061/tcp	
IoT agent Ultralight	4041/tcp, 4061/tcp, 7896/tcp	
IoT agent OPC-UA	4041/tcp, 9229/tcp	
IoT agent JSON	4041/tcp, 7896/tcp	
IoT manager	8002/tcp	
Kong Gateway	8000-8004/tcp, 8443-8447/tcp	80 (HTTP) ² , 443 (HTTPS), 1700 (UDP - Chripstack GB), 1883 (MQTT) ³
Keycloak	TBD	
MongoDB	27017/tcp	
Mosquitto	1883/tcp, 9001/tcp	
Orion-LD Context Broker	1026/tcp	
QuantumLeap	8668/tcp	
Redis	6379/tcp	
ThingsBoard	TBD	
ThingsBoard IoT Gateway	5000/tcp	
TimescaleDB	5432/tcp, 8008/tcp, 8081/tcp	

² Vrata 80 za HTTP se uporabljajo za Let's Encrypt HTTP-01 challenge. Ob uporabi DNS-01 challenge se lahko ta vrata zaprejo.

³ Vrata za MQTT se bodo kasneje spremenila iz 1883 (nekriptirana) v 8883 (kriptirana - MQTTS).

4.1.4 Postopek namestitve

To poglavje opisuje postopek začetne namestitve platforme na gostiteljski računalnik z uporabo Docker vsebnikov.

4.1.4.1 Predpogoji

Pred namestitvijo platforme je potrebno zagotoviti:

- gostiteljski računalnik z operacijskim sistem Ubuntu 20.04
- ustrezno nastavljen požarni zid
- javni IP naslov in domena z možnostjo urejanja DNS zapisov
- Docker 24.0.5
- docker-compose v2.20.3
- Python >= 3.6

4.1.4.2 Izvorna koda za namestitev platforme

Izvorna koda za namestitev platforme se nahaja v Git skladišču.

```
Git clone git@github.com:mabajec/iot-mju.git
```

4.1.4.3 Docker omrežje mju-iot

Vsebniki platforme so vključeni v skupno Docker omrežje **mju-iot** tipa **bridge**.

Pomembno:

V okviru Docker-ja premostitveno omrežje (angl. "bridge network") uporablja programski most (angl. "bridge"), ki omogoča komunikacijo med vsebniki, povezanimi z istim premostitvenim omrežjem, hkrati pa zagotavlja izolacijo od vsebnikov, ki niso povezani s tem premostitvenim omrežjem. Dockerjev premostitveni gonilnik (angl. "bridge driver") samodejno namesti ustrezna pravila v gostiteljski računalnik, tako da vsebniki v različnih premostitvenih omrežjih ne morejo neposredno komunicirati drug z drugim.

Ker gre za uporabniško definirano (angl. "user-defined") Docker omrežje, ga je potrebno najprej ustvariti:

```
docker network create --subnet 172.30.0.0/16 mju-iot
```

Parameter **--subnet** se v tem primeru uporablja za potrebe IP naslavljanja vsebnikov, ki potrebujejo fiksni IP naslov.

Opomba: Trenutno so zaradi poenostavitve vsi vsebniki vključeni v isto Docker omrežje. V produkcijski postavitvi je priporočljivo, da se vsebnike glede na njihove vloge razdeli v več različnih Docker omrežij.

4.1.4.4 Wildcard DNS zapis

DNS poskrbi za mapiranje domene v IP naslov gostiteljskega računalnika. Priporočljivo je uporabiti t.i. "wildcard" DNS zapis.

Tip DNS zapisa	Ime	IPv4 naslov
A	*	192.0.2.1


Zgornji primer bo preslikal domeno in vse njene poddomene (*) na podan IP naslov.

4.1.4.5 Nastavitve API prehoda Kong

Kong API gateway deluje kot vstopna točka v platformo, ki skladno z uporabniškimi pravicami zahteve preusmeri do ustreznih zalednih storitev.

Spodnje ukaze je potrebno izvesti v mapi kong-gateway.

Nastavitev okoljskih spremenljivk

 Pripraviti datoteko z okoljskimi spremenljivkami in urediti njeno vsebino.

```
cp -fr .env.prod .env
vim .env
```

Začetna vzpostavitev podatkovne baze

Pred prvim zagonom Kong-a je potrebno inicializirati podatkovno bazo.

```
# Zagon podatkovne baze
docker compose up -d kong-db
# Migracija podatkovne baze
docker compose --file kong-init-db.yml up && docker compose --file kong-
init-db.yml down
# Zagon Kong-a
docker compose up -d
```

Opomba: V produkciji je potrebno spremeniti dostopne podatke za podatkovno bazo ter jih varno shraniti.

Uveljavitev konfiguracije

S pomočjo skripte regenerate-kong-prod-paths.sh se prilagodi vsebina konfiguracijske datoteke na željeno domeno.

```
cd scripts
./regenerate-kong-prod-paths.sh domena.com
```

Z uporabo decK-a se iz posodobljene konfiguracijske datoteke deck/kong.prod.yml nastavitve uveljavijo v Kong podatkovni bazi.

Pomembno: deck pomaga upravljati konfiguracijo programa Kong na deklarativen način. To pomeni, da lahko upravljevec opredeli želeno stanje prehoda Kong Gateway (storitve, poti, vtičniki, itd.) nato pa deck poskrbi za izvajanje, brez da bi bilo potrebno vsak korak izvesti ročno, kot bi sicer to storili z vmesnikom API Kong Admin.

Najprej z ukazom ping preverimo, če je Kong dosegljiv deck-u:

```
# Ukaz mora vrniti 'Successfully connected to Kong!'
docker run --network mju-iot kong/deck --kong-addr http://kong:8333 ping
```

Nato s pomočjo t.i. suhega zagona z ukazom diff preverimo vsebino konfiguracije:

```
docker run -i \
-v $(pwd)/deck:/deck \
--network mju-iot \
kong/deck --kong-addr http://kong:8333 -s /deck/kong.local.yaml diff
```

Na koncu z ukazom **sync** uveljavimo konfiguracijo:

```
docker run -i \
-v $(pwd)/deck:/deck \
--network mju-iot \
kong/deck --kong-addr http://kong:8333 -s /deck/kong.prod.yaml sync
```

Vtičnik ACME

Ob uveljavitvi konfiguracije kong.prod.yaml se samodejno vključi **Kong ACME vtičnik** (Automatic Certificate Management Environment), ki poskrbi za avtomatsko pridobivanje in osveževanje **Let's Encrypt** certifikatov.

Certifikati se uporabljajo pri zagotavljanju varne kriptirane povezave s pomočjo kriptografskega protokla TLS, npr. HTTPS.

Seznam pridobljenih certifikatov je na voljo v Kong Manager nadzorni plošči pod API Gateway -> Certificates.

Začetna vzpostavitev TimescaleDB

Za pravilno delovanje shranjevanja podatkov v TimescaleDB s pomočjo QuantumLeap-a, slednji potrebuje ustrezno skonfigurirano podatkovno zbirko quantumleap.

Z zagonom QuantumLeap Timescale database boot/initialise/load script se izvede avtomatska vzpostavitev potrebnega začetnega stanja.

```
# Premik v ustrezno mapo
cd timescaledb
```

```
# Namestitev PostgreSQL Client-a
sudo apt install postgresql-client
# Zagon TimescaleDB
docker compose up -d
# Zagon inicializacijske skripte za QuantumLeap
python3 quantumleap-db-setup.py --pg-host 172.30.0.253 --pg-pass mju-1o7
```

Pomembno: Za potrebe quantumleap-db-setup.py sta v TimescaleDB docker-compose.yml definirana fiksni IP in geslo, ki se uporabita kot argumenta --pg-host in --pg-pass.

Zagon vseh storitev

Za avtomatski zagon se lahko uporabi skripta, ki avtomatsko zažene Docker vsebnike vseh storitev:

```
./run_all.sh
```

Prof. dr. Marko Bajec
