Devops Onboarding Tasks - Flare

Kaj je Flare?

Flare je L1 blockchain EVM veriga.

Na nivoju pametnih pogodb implementira posebne protokole kot so:

- FTSO: on-chain cene kriptovalut v USD, napovedujejo jih neodvisne entitete “data providerji” ki jih je okrog 100. Cene se posodabljajo vsake 3 minute.
- FTSO fast updates: podobno kot FTSO vendar se cene posodobijo na vsak blok, to je vsako sekundo. Providerji so vsak blok naključno izbrani in napovedo samo spremembo cene v delti (+, -, brez sprememb)
- FDC: prenos podatkov iz drugih verig in Web2 na Flare verigo. Izpraševalec postavi vprašanje npr. “Ali je bilo plačilo XY izvedeno na verigi bitcoin?”, neodvisni providerji preverijo stanje,vrnejo odgovor true/false in dosežejo konsenz.
- FASSETS: uporaba ne-pametnih žetonov (Xrp, Btc..) v pametnih pogodbah.

Glavna spletna stran: https://flare.network
Developer docs: https://dev.flare.network
Github: https://github.com/flare-foundation

Flare sestavljajo 4 med seboj neodvisne verige:
- Flare (mainnet)
- Songbird (canary)
- Coston (testnet za Songbird)
- Coston2 (testnet za Flare)

Explorerji: to so vizualizacijska orodja ki izpostavijo podatke verige na uporabniku bolj uporaben način:
https://flare-explorer.flare.network/
https://songbird-explorer.flare.network/
https://coston-explorer.flare.network/
https://coston-explorer.flare.network/

Tu lahko preverjamo stanje denarnic, delamo interakcijo s pametnimi pogodbami ipd.

Kaj je Flare node (go-flare)?

Flare Node je software ki bazira na odprti kodi Avalanchego. Flare node je jedro verige in poskrbi da veriga producira bloke in izpostavlja RPC APIje da lahko klienti komunicirajo z verigo in pridobijo podatke o blokih, transakcijah, denarnicah ipd.

Koda je na voljo na: https://github.com/flare-foundation/go-flare

Repozitorij je enak za vse 4 verige, razlika je samo v konfiguraciji.

Go-flare lahko teče v več različnih vlogah.

Validator

Validira in sestavlja nove bloke transakcij. Da nekdo postane validator mora zakleniti min 1.000.000 valute. Temu rečemo self-bond ali stake.
Ta self-bond da moč glasovanja v networku, večjo moč kot ima validator, večja je verjetnost da bo naključno izbran pri izbiri novega bloka.

Mainnet Flare veriga je sestavljena iz 100+ neodvisnih validatorjev po celem svetu. Vsaj 80% validatorjev mora biti “online” sicer se veriga upočasni ali preneha delovati.

RPC API node

Ta tip se imenuje tudi “observer” in se priklopi na verigo, izpostavi RPC/REST API in deluje kot vstopna točka za komunikacijo raznih web3 klientov. Ne dela validacij ampak samo posluša in si dopolnjuje bazo z novimi bloki in transakcijami

Komunicira z validatorji in mora biti povezan na minimalno 80% validatorjev da je v "konsenzu".

Dodatno so glede baze možne še tri konfiguracije ki vplivajo na zasedenost diska:
1. Full history node: drži celotno zgodovino transakcij in hkrati tudi state vsakega bloka. Dovoli npr. da RPC vrne odgovor kakšno je bilo stanje neke denarnice za poljuben blok N.
2. Pruning node: Drži celotno zgodovino transakcij ampak ne vmesnih stanj. Npr. na prejšnje vprašanje ne more dat odgovora ker bi moral narediti re-play vseh transakcij iz bloka 0 do bloka N da ugotovi dejansko stanje.
3. State-synced node: najhitrejši način za postavitev in tudi najmanjša baza. Pridobi samo trenutno stanje in potem nalaga transakcije v bazo od te točke dalje.

Devops Task

Orodja in tehnologije:

- Ansible
- Terraform
- Ubuntu/Linux
- GCP (priporočeno, lahko tudi poljuben cloud)

Cilj: namestitev go-flare observer node v GCP na Ubuntu VM po najboljših praksah.

Terraform

Vzpostavitev virtualke in ostalih potrebnih virov.

- Provisioning Ubuntu 24.04 kot individualen VM ali kot del Instance Group velikosti 1.
- Dodaten podatkovni disk za go-flare bazo
- Lasten (ne default) VPC network.
- Izpostavljen port 9650 za RPC, ostali inbound porti zaprti (z uporabo gcp firewall).
- Opcijsko. SSH port 22 dostopen samo prek IAP subneta 35.235.240.0/20 (https://cloud.google.com/compute/docs/connect/ssh-best-practices/network-access) - možne težave za uporabo z ansible
- Opcijsko: uporaba cloud NAT namesto javnega IPja (VM ima v tem primeru samo privatni IP)

Ansible

Postavitev verige coston2 v “state-sync” načinu (načeloma bi lahko katerokoli verigo ampak ta zasede najmanj prostora na disku, pruning ali full history način pa potrebuje več ur ali celo dni da pridobi vse podatke).

Namestitev z uporabo docker compose. Osnove postavitve so dokumentirane na https://dev.flare.network/run-node/using-docker

Slike se nahajajo na docker hub in pa na samem go-flare repozitoriju (ghcr).

Za state-sync način je potrebno v privzeti config.json dodati: **"state-sync-enabled": true**

Baza naj se shranjuje na dodatni podatkovi disk ki je bil ustvarjen z terraform.

Opcijsko: uporaba distroless docker slike
https://github.com/flare-foundation/go-flare/pkgs/container/go-flare

V Flare ekipi smo trenutno sredi prehoda docker slik na distroless in rootless slike za izboljšanje varnosti. Te slike imajo postfix -dless (npr. Latest je ghcr.io/flare-foundation/go-flare:dev-dless)

Pri teh slikah je pomembna sprememba ta da go-flare proces teče kot uporabnik z UID: 65532 Zato morajo biti configi in volume mount za bazo ustrezno lastniško popravljeni, npr: sudo chown -R 65532:65532 <volume-mountpoint>

Opcijsko:
Pred kratkim smo v ekipi dodali cosign podpisovanje Docker slik. Eden izmed končnih Ansible taskov bi lahko preveril ali se podpis lokalne slike ujema. Navodila za preverjanje podpisa:
https://github.com/flare-foundation/go-flare?tab=readme-ov-file#container-image

Uporaba RPC v praksi

Health lahko preverimo z: curl http://localhost:9650/ext/health

Podprta je večina standardnih Ethereum RPC metod. Npr. pridobivanje trenutnega bloka:

```
curl --location 'http://localhost:9650/ext/C/rpc' \
--header 'Content-Type: application/json' \
--data '{"jsonrpc":"2.0","method":"eth_blockNumber","params":[],"id":1}'
```

Opcijsko: Testiranje uporaba z blockchain denarnico

RPC endpoint lahko uporabimo skupaj z katero izmed blockchain denarnic. Ena najbolj znanih je MetaMask ki se namesti kot browser extension.

1. Na Chrome si namestimo MetaMask extension po teh navodilih:
https://dev.flare.network/network/getting-started#steps
Spremenimo samo Network URL da uporabimo lasten RPC in sicer: http://<ip virtualke ali NAT ip>:9650/ext/bc/C/rpc
MetaMask je denarnica in bo s tem uporabila naš lokalni RPC za komunikacijo z networkom, privzeto pa se sicer uporabi tudi javen RPC kot je prikazano v dokumentaciji.

2. Ustvarimo nov prazen naslov na omrežju coston2.
Ker je to testno omrežje obstajajo t.i. “Faucets” oz. “pipe” kjer lahko zastonj dobimo 100 žetonov. Iz dropdowna izberemo Coston2 verigo in vstavimo naslov denarnice:
https://faucet.flare.network/

3. Interakcija z pametno pogodbo
Interakcija je lahko čisto programska z uporabo raznih web3 knjižnic ali pa prek UI vmesnikov, med drugimi tudi ti explorerji ki sem jih prej poslal.
Npr. odpremo pogodbo WNat
https://coston2-explorer.flare.network/address/0xC67DCE33D7A8efA5FfEB961899C73fe01bCe9273?tab=write_contract

To je neka pametna pogodba ki "C2FLR" žetone pretvori v "Wrapped C2FLR". Kaj to točno je out of scope ampak recimo da se ta oblika žetona uporablja za neke posebne namene znotraj Flare sistemov.

Gremo na Write tab, se povežemo z MetaMask denarnico, potem pa pokličemo metodo "deposit", kjer noter vpišemo cifro "1000000000000000000" (to je enako 1 C2FLR), to je 10 na 18.
Potem potrdimo transakcijo z denarnico in če potem v explorerju odpreš naslov svoje denarnice (ki ga vidimo v Metamasku), bi moral imet stanje -1 FLR in +1 Wrapped FLR.