7. Akka Cluster

U ovom poglavlju nastavljamo s Akka.Clsuter modulom. Pogledati cemo sto je to klaster, kako se instalira Akka.Cluster modul te jednostavan primjer koristenja istog. Izgleda da trenutno ne postoji stabilna verzija modula, stoga cemo koristiti Akka.Cluster-1.0.7-18 beta.

7.1 Akka Cluster

U kontekstu racunarstva racunalni klaster se sastoji od skupa povezanih racunala koja zajedno funkcioniraju kako bi pruzila nekakvu uslugu, a mozemo ih smatrati jedinstvenim sustavom.

U kontekstu Akka.NET-a, klaster predstavlja decentraliziranu, otpornu na gresku i elasticnu peer-to-peer mrezu Akka.NET aplikacija bez jedne kriticne tocke (*single point of failure*) i uskog grla (*bottleneck*). Modul koji omogucuje izradu ovakvih aplikacija je Akka.Cluster.

Mogucnosti koje Akka.Cluster uvodi u Akka.NET ukljucuju:

- Jednostavnija izrada peer-to-peer sustava na mrezi;
- Omogucuje clanovima sustava automatsko otkrivanje novih cvorova i uklanjanje mrtvih bez izmjena u konfiguraciji;
- Klasama koje je korisnik definirao omogucuje da se pretplate na notifikacije o izmjenama o dostupnosti cvorova u klasteru;
- Uvodi koncept "uloge" kojim se omogucuje razlikovanje odvojenih Akka.NET aplikacija unutar klastera;
- Dozvoljava kreiranje klastera rutera, koji su ekstenzija ugradenih Akka.NET rutera, samo sto ovi ruteri automatski prilagodjavaju svoju listu pripadajucih actora kojima prosljedjuju poruke.

Prednosti ispravno dizajniranog klastera su sljedece:

- Otpornost na greske - klasteri se mogu elegantno oporaviti od pogresaka;
- Elasticnost - klasteri su sami po sebi elasticni te mogu skalirati po potrebi;
- Decentraliziranost - moguce je imati vise jednakih replika nekog mikroservisa i slicno;
- Peer-to-peer - novi cvorovi mogu stupiti u kontakst s postojecim clanovima na mrezi, biti obavijesteni o ostalim clanovima i u potpunosti se integrirati u mrezu bez konfiguracijskih izmjena;
- Nema jedne tocke kvara ni uskog grla - vise cvorova moze raditi za ispunjenje nekog zahtjeva cime se povecaje propusnost sustava i otpornost na pogreske.

Akka.Cluster bi trebali koristiti u scenarijima kada je potrebna visoka razina dostupnosti aplikacije. Ukratko, Akka.Cluster je dobar kandidat za koristenje u bilo kojem od sljedecih slucajeva:

- kada ocekujemo da ce nekakva aplikacija trpjeti veliku razinu prometa;
- ako zadace koje aplikacija mora obaviti nisu trivijalne;
- u slucajevima kada se ocekuje brz odgovor od aplikacije;
- kada nam je potrebna elasticnost - ako imamo peekove koristenja;
- mikroservisna arhitektura

Neki od slucajeva koristenja ukljucuju:

- Analitika
- Multiplayer igre
- Pracenje uredaja/Internet of things
- Sustavi za nadzor
- itd...

Osnovni koncepti:

- Cvor - logicki clan klastera;
- klaster - skup cvorova udruzenih u jednu cjelinu putem *membership* servisa. Vise Akka.NET aplikacija moze biti dio jednog klastera;
- *Gossip* - poruke u pozadini koje pokrecu sami klaster;
- *Leader* - cvor u klasteru koji dodaje i uklanja cvorove iz klastera;
- Uloga - imenovana odgovornost ili aplikacija unutar klastera. Klaster moze sadrzavati veci broj Akka.NET aplikacija u sebi, gdje svaka moze imati svoju ulogu;

- *Convergence* - kada se kvorum (vecina) gosip poruka slaze, potvrduje, izmjenu stanja nekog clana klastera.

7.2 Kako se Akka.Cluster razlikuje od Akka.Remote?

U pozadini Akka.Cluster-a se nalazi Remote, stoga sve sto se moze napraviti uz pomoc Akka.Remoting se ujedno moze napraviti i uz pomoc Akka.Cluster. Akka.Cluster koristi Remote s ciljem ispunjenja jedne strukture: klastera aplikacija.

Akka.Remote je modul koji sluzi kao komponente za pruzanje odredenih mogucnosti Akka.Clusteru i nekim drugim modulima. Akka.Remote, sam po sebi, bismo koristili u situacijama u kojima nam nije potrebna elasticnost i otpornost na pogreske.

7.3 Kako ukljuciti Akka.Cluster?

Ukljucivanje Akka.Cluster-a je dosta slicno kao i ukljucivanje Akka.Remote-a, a sastoji se od sljedecih koraka:

- Instalacija Akka.Cluster paketa
- Konfiguracija ClusterActorRefProvider-a
- Omoguci barem jedan Akka.Remote transport
- Konfiguriraj adresu za transport
- Omoguci barem 1 *seed* cvor
- Pokrenit actor sustav

Da biste instalirali Akka.Cluster, potrebno je otvoriti konzolu *nuget packet manager*-a. Istu mozete pronaci pod: Tools > Nuget Package Manager > Package Manager Console. Zatim u konzolu unesete:

Primjer 7.1: naredba za instalaciju

```
1          Install-Package Akka.Cluster -pre
```

Nakon instalacije Akka.Cluster paketa, potrebno je omoguciti ClusterActorRefProvider putem HOCON konfiguracije, konfigurirati Akka.Remote transport i omoguciti barem jedan seed cvor.

Primjer 7.2: Cvor koji je ujedno seed node

```
1  akka {
2      actor.provider = "Akka.Cluster.ClusterActorRefProvider,
           Akka.Cluster"
3      remote {
4          helios.tcp {
5              port = 8081
6              hostname = localhost
7          }
8      }
9      cluster {
10         seed-nodes = ["akka.tcp://ClusterSystem@127.0.0.1:8081"]
11     }
12 }
```

U 7.2 konfiguriramo trenutni cvor da je ujedno seed cvor.

Primjer 7.3: Ne seed konfiguracija

```
akka {
    actor.provider = "Akka.Cluster.ClusterActorRefProvider,
       Akka.Cluster"
    remote {
        helios.tcp {
            port = 0 #let os pick random port
            hostname = localhost
        }
    }
    cluster {
        seed-nodes = ["akka.tcp://ClusterSystem@127.0.0.1:8081"]
    }
}
```

U 7.3 konfiguriramo cvor koji ce se povezati na neki slucajno dodijeljen slobodan port, a seed nodom ce smatrati cvor stvoren u 7.2.

Dvije vazne napomenete:

- Preporuceno je imati barem 2-3 seed cvora. U slucaju da je samo 1 te se srusi, klaster ce i dalje funkcionirati, ali novi cvorovi se ne mogu pridruziti klasteru;
- Svim cvorovima u klasteru se actor system mora isto zvati, bez obzira radi li se o razlicitim aplikacijama.

7.4 Klaster Gossip

Ovo je najvazniji koncept unutar Akka.Cluster-a. Gossip je nacin na koji cvorovi saznaju o tome da se novi cvor pridruzio ili da je neki uklonjen bez izmjena u konfiguraciji.

Gossip je protok poruka medu cvorovima kojima se clanovima u klasteru osvjezava lista ostalih clanova klastera.

Sada dolazimo do onoga zasto su seed cvorovi vazni. Naime, ako se neki novi cvor zeli dodati u klaster, isti mora kontaktirati jednog od poznatih seed cvorova. Jednom kada je uspio kontaktirati seed cvor, zapocet ce primati gossip poruke koje sadrze informacije o ostalim clanovima klastera.

Dakle, razlika izmedu cvora i seed cvora je u tome sto je seed cvor (*seed node*) dobro poznata kontakt tocka koju novi cvor koji se zeli pridruziti klasteru mora kontaktirati. Seed cvorovi funkcioniraju kao *service-discovery* mehanizam za Akka.Cluster. S druge strane, cvor je samo obivna logicka jedinica klastera.

Tok gossip poruka cemo objasniti na primjeru sa slike 7.1. Dakle, u tom primjeru:

1. B zna seed cvor A i kontaktira ga te zatrazi da se pridruzi klasteru.
2. A oznaci cvor B kao podignut (spreman) i pocme dijeliti gossip poruke s cvorom B o ostalim cvorovima u klasteru, ali nema drugih cvorova koji su povezani u klaster u tom momentu.
3. C kontaktira B i zatrazi da se pridruzi klasteru.
4. A doda cvor C u klaster u pocme dijeliti gossip poruke s cvorom C.
5. B i C saznaju jedan o drugome od cvora A.
6. B i C se medusobno povezu te uspostave komunikaciju.

Gossip poruke se s vremenom javljaju svaki put kada neki od clanova promijeni svoje stanje u klasteru. Primjer koji cemo pogledati pokrece par actora u klasteru koji ispisuju ove poruke o clanstvu u klasteru.

7.5 Kako se klaster formira?

Ovdje cemo opisati kako izgleda proces spajanja cvora u klaster. Klasteri se inicijalno sastoje od dva razlicita dijela:

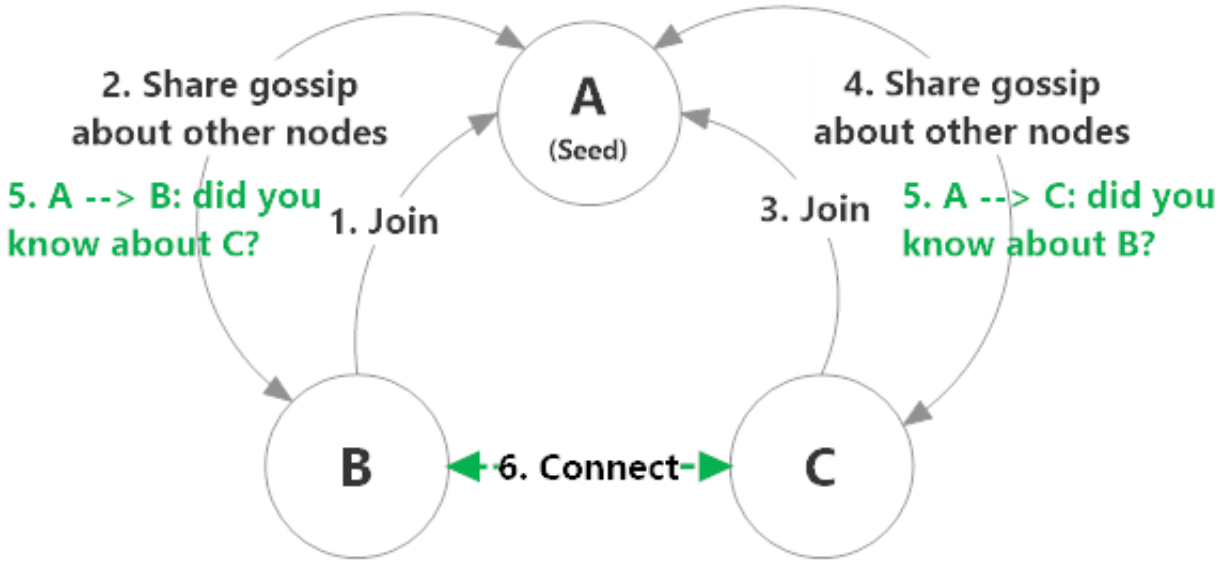

Figure 7.1: Tok gossip poruka, primjer

1. Seed cvorova: cvorovi koji se nalaze na dobro poznatim lokacijama na mrezi;
2. Obicni cvorovi: cvorovi cije inicijalne lokacije nisu poznate te ovi cvorovi kontaktiraju seed cvorove kako bi se formirao klaster.

7.5.1 Inicijalno stanje

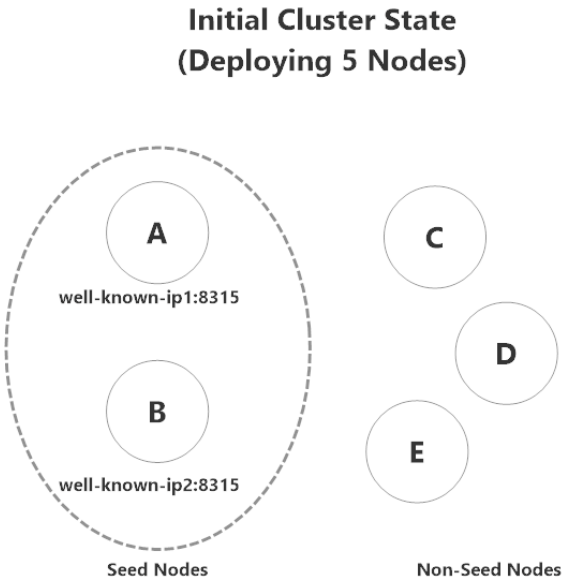

Figure 7.2: Kako se klaster formira 1

Inicijalno stanje klastera od 5 cvorova mozemo zamisliti da izgleda kao na slici 7.2.

A i B su inicijalno seed cvorovi te su vezani uz dobro poznate adrese (IP + port) koje su ugradjene u konfiguraciju cvorova C, D i E. A i B takoder znaju za jedan drugoga kako bi medusobno mogli komunicirati.

7.5.2 Stanje 1

Stanje 1 klastera je dano na slici 7.3.

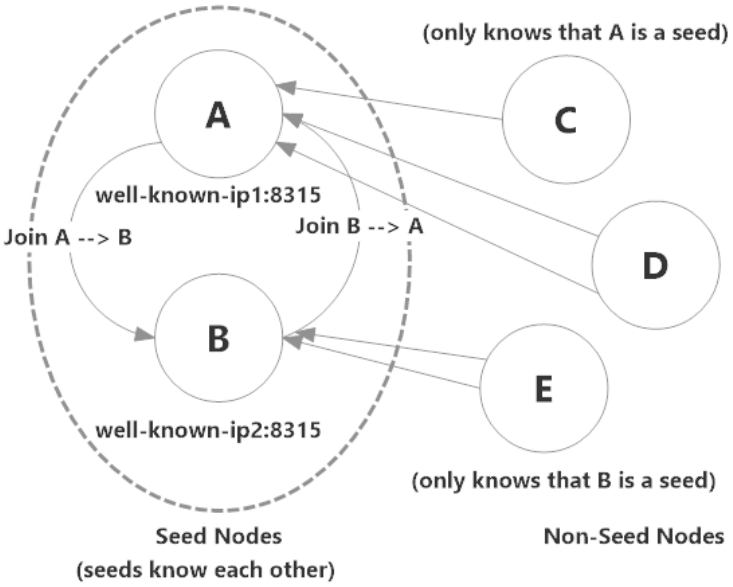

Figure 7.3: Kako se klaster formira 2

Svi cvorovi u ovom koraku pokusavaju konktaktirati seed cvor. U ovom primjeru cvorovi su konfigurirani na sljedeci nacin:

1. E zna kako kontaktirati B;
2. C i D znaju kako kontaktirati A;
3. A i B znaju kako kontaktirati jedan drugoga.

7.6 Stanje 2

Stanje 2 - podrazumjeva odabir vode (leader) te se cvorovi oznacuju kao spremni - zapocinje gossip.

Tijekom inicijalnog kontakta unutar klastera, mora se odabrati voda medu cvorovima. U primjeru 7.4 je odabran cvor A kao voda.

A ce poceti oznacavati cvorove kao podignute (spremne), pocevsi od cvorova za koje zna: A, B, C i D. Buduci da A jos uvijek ne zna za cvor E, isti nece biti oznacen kao podignut (spreman).

Sljedece se gossip informacije o clanstvima cvorova salju do svih cvorova klastera te ce se svi cvorovi poceti medusobno povezivati te stvarati isprepletenu mrezu.

7.6.1 Stanje 3

Stanje 3 - podrazumjeva da se gossip siri, formira se prsten - uspostavlja se komunikacija medu obicnim cvorovima (koji nisu seed).

Nakon sto je gossip imao priliku propagirati se po svim cvorovima te je voda (leader) oznacio sve cvorove kao spremne (podignute), svaki cvor ce biti povezan sa svakim te je tada klaster formiran. Svaki cvor sada moze sudjelovati u operacijama koje definira korisnik.

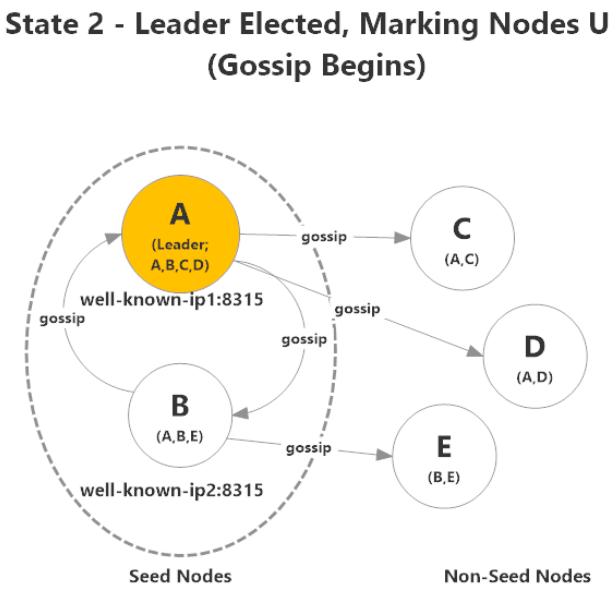

Figure 7.4: Kako se klaster formira 3

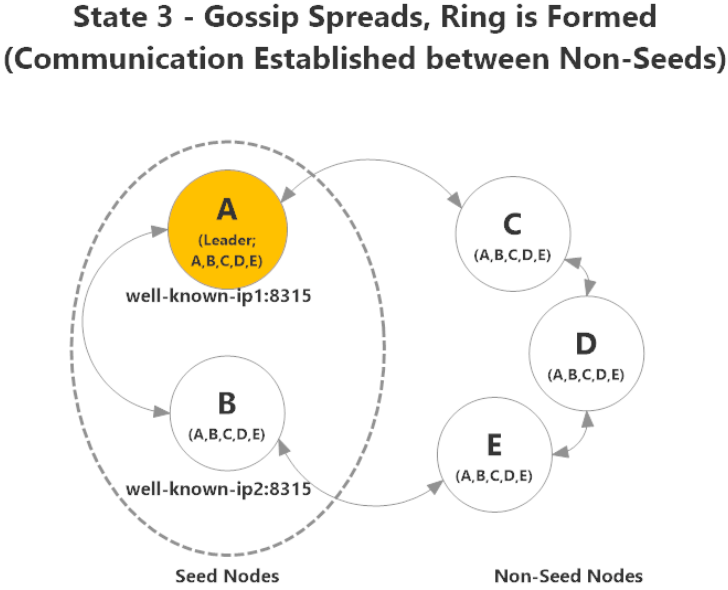

Figure 7.5: Kako se klaster formira 4