Kompassi

Kaikki muilla Tietojärjestelmät-sivun alasivuilla kerrotut asiat pätevät päällisin puolin Kompassiin, mutta Kompassi on sen verran monimutkainen eläin, että tässä vielä koosteena muutamia juttuja mitä siitä on hyvä tietää.

Kooditason dokumentaatio

Kannattaa perehtyä Kompassin lähdekoodin seassa oleviin *README.md*-tiedostoihin, luokkien ja funktioiden docstringeihin, koodin kommentteihin sekä tietokantakenttien *help_text*eihin. Kooditason dokumentaatio on auttamattoman puutteellista, pärjäile 🙂.

Kaikki sellainen, joka koskee vain Kompassia itseään avoimen lähdekoodin sovelluksena, kuuluu kooditason dokumentaatioon. Tälle sivulle kuuluvat ne asiat, jotka käsittelevät osoitteessa https://kompassi.eu sijaitsevaa virallista Kompassi-asennusta sekä Kompassin kehitysprosessia osana Traconia, Desuconia ja Ropeconia.

Kehitysprosessi

Backlog sijaitsee Jirassa. Boardille pääsemiseksi tarvitaan *kompassi-developers*-ryhmäjäsenyys, jonka saa Japsulta. ⚠️ **Ryhmän antaminen Kompassin taka-administa ei riitä, koska se ei päivitä ryhmäjäsenyyttä Crowdiin, vaan se pitää toistaiseksi päivittää käsin myös Crowdiin.**

Jirassa käytetään Scrum-moodia vain siksi, että se tarjoaa tehokkaammat filtterit kuin Kanban-moodi. Prosessilla ei ole mitään tekemistä Scrumin kanssa. Japsu saattaa käynnistää ja lopettaa sprinttejä sen mukaan, mitä kulloinkin haluaa nähdä varsinaisella lapputaululla.

Kompassin kehityksessä noudatetaan kevennettyä versiota Git Flow -prosessista. Kevennysten perusteena on se, että Japsu tekee suurimman osan kehitystyöstä eikä muita kehittäjiä ole aina saatavilla esimerkiksi koodikatselmointeja varten. Kevennykset ovat seuraavat:

- Japsu kehittää suoraan *development*-haarassa, jos työ ei jää pitkäksi aikaa kesken. Japsu ei siis välttämättä aina käytä *feature*-haaroja.
  - Vieraileville kehittäjille *feature*-haarat ovat pakollisia ja *developmentiin* mergetään Japsun koodikatselmoinnin kautta.
  - Katselmointia pyydetään lähettämällä GitHubissa *pull request*, joka kohdistuu kehittäjän omasta *feature*-haarasta virallisen repon *develop ment*-haaraan.
- Japsu tekee tarvittaessa hotfixejä suoraan *master*-haaraan ja mergeää suoraan *masterin developmentiin*.
  - Siinä epätodennäköisessä tapauksessa, että vieraileva kehittäjä tekee hotfixin, Japsu katselmoi sen ja mergeää sen ensin *masteriin* ja *de velopmentiin*.
- Releaseja ei tehdä GitFlow-tyyliin, vaan Japsu testattuaan uudet ominaisuudet *development*-haarassa ja Staging-ympäristössä mergeää *develop ment*-haaran suoraan *master*-haaraan ja sitä kautta deplaa sen tuotantoon.

Siinä vaiheessa, kun jatkuvaa kehitystyötä tekeviä kehittäjiä on useampia, näistä kevennyksistä voidaan siirtyä kohti "oikeaa" Git Flow'ta ja Japsu alistaa mielellään omatkin muutoksensa katselmoinnille. Kehitystyön ja tuotantoonviennin nopeus ei saa kuitenkaan merkittävästi kärsiä.

Ympäristöt

Kompassista on olemassa kaksi rinnakkaista asennusta:

Ympäristö	Osoite	Mihin se on tarkoitettu	Päivityspolitiikka
Tuotanto	https://kompassi.eu	Kompassin varsinainen tuotantoympäristö. Tätä oikeat tapahtumat ja ihmiset käyttävät.	Jatkuvan integraation ns. CI-putki päivittää *master*-haarasta
Staging	https://dev.kompassi.eu	Kompassin kehittämisen tueksi sekä esittelyyn muille tapahtumaorganisaatioille. Tänne voi antaa vapaammin pääkäyttäjätunnuksia joilla voi mellastaa vapaasti pääsemättä käsiksi henkilötietoihin tai aiheuttamatta tuhoa muille tapahtumille.	CI-putki päivittää *develop ment*-haarasta

Molemmat ympäristöt pyörivät Neula-palvelimella. Ympäristöjä hallitaan Ansible-konfiguraationhallinnan avulla.

Tuotantoonvienti

Olet siis saanut aikaan jotain valmista? Hyvä!

1. Rebaseta *feature*-haarasi GitHubin *tracon/kompassi*-repon *development*-haaran uusimman version päälle, jos se ei ole jo.
2. Puske feature-haarasi GitHubiin oman tunnuksesi alle luomaasi *kompassi*-repon forkkiin kuvaavalla haaranimellä, esim. *feature/condb-584*. Käytä haaran nimen osana joko Jira-tiketin numeroa tai ytimekästä kuvausta ominaisuudellesi, esim. *feature/automatic-shift-generation*.
3. Lähetä feature-haarastasi *pull requesttracon/kompassi*-repon *development*-haaraan.
4. Japsu katselmoi *pull requestisi*. Toteuta pyydetyt muutokset (tarvittaessa yhdessä Japsun kanssa).
5. Kun tarvittavat muutokset on tehty, Japsu mergeää *feature*-haarasi *development*-haaraan ja se pyörähtää ajoon Staging-ympäristöön.
6. Auta Japsua testaamaan muutoksesi Staging-ympäristössä.
7. Jos kaikki meni putkeen, seuraavaksi Japsu mergeää *development*-haaran *master*-haaraan eli "tekee julkaisun".
8. Muutoksesi ovat nyt tuotannossa. Onneksi olkoon!

Tietokantamigraatiot ja setup-skriptit

Tietokantamigraatiot tehdään Djangon omalla migraatiotyökalulla. Niiden vastuulla on tietokannan rakenteen (taulujen, kenttien ja indeksien) päivittäminen ajan tasalle.

Migraatioiden lisäksi käytetään ns. *idempotentteja setup-skriptejä*. Idempotentti tarkoittaa sitä, että setup-skriptiä voi ajaa samaan asennukseen uudestaan ja uudestaan, eikä se myöhemmillä ajokerroilla riko mitään saati hajoa omaan mahdottomuuteensa, vaan varmistaa vain haluttujen resurssien olevan olemassa.

Setup-skriptin vastuut pähkinänkuoressa:

1. luo organisaatiot, tapahtumat ja muun **rakenteellisen** informaation Kompassin tietokantaan
2. luo uudet ryhmät Atlassian-tuotteiden Crowd-kertakirjautumispalveluun, jos se on konfiguroitu

Omassa kehitysympäristössäsi voit ajaa setup-skriptin tai pelkät migraatiot Docker Composen avulla näin:

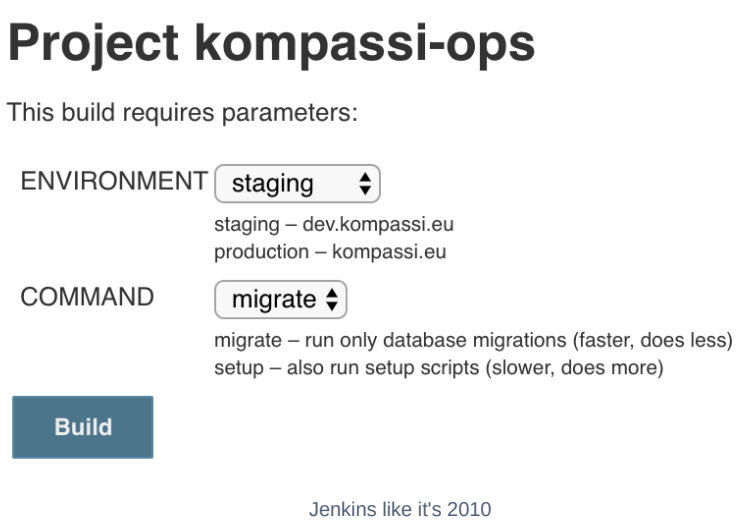
Migraatioiden ja setup-skriptin ajaminen lokaalissa kehitysympäristössä

```
# käynnistä ensin kehitysympäristö toisessa terminaali-ikkunassa
docker-compose up

# setup-skriptin ajaminen
docker-compose exec web python manage.py setup

# pelkkien tietokantamigraatioiden ajaminen
docker-compose exec web python manage.py migrate
```

Staging- ja tuotantoympäristöissä setup-skriptin ja migraatioiden ajoon käytetään Jenkinsistä löytyvää *kompassi-ops*-työtä. Jenkins käyttää puolestaan näiden ajamiseen Ansible-skriptejä. Kuha pääsee Jenkinsiin, voi ajaa tietokantamigraatiot tai setup-skriptin vaikka ei ole SSH-pääsyä tai paikallista Ansible-ympäristöä.



Jenkins like it's 2010

Tietokanta

Tuotanto- ja Staging-ympäristöissä tietokannan käyttöoikeudet on eriytetty siten, että tietokannan rakenteeseen vaikuttavat operaatiot – siis käytännössä migrate- ja setup-skriptit – ajetaan tietokannan omistajakäyttäjällä (ei kuitenkaan pääkäyttäjällä) ja sovellusta itseään pyöritetään matalammin oikeuksin. Lisää tästä voit lukea kooditason dokumentaatiosta.

Molempien tietokannat sijaitsevat tätä kirjoitettaessa vielä Neulan dockeroidussa PostgreSQL:ssä ja tätä lukiessasi ehkä jo Siilolla. Tietokantoja varmuuskopioidaan jatkuvasti Piiloon. Lue lisää tietokannoista.

Ympäristö	Tietokanta missä nyt (16.9.2017)	Tietokanta missä toivottavasti pian	Tavallinen käyttäjä (Sovelluksen ajo)	Tietokannan omistajakäyttäjä (Migrate- ja setup-skriptit)
Tuotanto	palvelin *neula.kompassi.eu* kontti *kompassi.eu-docker* (PostgreSQL 9.5) tietokanta *kompassi*	palvelin *siilo.tracon.fi* kontittamaton PostgreSQL 10 tietokanta *kompassi*	*kompassi*	*kompassi_ddl*

Staging	palvelin *neula.kompassi.eu* kontti *kompassi.eu-docker* (PostgreSQL 9.5) tietokanta *kompassidev*	palvelin *siilo.tracon.fi* kontittamaton PostgreSQL 10 tietokanta *kompassi*	*kompassidev*	*kompassidev_ddl*