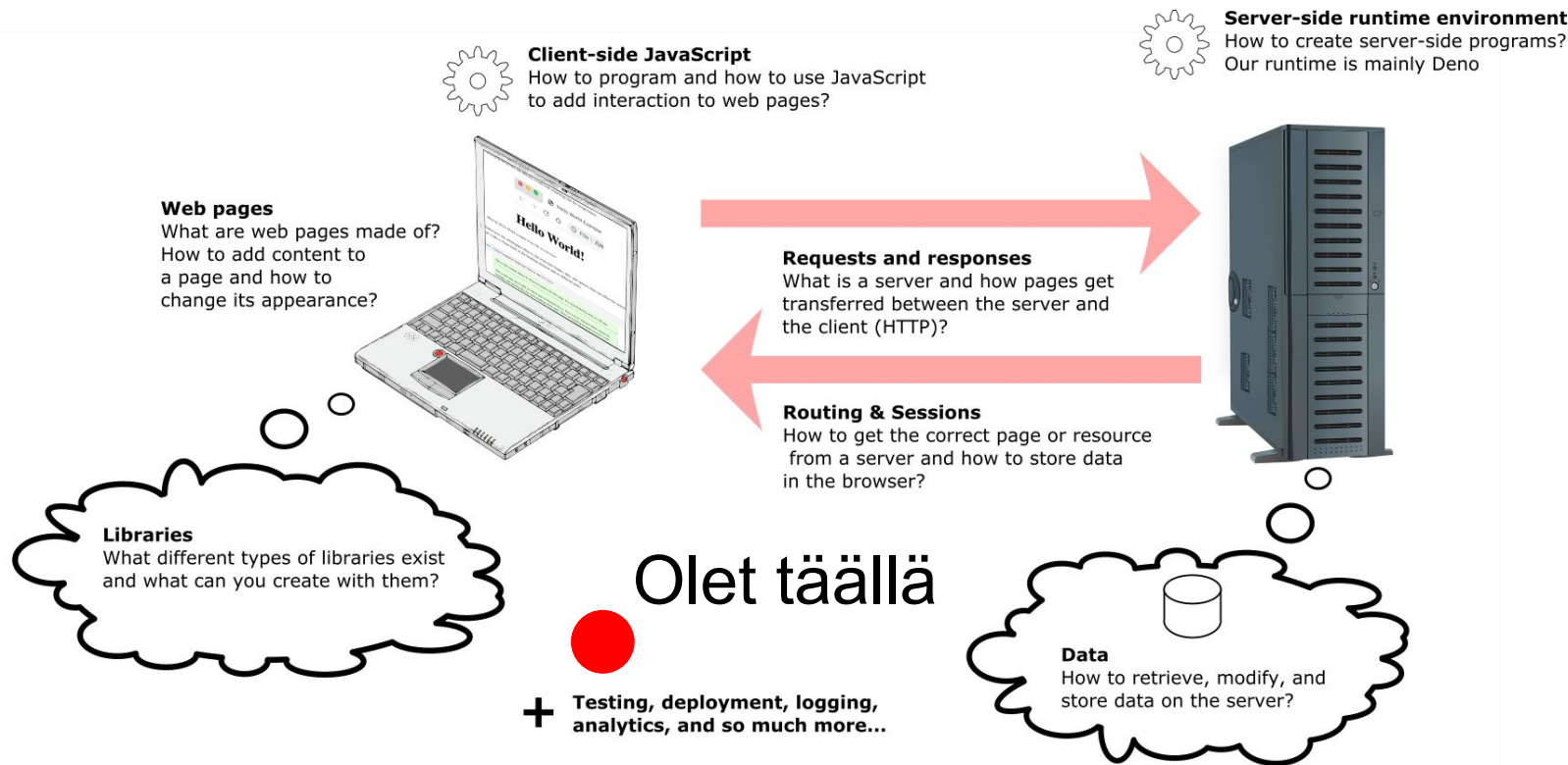


Päivä 8 - CI/CD, Serverless, Logitus, Monitorointi

2021-12-17

AaltoPRO - Websovelluskehitys

Web-sovellukset korkealla tasolla

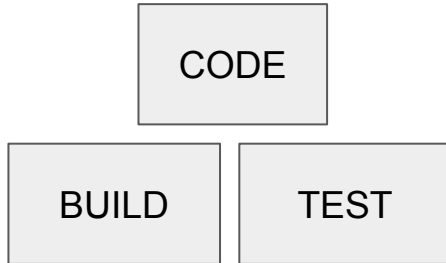


Päivä 8

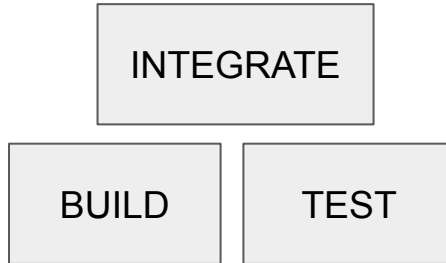
- 9-12 Aamupäivä
 - CI/CD, Deployment
 - Kahvitauko
 - Serverless, logitus, monitorointi
 - Yhteenveto
- 12:00 - 13:00 Lounas
- 13:00 - 16:00 Iltapäivä
 - Miniprojekti
 - Kahvitauko
 - Miniprojekti jatkuu
 - Yhteenveto

Continuous Integration, Continuous Delivery, ...

Oma kone



Esim. GitHub

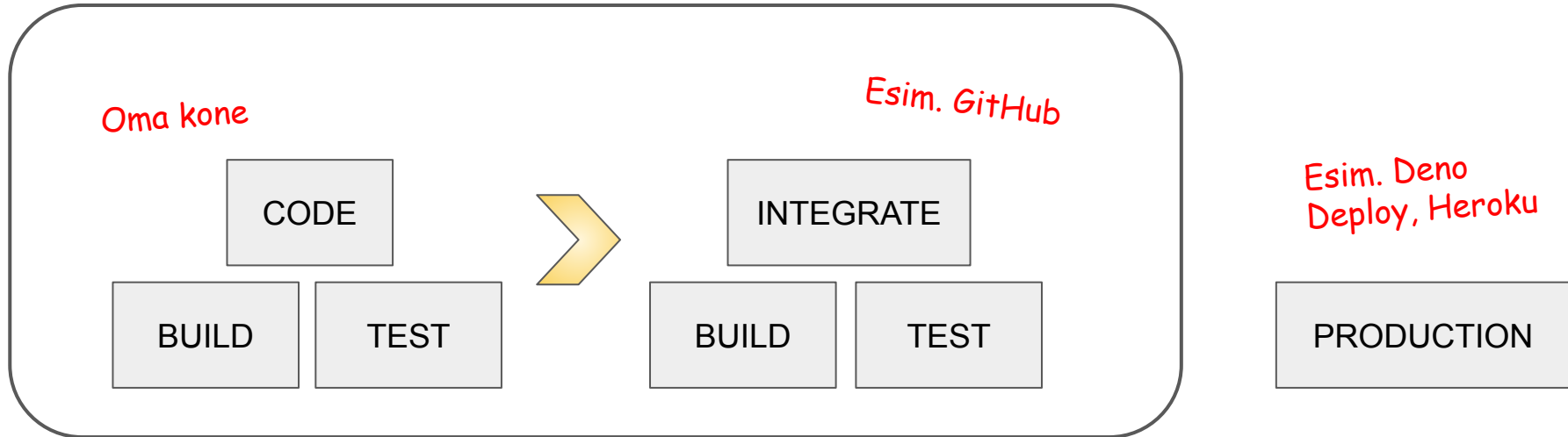


*Esim. Deno
Deploy, Heroku*



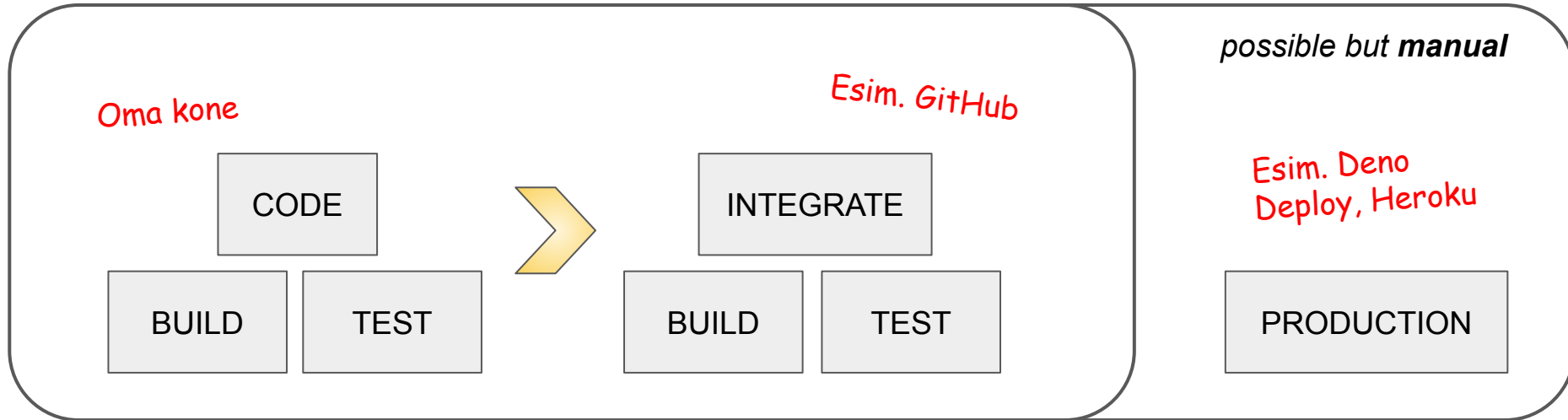
Continuous Integration, Continuous Delivery, ...

- Continuous Integration → *changes committed often, automated tests*



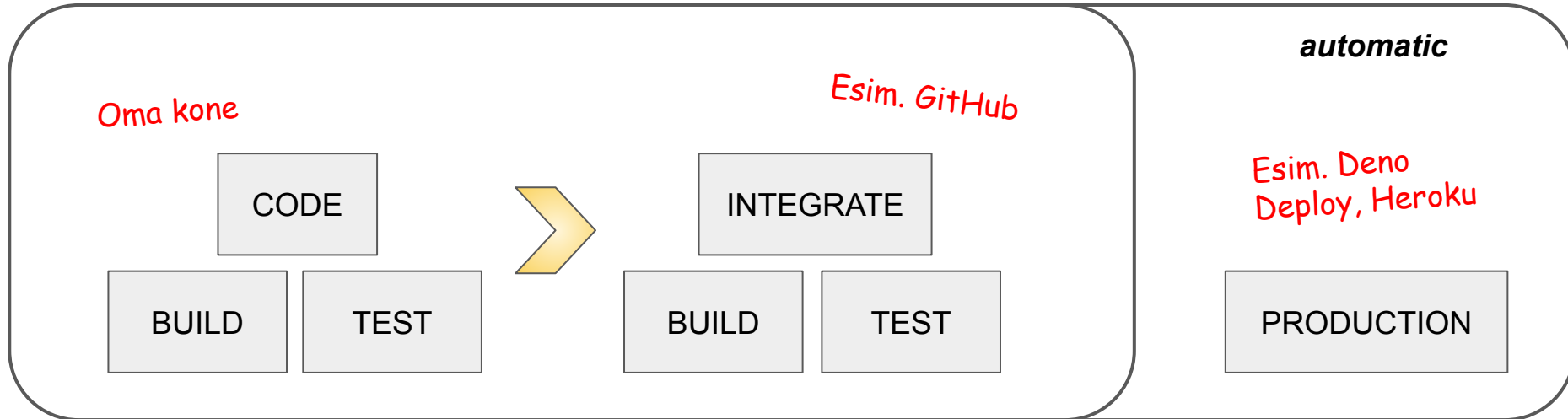
Continuous Integration, Continuous Delivery, ...

- Continuous Integration → *changes committed often, automated tests*
- Continuous Delivery → *software can be released to production at any time*



Continuous Integration, Continuous Delivery, ...

- Continuous Integration → *changes committed often, automated tests*
- Continuous Delivery → *software can be released to production at any time*
- Continuous Deployment → *latest version of the software is the live version*



CI / CD vaatii automatisaatiota

- Yksi vaihtoehto GitHub Actions: <https://github.com/features/actions>
 - Testien ajaminen
 - Koodin katselmointi
 - Sovelluksen vieminen tuotantoon
 - ...
- Myös muita vaihtoehtoja
 - Travis CI
 - Jenkins
 - CircleCI
 - ...

GitHub Actionsin käyttöönotto

- Oletus: projekti GitHubissa
- GitHubissa olevaan projektiin luodaan kansio `.github`
- Kansion `.github` sisälle luodaan kansio `workflows`
- Kansion `workflows` sisälle luodaan automatisoitavaa prosessia luova tiedosto, esimerkiksi `ci.yml`

Konfiguraatiotiedosto (kansiossa `.github/workflows`)

- YML-tiedosto (yet another markup file), joka sisältää ohjeet GitHub Actionsille
 - Ohjeiden nimi: `name`
 - Milloin ohjeita tulee noudattaa: `on`
 - Mitä tulee tehdä: `jobs`

Konfiguraatiotiedosto (kansiossa `.github/workflows`)

- YAML-tiedosto (yet another markup file), joka sisältää ohjeet GitHub Actionsille
 - Ohjeiden nimi: `name`
 - Milloin ohjeita tulee noudattaa: `on`
 - Mitä tulee tehdä: `jobs`

```
1 name: Run tests on project
2
3 on:
4   push:
5     branches: [main]
6
7 jobs:
8   test:
9     runs-on: ubuntu-latest
10
11     steps:
12       - name: Setup repo
13         uses: actions/checkout@v2
14
15       - name: Setup Deno
16         uses: denoland/setup-deno@v1
17         with:
18           deno-version: "v1.15.3"
19
20       - name: Run tests
21         run: deno test --allow-all
```

Konfiguraatiotiedosto (kansiossa `.github/workflows`)

- YAML-tiedosto (yet another markup file), joka sisältää ohjeet GitHub Actionsille
 - Ohjeiden nimi: `name`
 - Milloin ohjeita tulee noudattaa: `on`
 - Mitä tulee tehdä: `jobs`

```
1 name: Run tests on project
2
3 on:
4   push:
5     branches: [main]
6
7 jobs:
8   test:
9     runs-on: ubuntu-latest
10
11   steps:
12     - name: Setup repo
13       uses: actions/checkout@v2
14
15     - name: Setup Deno
16       uses: denoland/setup-deno@v1
17       with:
18         deno-version: "v1.15.3"
19
20     - name: Run tests
21       run: deno test --allow-all
```

Konfiguraatiotiedosto (kansiossa `.github/workflows`)

- YAML-tiedosto (yet another markup file), joka sisältää ohjeet GitHub Actionsille

- Ohjeiden nimi: `name`
- Milloin ohjeita tulee noudattaa: `on`
- Mitä tulee tehdä: `jobs`

```
1 name: Run tests on project
2
3 on:
4   push:
5     branches: [main]
6
7 jobs:
8   test:
9     runs-on: ubuntu-latest
10
11   steps:
12     - name: Setup repo
13       uses: actions/checkout@v2
14
15     - name: Setup Deno
16       uses: denoland/setup-deno@v1
17       with:
18         deno-version: "v1.15.3"
19
20     - name: Run tests
21       run: deno test --allow-all
```

Konfiguraatiotiedosto (kansiossa `.github/workflows`)

- YAML-tiedosto (yet another markup file), joka sisältää ohjeet GitHub Actionsille
 - Ohjeiden nimi: `name`
 - Milloin ohjeita tulee noudattaa: `on`
 - Mitä tulee tehdä: `jobs`

```
1 name: Run tests on project
2
3 on:
4   push:
5     branches: [main]
6
7 jobs:
8   test:
9     runs-on: ubuntu-latest
10
11   steps:
12     - name: Setup repo
13       uses: actions/checkout@v2
14
15     - name: Setup Deno
16       uses: denoland/setup-deno@v1
17       with:
18         deno-version: "v1.15.3"
19
20     - name: Run tests
21       run: deno test --allow-all
```

*Kun lähdekoodi
pusketaan GitHubiin
(päähaara)*

Konfiguraatiotiedosto (kansiossa `.github/workflows`)

- YAML-tiedosto (yet another markup file), joka sisältää ohjeet GitHub Actionsille

- Ohjeiden nimi: `name`
- Milloin ohjeita tulee noudattaa: `on`
- Mitä tulee tehdä: `jobs`

```
1 name: Run tests on project
2
3 on:
4   push:
5     branches: [main]
6
7 jobs:
8   test:
9     runs-on: ubuntu-latest
10
11   steps:
12     - name: Setup repo
13       uses: actions/checkout@v2
14
15     - name: Setup Deno
16       uses: denoland/setup-deno@v1
17       with:
18         deno-version: "v1.15.3"
19
20     - name: Run tests
21       run: deno test --allow-all
```

*Kun lähdekoodi
pusketaan GitHubiin
(päähaara)*

Konfiguraatiotiedosto (kansiossa `.github/workflows`)

- YAML-tiedosto (yet another markup file), joka sisältää ohjeet GitHub Actionsille

- Ohjeiden nimi: `name`
- Milloin ohjeita tulee noudattaa: `on`
- Mitä tulee tehdä: `jobs`

Yksi tehtävä, jonka nimeksi annettu test

```
1 name: Run tests on project
2
3 on:
4   push:
5     branches: [main]
6
7 jobs:
8   test:
9     runs-on: ubuntu-latest
10
11   steps:
12     - name: Setup repo
13       uses: actions/checkout@v2
14
15     - name: Setup Deno
16       uses: denoland/setup-deno@v1
17       with:
18         deno-version: "v1.15.3"
19
20     - name: Run tests
21       run: deno test --allow-all
```

*Kun lähdekoodi
pusketaan GitHubiin
(päähaara)*

Konfiguraatiotiedosto (kansiossa `.github/workflows`)

- YAML-tiedosto (yet another markup file), joka sisältää ohjeet GitHub Actionsille

- Ohjeiden nimi: `name`
- Milloin ohjeita tulee noudattaa: `on`
- Mitä tulee tehdä: `jobs`

Yksi tehtävä, jonka nimeksi annettu test

Ajetaan Ubuntulla

```
1 name: Run tests on project
2
3 on:
4   push:
5     branches: [main]
6
7 jobs:
8   test:
9     runs-on: ubuntu-latest
10
11   steps:
12     - name: Setup repo
13       uses: actions/checkout@v2
14
15     - name: Setup Deno
16       uses: denoland/setup-deno@v1
17       with:
18         deno-version: "v1.15.3"
19
20     - name: Run tests
21       run: deno test --allow-all
```

*Kun lähdekoodi
pusketaan GitHubiin
(päähaara)*

Konfiguraatiotiedosto (kansiossa `.github/workflows`)

- YAML-tiedosto (yet another markup file), joka sisältää ohjeet GitHub Actionsille

- Ohjeiden nimi: `name`
- Milloin ohjeita tulee noudattaa: `on`
- Mitä tulee tehdä: `jobs`

Yksi tehtävä, jonka nimeksi annettu test

Kolme askelta

Ajetaan Ubuntulla

```
1 name: Run tests on project
2
3 on:
4   push:
5     branches: [main]
6
7 jobs:
8   test:
9     runs-on: ubuntu-latest
10
11     steps:
12       - name: Setup repo
13         uses: actions/checkout@v2
14
15       - name: Setup Deno
16         uses: denoland/setup-deno@v1
17         with:
18           deno-version: "v1.15.3"
19
20       - name: Run tests
21         run: deno test --allow-all
```

Kun lähdekoodi pusketaan GitHubiin (päähaara)

Konfiguraatiotiedosto (kansiossa `.github/workflows`)

- YAML-tiedosto (yet another markup file), joka sisältää ohjeet GitHub Actionsille

- Ohjeiden nimi: `name`
- Milloin ohjeita tulee noudattaa: `on`
- Mitä tulee tehdä: `jobs`

1. Hae lähdekoodit

Yksi tehtävä, jonka nimeksi annettu test

Kolme askelta

Ajetaan Ubuntulla

```
1 name: Run tests on project
2
3 on:
4   push:
5     branches: [main]
6
7 jobs:
8   test:
9     runs-on: ubuntu-latest
10
11     steps:
12       - name: Setup repo
13         uses: actions/checkout@v2
14
15       - name: Setup Deno
16         uses: denoland/setup-deno@v1
17         with:
18           deno-version: "v1.15.3"
19
20       - name: Run tests
21         run: deno test --allow-all
```

Kun lähdekoodi
pusketaan GitHubiin
(päähaara)

Konfiguraatiotiedosto (kansiossa `.github/workflows`)

- YML-tiedosto (yet another markup file), joka sisältää ohjeet GitHub Actionsille

- Ohjeiden nimi: `name`
- Milloin ohjeita tulee noudattaa: `on`
- Mitä tulee tehdä: `jobs`

1. Hae lähdekoodit

2. Asenna Deno

Yksi tehtävä, jonka nimeksi annettu test

Kolme askelta

Ajetaan Ubuntulla

```
1 name: Run tests on project
2
3 on:
4   push:
5     branches: [main]
6
7 jobs:
8   test:
9     runs-on: ubuntu-latest
10
11     steps:
12       - name: Setup repo
13         uses: actions/checkout@v2
14
15       - name: Setup Deno
16         uses: denoland/setup-deno@v1
17         with:
18           deno-version: "v1.15.3"
19
20       - name: Run tests
21         run: deno test --allow-all
```

Kun lähdekoodi
pusketaan GitHubiin
(päähaara)

Konfiguraatiotiedosto (kansiossa `.github/workflows`)

- YAML-tiedosto (yet another markup file), joka sisältää ohjeet GitHub Actionsille

- Ohjeiden nimi: `name`
- Milloin ohjeita tulee noudattaa: `on`
- Mitä tulee tehdä: `jobs`

1. Hae lähdekoodit

2. Asenna Deno

3. Aja testit

Yksi tehtävä, jonka nimeksi annettu `test`

Kolme askelta

Ajetaan Ubuntuilla

```
1 name: Run tests on project
2
3 on:
4   push:
5     branches: [main]
6
7 jobs:
8   test:
9     runs-on: ubuntu-latest
10
11     steps:
12       - name: Setup repo
13         uses: actions/checkout@v2
14
15       - name: Setup Deno
16         uses: denoland/setup-deno@v1
17         with:
18           deno-version: "v1.15.3"
19
20       - name: Run tests
21         run: deno test --allow-all
```

*Kun lähdekoodi
pusketaan GitHubiin
(päähaara)*

Demo

- Demo
 - Uuden projektin luominen GitHubiin
 - Sovellus, joka palauttaa viestin “Hei maailma!”
 - Testi sovellukselle
 - GitHub Actions -konfiguraatio (kansioon `.github/workflows`)
 - Projektin lisääminen GitHubiin
- Otetaan pohjaksi project-08-starter (pohjasta kannattaa kommentoida tietokantaan liittyvät asiat pois)
- (+ hands-on nyt vai kohta – äänestys!)

Hieman hifistelyä

- Mahdollisuus lisätä dokumentaatioon (README.md) tieto testien läpimenosta
- <https://docs.github.com/en/actions/monitoring-and-troubleshooting-workflows/adding-a-workflow-status-badge>

Kahvitauko?

Sovellukset verkossa, monta vaihtoehtoa

- Software as a Service (SaaS), sovellus ostetaan palveluna

Sovellukset verkossa, monta vaihtoehtoa

- Software as a Service (SaaS), sovellus ostetaan palveluna
- Platform as a Service (PaaS), alusta ostetaan palveluna (Heroku, Deno Deploy)

Sovellukset verkossa, monta vaihtoehtoa

- Software as a Service (SaaS), sovellus ostetaan palveluna
- Platform as a Service (PaaS), alusta ostetaan palveluna (Heroku, Deno Deploy)
- Infrastructure as a Service (IaaS), palvelimen tai virtuaalipalvelimen ostaminen palveluna

Sovellukset verkossa, monta vaihtoehtoa

- Software as a Service (SaaS), sovellus ostetaan palveluna
- Platform as a Service (PaaS), alusta ostetaan palveluna (Heroku, Deno Deploy)
- Infrastructure as a Service (IaaS), palvelimen tai virtuaalipalvelimen ostaminen palveluna
- Oma infra – oma palvelin (ml. ylläpito, varmuuskopiot, ...)

Sovellukset verkossa, monta vaihtoehtoa

- Software as a Service (SaaS), sovellus ostetaan palveluna
- Platform as a Service (PaaS), alusta ostetaan palveluna (Heroku, Deno Deploy)
- Infrastructure as a Service (IaaS), palvelimen tai virtuaalipalvelimen ostaminen palveluna
- Oma infra – oma palvelin (ml. ylläpito, varmuuskopiot, ...)



SaaS

PaaS

IaaS

“Oma infra”

PaaS ja Serverless

- Platform as a Service
 - Sovelluskehittäjä voi (usein) vaikuttaa ympäristöön (ytimet, muisti, ym)
 - Sovelluskehittäjä määrittelee palvelinten lukumäärän määrän
 - Sovelluksen käynnistymisessä voi kestää hetki

PaaS ja Serverless

- Platform as a Service
 - Sovelluskehittäjä voi (usein) vaikuttaa ympäristöön (ytimet, muisti, ym)
 - Sovelluskehittäjä määrittelee palvelinten lukumäärän määrän
 - Sovelluksen käynnistymisessä voi kestää hetki
- Serverless
 - Ympäristöön ei tyypillisesti voi vaikuttaa
 - Palvelinten lukumäärä muuttuu automaattisesti tarvittaessa
 - Sovelluksen käynnistyminen tyypillisesti nopeaa

PaaS ja Serverless

- Platform as a Service

- Sovelluskehittäjä voi (usein) vaikuttaa ympäristöön (ytimet, muisti, ym)
- Sovelluskehittäjä määrittelee palvelinten lukumäärän määrän
- Sovelluksen käynnistymisessä voi kestää hetki

- Serverless

- Ympäristöön ei tyypillisesti voi vaikuttaa
- Palvelinten lukumäärä muuttuu automaattisesti tarvittaessa
- Sovelluksen käynnistyminen tyypillisesti nopeaa

*Ei tarkoita että palvelinta
tai palvelimia ei olisi, niistä
vain ei tarvitse välittää.*

PaaS ja Serverless

- Platform as a Service

- Sovelluskehittäjä voi (usein) vaikuttaa ympäristöön (ytimet, muisti, ym)
- Sovelluskehittäjä määrittelee palvelinten lukumäärän määrän
- Sovelluksen käynnistymisessä voi kestää hetki

- Serverless

- Ympäristöön ei tyypillisesti voi vaikuttaa
- Palvelinten lukumäärä muuttuu automaattisesti tarvittaessa
- Sovelluksen käynnistyminen tyypillisesti nopeaa

*Ei tarkoita että palvelinta
tai palvelimia ei olisi, niistä
vain ei tarvitse välittää.*

- Yhteistä

- Keskitytään sovelluksen luomiseen (ei esimerkiksi palvelimen konfigurointiin)

PaaS ja Serverless

*Kustannus => palvelinten
päälläoloaika*

- Platform as a Service

- Sovelluskehittäjä voi (usein) vaikuttaa ympäristöön (ytimet, muisti, ym)
- Sovelluskehittäjä määrittelee palvelinten lukumäärän määrän
- Sovelluksen käynnistymisessä voi kestää hetki

- Serverless

- Ympäristöön ei tyypillisesti voi vaikuttaa
- Palvelinten lukumäärä muuttuu automaattisesti tarvittaessa
- Sovelluksen käynnistyminen tyypillisesti nopeaa

*Ei tarkoita että palvelinta
tai palvelimia ei olisi, niistä
vain ei tarvitse välittää.*

- Yhteistä

- Keskitytään sovelluksen luomiseen (ei esimerkiksi palvelimen konfigurointiin)

PaaS ja Serverless

- Platform as a Service
 - Kustannus => palvelinten
päälläoloaika*
 - Sovelluskehittäjä voi (usein) vaikuttaa ympäristöön (ytimet, muisti, ym)
 - Sovelluskehittäjä määrittelee palvelinten lukumäärän määrän
 - Sovelluksen käynnistymisessä voi kestää hetki
- Serverless
 - Kustannus => pyyntöjen
lukumäärä*
 - Ympäristöön ei tyypillisesti voi vaikuttaa
 - Palvelinten lukumäärä muuttuu automaattisesti tarvittaessa
 - Sovelluksen käynnistyminen tyypillisesti nopeaa
- Yhteistä
 - Keskitytään sovelluksen luomiseen (ei esimerkiksi palvelimen konfigurointiin)

*Ei tarkoita että palvelinta
tai palvelimia ei olisi, niistä
vain ei tarvitse välittää.*

Serverless - hyödyt ja haitat

- Hyötyjä
 - Ei tarvitse välittää palvelimen konfiguroinnista (jne)
 - Kustannukset parhaimmillaan todella maltillisia
 - Skaalautuu automaattisesti

Serverless - hyödyt ja haitat

- Hyötyjä
 - Ei tarvitse välittää palvelimen konfiguroinnista (jne)
 - Kustannukset parhaimmillaan todella maltillisia
 - Skaalautuu automaattisesti *Myös alas!*

Serverless - hyödyt ja haitat

- Hyötyjä

- Ei tarvitse välittää palvelimen konfiguroinnista (jne)
- Kustannukset parhaimmillaan todella maltillisia
- Skaalautuu automaattisesti *Myös alas!*

- Haasteita

- “Cold start” – sovelluksen ensimmäinen käynnistyminen pidempi
- Palvelimet tulevat ja menevät - tarvitsee “uusia” välineitä monitorointiin
- Voi tulla kalliiksikin... (vrt. jatkuvasti kovassa käytössä olevat palvelimet)

Serverless - hyödyt ja haitat

- Hyötyjä

- Ei tarvitse välittää palvelimen konfiguroinnista (jne)
- Kustannukset parhaimmillaan todella maltillisia
- Skaalautuu automaattisesti *Myös alas!*

- Haasteita

- “Cold start” – sovelluksen ensimmäinen käynnistyminen pidempi
- Palvelimet tulevat ja menevät - tarvitsee “uusia” välineitä monitorointiin
- Voi tulla kalliiksikin... (vrt. jatkuvasti kovassa käytössä olevat palvelimet)

*Huom! Palvelimissa
mahdollisesti myös muita kuluja
kuten verkkoliikenne!*

Demo

- Jatketaan aiemman projektin parissa – siirretään sovellus Serverless-ympäristöön

Demo

- Jatketaan aiemman projektin parissa – siirretään sovellus Serverless-ympäristöön

*Tuttu Deno
Deploy!*

Demo

- Jatketaan aiemman projektin parissa – siirretään sovellus Serverless-ympäristöön *Tuttu Deno Deploy!*
- Käynnistymisen nopeuttamiseksi haluamme hieman tiivistää sovellusta: komento `deno bundle` kopioi sovelluksen ja sen riippuvuudet yhteen tiedostoon.

Demo

- Jatketaan aiemman projektin parissa – siirretään sovellus Serverless-ympäristöön *Tuttu Deno Deploy!*
- Käynnistymisen nopeuttamiseksi haluamme hieman tiivistää sovellusta: komento `deno bundle` kopioi sovelluksen ja sen riippuvuudet yhteen tiedostoon.
- Kokeillaan paikallisesti:
 - `deno bundle app-launch.js app.bundle.js`
 - `deno run -allow-all app.bundle.js`

Demo

- Jatketaan aiemman projektin parissa – siirretään sovellus Serverless-ympäristöön
- Käynnistymisen nopeuttamiseksi haluamme hieman tiivistää sovellusta: komento `deno bundle` kopioi sovelluksen ja sen riippuvuudet yhteen tiedostoon.
- Kokeillaan paikallisesti:
 - `deno bundle app-launch.js app.bundle.js`
 - `deno run -allow-all app.bundle.js`

*Tuttu Deno
Deploy!*

*Sovellus käynnistetään
tämän kautta
deno run -komennolla*

jobs:

test: // täällä yhä vanha sisältö

deploy:

runs-on: ubuntu-latest

needs: test

steps:

- name: Setup repo

uses: actions/checkout@v2

- name: Setup Deno

uses: denoland/setup-deno@v1

with:

deno-version: "v1.15.3"

- name: Bundle

run: deno bundle app-launch.js app.bundle.js

- name: Commit to branch

run: |

git ...

git ...

Hahmotelma: CD

- Sovelluksen vieminen verkkoon (Deno Deploy + GitHub Actions -yhdistelmä)
- Asetetaan Deno Deploy viemään production-nimisestä haarasta versiot tuotantoon
- Kopioidaan muutokset production-haaraan mikäli testit menevät läpi
- Oikealla oleva ehdotus [StackOverflowsta](#)

Hahmotelma: CD

- Sovelluksen vieminen verkkoon (Deno Deploy + GitHub Actions -yhdistelmä)
- Asetetaan Deno Deploy viemään production-nimisestä haarasta versiot tuotantoon
- Kopioidaan muutokset production-haaraan mikäli testit menevät läpi
- Oikealla oleva ehdotus [StackOverflowsta](#)

jobs:

```
test: // täällä yhä vanha sisältö
```

deploy:

```
runs-on: ubuntu-latest
```

```
needs: test
```

```
steps:
```

```
- name: Setup repo
```

```
uses: actions/checkout@v2
```

```
- name: Setup Deno
```

```
uses: denoland/setup-deno@v1
```

```
with:
```

```
deno-version: "v1.15.3"
```

```
- name: Bundle
```

```
run: deno bundle app-launch.js app.bundle.js
```

```
- name: Commit to branch
```

```
run: |
```

```
git ...
```

```
git ...
```

*Vain mikäli testit
menevät läpi*

Serverless: Monitorointi ja logitus

- Serverless-teknologian ytimessä se, että palvelin saatetaan käynnistää vain hetkeksi \Rightarrow logien kerääminen palvelimelle käytännössä mahdotonta
- Monitorointi – sovelluksen tilaa ja toimintaa kuvaavan logidatan keruu – tulee käytännössä toteuttaa toiseen sovellukseen, johon Serverless-teknologiaan perustuva sovellus lähettää tietoa.

Monitorointi ja logitus

- Tähän on monia kolmannen osapuolen palveluita
 - Pingdom, <https://www.pingdom.com/>
 - Datadog, <https://www.datadoghq.com/>
 - Zabbix, <https://www.zabbix.com/>
 - ApiDeck, <https://www.apideck.com/products/proxy/deno-land>
 - ...
- Käytännössä omassa koodissa on kutsuja kolmannen osapuolen palveluihin, esimerkiksi pluginien kautta
 - esim . Zabbix
 - https://www.zabbix.com/container_images
 -

Demo

- Projekti “project-08-monitoring” sisältää (hyvin) yksinkertaisen monitorointipalvelun toteutuksen.
 - API havaintojen lisäämiseen
 - Mahdollisuus viimeisinten havaintojen listaamiseen
- Sovellus tarkasteltavissa osoitteessa:
 - <https://deno-simple-monitoring.herokuapp.com>
 - API (POST): <https://deno-simple-monitoring.herokuapp.com/api/entries>
- API vastaanottaa JSON-muotoista dataa, jossa merkkijonomuotoinen sivu (site) sekä JSON-objekti data (data). Esim.

```
{"site": "my-site", "data": {"method": "GET"}}
```

Demo

- Projekti “project-08-monitoring” sisältää (hyvin) yksinkertaisen monitorointipalvelun toteutuksen.
 - API havaintojen lisäämiseen
 - Mahdollisuus viimeisinten havaintojen listaamiseen
- Sovellus tarkasteltavissa osoitteessa:
 - <https://deno-simple-monitoring.herokuapp.com>
 - API (POST): <https://deno-simple-monitoring.herokuapp.com/api/entries>
- API vastaanottaa JSON-muotoista dataa, jossa merkkijonomuotoinen sivu (site) sekä JSON-objekti data (data). Esim.

```
{"site": "my-site", "data": {"method": "GET"}}
```

*Näistä lisää
kevällä*

Demo

- Voidaan kokeilla komentoriviltä cURL-komennon avulla (<https://curl.se/download.html>) – alla tehty paikallisesti

*Kaikki
yhdelle
riville!*

```
curl -X POST "http://localhost:3000/api/entries" -H "Content-Type:  
application/json" -d "{\"site\":\"my-site\",\"data\":{\"method\":\"GET\"}}"
```

Demo

- API-pyyynnön tekeminen sovelluksessa `fetch`-komennolla
 - https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch

Demo

- API-pyyynnön tekeminen sovelluksessa `fetch`-komennolla
 - https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch
- Komennolla määritellään osoite, metodi, otsaketiedot, ja pyynnön runko.

Demo

- API-pyynnön tekeminen sovelluksessa `fetch`-komennolla
 - https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch
- Komennolla määritellään osoite, metodi, otsaketiedot, ja pyynnön runko.

```
const data = { site: "my-site", data: { method: "GET" } };
```

```
fetch("osoite", {  
  method: "POST",  
  headers: {  
    "Content-Type": "application/json",  
  },  
  body: JSON.stringify(data),  
});
```

Demo

- API-pyynnön tekeminen sovelluksessa `fetch`-komennolla
 - https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch
- Komennolla määritellään osoite, metodi, otsaketiedot, ja pyynnön runko.

```
const data = { site: "my-site", data: { method: "GET" } };
```

- Minne laitetaan?

```
fetch("osoite", {  
  method: "POST",  
  headers: {  
    "Content-Type": "application/json",  
  },  
  body: JSON.stringify(data),  
});
```

Demo

- API-pyynnön tekeminen sovelluksessa `fetch`-komennolla
 - https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch
- Komennolla määritellään osoite, metodi, otsaketiedot, ja pyynnön runko.

```
const data = { site: "my-site", data: { method: "GET" } };
```

- Minne laitetaan?

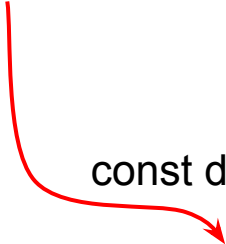
Middleware!

```
fetch("osoite", {  
  method: "POST",  
  headers: {  
    "Content-Type": "application/json",  
  },  
  body: JSON.stringify(data),  
});
```


Demo + hands-on: muokataan sovellusta

- Monitoring middlewaren toteutus – lähetä tieto jokaisesta pyynnöstä osoitteeseen <https://deno-simple-monitoring.herokuapp.com/api/entries>

- Huom! Keksi sovelluksellesi oma nimi (site) ja kokeile lähettää muutakin tietoa – esim. haettu polku.



```
const data = { site: "my-site", data: { method: "GET" } };  
  
fetch("osoite", {  
  method: "POST",  
  headers: {  
    "Content-Type": "application/json",  
  },  
  body: JSON.stringify(data),  
});
```

Edellä käsitelty palvelinpuolta...

- Mutta, sama ongelma myös selainpuolella
 - Sentry, <https://sentry.io/>
 - DataDog, <https://www.datadoghq.com/>
 - LogRocket, <https://logrocket.com/for/how-to-log-client-side-javascript-errors/>

Hyvä muistaa!

- Mikäli sivusto käyttää kolmannen osapuolen palveluita, siitä on hyvä kertoa käyttäjille.
- Sivuille “privacy policy” → kertoo mitä dataa kerätään, miten dataa kerätään, miten dataa käytetään, mihin dataa jaetaan, ymym. Myös perustelut *miksi* dataa kerätään.

Lounas

Tietokanta pilvipalveluita käytettäessä

- Pilvipalveluita käytettäessä tietokanta myös usein pilvipalveluna
- Eräs vaihtoehto ElephantSQL <https://www.elephantsql.com/>
 - Tarkastellaan tätä lyhyesti
- Kun sovellus lähetetään verkkoon, tietokannan tiedot syötetään “salaisuuksina”
 - GitHubiin eri tietokanta kuin Deno Deployhyn →
Toinen testitietokanta, toinen “tuotanto”

Tietokanta pilvipalveluita käytettäessä

Demo

- Pilvipalveluita käytettäessä tietokanta myös usein pilvipalveluna
- Eräs vaihtoehto ElephantSQL <https://www.elephantsql.com/>
 - Tarkastellaan tätä lyhyesti
- Kun sovellus lähetetään verkkoon, tietokannan tiedot syötetään “salaisuuksina”
 - GitHubiin eri tietokanta kuin Deno Deployhyn →
Toinen testitietokanta, toinen “tuotanto”

Miniprojekti: lahjalista

Lahjalista-sovellus pitää kirjaa lahjatoiveista ja niiden esittäjistä. Sovellukseen tulee voida lisätä lahjatoive (lisättäessä lisätään myös lahjatoiveen esittäjä). Sovelluksen tulee listata lahjatoiveet kolmella eri sivulla (jokaisella sivulla lista). Ensimmäisellä sivulla näytetään lahjatoiveet, joita ei ole vielä hankittu. Toisella sivulla näytetään lahjatoiveet, jotka on hankittu, mutta joita ei ole vielä toimitettu. Kolmannella sivulla näytetään lahjatoiveet, jotka on hankittu ja jotka on toimitettu. Ensimmäisellä sivulla näytetään jokaisen lahjatoiveen yhteydessä nappi "Hankittu", jonka painaminen siirtää lahjan seuraavalle sivulle. Toisella sivulla näytetään jokaisen lahjatoiveen yhteydessä nappi "Toimitettu", jonka painaminen siirtää lahjan kolmannelle sivulle.

Miniprojekti: lahjalista

1. Tee sovellus ensin ilman tietokantaa.
2. Sovellus tulee lisätä GitHubiin ja verkkoon (Deno Deploy).
3. Sovellukselle tulee ajaa testit GitHubissa (CI).
4. Kun koko sovellus toimii, tietokannan lisääminen (ElephantSQL)

Lahjalista-sovellus pitää kirjata lahjatoiveista ja niiden esittäjistä. Sovellukseen tulee voida lisätä lahjatoive (lisättäessä lisätään myös lahjatoiveen esittäjä). Sovelluksen tulee listata lahjatoiveet kolmella eri sivulla (jokaisella sivulla lista). Ensimmäisellä sivulla näytetään lahjatoiveet, joita ei ole vielä hankittu. Toisella sivulla näytetään lahjatoiveet, jotka on hankittu, mutta joita ei ole vielä toimitettu. Kolmannella sivulla näytetään lahjatoiveet, jotka on hankittu ja jotka on toimitettu. Ensimmäisellä sivulla näytetään jokaisen lahjatoiveen yhteydessä nappi "Hankittu", jonka painaminen siirtää lahjan seuraavalle sivulle. Toisella sivulla näytetään jokaisen lahjatoiveen yhteydessä nappi "Toimitettu", jonka painaminen siirtää lahjan kolmannelle sivulle.

Hyvä paikka TDD:n harjoittelulle!

Yhteenveto päivästä

Seuraava lähipäivä lyhyesti