

Hello

How to become a developer in less than 60 minutes

Hello

How to become a developer in less than 60 minutes
(jk)

Lars Erik Græsdal-Knutrud - <https://larserik.dev>

- M.Sc. Informatics @ NTNU Trondheim
- Acando/CGI
- Temporal AS
- Backend / Tech. project lead / Fullstack
- Currently on contract with Scoutdi - scoutdi.com





Inspection Live Stream

Drone inspection operations can be live-streamed through the Scout Web Portal when the drone ground station is connected to the internet. Remote viewing the inspection can take place anywhere in the world and reduces the need for people to be physically present in the inspection asset.



Data Visualization

The portal's split screen view allows you to see both the video and 3D Lidar point cloud allowing you to maintain your situational awareness while viewing the inspection data.



Inspection Report Generation

Inspection reports in MS Word format can be create and tailored to your specific communication needs.



Cloud Data Storage

With cloud storage you no longer need to worry about running out of space or what to do with your archived data. We provide a worry-free inspection data storage solution.



Data Analysis

Advanced machine learning and AI tools can be integrated with the portal to assist in identifying crack, corrosion, and surface anomalies.



API Access

Can be used when your data systems need to talk to or access data directly from the Scout Portal.

Agenda

- Consultant life
- Web development basics
- Development lifecycle
 - Design / UX
 - Architecture
 - Development / OPS
- Common architecture components
- Open Source
- Questions and hopefully, answers

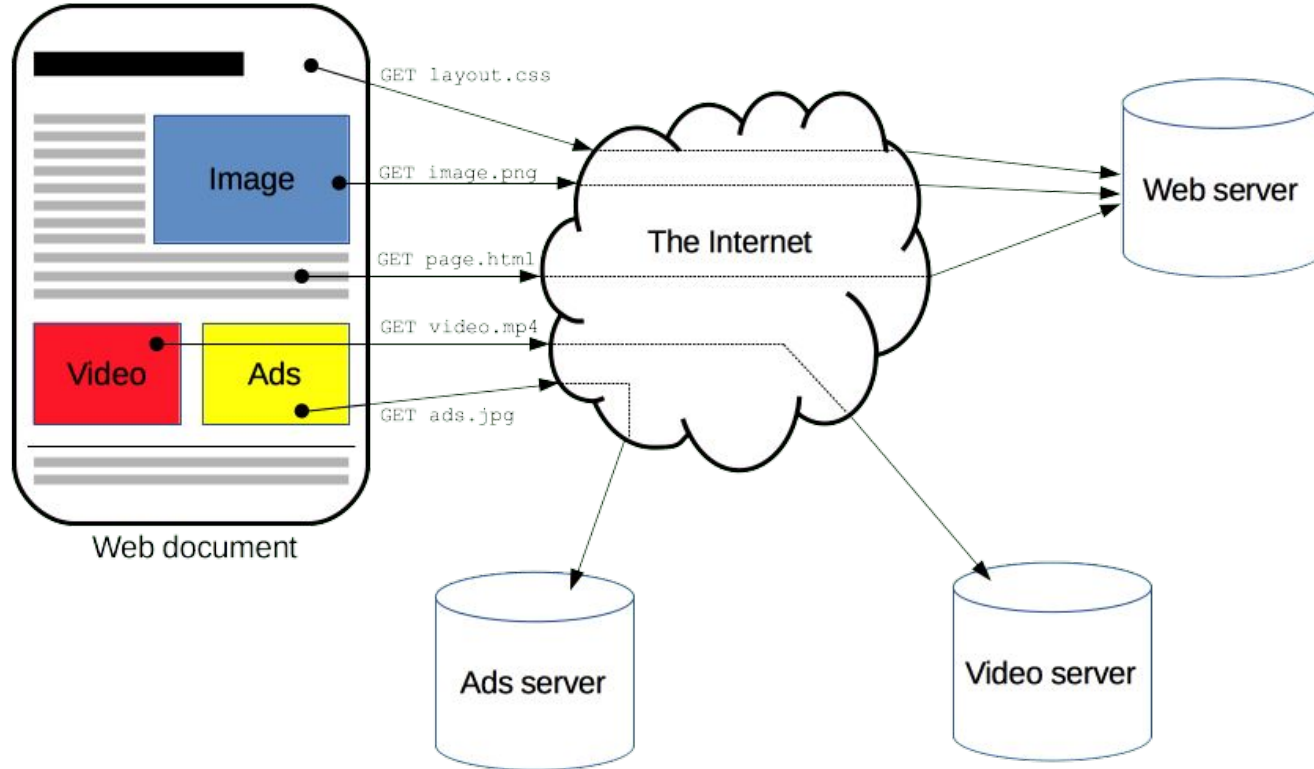
Consultant life

- Being on the team vs. being part of the team
- Startup life vs. enterprise life
- Contracts and predictability
- Skills, certifications and experience
- A consultant is also an advisor

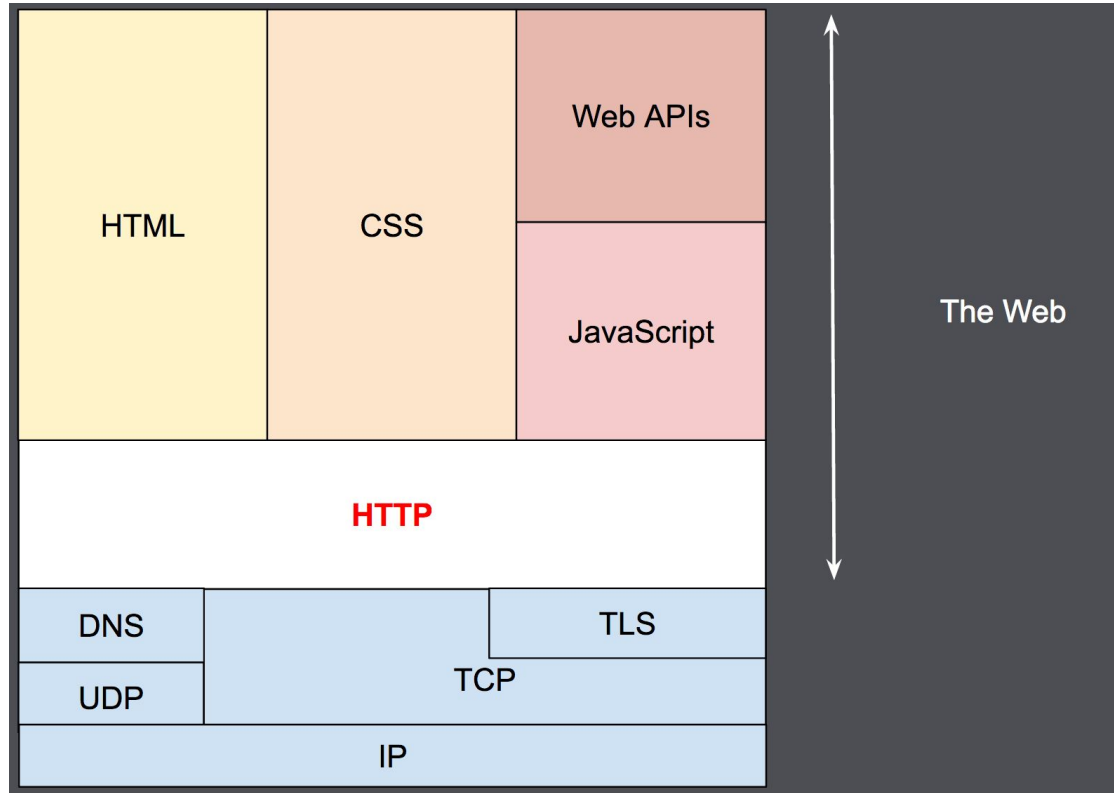
Web development basics

- Single Page Application (SPA) + Back End = <3
- CRUD (Create, Read, Update, Delete)
- Frontend is usually React, Vue or Angular
- HTTP most important protocol in the stack
- JSON / XML used for data transport

HTTP introduction and basics



HTTP and the web



HTTP methods and usage

- GET
 - The GET method requests a representation of the specified resource. Requests using GET should only retrieve data.
- POST
 - The POST method submits an entity to the specified resource, often causing a change in state or side effects on the server.
- PUT / PATCH
 - The PUT method replaces all current representations of the target resource with the request payload. The PATCH method applies partial modifications to a resource.
- DELETE
 - The DELETE method deletes the specified resource.

Real life HTTP examples

POST ▼ <https://portal.scoutdi.com/api/v1/assets/>

Params Authorization ● Headers (10) **Body** ● Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL **JSON** ▼

```
1
2     ... "name": "Test asset",
3     ... "description": "Presentation for FE",
4     ... "url": "https://example.com/"
5
```

```
1
2     "id": "4674ca55-7d8a-445f-82e2-3eb93810599b",
3     "name": "Test asset",
4     "description": "Presentation for FE",
5     "url": "https://example.com/",
6     "image": null,
7     "group": 1
8
```


PATCH ▼ <https://portal.scoutdi.com/api/v1/assets/4674ca55-7d8a-445f-82e2-3eb93810599b/>


Params Authorization ● Headers (10) **Body** ● Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL **JSON** ▼

```
1  [25]
2  ... "url": "https://firstengineers.com/"
3  [26]
```

```
1  [25]
2  "id": "4674ca55-7d8a-445f-82e2-3eb93810599b",
3  "name": "Test asset",
4  "description": "Presentation for FE",
5  "url": "https://firstengineers.com/",
6  "image": null
7  [26]
```

DELETE  <https://portal.scoutdi.com/api/v1/assets/4674ca55-7d8a-445f-82e2-3eb93810599b/>

Params Authorization  Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

Body Cookies Headers (10) Test Results  Status: 204 No Content

Pretty Raw Preview Visualize Text  

1

Development lifecycle













- Architecture vision
- Design / Ux
- Architecture work
- Development / operations


User experience (UX) and design

- Manage stakeholders
- Create and understand personas
- Create sketches and designs
- User testing
- Iterate

Fuzzy design

Sharing settings for "To be shared"

| | | | | |
|---|--------------------|-----------------|---------------|---|
|  | Ola Normann | user@domain.com | Read |  |
|  | Ola Normann | user@domain.com | Read Write |  |
|  | Ola Normann | user@domain.com | Write |  |
|  | Equinor | 12 users | Read |  |
|  | Pending invitation | 0000-0000-0000 | Read |  |
|  | Pending invitation | user@domain.com | Write |  |

 Resend invitation

Remove all

Share with user/group

Email or group sharing token

Access

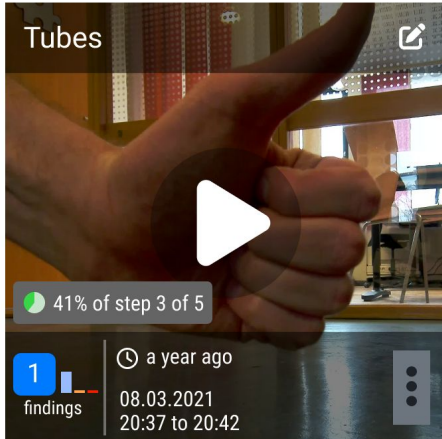
Concrete design and design systems

The screenshot displays the SCOUTDI web application interface. At the top, a dark navigation bar contains the SCOUTDI logo and menu items: LIVE SESSIONS, ASSETS, TARGETS, INSPECTIONS, DOCUMENTATION, USER MANAGEMENT, and APPLICATION MANAGEMENT. Below this, a light grey header area features three dropdown menus for 'Asset' (containing 'Test Asset'), 'Target' (containing 'Test Target'), and 'Inspection' (containing 'Test Inspection'). A red '+' icon is positioned to the right of the 'Inspection' dropdown. The main content area is dark grey and includes a large white rectangular placeholder on the left. On the right side of this area, there is a red toolbar with four icons: 'Edit Inspection', 'Upload Session', 'Reports', and 'Stream Code'. Below the toolbar, a 'Sessions' panel shows '3 sessions in inspection'. It contains two video thumbnails, each with a play button and a timestamp of 10:54:18. A context menu is open over the bottom thumbnail, listing actions: 'Download data', 'Move to inspection', 'Delete', 'Show analyses', and 'Show visualizations'. The bottom thumbnail also includes a '3 year ago' label and a date range from 08.03.2021 to 20.12.2021.

Interactive prototypes

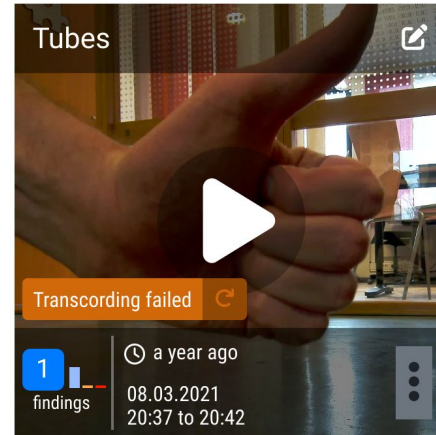
Step 3

Amidst a processing step that we can give a % progress on.
eg: video transcoding



Step 4

When any step fails, we show the user what step failed.

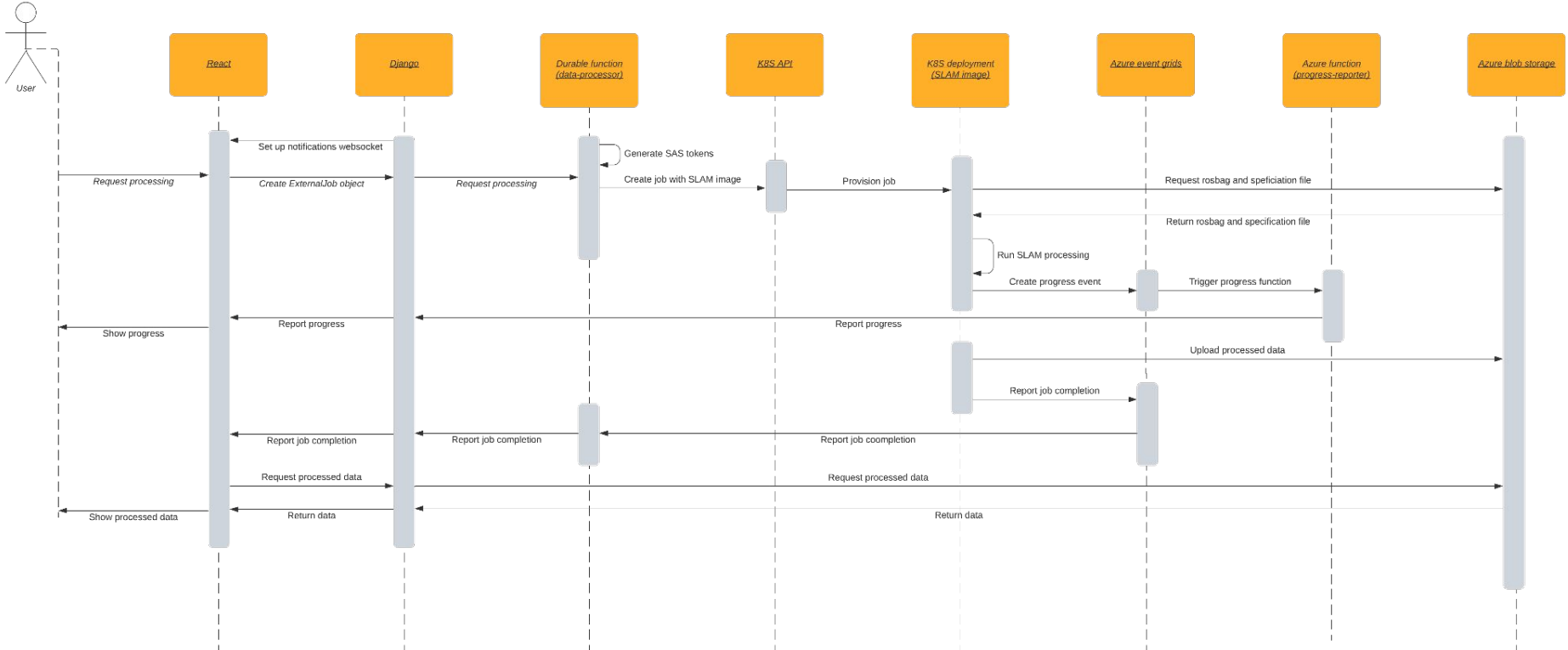


Architecture

- Set principles for development
- Map system needs
- Plan development
- Communicate intent

Diagrams

Infrastructure and flow for SLAM processing



Development

- Realize goals through architecture
 - Update architecture if needed
- Make informed decisions on tech debt
- Solve problems not seen in architecture

Operations

- DevOps
- Security updates
- Logging, Monitoring and alerts

Common architectural components / buzzwords

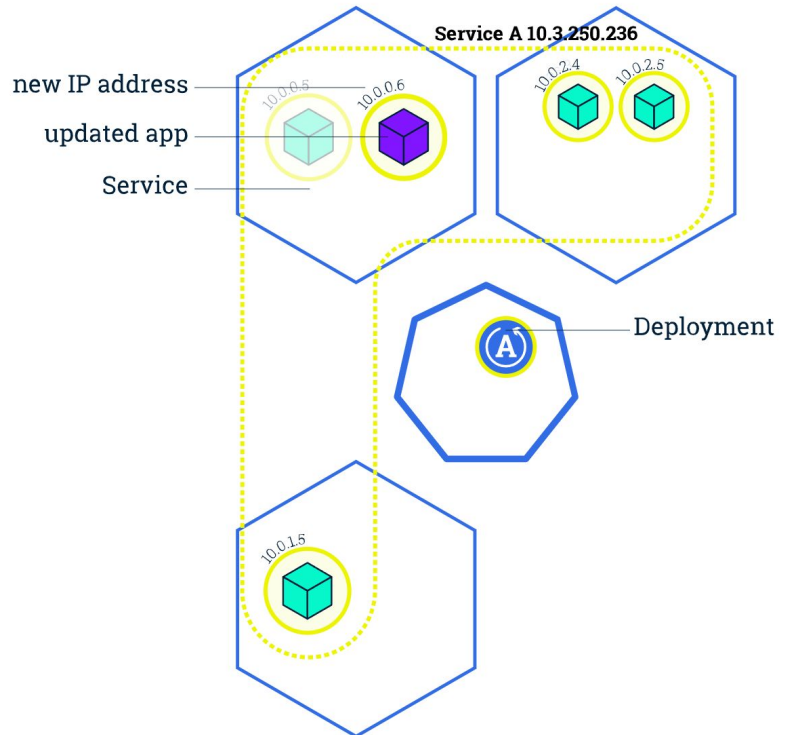
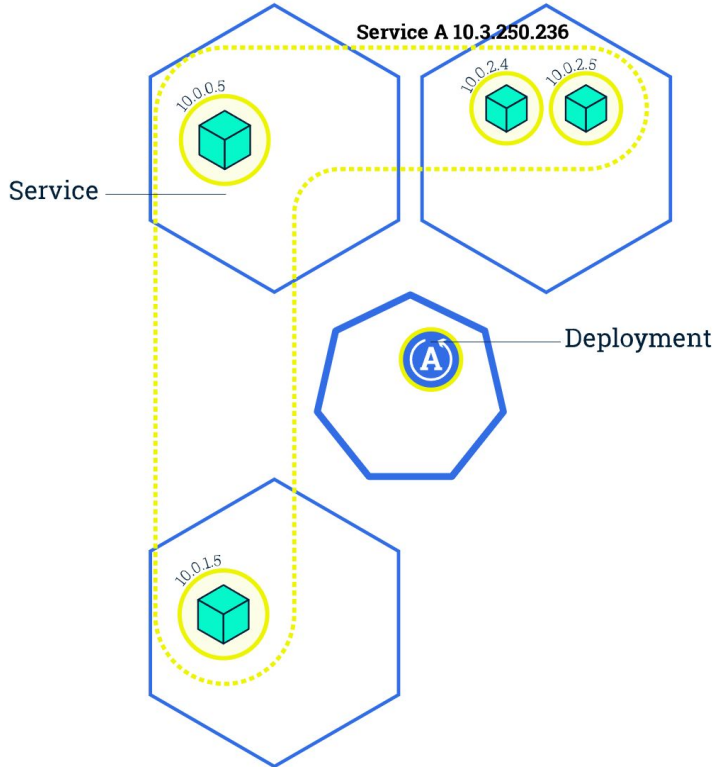
- Containers (Docker)
 - Pack dependencies with application
 - Promotes reuse
 - Cross platform
 - Deployable artifacts

Kubernetes

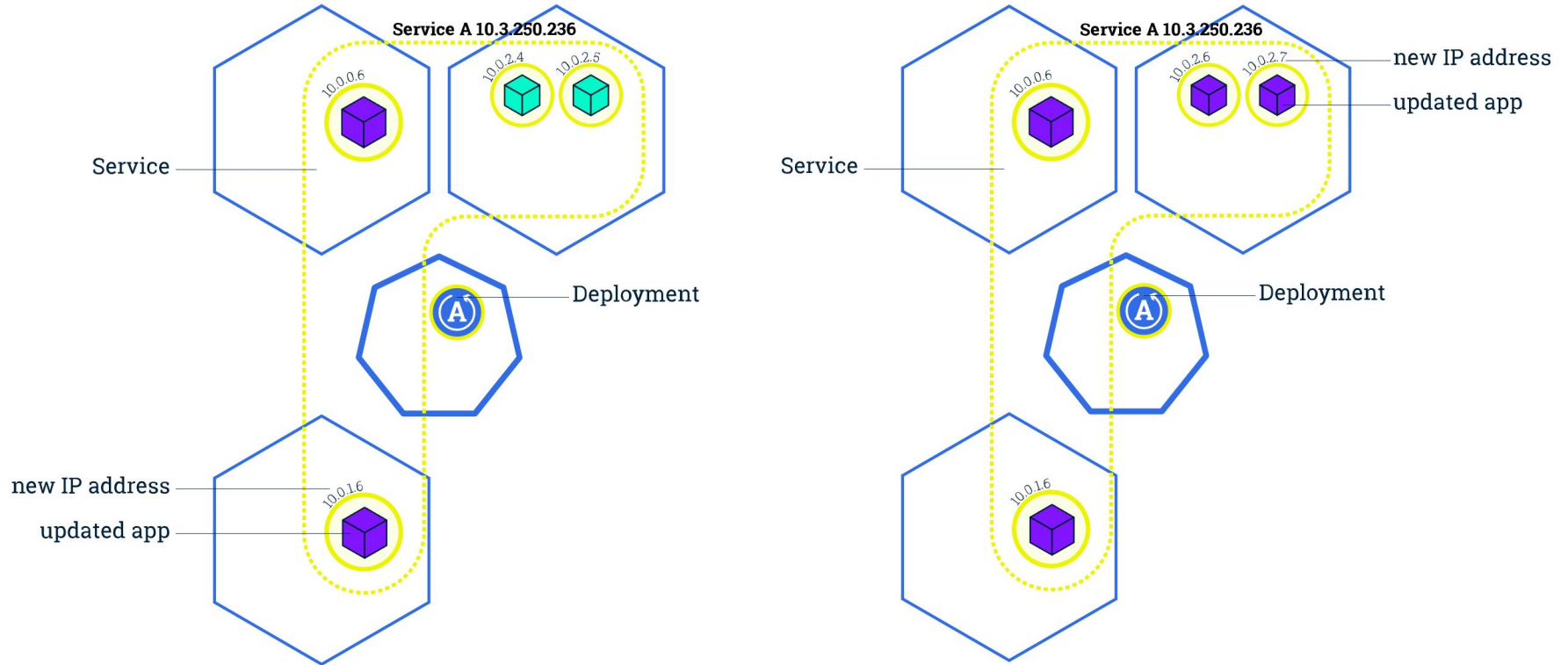
- Container orchestrator
- Open source
- Extensible



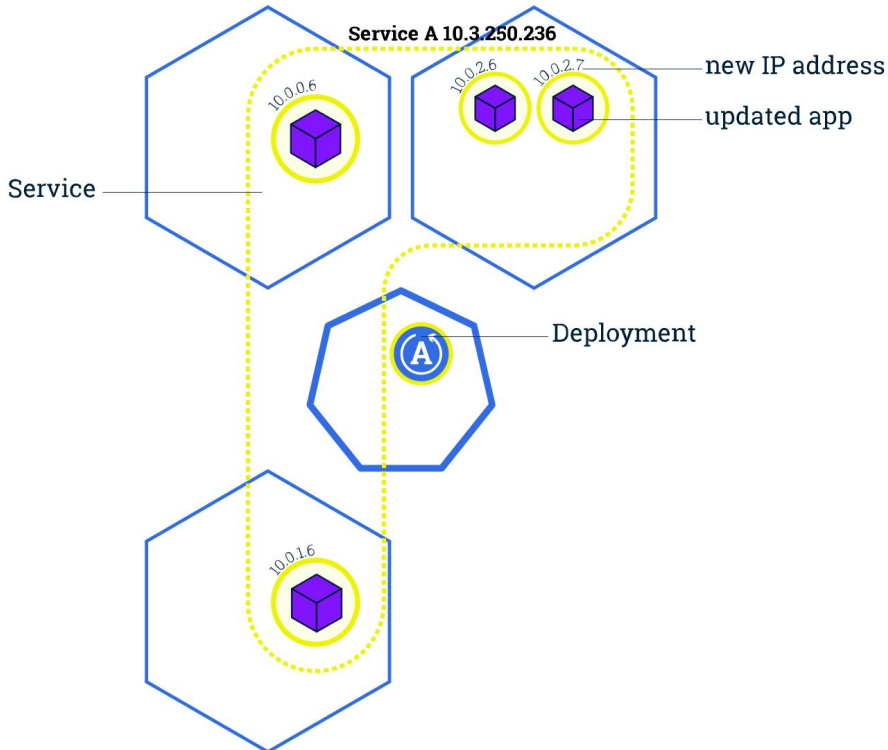
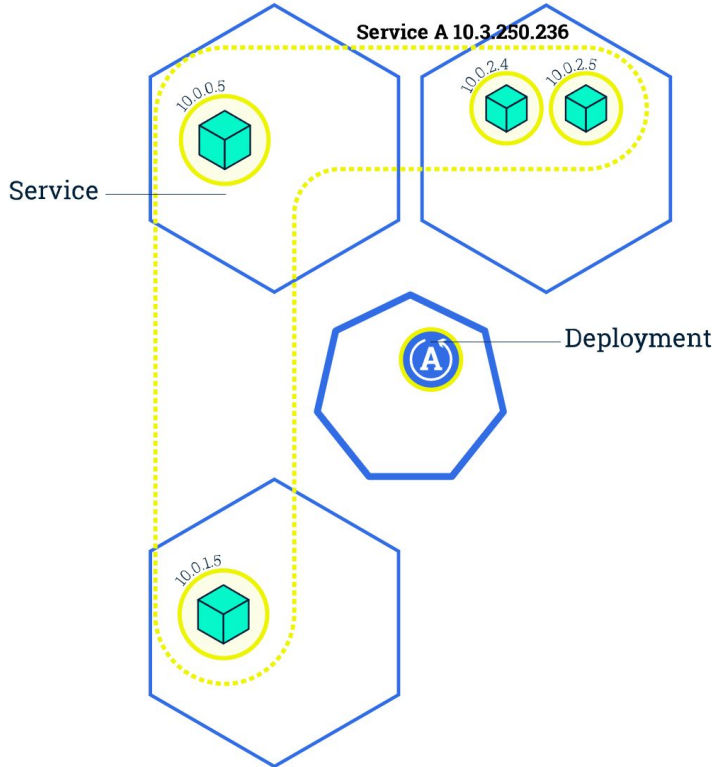
Rolling deployments - initial state



Rolling deployments - final state



Rolling deployments - diff



Infrastructure as code

- Reproducible infrastructure
- Built in audit log
- Tools like Helm give great modularity
- No more miss-clicks

Monolith vs. microservices

Monolith pros:

- Low infrastructure overhead
- Simple architecture
- Easy to test

Microservice pros:

- Language flexibility
- Individually scalable
- Services can be deployed separately

Serverless

- GCS Serverless, Azure Functions and AWS Lambda
- Allows for flexibility in language choices
- No to little time spent on infrastructure
- Might be complex for an entire application

Open Source

From: Linus Benedict Torvalds
Newsgroups: comp.os.minix
Subject: What would you like to see most in minix?
Date: 25 Aug 91 20:57:08 GMT
Organization: University of Helsinki

Hello everybody out there using minix -

I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things).

I've currently ported bash(1.08) and gcc(1.40), and things seem to work. This implies that I'll get something practical within a few months, and I'd like to know what features most people would want. Any suggestions are welcome, but I won't promise I'll implement them :-)

Linus

PS. Yes - it's free of any minix code, and it has a multi-threaded fs. It is NOT protable (uses 386 task switching etc), and it probably never will support anything other than AT-harddisks, as that's all I have :-).

Questions

Sources

- scoutdi.com (Scout promo material)
- developer.mozilla.org (HTTP info and diagrams)
- kode24.no (Article about Brønnøysundregisteret and log4shell)
- kubernetes.io (Rolling deployment diagrams)
- github.com/nais/doc/ (NAIS)
- [techradar](#) (NPM Colors story)