CS-E4004 Project report

Radiodiodi infrastructure containerization

Jan Tuomi & Aarni Halinen
2019 – 2020

Table of contents

Table of contents 1

Abstract 2

Introduction 2

Background 3

Approach and methods 4

Results 7

Conclusions 10

References 11

Appendix 1. 13

Abstract

Cloud-native computing is an approach to building and running containerized, dynamically orchestrated and microservice-oriented software. The approach emphasizes easy reproducibility, resource isolation and utilization, and maintainability.

Radiodiodi is a non-profit seasonal radio station which manages all technological aspects of a radio station. The infrastructure has been in growing hard to understand and maintain, since the services are deployed on a number of non-standardized server machines, operating systems and in complicated networking setups. This study was conducted to introduce comparison of different cloud-native technologies and to present a working implementation to solve the previously mentioned issues.

Introduction

Radiodiodi is a student-driven non-profit seasonal radio station that maintains its own technical infrastructure in the studio, on the web, and in the server room. The Radiodiodi team, which consists only of Aalto students, manages all technical aspects of the project.

The team changes every few years. To keep everything under control for the new experts, the technical handover from the old team to the new should be as painless as possible.

The infrastructure that has been in use for a few years now is growing hard to understand and maintain. Services are deployed on a number of non-standardized server machines, running on a number of different operating systems and environments and in complicated networking setups. Without intervention, such infrastructure will not be at all painless to maintain by new members of the team.

To combat these future issues, Radiodiodi has considered moving to a centralized, cloud-native infrastructure solution that would allow technical team members to manage the whole fleet of deployed services using one system.

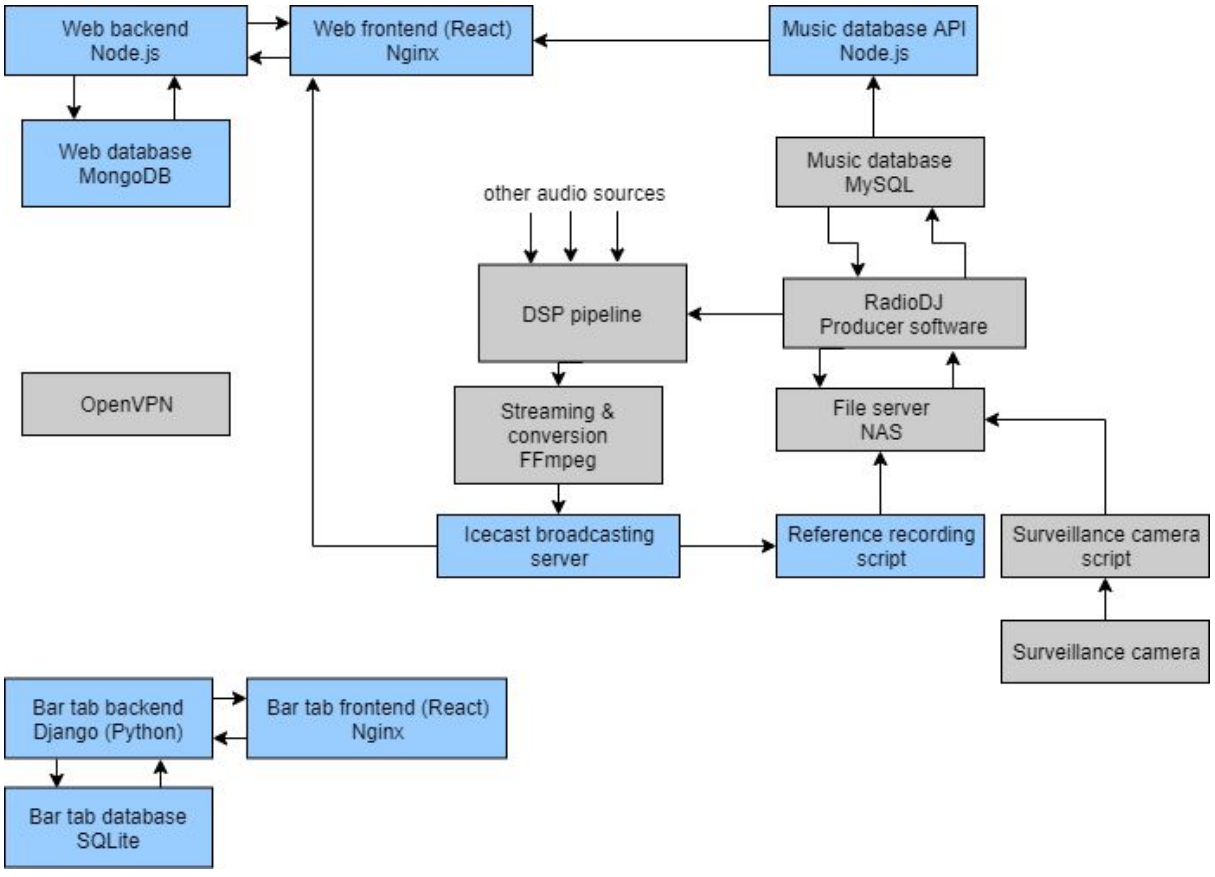
This report will study different options for setting up a centralized service deployment platform for Radiodiodi using cloud-native technologies such as Docker and Kubernetes. The report will consider if such options add value to the organization by evaluating qualitative metrics such as maintainability, ease of use and adaptability.

After evaluating the available options, the report will present a complete practical solution, including a description of adopted technologies and configurations. The practical solution will be presented in the form of version controlled source code, as well as technical documentation. Both will be hosted online and linked to from this report.

Background

Cloud-native computing is an approach to building and running containerized, dynamically orchestrated and microservice-oriented software. Containerization is OS-level virtualization in which a container holds packaged, self-contained, ready-to-deploy parts of applications [12]. The application and the resources needed are bundled together to provide easy reproducibility and resource isolation. Orchestration is automated configuration and management of computer systems and software, and especially in the container context refers to constructing and managing clusters of container-based applications [12]. Notable technologies are Docker and containerd for containerization and Kubernetes and Docker Swarm for orchestration.

The cloud-native approach provides easier deploying of multi-container applications, maintainability with monitoring and automatic fault-tolerance and availability with scaling. Specially for Radiodiodi project, the orchestration of already microservice-oriented architecture provides better networking between the services.



Current topology of Radiodiodi services. Blue-colored services are included in the implementation.

Radiodiodi services can be roughly separated in four different macro-level applications.

- On-premise services, which consist of RadioDJ, DSP pipeline, streaming & conversion FFmpeg and surveillance camera, are running in near vicinity of the studio.
- Icecast broadcasting service for hosting the radio stream
- Website services, which consist of React frontend radiodiodi.fi and Express backend api.radiodiodi.fi and their databases
- Bar tab, which is a small web service for tracking beverage consumption of individual Radiodiodi team members

On-premise services are hard to scale, may require low latency and especially the DSP service utilize proprietary software, so running them in cloud would be counter-intuitive.

The streaming service is an Icecast streaming application installed on Digital Ocean droplet. Reference recording bash script is located on the same droplet.

Another Digital Ocean droplet is used for deploying website and bar tab services. All of the services are behind a Caddy HTTP server. The databases are running in Docker containers, while NodeJS application process manager pm2 is used for deploying the frontend and backend applications.

The deployment process for the services is manual. The user connects to the server with SSH and runs deployment scripts there. The scripts trigger build processes and handle restart of server applications. Environment variables and secrets are located in .env files found on the server repository folders.

In addition to the database Docker images, the website frontend and backend services are also built into Docker images. However, these images are not yet used for other than development purposes. Authors have also experience with Docker containers on Guild of Electrical Engineering's website rewrite project, in which Docker images are used with docker-compose and Docker Swarm for development and production server deployments.

Approach and methods

This project looks to find an answer to the research question: *Which cloud-native technology choices and configurations are most suitable for use in Radiodiodi?* The suitability of options is evaluated qualitatively, using the metrics listed under Metrics.

This report will consider a number of technology options. Comparison of cloud service providers and platforms, such as Google Cloud Platform and Amazon Web Services, is not in the scope of this project. Therefore, business aspects like pricing, resource quotas and organization management are not considered in the evaluation. In addition, a good solution

ought to be cloud service provider independent, so that the infrastructure would not be vendor locked in the event of possible migration in the future.

Evaluation of options will be conducted by qualitatively studying alternatives in the space of plausible technologies, as well as by conducting a questionnaire study to reinforce or find issues in the discovered solution.

Metrics

The following qualitative metrics will be considered.

- Maintainability
    - Is the solution easy to maintain?
    - Is it possible to resolve issues quickly?
- Ease of use
    - Is the solution easy to manage and understand?
- Steepness of learning curve (low > high)
    - How hard is it to learn to use the solution?
- Adaptability
    - Can the solution solve future problems without major issues?
- Debuggability
    - How possible is it to pinpoint the source of a problem?

Qualitative evaluation

The following technologies, grouped by purpose, were considered for the solution. The solution contains one technology from each group.

All options in the list are open-source and suitably licenced for use in the non-profit, low-cost context of Radiodiodi.

Container runtime

- *No container runtime*
- Docker
- containerd
- CRI-O

Orchestrator

- *No orchestrator*
- docker-compose
- Docker Swarm
- Kubernetes

- Kontena Classic
- HashiCorp Nomad

The existing architecture used in Radiodiodi contained two approaches to containerization: some services used the Docker container runtime, deployed using docker-compose, while some services were deployed without using containerization at all.

Out of the available options, Docker was chosen as the best container runtime for the project. Docker, using containerd in its core, has attained a large user base and maturity since its first release in 2013. Extensive, beginner-friendly documentation, widespread use globally and personal experience with the technology lifted it above the other options in the evaluation.

The existing architecture did not make use of an orchestrator. Production deployments were done using docker-compose, which would be better categorized as a multi-container runner instead of an orchestrator, since it is limited in terms of production-grade features. The documentation of docker-compose indicates that the tool is not suitable for production, and as such, it was not considered as orchestrator in the project.

Kontena Classic and HashiCorp Nomad were the most obscure options considered. Neither author had experience with the technologies, and the feature list of either option did not lift them above the rest of the alternatives, i.e., Kubernetes and Docker Swarm, with which the authors had more experience. On the other hand, both Kontena Classic and HashiCorp Nomad are widely used and backed by reputable companies in the cloud technology industry.

Kontena Classic, albeit promising on the surface, is declared *deprecated* on the project's Github page, and as such, is not recommended for new projects. The disadvantages of HashiCorp Nomad were not so clear cut, but having no clear advantage over the more popular alternatives made it a non-optimal choice.

Making a choice between the remaining available options, i.e. Docker Swarm and Kubernetes, turned out to be difficult. Comparison of the two technologies can be found under Results, alongside the outcomes of the questionnaire study.

Questionnaire study

To challenge the authors' conclusions on the research problem, a small-scale questionnaire study was conducted. The study consisted of questions related to the technical infrastructure of Radiodiodi and possible adoptable technologies, such as those presented under Evaluation. The study was aimed towards people with experience of being in the Radiodiodi team and with knowledge about cloud-native technologies. The study was conducted in Finnish.

The questionnaire study presented a possibility to give a numeric rating for suitability to a number of cloud-native technologies. This rating tells how suitable the subject thinks that technology would be for the project.

The structure of the questionnaire can be found in Appendix 1. The results of the study are presented under Results.

Results

Qualitative evaluation results

Qualitative evaluation (under Approach and Methods) yielded two best options, i.e. Kubernetes and Docker Swarm. These options were compared to each other. The pros and cons and a comparison of qualitative metrics of both technologies are presented in the tables below.

	Docker Swarm	Kubernetes
Pros	+ Included with Docker installation + Same API as Docker + Faster deployment and scaling	+ De facto industry-standard tool + Large ecosystem of plugins
Cons	- Limited in secret management - Limited in rollout features	- Steep learning curve - Usually deemed too complex for small projects - Choosing a distribution non-trivial

Table. Pros and Cons of Docker Swarm and Kubernetes. [3] [4]

Metric	Docker Swarm	Kubernetes
Maintainability	Well documented features. Common issues solved readily with community content. Low apparent system complexity. Frequent updates. Easy to maintain.	Well documented features. Common issues solved with community content. High apparent system complexity. Frequent updates. More difficult to maintain.
Ease of use	Uses same CLI as Docker, same common language.	Requires knowing of Kubernetes CLI tool in addition to Docker CLI.
Learning curve	Gentle learning curve.	Steeper learning curve because of apparent complexity.

Adaptability	Uses same, simple, YAML configuration as docker-compose. Secrets have to be read from files on container start.	More complex YAML configurations. Secrets are usable at runtime from environment variables.
Debuggability	Local execution possible after setupping swarm node locally. Monitoring services exist, but require some networking setups.	Local execution possible with local cluster, shipped with some Docker installations. Dashboard container created by Kubernetes authors.

Table. Qualitative properties of Docker Swarm and Kubernetes. [1] [2] [3] [4]

Questionnaire study results

The questionnaire study resulted in the following data (N = 4).

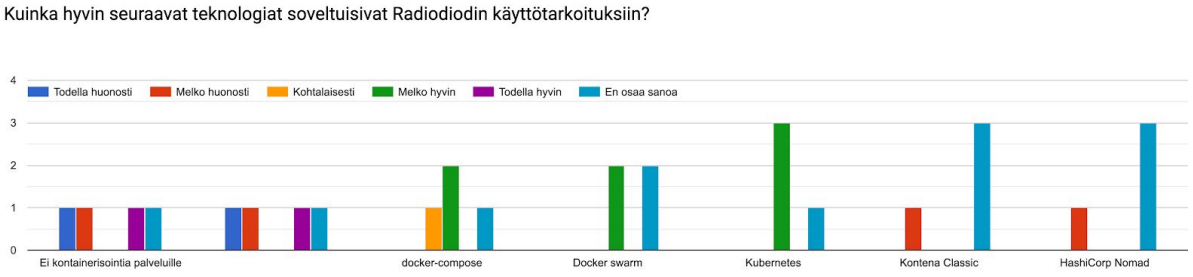


Figure. A bar plot of the questionnaire study results.

The collected results can be seen in the table below, translated to English. Answers were scored 1–5, with 1 being *very badly* and 5 being *very well*. *I don't know* votes do not count towards the average score or the number of votes.

How well would the following technologies suit the needs of Radiodiodi?

Technology name	Average suitability score	# of votes
No containerization	2,67	3
Docker	2,67	3
docker-compose	3,67	3
Docker Swarm	4,00	2
Kubernetes	4,00	3

Kontena Classic	2,00	1
HashiCorp Nomad	2,00	1

Table. Average suitability of technologies based on the results of the questionnaire study.

Since the sample size is so low, it is not reasonable to calculate statistical properties, such as p-values, effect sizes or standard error. However, we can informally compare the results with the qualitative evaluation results to see if they are clearly in conflict.

By comparing the average suitability scores, the questionnaire study yielded two best options: Docker Swarm and Kubernetes. The worst options were Kontena Classic and HashiCorp Nomad. This is similar to the results of the authors' qualitative evaluation, so it is not in conflict with the authors' conclusions.

Outcome

Based on qualitative evaluation and the results of the questionnaire study, the technologies selected for the project are the following.

Container runtime
Docker
Orchestrator
Kubernetes

These technologies will be used to construct a working solution.

Implementation

The solution was implemented in three steps. First every application and shell script was containerized with Docker. Some shell scripts were only found on server storage, so this step also included copying these to version control. Also some modifications were done to existing Docker containers, since these were built only for one execution context. Information about the context is passed to the containers with runtime variables [13].

Most of the Dockerized applications are open source, and can be found at https://gitlab.com/radiodiodi.

Kubernetes configuration can be found in a public repository at https://gitlab.com/radiodiodi/k8s-conf-public. The original development repository is kept private, since the Git history contains secrets.

Kubernetes local cluster was used for developing the first iteration of Kubernetes configuration. Dockerized applications were hosted in Gitlab container registry. A continuous integration pipeline launched by Git tags was used for deploying images to the registry.

A lightweight Kubernetes distribution k3s was chosen for Cloud prototype and installed on Digital Ocean droplet. Since resources such as volumes and ingress networking differed from local development, the configurations were branched at this point. The configurations are located in own directories in the k8s-conf repository.

Conclusions

Research question

This project looked to find an answer to the research question: *Which cloud-native technology choices and configurations are most suitable for use in Radiodiodi?*

Through multiple forms of evaluation, the report arrived at a candidate solution for the problem. The solution was analysed methodically and was found to be suitable for the use case.

Development on the solution will continue after this project to adapt it to future challenges.

References

[1] *Swarm mode overview*. Official Docker documentation. https://docs.docker.com/engine/swarm/. Visited 19.11.2019.

[2] *Production-Grade Container Orchestration*. Official Kubernetes website. https://kubernetes.io/. Visited 19.11.2019.

[3] Hind Naser, *Kubernetes Vs. Docker Swarm: A Comparison of Containerization Platforms*. Vexxhost blog. https://vexxhost.com/blog/kubernetes-vs-docker-swarm-containerization-platforms/, Visited 19.11.2019.

[4] Soumyajit Dutta, Kuberne*tes vs Docker Swarm. Who's the bigger and better?* Medium. https://medium.com/faun/kubernetes-vs-docker-swarm-whos-the-bigger-and-better-53bbe76b9d11. Visited 19.11.2019.

[5] Official containerd website. https://containerd.io/. Visited 19.11.2019.

[6] *Kontena Classic – The classic, developer friendly container platform*. Official Kontena website. https://www.kontena.io/classic. Visited 19.11.2019.

[7] *HashiCorp Nomad: Deploy and Manage Any Containerized, Legacy, or Batch Application*. Official HashiCorp Nomad website. https://www.nomadproject.io/. Visited 19.11.2019.

[8] *CRI-O: Lightweight container runtime for Kubernetes.* Official CRI-O website. https://cri-o.io/. Visited 19.11.2019.

[9] *Docker overview*. Official Docker documentation. https://docs.docker.com/engine/docker-overview/. Visited 19.11.2019.

[10] *Overview of Docker Compose.* Official Docker documentation. https://docs.docker.com/compose/. Visited 19.11.2019.

[11] David Linthicum, *The essential guide to software containers for application development.* TechBeacon. https://techbeacon.com/enterprise-it/essential-guide-software-containers-application-development. Visited 19.11.2019.

[12] Claus Pahl et al., *Cloud Container Technologies: a State-of-the-Art Review.* IEEE. 2017.

[13] Jan Tuomi, *How to access environment variables from a built frontend application in an Nginx container.* Medium. https://medium.com/@jans.tuomi/how-to-use-environment-variables-in-a-built-frontend-application-in-an-nginx-container-c7a90c011ec2. Visited 11.1.2020.

Appendix 1

Questionnaire study structure

Name (not mandatory)

How well do you know the technical architecture of Radiodiodi? (such as servers, network, services, resources)

List three concrete problems in the Radiodiodi infrastructure. The problems can be related to any part of the whole.

What do you think of the plan to migrate the majority of Radiodiodi services to 3rd party cloud platform providers?

How well do you know products/solutions based on container technology? (e.g. Docker, Docker Swarm, Kubernetes)

What do you think of the plan to migrate the majority of Radiodiodi services onto a container-based production platform?

How well would the following technologies suit the needs of Radiodiodi?

1. Very badly
2. Somewhat badly
3. Neutral
4. Somewhat well
5. Very well
6. I don’t know

Was some option missing?

If you presented strong support or opposition toward some options, provide arguments.