Physics and Imaging in Radiation Oncology 31 (2024) 100607

Contents lists available at ScienceDirect

Physics and Imaging in Radiation Oncology

journal homepage: www.sciencedirect.com/journal/physics-and-imaging-in-radiation-oncology

# RadDeploy: A framework for integrating in-house developed software and artificial intelligence models seamlessly into radiotherapy workflows

Mathis Ersted Rasmussen [a,b,c,*], Casper Dueholm Vestergaard [a,c], Jesper Folsted Kallehauge [a], Jintao Ren [a,c], Maiken Haislund Guldberg [a], Ole Nørrevang [a], Ulrik Vindelev Elstrøm [a], Stine Sofia Korreman [a,b,c]

[a] Danish Centre for Particle Therapy, Aarhus University Hospital, Palle Juul-Jensens Boulevard 25, 8200 Aarhus N, Denmark
[b] Department of Oncology, Aarhus University Hospital, Palle Juul-Jensens Boulevard 35, 8200 Aarhus N, Denmark
[c] Department of Clinical Medicine, Aarhus University Hospital, Palle Juul-Jensens Boulevard 99, 8200 Aarhus N, Denmark

ARTICLE INFO

*Keywords:*
Radiotherapy
In-house
Deploy
Container
DAG
Artificial intelligence

ABSTRACT

The use of and research in automation and artificial intelligence (AI) in radiotherapy is moving with incredible pace. Many innovations do, however, not make it into the clinic. One technical reason for this may be the lack of a platform to deploy such software into clinical practice. We suggest RadDeploy as a framework for integrating containerized software in clinical workflows outside of treatment planning systems. RadDeploy supports multiple DICOM as input for model containers and can run model containers asynchronously across GPUs and computers. This technical note summarizes the inner workings of RadDeploy and demonstrates three use-cases with varying complexity.

## 1. Introduction

The clinical use of and research in computers for automation and artificial intelligence (AI) in radiotherapy is moving with incredible pace. AI is currently revolutionizing auto-segmentation of organs-at-risk [1–5] and holds great promise for auto-segmentation of target structures [6–9], dose prediction [10–14], outcome prediction [15] and many other complex tasks in radiotherapy. While some of these technologies are available from commercial vendors, the cutting edge of research is not. Furthermore, commercial products might not comply with local standards and preferences. These issues are sometimes solved with in-house developed tools.

Many innovations in healthcare do not make it into clinical practice [16]. In radiotherapy, a technical reason for this may be the lack of a platform to deploy the in-house developed software. Many treatment planning systems (TPSs) have scripting capabilities, through which some software can run. Using such tools might lead to implementations being tightly coupled to the ecosystem of that particular TPS and other local IT infrastructures. Hence, if a piece of software is successfully implemented locally it might not be transferable to other institutions which may inhibit development and validation of the tool.

The challenge of executing software reproducibly in different computational environments is not confined to radiotherapy. During the last decade, *containers* have revolutionized the way software is developed and deployed [17]. Containers can be thought of as "frozen" computer environments which execute the same way on different computers [18]. While containers solve the reproducibility issue, they do, however, not offer a solution for implementation – this must be tailored to the specific domain.

We propose RadDeploy as a simple-to-configure, yet flexible and scalable framework to facilitate seamless implementations of in-house developed software encapsulated in containers. The aim of this work was to investigate the usability of RadDeploy for deploying three in-house developed AI-models with varying complexity in clinical practice.

## 2. Material and methods

### 2.1. Terminology

In RadDeploy, a *model* refers to a unit of software which is

* Corresponding author at: Danish Centre for Particle Therapy, Aarhus University Hospital, Palle Juul-Jensens Boulevard 25, 8200 Aarhus N, Denmark.
*E-mail addresses:* mathis.rasmussen@rm.dk (M. Ersted Rasmussen), casper.dueholm.vestergaard@rm.dk (C. Dueholm Vestergaard), jespkall@rm.dk (J. Folsted Kallehauge), jintaoren@clin.au.dk (J. Ren), maikgu@rm.dk (M. Haislund Guldberg), ole.noerrevang@auh.rm.dk (O. Nørrevang), ulrik.vindelev.elstroem@auh.rm.dk (U. Vindelev Elstrøm), stine.korreman@clin.au.dk (S. Sofia Korreman).

https://doi.org/10.1016/j.phro.2024.100607
Received 29 April 2024; Received in revised form 20 June 2024; Accepted 27 June 2024
Available online 2 July 2024

encapsulated in a Docker container [19]. A model could be an in-house developed AI model for auto-segmentation, dose prediction or optimization but, in fact, any piece of software that can be wrapped in a Docker container qualifies as a model.

A *flow* refers to the passage of a set of DICOM files through the system. A flow consists of three major parts: *triggers*, *models* and *destinations*. *Triggers* define DICOM tag values which, when found in the received files, will trigger the flow. *Models* define which and how Docker containers should be executed. *Destinations* specify DICOM nodes to which outputs should be sent.

### 2.2. Architecture and services

RadDeploy consists of microservices each dedicated to do isolated tasks. Services communicate through message queues provided by a message broker. Hence, services receive messages from specified queues, perform some operations, and publish results to other queues. More details may be found in the repository wiki [20].

#### 2.2.1. Storage service class provider (SCP)

The SCP is the entrypoint for DICOM files into RadDeploy (Fig. 1). It acts like any other SCP to which DICOM files can be sent from TPSs and alike. It puts all received files in a tarball [21] (type of archive file) and sends it to a file storage service (not to be discussed further). Subsequently, the SCP publishes a unique identifier (UID) of the tarball.

#### 2.2.2. Fingerprinter

This service subscribes to messages coming from the SCP. When it receives a message, it determines which of the DICOM files from the tarball are eligible for which flows. When a set of files are found to match the triggers for a particular flow, a new tarball containing only the relevant files is sent to the file storage and the corresponding tarball UID is published along with the flow definition (see section 2.1).

#### 2.2.3. Scheduler

This service receives messages from the fingerprinter and consumers (see section 2.2.4). It orchestrates the queuing of models eligible for execution and determines when a flow is finished. It publishes finished flows and the UID of the resulting tarball.

#### 2.2.4. Consumer

This service handles an arbitrary number of workers, whose purpose is to execute models as specified in the flow definition. Messages are retrieved from CPU/GPU queues to which the scheduler publishes.

#### 2.2.5. Storage service class user (SCU)

The SCU subscribes to finished flows published by the scheduler. It sends all DICOM files contained in the resulting tarball to the DICOM nodes (SCPs) defined in *destinations* in the flow definition. Such destinations are usually TPSs, Picture Archiving and Communication Systems (PACSs) or even the SCP of RadDeploy itself.

### 2.3. Deployment

RadDeploy can be deployed with a Docker compose file (supplementary figure S1). For this, the host machine must have the Docker engine installed with linux as the backend. If models require GPU acceleration, NVIDIA drivers and the nvidia-container-toolkit [22] must be installed. An example compose file can be downloaded off Gitlab and RadDeploy may be spun up with a single command in a terminal. A detailed guide can be found in the Wiki of the RadDeploy repository [20].

## 3. Results

At the time of writing, RadDeploy is a relatively young platform. However, it has already proven capable in a variety of tasks. In this section, we present three use-cases of increasing complexity and demonstrate how RadDeploy integrates these in our clinical/research environment.

### 3.1. Single modality auto-segmentation

Auto-segmentation models are often executed on a single image modality such as CT or MRI. In our clinic, we run several in-house developed auto-segmentation models to produce structures that are used as templates for clinical organs-at-risk contouring. When a patient is scanned, the DICOM image files are automatically exported from the CT/PET or MRI scanners to a standalone server running RadDeploy. Here, relevant flows are triggered and the resulting DICOM Radiotherapy Structure Sets (RTSS) are sent to the TPS. An example of the flow definition for this may be found in supplementary figure S2.

As a further example, the wiki provides a guide on how to setup RadDeploy to run TotalSegmentator [23] for single modality auto-

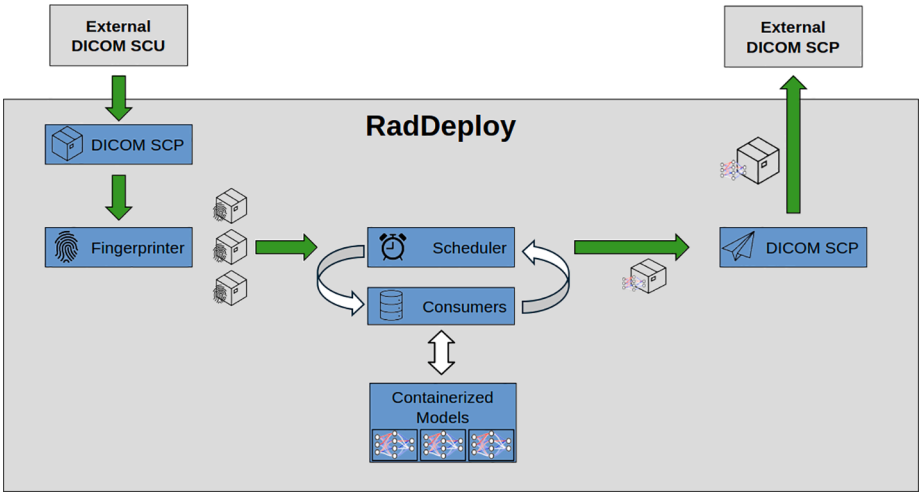


**Fig. 1.** Schematic representation of RadDeploy. Gray boxes represent logical computer units and blue boxes represent services. Arrows indicate the message flow through the stack. Note that under the hood, messages published to and subscribed from the underlying message broker (not shown) and files are posted and retrieved from a file storage-service (not shown). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

segmentation [24].

### 3.2. Multi modality auto-segmentation

In multi modality auto-segmentation several co-registered images are used to predict contours. Multi modality auto-segmentation is generally not offered commercially and are therefore often in-house developments.

A prospective randomized controlled trial is being conducted at our institution to evaluate the usefulness of deep learning contours as templates for delineation of primary and nodal gross tumor volumes (GTV-T and −N) of head-and-neck cancer patients [25]. The model input is four imaging modalities (CT, PET, T1- and T2-weighted MRI) which are manually pushed to RadDeploy from Aria Eclipse (Varian Medical Systems, Palo Alto, CA, USA). The model consists of multiple steps. Firstly, MRIs are deformably registered to the CT, secondly, the GTVs are predicted, and thirdly, a randomization determines if the GTV predictions or empty structures are sent to the TPS for clinical contouring. Additionally, patient identifiers and model data is pushed to a RedCap database via the HTTP API. An example of the flow definition may be found in supplementary figure S3.

### 3.3. Synthetic CT generation

Proton radiotherapy is highly sensitive to anatomical changes which may arise during the period of treatment [26,27]. Daily cone beam CTs (CBCTs) are obtained at the gantry for the purpose of patient positioning and evaluation of anatomical changes. Unfortunately, the image quality of CBCTs is often insufficient for performing proton dose calculations accurate enough to assess the need for replanning. Therefore, the quality of proton dose calculations on synthetic CT scans (sCT) made from daily CBCT scans is currently being investigated at our department using an in-house developed deep learning model.

RadDeploy triggers the flow when receiving a planning CT, a CBCT and a rigid registration file. The output is a DICOM sCT containing an inferred anatomy of the patient. An example of the flow definition may be found in supplementary figure S4.

## 4. Discussion

In this technical note, we have presented RadDeploy which is a framework for deploying containerized software seamlessly into clinical workflows. We have demonstrated three use-cases which vary in complexity and maturity and RadDeploy has proven able to perform stably across these.

Apart from RadDepoy, at least one other framework exists for integrating containerized models [28]. While the purpose is the same, it is achieved differently. Hence, RadDeploy should be seen as an alternative solution with its own set of pros and cons.

A unique feature of RadDeploy is the microservice architecture. Microservices provide a high degree of decoupling. As long as the structure of messages that are flowing between the services is strictly followed, maintenance of existing and development of new services are possible without interfering with the rest of the system. Furthermore, services like the SCP, fingerprinter, consumer and SCU are stateless, and may therefore be scaled horizontally across multiple computers and GPUs. RadDeploy can easily be extended with new services providing additional functionality as these can seamlessly be “mounted” into the stack by subscribing/publishing to specific queues. A notable disadvantage of the microservice architecture is that there is no central database or service that knows the entire state of the system. This increases the risk of messages being lost unnoticed due to power outage or network failure. Furthermore, it is essentially impossible to test every state of the system as messages flow asynchronously between services.

Although not demonstrated here, flows may be designed as Directed Acyclic Graphs (DAGs) represented as models being nodes and in- and output mounts being edges. When flows are designed as DAGs, model containers are scheduled for execution asynchronously when the required inputs are available. This would be useful in the multi-modality auto-segmentation flow as it contains two time consuming and computationally heavy tasks: (1) CPU-based image registration, and (2) GPU-based five-fold prediction of target structures. Hence, the current all-in-one container could be decomposed into a CPU container and five GPU containers, which would be executed asynchronously and potentially across multiple computers/GPUs and thereby ensuring better utilization of computational resources. It is, however, not done currently for the trial, as the model container was developed before DAG scheduling was implemented in RadDeploy. An illustrative flow definition of DAG scheduling can be found in supplementary figure S5.

There is still room for improvement of RadDeploy, and future developments might include a graphical user interface for setup, configuration and maintenance. Currently, the database of the scheduler can be visualized using a dashboard service like Grafana [29] (Fig. 2) which is useful for end-users to keep track of flows but it does not provide any features for controlling jobs. Furthermore, distributed DAG scheduling is a complex task which in the future might be handled better by a third party tool such as Argo Workflows [30], Apache AirFlow [31] or Celery [32].

All stable versions of RadDeploy are made available as public Docker images in the Gitlab container registry [20]. Thus, it is possible to set up the default stack within minutes. Thereby, resources can be put into development, testing, validation and documentation of models without worrying about deployment; as long as the software can be wrapped in a Docker container and has DICOM files as input, RadDeploy can be used. If the default stack does not accommodate the research/clinical requirements, RadDeploy is open source and may freely be modified and extended with new services. Feature requests, requests for support and contributions to the main repository and the wiki are also warmly welcome.

In conclusion, RadDeploy is a microservice-based framework which allows deployment of in-house developed software as Docker containers in radiotherapy workflows. It offers a comprehensive triggering functionality which enables execution of multi-modality models. Furthermore, flows can contain multiple model containers as directed acyclic graphs which allows asynchronous model execution across multiple GPUs and computers.

## Funding

This work is funded with salary for MER from Aarhus University and Aarhus University Hospital.

## CRediT authorship contribution statement

**Mathis Ersted Rasmussen:** Conceptualization, Methodology, Software, Writing – original draft, Visualization, Project administration. **Casper Dueholm Vestergaard:** Software, Methodology, Validation, Writing – original draft. **Jesper Folsted Kallehauge:** Conceptualization, Methodology, Software, Validation. **Jintao Ren:** Methodology, Writing – original draft, Visualization. **Maiken Haislund Guldberg:** Investigation, Software. **Ole Nørrevang:** Writing – original draft, Funding acquisition. **Ulrik Vindelev Elstrøm:** Methodology, Validation. **Stine Sofia Korreman:** Conceptualization, Writing – original draft, Supervision, Project administration, Funding acquisition.

## Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

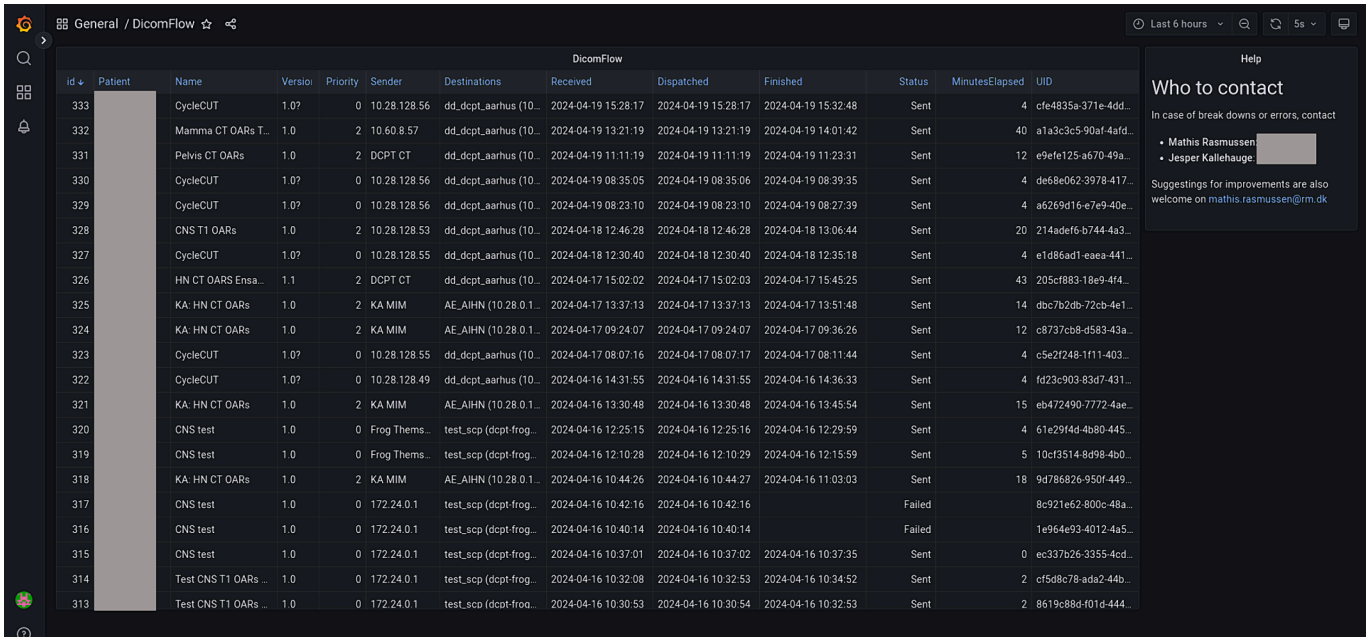


**Fig. 2.** An example of a Grafana dashboard made from the database of the scheduler service.

## Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.phro.2024.100607.

## References


[1] Chen W, Li Y, Dyer BA, Feng X, Rao S, Benedict SH, et al. Deep learning vs. atlas-based models for fast auto-segmentation of the masticatory muscles on head and neck CT images. Radiat Oncol Lond Engl 2020;15:176. https://doi.org/10.1186/s13014-020-01617-0.

[2] Brouwer CL, Boukerroui D, Oliveira J, Looney P, Steenbakkers RJHM, Langendijk JA, et al. Assessment of manual adjustment performed in clinical practice following deep learning contouring for head and neck organs at risk in radiotherapy. Phys Imaging Radiat Oncol 2020;16:54–60. https://doi.org/10.1016/j.phro.2020.10.001.

[3] Choi MS, Choi BS, Chung SY, Kim N, Chun J, Kim YB, et al. Clinical evaluation of atlas- and deep learning-based automatic segmentation of multiple organs and clinical target volumes for breast cancer. Radiother Oncol J Eur Soc Ther Radiol Oncol 2020;153:139–45. https://doi.org/10.1016/j.radonc.2020.09.045.

[4] Lustberg T, van Soest J, Gooding M, Peressutti D, Aljabar P, van der Stoep J, et al. Clinical evaluation of atlas and deep learning based automatic contouring for lung cancer. Radiother Oncol 2018;126:312–7. https://doi.org/10.1016/j.radonc.2017.11.012.

[5] Costea M, Zlate A, Durand M, Baudier T, Grégoire V, Sarrut D, et al. Comparison of atlas-based and deep learning methods for organs at risk delineation on head-and-neck CT images using an automated treatment planning system. Radiother Oncol J Eur Soc Ther Radiol Oncol 2022;177:61–70. https://doi.org/10.1016/j.radonc.2022.10.029.

[6] Ren J, Eriksen G, Nijkamp J, Stine &, Korreman S, Grau Eriksen J, et al. Comparing different CT, PET and MRI multi-modality image combinations for deep learning-based head and neck tumor segmentation n.d. https://doi.org/10.1080/0284186X.2021.1949034.

[7] Kunkyab T, Bahrami Z, Zhang H, Liu Z, Hyde D. A deep learning-based framework (Co-ReTr) for auto-segmentation of non-small cell-lung cancer in computed tomography images. J Appl Clin Med Phys 2024;25:e14297.

[8] Cho H, Lee JS, Kim JS, Koom WS, Kim H. Empowering Vision Transformer by Network Hyper-Parameter Selection for Whole Pelvis Prostate Planning Target Volume Auto-Segmentation. Cancers 2023;15:5507. https://doi.org/10.3390/cancers15235507.

[9] Wang Y, Lombardo E, Huang L, Avanzo M, Fanetti G, Franchin G, et al. Comparison of deep learning networks for fully automated head and neck tumor delineation on multi-centric PET/CT images. Radiat Oncol Lond Engl 2024;19:3. https://doi.org/10.1186/s13014-023-02388-0.

[10] Zhang Y, Li C, Zhong L, Chen Z, Yang W, DoseDiff WX. Distance-aware Diffusion Model for Dose Prediction in Radiotherapy. IEEE Trans Med Imaging 2024;PP.. https://doi.org/10.1109/TMI.2024.3383423.

[11] Gao Y, Gonzalez Y, Nwachukwu C, Albuquerque K, Jia X. Predicting treatment plan approval probability for high-dose-rate brachytherapy of cervical cancer using adversarial deep learning. Phys Med Biol 2024. https://doi.org/10.1088/1361-6560/ad3880.

[12] Irannejad M, Abedi I, Lonbani VD, Hassanvand M. Deep-neural network approaches for predicting 3D dose distribution in intensity-modulated radiotherapy of the brain tumors. J Appl Clin Med Phys 2024;25:e14197.

[13] Roberfroid B, Barragán-Montero AM, Dechambre D, Sterpin E, Lee JA, Geets X. Comparison of Ethos template-based planning and AI-based dose prediction: General performance, patient optimality, and limitations. Phys Medica PM Int J Devoted Appl Phys Med Biol Off J Ital Assoc Biomed Phys AIFB 2023;116:103178. https://doi.org/10.1016/j.ejmp.2023.103178.

[14] Barragán-Montero AM, Thomas M, Defraene G, Michiels S, Haustermans K, Lee JA, et al. Deep learning dose prediction for IMRT of esophageal cancer: The effect of data quality and quantity on model performance. Phys Medica PM Int J Devoted Appl Phys Med Biol Off J Ital Assoc Biomed Phys AIFB 2021;83:52–63. https://doi.org/10.1016/j.ejmp.2021.02.026.

[15] Appelt AL, Elhaminia B, Gooya A, Gilbert A, Nix M. Deep Learning for Radiotherapy Outcome Prediction Using Dose Data - A Review. Clin Oncol R Coll Radiol G B 2022;34:e87–96. https://doi.org/10.1016/j.clon.2021.12.002.

[16] Jacobs SR, Weiner BJ, Reeve BB, Hofmann DA, Christian M, Weinberger M. Determining the predictors of innovation implementation in healthcare: a quantitative analysis of implementation effectiveness. BMC Health Serv Res 2015;15:1–13. https://doi.org/10.1186/s12913-014-0657-3.

[17] Randal A. The Ideal Versus the Real: Revisiting the History of Virtual Machines and Containers. pp. 1–5:31 ACM Comput Surv 2020;53(5). https://doi.org/10.1145/3365199.

[18] Bentaleb O, Belloum ASZ, Sebaa A, El-Maouhab A. Containerization technologies: taxonomies, applications and challenges. J Supercomput 2022;78:1144–81. https://doi.org/10.1007/s11227-021-03914-1.

[19] Docker: Accelerated Container Application Development 2022. https://www.docker.com/ (accessed April 5, 2024).

[20] Home · Wiki · Aarhus RadOnc AI / repos / RadDeploy · GitLab. GitLab 2024. https://gitlab.com/aarhus-radonc-ai/repos/raddeploy/-/wikis/home (accessed May 30, 2024).

[21] Tar - GNU Project - Free Software Foundation n.d. https://www.gnu.org/software/tar/ (accessed June 20, 2024).

[22] NVIDIA/nvidia-container-toolkit: Build and run containers leveraging NVIDIA GPUs n.d. https://github.com/NVIDIA/nvidia-container-toolkit (accessed April 18, 2024).

[23] Wasserthal J, Breit H-C, Meyer MT, Pradella M, Hinck D, Sauter AW, et al. TotalSegmentator: robust segmentation of 104 anatomical structures in CT images 2023. https://doi.org/10.48550/arXiv.2208.05868.

[24] Set Up RadDeploy · Wiki · Aarhus RadOnc AI / repos / RadDeploy · GitLab. GitLab 2024. https://gitlab.com/aarhus-radonc-ai/repos/raddeploy/-/wikis/home/Guides/Set-Up-RadDeploy (accessed June 4, 2024).

[25] Aarhus RadOnc AI / projects / PROSA-GTV · GitLab. GitLab 2024. https://gitlab.com/aarhus-radonc-ai/projects/prosa-gtv (accessed April 9, 2024).

[26] Paganetti H, Botas P, Sharp GC, Winey B. Adaptive proton therapy. Phys Med Biol 2021;66. https://doi.org/10.1088/1361-6560/ac344f.

[27] Bobić M, Lalonde A, Sharp GC, Grassberger C, Verburg JM, Winey BA, et al. Comparison of weekly and daily online adaptation for head and neck intensity-modulated proton therapy. Phys Med Biol 2021. https://doi.org/10.1088/1361-6560/abe050.

[28] MONAI Deploy - Product Page n.d. https://monai.io/deploy.html (accessed April 3, 2024).
[29] Grafana: The open observability platform | Grafana Labs n.d. https://grafana.com/ (accessed April 3, 2024).
[30] Argo Workflows - The workflow engine for Kubernetes n.d. https://argo-workflows.readthedocs.io/en/release-3.5/ (accessed June 12, 2024).
[31] Apache Airflow. Apache Airflow n.d. https://airflow.apache.org/ (accessed June 12, 2024).
[32] Celery - Distributed Task Queue — Celery 5.4.0 documentation n.d. https://docs.celeryq.dev/en/stable/ (accessed June 12, 2024).