EPI Framework: Developing and Executing Multi-site applications in the Medical Domain

Organizers

- Tim Muller
- Jamila Alsayed Kassem
- Adam Belloum

Location

- Date: April 20
- Time: 12:30 - 14:30
- Room: Room 112
- Location: Jaarbeurs, Utrecht

[Registration now]

Program

- *12:30 - 12:45*: **Introduction** (presentation)
- *12:45 - 13:30*: **Part1: Hello, world!** (guided hands-on)
- *13:30 - 13:45*: **Break**
- *13:45 - 14:15*: **Part 2: A workflow for Disaster Tweets** (hands-on)
- *14:15 - 14:30*: **Evaluation**

Introduction

In this tutorial, participants will get hands-on experience with the EPI Framework, a workflow execution system that aims to unlock data silos in the medical domain. There are many aspects of the framework worth exploring, but this tutorial will focus on being an end-user (domain scientist) and software engineer for the system. Participants will get hands-on experience with writing software packages and workflows, and will experience submitting them to an already running instance of the framework.

Important:
Because this tutorial is hands-on, it is required that participants bring their own laptop. Currently, Windows, macOS (Intel, M1/M2) and Linux (Debian-based, Arch Linux) are supported. Other operating systems, and in particular other Linux distributions, may work, but we make no guarantees. Moreover, participants should be OK with installing Docker[1] (how to: Windows, macOS, Ubuntu, Arch Linux) and the Docker Buildx plugin[2] (how to) on their laptop, which they are encouraged to do beforehand.

Background

The EPI Framework is being developed as part of the *Enabling Personalized Interventions* (EPI) project, which aims to introduce *Digital Health Twins* (DHT) [1] to the healthcare domain with the goal of personalizing medicine. In particular, using DHTs will allow predictions and suggestions to be made that are tailored to the patient in question. However, this is hard to implement in practice, because the data necessary to build a DHT is locked in various healthcare institutions, each with their own, strict privacy requirements on the datasets. Moreover, these requirements are often dynamic, because they rely on whether a patient has given consent, which may also be retracted at any point in time.

The EPI Project investigates multiple aspects to addressing these issues. The one this tutorial focuses on is the introduction of the EPI Framework, which employs the *algorithm-to-data* paradigm in order to minimize moments in which data has to be shared outside of a particular healthcare domain. Instead, only results need to be shared, which supports recent advances in federated machine learning [2]. In addition, the participating institutions, or *domains*, can express high-level policies that express what the resource can execute and who can get access to the resource's datasets [3]. Unique to the EPI Framework, these policies are hidden behind a minimal interface, allowing them to both be *private*, which is desired in the case of patients giving consent, and facilitating more complex means of

[1] https://www.docker.com/
[2] https://github.com/docker/buildx

expressing policies, such as functional programming languages like eFLINT [4]. This latter allows the EPI Framework to capture complex concepts, or even norms, which allows the framework to enforce existing contracts or laws with minimal adaptation.

In addition, the EPI Framework also attempts to simplify deployment and adaptation of the underlying, physical infrastructure, by managing networks through *virtualized network functions* [5]. This is done through the Bridging Function Chain Orchestrator (BFC Orchestrator)[3] [6], which orchestrates and provisions the networking containers across the resources of the healthcare domains. This complements the higher-level privacy policies by allowing low-level network and security policies to be expressed.

Aside from the data side, the EPI Framework is based on BRANE [7], which was originally developed to be used in general High-Performance Computing use-cases. BRANE attempts to simplify access to domain experts by introducing a separation of concerns: software engineers develop software packages, which are then composed in natural language-like DSLs by the domain experts. Then there are also system administrators, in charge of managing the resources available to the EPI Framework, and policy experts, who write the policies that restrict what a resource executes and shares.

Infrastructure

Participants in this tutorial will primarily use their own devices to run the client part of the EPI Framework. The server part, however, is hosted on virtual machines owned by two domains: SURF[4] and the University of Amsterdam[5]. Both will already have a running EPI Framework worker nodes, which can execute tasks defined in a workflow; the workflow itself can be submitted to a central node running at the University of Amsterdam, which will then orchestrate the work over the worker nodes.

Part 1: Hello, world!

In the first half of the tutorial, participants will install the framework client on their laptops, and then learn the basics of using the framework to develop and deploy a traditional "Hello, world!" package.

This session will be done in a centralized fashion, where everyone will go through the steps together under guidance of the tutorial hosts.

Guide

You can find the detailed step-by-step process for the first part here:
https://wiki.enablingpersonalizedinterventions.nl/user-guide/appendix/tutorials/2023-04-20/p1_hello_world.html

---

[3] The BFC Orchestrator was previously known as "EPI Framework". However, this term has been generalized to include the entire data-sharing framework, of which the BFC Orchestrator is a component.
[4] https://www.surf.nl/en
[5] https://uva.nl

Part 2: A workflow for Disaster Tweets

In the second half of the tutorial, participants will use an existing software package[6] to implement a workflow that can be used to train a classifier on the Disaster Tweets[7] dataset. Specifically, the software package implements a Bernoulli Naive Bayes classifier, as implemented by the sklearn package[8] [8].

This part uses the EPI Framework's custom DLS, BraneScript. You can refer to the wiki[9] for more information.

This session will be done in a decentralized manner, where participants have a chance to experiment with the framework for themselves. The tutorial hosts will provide support as required.

Guide

You can find the detailed step-by-step process for the second part here:
https://wiki.enablingpersonalizedinterventions.nl/user-guide/appendix/tutorials/2023-04-20/p2_disaster_tweets.html

References and links

[1] Koen Bruynseels, Filippo Santoni de Sio, and Jeroen van den Hoven. *Digital twins in health care: Ethical implications of an emerging engineering paradigm.* Frontiers in Genetics, 9, 2018. doi:10.3389/fgene.2018.00031

[2] Jakub Konečný, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. *Federated learning: Strategies for improving communication efficiency*. arXiv preprint arXiv:1610.05492, 2016.

[3] C. A. Esterhuyse, T. Müller, L. T. Van Binsbergen and A. S. Z. Belloum, *Exploring the Enforcement of Private, Dynamic Policies on Medical Workflow Execution*, 2022 IEEE 18th International Conference on e-Science (e-Science), Salt Lake City, UT, USA, 2022, pp. 481-486, doi: 10.1109/eScience55777.2022.00086.

[4] L. Thomas van Binsbergen, Lu-Chi Liu, Robert van Doesburg, and Tom van Engers. 2020. *EFLINT: a domain-specific language for executable norm specifications.* In Proceedings of the 19th ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences (GPCE 2020). Association for Computing Machinery, New York, NY, USA, 124–136. https://doi.org/10.1145/3425898.3426958

[5] B. Han, V. Gopalakrishnan, L. Ji and S. Lee, *Network function virtualization: Challenges and opportunities for innovations*, in IEEE Communications Magazine, vol. 53, no. 2, pp. 90-97, Feb. 2015, doi: 10.1109/MCOM.2015.7045396.

[6] https://github.com/epi-project/brane-disaster-tweets-example
[7] https://www.openml.org/search?type=data&status=active&id=43395&sort=runs
[8] https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.BernoulliNB.html
[9] https://wiki.enablingpersonalizedinterventions.nl/user-guide/branescript/introduction.html

[6] J. A. Kassem, O. Valkering, A. Belloum and P. Grosso, *EPI Framework: Approach for Traffic Redirection Through Containerised Network Functions*, 2021 IEEE 17th International Conference on eScience (eScience), Innsbruck, Austria, 2021, pp. 80-89, doi: 10.1109/eScience51609.2021.00018.

[7] Valkering, Onno & Cushing, Reginald & Belloum, A.. (2021). *Brane: A Framework for Programmable Orchestration of Multi-Site Applications*. 277-282. 10.1109/eScience51609.2021.00056.

[8] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. *Scikit-learn: Machine learning in python*. Journal of Machine Learning Research, 12(85):2825–2830, 2011. URL: http://jmlr.org/papers/v12/pedregosa11a.html

BRANE wiki: https://wiki.enablingpersonalizedinterventions.nl/
BRANE user guide: https://wiki.enablingpersonalizedinterventions.nl/user-guide
EPI Framework repository: https://github.com/epi-project/brane
Disaster Tweets Package repository: https://github.com/marinoandrea/disaster-tweets-brane