The CLARIN ERIC deployment infrastructure and its applicability to reproducible research

Background

- CLARIN infrastructure goals:
  - **Portability, deployability** and **evaluability** of services
  - **Immutability** of the infrastructure
- Implemented by:
  - Containerization based on docker
  - Customized build and deploy workflow

Application Deploy Workflow

- Project **containers** and the **associated configuration** are grouped together using **docker-compose**
- Managed in git repositories
  - Sensible defaults are supplied to support running out of the box
  - Overriding of defaults enabled via `.env` file stored outside the git repository
- Custom tooling to deploy and control projects
  - `deploy` user unable to directly modify the filesystem to avoid untracked changes. Only remote git tags can be deployed
  - **Deploy script** to quickly deploy, update and roll back the project
    - [https://gitlab.com/CLARIN-ERIC/deploy-script](https://gitlab.com/CLARIN-ERIC/deploy-script)
    - Example:
      
      ```
      $ ./deploy.sh --name test --git test --tag v1
      $ ./control.sh test init
      $ ./control.sh test start | stop | restart
      ```

Docker Image Organisation

- Based on **Alpine Linux**
- Not following the single process approach
  - **Cron** daemon to run periodic tasks
  - **Fluentd** daemon to aggregate and tag logs into stdout
  - **Supervisord** daemon to manage process
  - Application process(es)

Docker Image Build workflow

- Unified workflow between local and remote CI environments, integrated image testing provided by the **build script**: [https://gitlab.com/CLARIN-ERIC/build-script](https://gitlab.com/CLARIN-ERIC/build-script)

- Easy local running of images provided by the **CLARIN control script**: [https://gitlab.com/CLARIN-ERIC/control-script](https://gitlab.com/CLARIN-ERIC/control-script)

- **Coupling of files** and **Docker images** via **git repository**, and a connected Continuous Integration (CI) environment that feeds into a Docker registry.

Reproducibility of Research

- The **output** of processing pipelines for compiling language data **depends on the configuration**
- But **configurations change over time**, e.g. when new versions or different tools become available
- **Containers** mitigate this by encapsulating fully functional units that can be kept for later re-use ensuring **identical behaviour**

Authors

Andre Moreira, Willem Elbers (CLARIN ERIC, the Netherlands), Egon W. Stemle, Alexander König, Luca Cattani, Martin Palma (Eurac Research, Italy)