LaMachine

Maarten van Gompel, KNAW HuC / CLST

June 24th, 2021

LaMachine

Context

- since 2015
- Increasingly complex software stack: Timbl, Frog, ucto, libfolia
- Mostly developed in CLARIN/CLARIAH WP3
- C++ code with dependencies that was non-trivial to compile for most people
- and a stack of Python-based software
- Multiple interfaces and users on each level:
  - command-line interface
  - C++ library / Python bindings
  - RESTful webservice (via CLAM)
  - web-application (via CLAM)

What is LaMachine

A **meta-distribution**:

- A solution for the distribution and deployment of software and software services
- Installation and configuration recipes
- ..for a limited set of (often interconnected) NLP software
- WP3 software stack from Radboud University / KNAW HuC
- No new repository; relies on established software repositories
- Builds on existing technologies
- A fairly standalone infrastructure in the absence of a larger CLARIAH-wide one

Different “flavours”

Offer a similar environment in different flavours:

- Native, in a local user environment
- Native, globally on dedicated system *(local or remote)*:
  - Linux
  - Windows Subsystem for Linux (limited)
  - macOS (limited)
- As a virtual machine
- As a container

Technologies

- **Provisioning** (Installation and configuration recipes):** Ansible
  - used for all flavours
- **Virtualisation:** Vagrant and Virtualbox
- **Containerisation:**
  1. Docker
     - No need to write your own `Dockerfile`
     - “Fat” container may be at odds with Docker’s paradigm
  2. LXC
  3. Singularity

Target audience

- data scientists / researchers
- developers
- service hosting providers (e.g. CLARIAH centres)
- high-performance computing cluster providers (e.g. universities)

Target interfaces

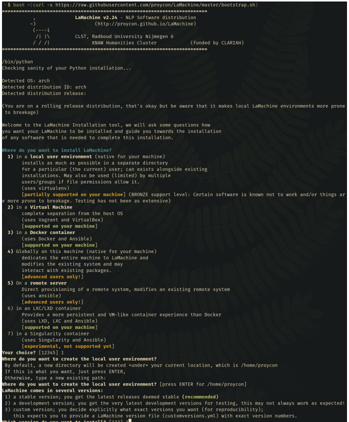
- Command-line shell (possibly over ssh)
  - Direct access to installed software
- Web applications (through the browser)
- Web services (REST)
- Web-based IDE and Notebooks (Jupyter Lab)
  - Direct access to isntalled modules

Target platforms and support levels

- Gold support
  - Debian 10 (buster, stable) *(Docker default)*
  - Ubuntu 20.04 LTS *(VM default, lxc default)*
- Silver support
  - Debian 9 (stretch, oldstable)
  - Ubuntu 18.04 LTS
  - CentOS 8 / RedHat Enterprise Linux 8 -
- Bronze support
  - Debian testing / Debian unstable
  - Ubuntu non-LTS after last LTS
  - macOS (latest version)
  - Arch Linux
  - Linux Mint
  - Fedora Linux

Bootstrap

- Start from a single executable (shell script) and build a LaMachine environment from scratch (any flavour): `bash <(curl -s https://raw.githubusercontent.com/proycon/LaMachine/mas`
- Start from the latest Docker base image (`Dockerfile`)
- Start from the latest VM image (`Vagrantfile`)



```
└$ bash <(curl -s https://raw.githubusercontent.com/proycon/LaMachine/master/bootstrap.sh)
=========================================================================
         ,              LaMachine v2.24 - NLP Software distribution
        ~)                     (http://proycon.github.io/LaMachine)
         (----i
          /| |\        CLST, Radboud University Nijmegen &
         / / /|              KNAW Humanities Cluster            (funded by CLARIAH)
=========================================================================

/bin/python
Checking sanity of your Python installation...

Detected OS: arch
Detected distribution ID: arch
Detected distribution release:

(You are on a rolling release distribution, that's okay but be aware that it makes local LaMachine environments more prone
 to breakage)

Welcome to the LaMachine Installation tool, we will ask some questions how
you want your LaMachine to be installed and guide you towards the installation
of any software that is needed to complete this installation.

Where do you want to install LaMachine?
  1) in a local user environment (native for your machine)
       installs as much as possible in a separate directory
       for a particular (the current) user; can exists alongside existing
       installations. May also be used (limited) by multiple
       users/groups if file permissions allow it.
       (uses virtualenv)
       [partially supported on your machine] (BRONZE support level: Certain software is known not to work and/or things ar
e more prone to breakage. Testing has not been as extensive)
  2) in a Virtual Machine
       complete separation from the host OS
       (uses Vagrant and VirtualBox)
       [supported on your machine]
  3) in a Docker container
       (uses Docker and Ansible)
       [supported on your machine]
  4) Globally on this machine (native for your machine)
       dedicates the entire machine to LaMachine and
       modifies the existing system and may
       interact with existing packages.
       [advanced users only!]
  5) On a remote server
       Direct provisioning of a remote system, modifies an existing remote system
       (uses ansible)
       [advanced users only!]
  6) in an LXC/LXD container
       Provides a more persistent and VM-like container experience than Docker
       (uses LXD, LXC and Ansible)
       [supported on your machine]
  7) in a Singularity container
       (uses Singularity and Ansible)
       [experimental, not supported yet]
Your choice? [12345] 1
Where do you want to create the local user environment?
 By default, a new directory will be created *under* your current location, which is /home/proycon
 If this is what you want, just press ENTER,
 Otherwise, type a new existing path:
Where do you want to create the local user environment? [press ENTER for /home/proycon]
LaMachine comes in several versions:
 1) a stable version; you get the latest releases deemed stable (recommended)
 2) a development version; you get the very latest development versions for testing, this may not always work as expected!
 3) custom version; you decide explicitly what exact versions you want (for reproducibility);
    this expects you to provide a LaMachine version file (customversions.yml) with exact version numbers.
Which version do you want to install? [123] 1
```

Development vs Production

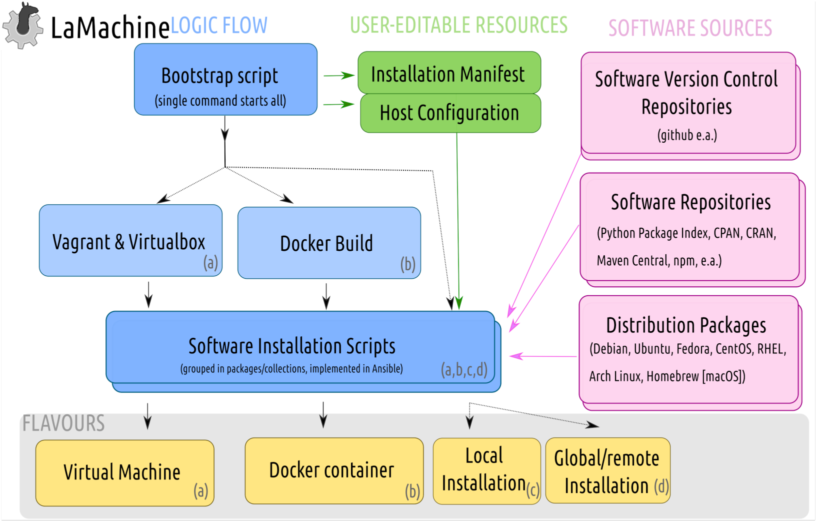
Two *channels*:

- **Stable** - Will pull in the latest released ‘stable’ versions of all software
- **Development** - Will pull in the latest development versions of all software (from git), may break.

Modularity and Configurability

- LaMachine defines a limited number of *software meta-packages* of participating software
  - (these ‘packages’ are implemented as ansible *roles*)
- The user decides which to install, packages can be also be added later at will (but not removed)

Architecture Overview


LaMachine
LOGIC FLOW
USER-EDITABLE RESOURCES
SOFTWARE SOURCES
Bootstrap script
(single command starts all)
Installation Manifest
Host Configuration
Software Version Control Repositories
(github e.a.)
Software Repositories
(Python Package Index, CPAN, CRAN, Maven Central, npm, e.a.)
Distribution Packages
(Debian, Ubuntu, Fedora, CentOS, RHEL, Arch Linux, Homebrew [macOS])
Vagrant & Virtualbox (a)
Docker Build (b)
Software Installation Scripts
(grouped in packages/collections, implemented in Ansible)
(a,b,c,d)
FLAVOURS
Virtual Machine (a)
Docker container (b)
Local Installation (c)
Global/remote Installation (d)

CLARIAH WP3 Software

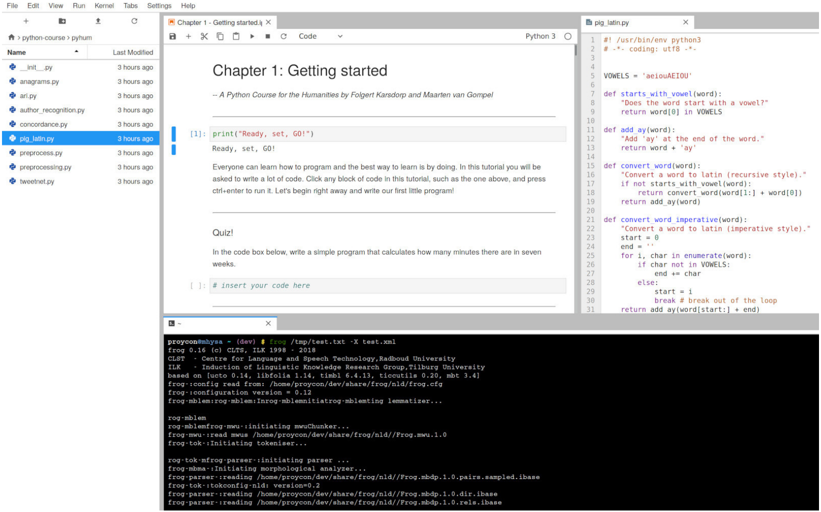
- Frog, ucto, libfolia (NLP software for dutch)
  - timbl, mbt
- foliapy, python-frog, python-ucto
- folia tools/utilities
- Deepfrog, folia-rust
- FLAT: FoLiA Linguistic annotation Tool
- PICCL: OCR and OCR post-correction/normalisation
- CLAM: Facilitates building webservices
- Piereling: Webservice for conversion between document formats
- Alpino (Rijksuniversiteit Groningen)

Software from related projects

- kaldi-nl (Stichting Openspraaktechnologie): Dutch Speech Recognition
- Colibri Core (Radboud University): pattern detection
- T-Scan (Utrecht University): Analytics for dutch texts
- Gecco & Valkuil (Radboud University): Dutch context-sensitive spelling correction

Third party software

- LaMachine includes (optionally) a lot of third party software common in the field:
  - Jupyter hub/lab/notebooks
  - Tesseract (OCR)
  - pytorch (DL), tensorflow, fasttext
  - Kaldi (ASR)
  - SpaCy (NLP), CoreNLP (Stanford)
  - Moses (SMT)
  - FLAIR , fasttext
  - Nextflow
  - Lots of generic Python libs (numpy, nltk, scikit-learn etc)..
- Common languages: C/C++, Python, JS, R, Go, Rust, Java, Julia



Name	Last Modified
__init__.py	3 hours ago
anagrams.py	3 hours ago
ari.py	3 hours ago
author_recognition.py	3 hours ago
concordance.py	3 hours ago
pig_latin.py	3 hours ago
preprocess.py	3 hours ago
preprocessing.py	3 hours ago
tweetnet.py	3 hours ago

Chapter 1: Getting started

-- A Python Course for the Humanities by Folgert Karsdorp and Maarten van Gompel

```
[1]: print("Ready, set, GO!")
```

```
Ready, set, GO!
```

Everyone can learn how to program and the best way to learn is by doing. In this tutorial you will be asked to write a lot of code. Click any block of code in this tutorial, such as the one above, and press ctrl+enter to run it. Let's begin right away and write our first little program!

Quiz!

In the code box below, write a simple program that calculates how many minutes there are in seven weeks.

```
[ ]: # insert your code here
```

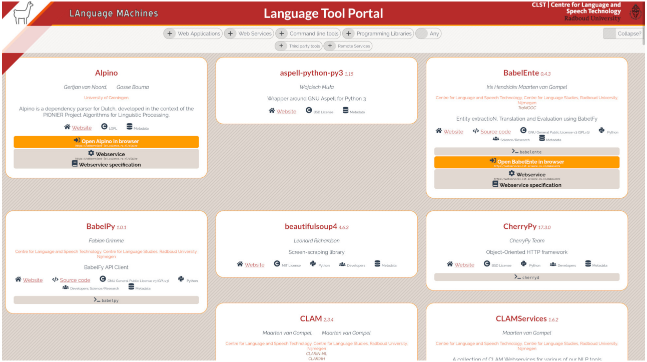
```python
#! /usr/bin/env python3
# -*- coding: utf8 -*-


VOWELS = 'aeiouAEIOU'

def starts_with_vowel(word):
    "Does the word start with a vowel?"
    return word[0] in VOWELS

def add_ay(word):
    "Add 'ay' at the end of the word."
    return word + 'ay'

def convert_word(word):
    "Convert a word to latin (recursive style)."
    if not starts_with_vowel(word):
        return convert_word(word[1:] + word[0])
    return add_ay(word)

def convert_word_imperative(word):
    "Convert a word to latin (imperative style)."
    start = 0
    end = ''
    for i, char in enumerate(word):
        if char not in VOWELS:
            end += char
        else:
            start = i
            break # break out of the loop
    return add_ay(word[start:] + end)
```

```
proycon@mhysa ~ (dev) $ frog /tmp/test.txt -X test.xml
frog 0.16 (c) CLTS, ILK 1998 - 2018
CLST  - Centre for Language and Speech Technology,Radboud University
ILK   - Induction of Linguistic Knowledge Research Group,Tilburg University
based on [ucto 0.14, libfolia 1.14, timbl 6.4.13, ticcutils 0.20, mbt 3.4]
frog-:config read from: /home/proycon/dev/share/frog/nld/frog.cfg
frog-:configuration version = 0.12
frog-mblem:rog-mblem:Inrog-mblemnitiatrog-mblemting lemmatizer...

rog-mblem
rog-mblemfrog-mwu-:initiating mwuChunker...
frog-mwu-:read mwus /home/proycon/dev/share/frog/nld//Frog.mwu.1.0
frog-tok-:Initiating tokeniser...

rog-tok-mfrog-parser-:initiating parser ...
frog-mbma-:Initiating morphological analyzer...
frog-parser-:reading /home/proycon/dev/share/frog/nld//Frog.mbdp.1.0.pairs.sampled.ibase
frog-tok-:tokconfig-nld: version=0.2
frog-parser-:reading /home/proycon/dev/share/frog/nld//Frog.mbdp.1.0.dir.ibase
frog-parser-:reading /home/proycon/dev/share/frog/nld//Frog.mbdp.1.0.rels.ibase
```

Upgrade procedure

- Running `lamachine-update` inside a Lamachine environment will update an existing installation and and all software in it
  - (simply invokes ansible again)
- Or pull fresh new image from your image repository (Docker/Vagrant)

Portal



Alpino

Gertjan van Noord, Gosse Bouma

University of Groningen

Alpino is a dependency parser for Dutch, developed in the context of the PIONIER Project Algorithms for Linguistic Processing.

Website LGPL Metadata

Open Alpino in browser https://webservices-lst.science.ru.nl/alpino

Webservice https://webservices-lst.science.ru.nl/alpino

Webservice specification

aspell-python-py3 1.15

Wojciech Muła

Wrapper around GNU Aspell for Python 3

Website BSD License Metadata

BabelEnte 0.4.3

Iris Hendrickx Maarten van Gompel

Centre for Language and Speech Technology, Centre for Language Studies, Radboud University, Nijmegen
TraMOOC

Entity extractioN, Translation and Evaluation using BabelFy

Website Source code GNU General Public License v3 (GPLv3) Python Science/Research Metadata

`babelente`

Open BabelEnte in browser https://webservices-lst.science.ru.nl/babelente

Webservice https://webservices-lst.science.ru.nl/babelente

Webservice specification

BabelPy 1.0.1

Fabian Grimme

Centre for Language and Speech Technology, Centre for Language Studies, Radboud University, Nijmegen

BabelFy API Client

Website Source code GNU General Public License v3 (GPLv3) Python Developers; Science/Research Metadata

`babelpy`

beautifulsoup4 4.6.3

Leonard Richardson

Screen-scraping library

Website MIT License Python Developers Metadata

CherryPy 17.3.0

CherryPy Team

Object-Oriented HTTP framework

Website BSD License Python Developers Metadata

`cherryd`

CLAM 2.3.4

Maarten van Gompel, Maarten van Gompel

Centre for Language and Speech Technology, Centre for Language Studies, Radboud University, Nijmegen
CLARIN-NL
CLARIAH

CLAMServices 1.6.2

Maarten van Gompel

Centre for Language and Speech Technology, Centre for Language Studies, Radboud University, Nijmegen

A collection of CLAM Webservices for various of our NLP tools

- Lists all included software and services
- Provides access to included services
- Each LaMachine intstallation can automatically provide such a portal
- Dynamically generated, contents are derived from harvested software metadata
- List is also accessible on command-line through `lamachine-list`

CodeMeta as a Software Metadata scheme

> *"With codemeta, we want to formalize the schema used to map between the different services (GitHub, figshare, Zenodo) to help others plug into existing systems. Having a standard software metadata interoperability schema will allow other data archivers and libraries join in. This will help keep science on the web shareable and interoperable!"* [from https://codemeta.github.io]

Codemeta:

- is simple and minimalistic
- aimed at research software and enabling citability (DOI)
- uses Linked Open Data
  - serialises to JSON-LD
  - re-uses and collaborates with schema.org
- is an existing third-party effort, grew out of *Code as a Research Object*, a Mozilla Science project with Github and Figshare
  - provides a mapping to other systems (DOAP, Debian Packages, DataCite, WikiData, Maven, NodeJS, Python distutils, R, Ruby gems)

```
{
    "@context": [
        "https://doi.org/10.5063/schema/codemeta-2.0",
        "http://schema.org"
    ],
    "@type": "SoftwareSourceCode",
    "identifier": "lamachine",
    "name": "LaMachine",
    "version": "2.24",
    "description": "LaMachine is a unified software distrib
    "license": "https://spdx.org/licenses/GPL-3.0",
    "url": "https://proycon.github.io/LaMachine",
    "author": ...,
    "codeRepository": "https://github.com/proycon/LaMachine
    "issueTracker": "https://github.com/proycon/LaMachine/i
```

Software Metadata in LaMachine

During installation/bootstrapping/updating, LaMachine:

- Takes the software metadata from each tool's source repository if available
- Otherwise: converts metadata from the upstream package *(Python Package Index, CRAN, CPAN, Maven Central)*
- Augments the metadata where needed with installation/deployment specific information:
    - to register web-based entrypoints as provided by LaMachine
    - with extra information specified in the (Ansible) build recipes
- Builds a software registry of all installed software *(JSON-LD graph)*
- Provides a portal web-application on the basis of this metadata (Labirinto)
    - Example: https://webservices.cls.ru.nl
- *Note:* CodeMeta describes software metadata, not APIs

What is LaMachine *NOT*?

- **NOT** an NLP pipeline/workflow system; rather it may install such systems or components required by such systems.
  - *e.g PICCL (powered by Nextflow), Frog
- **NOT** a system for archiving/preserving legacy software
  - software **MUST** be maintained
- **NOT** only for Nijmegen software
- **NOT** a portal to search/access data collections
  - with LaMachine you can bring the tools to the data
- **NOT** a traditional Linux distribution

Authentication

- LaMachine can be configured to connect to **external** OAuth2/OpenID Connect for authentication.
- Provide the configuration once for all of LaMachine and LaMachine propagates it to participating software.
- LaMachine also works fine at the single-user level (or shared)

Limitations

For service providers, these are explicitly out-of-scope:

- **Scalability**: A single LaMachine installation does not scale for long, you can spin up multiple instances in the docker/VM flavour but have to handle the load balancing yourself.
- **Container Orchestration:** LaMachine does not cover multiple containers and does no container orchestration
- **Encryption:** LaMachine does not handle SSL certificates, you need to handle that in your own reverse proxy

For participating software providers:

- Software must remain maintained/up to date, participants must make an effort to support all flavours, target platforms and channels
- Software is limited to NLP/Data-science
- LaMachine is **not** a substitute for not providing source repositories or ecosystem packages
- Dependencies may not cause major conflicts between participating software (shared environment)
- No nested containers!

For all:

- **Rolling release** (latest version), limited support for rebuilding older versions
- **Data-agnostic**, does not tie into any large external data collections

Strengths and Weaknesses

Strengths:

- Highly **flexible** solution (many flavours, serves many different audiences)
- Does not focus exclusively on a service oriented architecture nor *Software as a Service*
- The software provider needs to know only a subset of Ansible and no specific knowledge of Docker, LXC, Vagrant is required.
- Really brings software to the users
- Proven track record; real users
- Builds on standard solutions, propagates software freedom

Weaknesses:

- **Complexity & Maintainability**: supporting many target distributions, flavours and channels, in continuously moving ecosystems is not easy.
- **Fat containers** may be at odds with the Docker paradigm