decode

D4.4 DECODE OS First Release

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 732546

Project no. 732546

DECODE

DEcentralised Citizens Owned Data Ecosystem

D4.4 First Release of the DECODE OS

Version Number: v1.0.0

Lead beneficiary: Dyne.org

Due Date: September 2017

Authors: Denis Roio and Ivan Jelinčić (Dyne.org)

Editors and reviewers: [Name and organisation of reviewers]

Dissemination level: PU (Public)		
PU	Public	
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Approved by: [First Name Last Name (Organisation)]

Date: [DD/MM/YYYY]

This report is currently awaiting approval from the EC and cannot be not considered to be

Contents

Introduction and scope 4
Main objectives of the DECODE OS 5
UNIX-like Architecture and process separation 6
The first release of DECODE OS 9
Steps to build the first DECODE OS release 9
Base components of the DECODE OS 11
Run the DECODE OS devops environment 13
Future plans 15

Introduction and scope

The DECODE OS is the base operating system running all software designed, developed and deployed for the DECODE project. This operating system is based on the renown Devuan GNU+Linux distribution, a fork of the now 20 years old Debian distribution, maintained by the Dyne.org foundation and an open community of volunteers. Devuan forked Debian to preserve the simplicity and minimalism of the SystemV tradition in UNIX systems, still running modern software applications and inheriting the security patches from Debian.

This document accompanies the first running release of the DECODE OS: a reference platform for building and testing any software made for the DECODE project. The deliverable 4.1 "" completes this documentation by detailing the process of development of packages to be included in the OS, as well provides an operational manual for the Simple Distro Kit (SDK) tools used to make OS releases.

The first release of the DECODE OS is made available at the Internet address https://files.dyne.org/decode/OS and it incudes fully functioning virtual machine images (Vagrant/vbox and Qcow2/Qemu formats) implementing a base architecture (kernel and base system tools) to build, run and connect a diverse range of future applications. The DECODE OS is provided to developers of the DECODE project along with comprehensive documentation and tools to reproduce its build, benchmark and audit its functioning, as well with a dashboard for live monitoring.

The scope of this document is limited to the documentation of a base system which is offered as reference for tests and benchmarks for other developers engaged in the DECODE project. In the following chapters are provided lists of software included and the versions of each core component at the base of DECODE OS. This is a living document updated across the course of the DECODE project and in particolar with the iteration of deliverable 4.6 "Integration and deployment of the DECODE OS and HUB platform" on month 12 and of deliverable 4.11 "Stable DECODE OS release" on month 24.

Main objectives of the DECODE OS

The primary goal of the DECODE OS can be explained in brief by defining it as a "**controlled execution environment**" where, from the making of its base to the execution of every single application, all steps are recorded on a ledger of events that can be saved, analysed and shipped along with every instance of the operating system. A secondary goal of this development is that of making the results of such a recorded sequence of operations **reproducible** (see D4.1).

The DECODE OS both as a product but also as a well defined process leading to its release aims primarily at satisfying the following main objectives:

- Adopt **free and open source software** compliant with any of DECODE's approved licenses. Make the source-code of any running software immediately available for audit. Avoid any dependency from proprietary software.
- Rely on widely used, peer-reviewed and stable applications and standards from **GNU/Linux/BSD software traditions** for its base functions.
- Make the whole **building process transparent and auditable**, providing schematic information about all adopted components that can be short and to the point, as well machine parsable.
- Realise a minimal, lean and resource aware operating system running as less processes as possible, to **avoid complexity** and to facilitate the controlled execution of **micro-services**.
- Rely on a **continuous-integration infrastructure** (see D4.2) able to integrate the dependencies and applications of DECODE developers with as less friction as possible.
- Provide an optimal **developer experience** by providing two distinct profiles: "-dev" for development and "-prod" for production use and a way to switch between development and production that is seamless.
- Maintain an **exact history of changes** made to the OS as well builds triggered, reflected in the final product and **auditable according to a map of liabilities** drawn during the build process.
- Integrate **benchmarking tools** and a **live monitoring dashboard** to facilitate quality assurance assessments and the review of resource management under different conditions. Also allow the customisation of the dashboard to include future application specific indicators.

- Target a **wide range of hardware architectures** and well-established chipset standards, virtual machines and cloud providers, without requiring any significant change in the base system, just a different setup of the continuous-integration pipeline.
- Refer to **well established UNIX standards** for the configuration of the system and process control. Avoid any binary configuration format and adopt as much as possible input and output formats that can be both read by humans and easily parsed by machines.
- Enforce **full-spectrum process separation** between all different applications running on the DECODE OS, making it possible to isolate problems and eventually recover functionality.

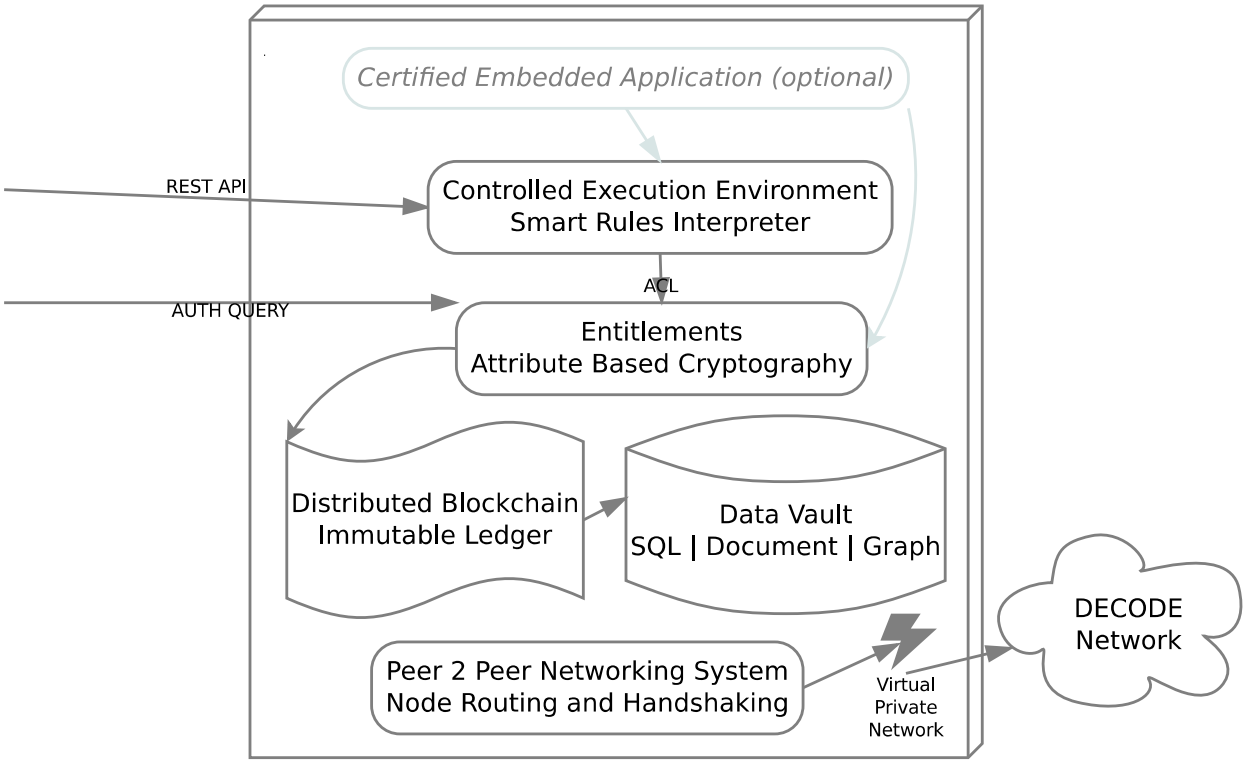
UNIX-like Architecture and process separation

DECODE's implementation of a distributed computational system aims to be solid and fit for mission critical purposes by leveraging well established standard practices in the UNIX world. Contrary to the monolithic applications implementing blockchain functionalities in a single runtime environment running in application space, our implementation of a “DECODE NODE” (see D1.1) is a controlled execution environment unit for Smart Rules grafted on the classic concept of a UNIX-like operating system, keeping POSIX.1b and SystemV compatibilty. This approach brings several advantages:

- The system is familiar to system administrators and can be monitored and managed using existing well known tools.
- A DECODE NODE is backward compatible with the vast majority of existing entreprise infrastructure.
- Planned and documented integration between application specific functions and the underlying OS, verticalising the full stack and providing a fully certified environment for smart rule execution.
- The security of the DECODE NODE is granted by means of stable Linux based access control lists (ACL) and firewalling.
- The diagnosis of problems occurring on running nodes is fully compatible with already established practices of logging, tracing, profiling and debugging processes.
- Components can be mixed in a modular fashion, allowing reusal of existing implementations in different roles, including production ready

software for network orchestration, p2p networking, data storage presentation, cryptographic operations and immutable ledger functions.

Considering a first abstraction of functionalities, mostly resulting from the study of user-cases and currently adopted pilots in DECODE, it is already possible to envision a set of modules whose functionalities can be satisfied by a considerable number of stable software implementations. The following figure shows the functionalities that can be contained in a DECODE node, each of them can be a different software application running as an isolated process, whose communication channels can be specified at the time of building the DECODE OS and cannot modified unless the design of the OS is re-negotiated according to clear needs and specified patterns.



The modules listed are:

- Smart Rules language interpreter and controlled execution environment
- Entitlements ACL layer based on attribute based cryptography
- Distributed Blockchain providing an immutable ledger
- Data storage (Vault) in different possible formats (Document or Graph)
- Peer to Peer Networking Orchestration System and VPN

The modules listed can be implemented by new software developed by the DECODE consortium as well as by existing and well established open source standard implementations. Being this the first release of the DECODE OS it is well out of the scope of this document to envision which software will satisfy these roles, as well the listed modules may be subject to further adjustments as new needs and refinements become manifest in the coming research iterations.

What matters to us now is the fact that there is a clear way to include such SOFTWARE modules inside a coherent and well documented system built from source and that the overhead to substitute a module with another implementation is minimum. Each application is an installed software package that is ran at startup, whose health is monitored by the system and whose failure does not propagates to other system components.

This architecture also allows to map the inner architecture of a NODE as well to envision the possibility to produce different NODES that are tailored ad-hoc to different use-cases. As a last note, the architecture also includes the possibility to have an optional application to be ran inside the NODE and interact with some of the components inside it: this is an opportunity DECODE OS can leverage for special applications requiring intensive communication with inner components of the DECODE NODE.

The first release of DECODE OS

This document accompanies the first release of the DECODE OS, a release that is mainly targeted at developers of the DECODE project, yet open for public review. At this stage the DECODE OS builds made available only target “devops” architectures that facilitate the testing across different development environments without introducing any hardware requirement.

This section will proceed providing technical details on:

- the steps needed to build this first release targeting virtual environments
- the software components adopted for the base system
- how to run the first release of the DECODE OS in a virtual environment

Steps to build the first DECODE OS release

The collection of tools used to build the DECODE OS is called “Simple Distro Kit” (SDK) and is comprised of shell scripts (Zsh) and plain text configuration files.

To run through the build steps one must start from a clean Devuan 2.0 “ASCII” installation, which is Devuan's “testing” at the moment of writing, coupled to the package collection of Debian's currently “stable” release.

To install ASCII the best course of action is to first install Devuan 1.0 “Jessie” and then proceed to upgrade it to ASCII.

- Short instructions to the installation of Devuan Jessie
- Comprehensive guides for the installation of Devuan Jessie
- Translated installation guides (available in ES, IT, DE, PL, FR)

Once completed the Devuan ASCII installation, the following dependencies must be added to the base system, directly available from the package installation system “APT”:

```
# apt-get install zsh debootstrap sudo kpartx cgpt xz-utils qemu qemu-utils
```

In addition to these dependencies, the “Vagrant” and the “VirtualBox” software applications must be installed by downloading the latest version directly from the vendors:

- Vagrant (by Hasciicorp)
- VirtualBox (by Oracle)

At this point all dependencies needed to build the DECODE OS are installed.

To download the latest version of the DECODE SDK using git:

```
$ git clone https://github.com/DECODEproject/os-build-system
```

Then enter the os-build-system directory and switch to the SDK interactive console by running: ./console.sh

At this point the prompt will change and the SDK commands will be available with completion. To list the build commands one can type: build_[tab].

The first release of DECODE OS currently supports the "build_vagrant_dist" target, while other targets are in development and may not yet work as desired.

Each build command abbreviates a sequence of operations needed for the build: "build_vagrant_dist" implies the following sequence of commands:

1. image_${imageformat}_as_strapdir
2. bootstrap_complete_base
3. vm_inject_overrides
4. blend_preinst
5. vm_setup_grub
6. blend_postinst
7. vm_umount_${imageformat}
8. vm_vbox_setup
9. vm_vagrant_package
10. vm_pack_dist

Each of the above build steps are implemented as shell functions inside the SDK, to review their code one can refer to the source-code of "vm-sdk/sdk" which in turn makes use of functions provided by the "libdevuansdk" base library and in particular its "vm" module which is found inside the SDK at the path: "vm-sdk/lib/libdevuansdk/zlibs/vm".

All source-code components and libraries named above are publicly available via git and via web at the address: https://git.devuan.org/sdk

Base components of the DECODE OS

DECODE OS has been engineered to optimise resource usage and provide a lean environment that avoids excessive dependencies and strips base functionalities to the bone as a starting point, with the plan to later increment the tooling as required by the development in application space. The first release of the 64bit base system fits a compressed virtual-machine image of 1.4GB and runs using only 50MB or RAM.

The DECODE OS is a GNU+Linux base system runs on a standard build of the Linux kernel 4.9.30 (as found in Devuan ASCII) with stable support for cgroups ACL, LXC2 containers and advanced firewalling functionalities. At this stage of development no optimisation has been applied yet, but later on in the course of the project the kernel will be optimised with an ad-hoc build.

The base components found in DECODE OS that are most significant for the purpose of the DECODE project are listed below with their version and the most recent release of each software has been choosen for its renown stability:

GNU Binutils 2.28-5

GNU Core Utils 8.26-3

GNU C Compiler 6.3.0

GNU LibC 2.24

OpenJDK 8u131

IPTables 1.4.21

LibFFI 3.2.1-6

LibGLIB 2.50.3

LibGMP 6.1.2

LibGNUTLS 3.5.8-5

Cryptsetup 1.7.3

GNUPG 2.1.18

OpenSSH 7.4p1

OpenSSL 1.1.0f

Perl 5.24.1

Python 2.7.13 and 3.5.3

SysVInit 2.88

LXC 2.0.7-2

Bash 4.4-5

ZSH 5.3.1

In addition to the standard components above, the DECODE OS is shipped with the GO language interpreter version 1.8.3 to satisfy the needs the development ongoing on the device-hub lead by partner Thingful (see deliverable 3.2).

For encrypted storage that is fully compliant with ISO/IEC 18033-1:2015 and 18033-3:2010 the underlying device-mapper system of the Linux kernel is managed using cryptsetup and the usability wrapper Tomb 2.4 also included.

To provide secure builds that are fully static, for instance for the privilege escalation tool 'sup' described in deliverable 4.1, the musl-libc is adopted at version 1.1.16.

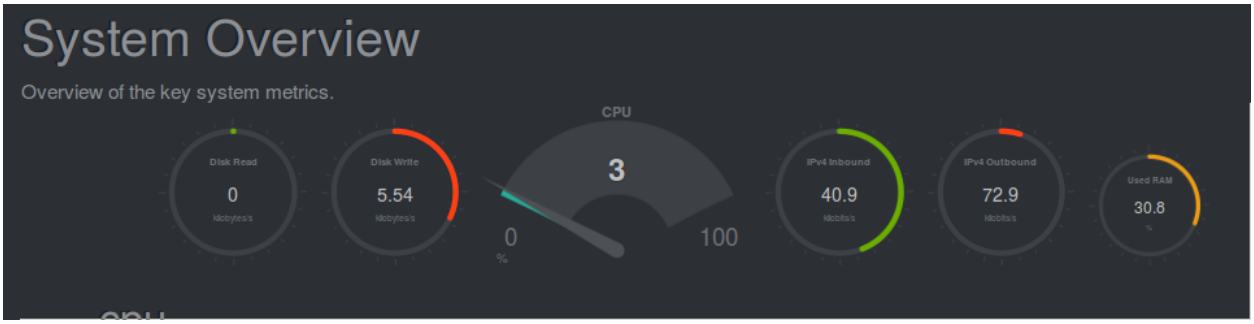
Run the DECODE OS devops environment

When thinking of the format to present the first release of the DECODE OS it became clear that, to facilitate the development cycle and to leverage the involvement even of casual adopters in the open source community, we need a format that is easy to test and work on any machine. Therefore DECODE OS is first packaged as a virtual machine both in Vagrant/VBOX format and in QCow2/QEMU, with the possibility to convert it to other virtual-machines.

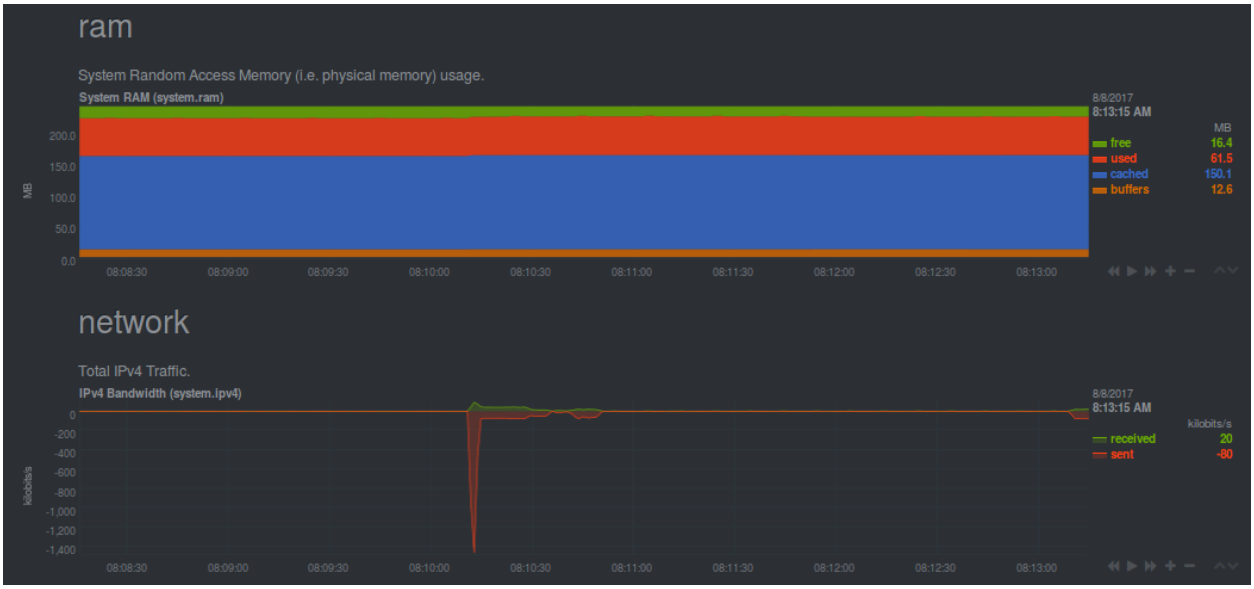
To test DECODE OS is necessary to have installed the latest Vagrant and Virtualbox versions (as described by the previous chapter about dependencies) and then create a 'Vagrantfile' in the current directory with the following contents:

```
Vagrant.configure(2) do |config|
  config.vm.box = "https://files.dyne.org/decode/OS/decode_1.0.0_amd64_vagrant.box"
  config.ssh.username = "root"
  config.ssh.password = "toor"
  config.vm.guest = :debian
  config.vm.synced_folder ".", "/vagrant", disabled: true
  config.vm.define "decode", primary: true do |os|
    os.vm.network :forwarded_port, guest: 19999, host: 19999,
                  id: "netdata", auto_correct: false
  end
  config.vm.provider 'virtualbox' do |vb|
   vb.customize ["modifyvm", :id, "--cableconnected1", "on"]
  end
end
```

Once ready one can simply run `vagrant up` to start the DECODE OS and `vagrant ssh` to log into its console. It is also possible to monitor the performance and load of the system while running different applications by connecting to port 19999 of the host machine, on which the DECODE OS dashboard is provided, here below a screenshot:



The dashboard provides widgets to monitor every aspect of the DECODE OS running in real time, which is a very important feature to facilitate the development of well performing software with an immediate feedback on the resources its claiming. The widgets provided in the first released dashboard are monitoring base aspects of the system, but as we progress with the development of DECODE application components new widgets will be created in order to monitor application specific usage.



Future plans

According to plans, with the early release of the DECODE OS on month 8 of the project, we intended to provide other delivery partners in the consortium a base system to test developments from the very beginning, to allow iterations of a LEAN development experience to happen across the life-span of the rest of the project. As we succesfully achieved this ambitious goal and while we have observed the system presents no issues, the development of the DECODE OS will continue in more ambitious directions with the upcoming release of deliverable 4.6 “Integration and deployment of the DECODE OS and HUB platform”.

We plan to complete all target builds, especially for the platforms envisioned to be most useful for the DECODE project purposes, in particular cloud-ready virtual machines, Docker images and for the provisional hardware platform indicated by partner Arduino in deliverable 4.3.

We also plan to extend the functionalities of the dashboard, apply optimisations to the system and offer to the public base images of the DECODE OS that are easy to manage in any virtualised and micro-service environment, leveraging the open source community and industry interest in our work.

At last, as soon as the results of deliverable 1.8 “Legal frameworks for digital commons and DECODE OS legal guidelines” will be manifest to the consortium, we intend to run an extensive techno-political analysis of the compliancy of DECODE OS with the privacy by design approach, also detailed in deliverable 1.2 “Privacy Design Strategies for the DECODE architecture”, and provide by all means a framework offering tools in full compliancy with the upcoming General Data Protection Regulation of the European Parliament (EU) 2016/679, to facilitate professional adoption and maintainance of the DECODE OS for DECODE itself as well for similar projects needing to pay attention to this new regulation, with the intention to avoid duplicate effort and provide a certified platform adhering to well established industry standards that is liberally licensed and not encumbered by patents.

For any suggestion, enquiry or request, please consider this deliverable is a living document and a software in course of development, its primary maintainer can be contacted with a mail to Denis Roio <jaromil@dyne.org> preferably using PGP, public key 0x4ACB7D10 Fingerprint: 6113D89C A825C5CE DD02C872 73B35DA5 4ACB7D10