Uni Web Topics Presentation

Felix Pojtinger

October 21, 2021

Introduction

Contributing

Contributing

These study materials are heavily based on professor Heuzeroth's "Spezielle Themen für Web-Anwendungen" lecture at HdM Stuttgart.

Found an error or have a suggestion? Please open an issue on GitHub (github.com/pojntfx/uni-webtopics-notes):

Figure 1: QR code to source repository

If you like the study materials, a GitHub star is always appreciated

License

License

Figure 2: AGPL-3.0 license badge

Overview

Overview

- What is DevOps?
- Which parts of the software lifecycle does it cover?
  - Development
  - Distribution (I will focus on this today)
  - Operation
- What is “cloud native”?
- Why are “traditional” distribution methods still relevant?

Development

Development

- DevOps: Also includes development!
- Modern development should not be bound to any client attributes
- It should not matter if the client is a RISC-V Linux machine, a locked-down Windows workstation or an Android phone
- Development should be possible from any platform, for any platform
- The only truly cross-platform application framework is the web
- PWAs make it possible for web apps to have all the features native apps have
- PWAs work offline by default
- Why not make our development environments PWAs?
- Virtual machines and user-friendly hypervisors and containers make it possible to run the editor's backend locally too

Distribution

Basic Distribution Principles

Basic Distribution Principles

- Binaries
  - Compiled forms of software
  - On Linux: ELF binaries, PE binaries on Windows and MACH-O binaries on macOS
  - Binaries can be statically or dynamically linked
    - Statically linked: Since the Linux ABIs are stable, one can depend on them not changing - this allows not linking against any specific C library and makes the resulting binary portable across distributions. It also allows including all external dependencies into the binary, effectively making it a “single-file” distribution method
    - Dynamically linked: Thanks to `dlopen` and package management, dynamic linking can also be used. Most of the time (especially on non-Linux OSes), at least the C library and external dependencies (i.e. `SQLite`) thus need to be available in `LD_LIBRARY_PATH` at runtime; if they are not, the application can’t continue. This makes the binaries non-portable across distributions; for example, if a binary is built on a Debian 11 host, it most probably won’t run on a Debian 10 host due to the different versions of the GNU C library used. This does however also have a few big

Pipelines

Pipelines

- Bagop
- Hydrun
- GitHub Actions
- Semantic Release

Packaging Overview

Packaging Overview

- Package manager
- Source packages and tarballs
- Binary packages
- Documentation packages
- Dependencies (build-time, runtime, one-of-many i.e. multiple OpenSSL implementations)
- AppStream metadata and `.desktop` files
- systemd services

Distribution to RedHat Linux

Distribution to RedHat Linux

- RPM packages

Distribution to Debian GNU/Linux

Distribution to Debian GNU/Linux

- DEB package
- APT repository
- Yum repository

Distribution to Linux (universal)

Distribution to Linux (universal)

- Flatpak
- Flatpak repository

Distribution to Android

Distribution to Android

- APK
- F-Droid repository

Distribution to Windows

Distribution to Windows

- MSI package with auto-updates

Distribution to macOS

Distribution to macOS

- DMG package with auto-updates

Distribution to Kubernetes/the Cloud

Distribution to Kubernetes/the Cloud

- Docker
- Kubernetes
- Helm
- Skaffold

Distribution to WebAssembly

Distribution to WebAssembly

- WASM-Binary
- WASI/wasm_exec equivalents

Operation

Operation

- Sentry
- OpenTelemetry
- Prometheus
- Grafana