Uni Web Topics Presentation

Felix Pojtinger

October 14, 2021

Contents

Uni Web Topics Presentation

1 Introduction

1.1 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 :)

1.2 License

Figure 2: AGPL-3.0 license badge

2 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?

3 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

- Source code can for example never leave the company's system

- Development environments can be quickly updated and tightened to prevent supply chain attacks and increase reproducibility

- Imagine: You find a Free Software project, and all you have to do in order to contribute is press "."!

- Onboarding new developers becomes much easier

- Independence of client choice enables the use of much cheaper or constrained client devices

- Open standards and web technologies enable the adoption of new client and server hardware (i.e. RISC-V chips) easier and enables the easy use of and testing on multiple architectures

- Autoscaling, ballooning etc. can be used server-side: There is no need to provision lots of development servers if no one is using them, and if there is a need for a lot of resources (for example if someone is compiling say a

C++ project) the provisioner (i.e. Kubernetes) can dynamically decide to scale up the container or VM

- There is no need to trust a project's build system, everything can be sandboxed!

- GitPod

- Codespaces

- Coder

- pojde

- But what if we want to develop things that one can't normally develop remotely?

    - Apps which require Android devices as a target (i.e. Android apps)
    - Apps which require USB or Bluetooth and are not using Web Bluetooth/Web Serial (i.e. smart home projects)
    - Apps which require a Wayland compositor/a screen (i.e. desktop Linux apps, GTK/QT apps)
    - Apps which target AVRs or require a programmer (i.e. IoT devices)

- Waypipe

- USB forwarding

- Port forwarding

- DBus forwarding

4   Distribution

4.1   Basic Distribution Principles

- Binaries
- GPG signing and Gridge
- Cosign
- Portability
- Reproducibility
- Why we need more than "just binaries"

4.2   Pipelines

- Bagop
- Hydrun
- GitHub Actions
- Semantic Release

4.3 Distribution to RedHat Linux

- RPM packages

4.4 Distribution to Debian GNU/Linux

- DEB package
- APT repository
- Yum repository

4.5 Distribution to Linux (universal)

- Flatpak
- Flatpak repository

4.6 Distribution to Android

- APK
- F-Droid repository

4.7 Distribution to Windows

- MSI package with auto-updates

4.8 Distribution to macOS

- DMG package with auto-updates

4.9 Distribution to Kubernetes/the Cloud

- Docker
- Kubernetes
- Helm
- Skaffold

4.10 Distribution to WebAssembly

- WASM-Binary
- WASI/wasm_exec equivalents

5 Operation

- Sentry
- OpenTelemetry
- Prometheus
- Grafana