Docker Grundlagen

Einführung in die Basics von Containerization

Linux-Cafe 2018-05-07

Bernd Strößenreuther
mailto:linux-cafe@stroessenreuther.net

Lizenz

Containerization: Wozu?

1. Leichtgewichtige Alternative zur Server-Virtualisierung
2. Einheitliches Format zur Auslieferung von Anwendungen

Agenda

- Warum brauchen wir eine Alternative zur Server-Virtualsierung?
- Hands-on Teil 1: Installieren und ausprobieren
- Hands-on Teil 2: Wir bauen eigene Docker-Images
- Entwickler aufgepasst: Umdenken!
- Docker als einheitliches Format zur Auslieferung von Anwendungen
  - Kleines Hands-on: Container anderer Leute
- Ausblick: Container-Management
  - Die Geschichte von Phippy

Agenda

- Warum brauchen wir eine Alternative zur Server-Virtualsierung?
- Hands-on Teil 1: Installieren und ausprobieren
- Hands-on Teil 2: Wir bauen eigene Docker-Images
- Entwickler aufgepasst: Umdenken!
- Docker als einheitliches Format zur Auslieferung von Anwendungen
- Ausblick: Container-Management

Warum Isolation?

Verschiedene Prozesse sollen getrennt von einander laufen weil...

- unterschiedlicher Schutzbedarf
  z. B. Web-Anwendung und Finanzbuchhaltung
- Anwendungen unterschiedlicher Kunden
- unterschiedliche Administratoren
- Ressourcenverbrauch soll begrenzt werden
- ...

Anwendung A  Anwendung B

Icons von https://openclipart.org/ - Public Domain

Geschichte (1)

Woher kommen wir?

- Mainframes und große Unix-Systeme: Seit je her mehrere logisch getrennte Systeme auf einer Hardware:
  - S/360 ... S/390, z/OS, AIX: LPAR
  - z/VM
  - Solaris: Zones

https://openclipart.org/detail/20931/mainframe
Public Domaain

Geschichte (2)

- PC: Personal Computer
  - Ursprünglich für einen Nutzer ausgelegt
- Leistungsfähige PCs langweilen sich
- Virtualisierung: Mehrere OS-Instanzen auf einer Hardware
  - VMware Workstation / VMware Player
  - VMware ESX / ESXi
  - VirtualBox
  - XEN
  - KVM

Virtualisierung: Wie?

- Vollvirtualisierung = Emulation
  alles in Software
- Paravirtualisierung
  Gastsystem muss wissen, dass es virtualisiert läuft und in bestimmten Fällen den Hypervisor fragen, statt auf Hardware zuzugreifen
- Hardwareunterstützte Virtualisierung
  - Intel: VT-x
    ```
    grep --color vmx /proc/cpuinfo
    ```
  - AMD: AMD-V
    ```
    grep --color svm /proc/cpuinfo
    ```

https://de.wikipedia.org/wiki/X86-Virtualisierung

Heute: Cloud Computing

- Anwendung (oft Webanwendung) soll dynamisch skalieren
- Compute-Power soll preisgünstig angeboten werden
- Hohe Packungsdichte erforderlich
- Für jede Instanz ein eigenes Betriebssystem hochfahren? Zu viel Overhead!
- Daher: Container

Was macht einen Container aus?

Wir nutzen eigentlich nur ein paar Features des Linux-Kernels zur Ressourcen-Isolation, die schon immer da waren, um ein paar Prozesse ein bisschen einzusperren!

- cgroups
- Kernel Namespaces
- capabilities
- OverlayFS

Der (einzige) Verdienst von Docker:

- hat diese Features zusammen gepackt
- einfach (!!) nutzbar gemacht
- populär gemacht

Docker kocht auch nur mit Wasser!

Exkurs: Capabilities

- Capabilities: root-Rechte feingranular

```
man capabilities

grep CapEff /proc/self/status

capsh --decode=0000003fffffffff
```

Docker: Einsatzzweck

- Server-Anwendungen
- Docker ist nicht für Desktop-Anwendungen (mit GUI) gedacht
- Docker stellt Netzwerkports als Schnittstelle nach außen zur Verfügung
- Desktop-Anwendungen einsperren? → snaps
  https://wiki.ubuntuusers.de/snap/

Agenda

- Warum brauchen wir eine Alternative zur Server-Virtualsierung?
- Hands-on Teil 1: Installieren und ausprobieren
- Hands-on Teil 2: Wir bauen eigene Docker-Images
- Entwickler aufgepasst: Umdenken!
- Docker als einheitliches Format zur Auslieferung von Anwendungen
- Ausblick: Container-Management

Install

- Aktuelles Release von docker.com
- https://docs.docker.com/install/
  → Docker CE
  → Install using the repository
- https://docs.docker.com/install/linux/linux-postinstall/
  - Eigenen User in die Gruppe docker (als root ausführen)
    `usermod -aG docker <user>`
    Achtung: User bekommt damit sehr weitreichende Rechte! (fast wie sudo)
  - Prüfen (unter eigener User-ID)
    ```
    id | grep --color docker
    docker version
    ```

Funktionstest

```
docker run hello-world
```

Ausprobieren

```
~# docker run ubuntu echo "hallo welt"
hallo welt
~# docker ps
CONTAINER ID    IMAGE    COMMAND    CREATED    STATUS    PORTS    NAMES
~# docker ps -a
CONTAINER ID    IMAGE    COMMAND                [...] NAMES
96933722a10d    ubuntu   "echo 'hallo welt'"    [...] focused_wescoff
~# docker rm focused_wescoff
focused_wescoff
~# docker ps -a
CONTAINER ID    IMAGE    COMMAND    CREATED    STATUS    PORTS    NAMES
```

Container erkunden

```
~# docker run -ti --name first  ubuntu:latest bash

root@a956e3c2d0e2:/# ps uax
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0  18232  3240 pts/0    Ss   07:02   0:00 bash
root        11  0.0  0.0  34420  2896 pts/0    R+   07:02   0:00 ps uax

root@a956e3c2d0e2:/# ip a
bash: ip: command not found

root@a956e3c2d0e2:/# grep CapEff /proc/self/status
CapEff: 00000000a80425fb

root@a956e3c2d0e2:/# exit
exit

~# docker ps
CONTAINER ID    IMAGE    COMMAND    CREATED    STATUS    PORTS    NAMES

~# docker ps -a
CONTAINER ID  IMAGE          COMMAND  CREATED         STATUS         NAMES
a956e3c2d0e2  ubuntu:latest "bash"   A minute ago    Exited (127)   first

~# docker rm first
first
```

Image versus Container

- Ein Image ist die Basis jedes Containers
- Wir können beliebig viele Container starten, die das gleiche Image verwenden

```
docker search busybox
docker pull busybox
docker images
docker run -d busybox sleep 300
docker ps
docker run -d busybox sleep 400
docker ps
```

- Image lokal wieder löschen

```
docker rmi busybox
```

- Images näher untersuchen

```
docker history ubuntu
docker inspect ubuntu
```

Agenda

- Warum brauchen wir eine Alternative zur Server-Virtualsierung?
- Hands-on Teil 1: Installieren und ausprobieren
- Hands-on Teil 2: Wir bauen eigene Docker-Images
- Entwickler aufgepasst: Umdenken!
- Docker als einheitliches Format zur Auslieferung von Anwendungen
- Ausblick: Container-Management

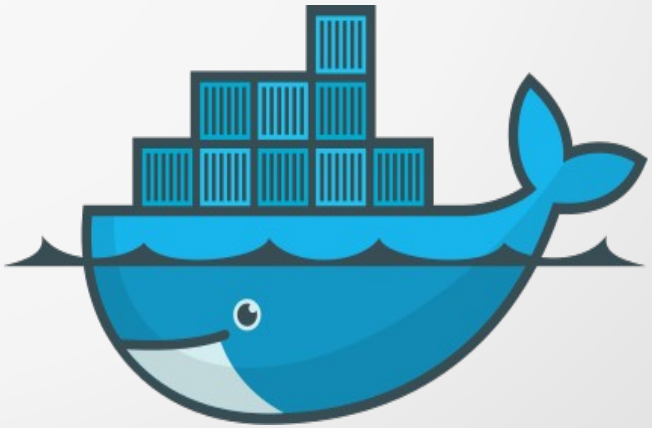
Ein eigenes Image

- Wir wollen irgendetwas „einsperren“ und bauen uns dazu ein eigenes Image

```
mkdir ~/Date
cd ~/Date
vim Dockerfile
```

```
FROM ubuntu:latest
MAINTAINER hans.wurst@example.com
CMD ["date"]
```

- Bauen
  `docker build -t date .`
- Kontrollieren
  `docker images`
- Laufen lassen
  `docker run --rm date`



Jetzt mal richtig!

- Dockerfile:

```
FROM ubuntu:latest
MAINTAINER hans.wurst@example.com
ENV DEBIAN_FRONTEND noninteractive
RUN apt-get update
RUN apt-get install -y nginx-light
RUN echo "daemon off;"  >>/etc/nginx/nginx.conf
RUN echo "Hallo Linux-Cafe! Hallo Welt!" > /var/www/html/index.html
RUN sed -i /etc/nginx/nginx.conf -e "s/access_log.*/access_log
\/dev\/stdout;/" -e "s/error_log.*/error_log \/dev\/stdout info;/"
EXPOSE 80
ENTRYPOINT ["/usr/sbin/nginx"]
```

- Bauen, laufen lassen, testen:

```
docker build -t nginx_hallo_welt .
docker run --rm -p 8080:80 -d nginx_hallo_welt
curl http://127.0.0.1:8080/
docker logs <containername>
```

Agenda

- Warum brauchen wir eine Alternative zur Server-Virtualsierung?
- Hands-on Teil 1: Installieren und ausprobieren
- Hands-on Teil 2: Wir bauen eigene Docker-Images
- Entwickler aufgepasst: Umdenken!
- Docker als einheitliches Format zur Auslieferung von Anwendungen
- Ausblick: Container-Management

Entwickler aufgepasst!

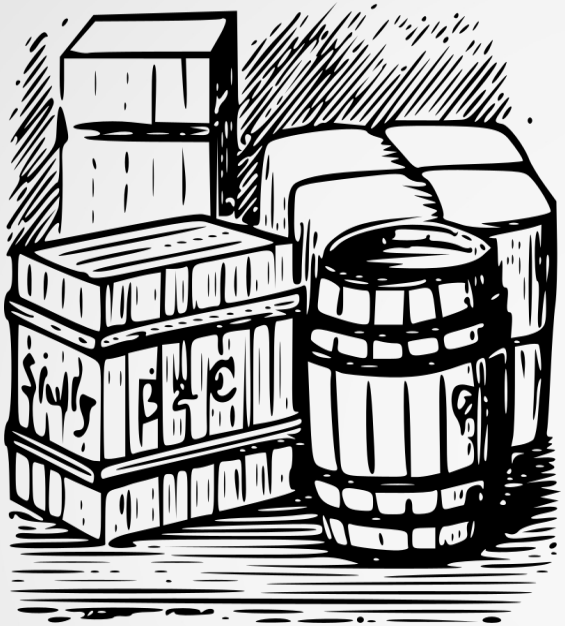
Umdenken ist angesagt! Alte Denkmuster komplett über Bord werfen!

- Container sind grundsätzlich stateless!
- Cattle versus Pets
- (Web-)Anwendungen gleich Cloud-Native entwickeln
- 12-factor Apps: https://12factor.net/
  - Stateless
  - Konfiguration per Environment Variablen
  - Loggen nach STDOUT
  - ...
- Idealerweise: 1 App pro Container
- Für Datenhaltungsschicht: Persistent Volumes

Agenda

- Warum brauchen wir eine Alternative zur Server-Virtualsierung?
- Hands-on Teil 1: Installieren und ausprobieren
- Hands-on Teil 2: Wir bauen eigene Docker-Images
- Entwickler aufgepasst: Umdenken!
- Docker als einheitliches Format zur Auslieferung von Anwendungen
  - Kleines Hands-on: Container anderer Leute
- Ausblick: Container-Management

Transport von Waren und Gütern



https://openclipart.org/detail/196313/goods
Public Domain

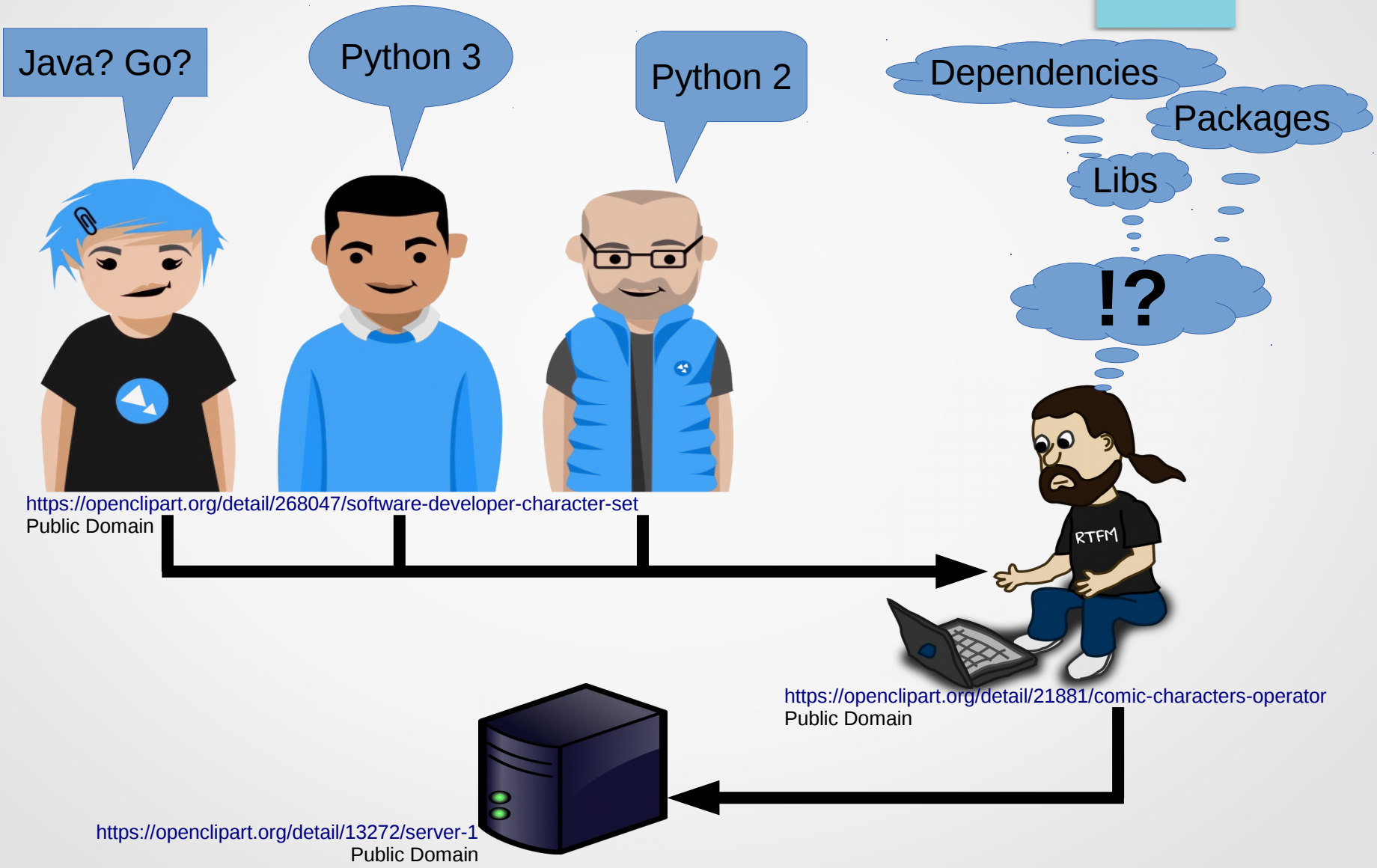
Transport
von Stückgut

Standardisierung

Transport von Containern
einheitlicher Größe

https://commons.wikimedia.org/wiki/File:Hamburg_Sud_container.jpeg
Creative Commons CC0 1.0 Universal Public Domain Dedication

Software Delivery



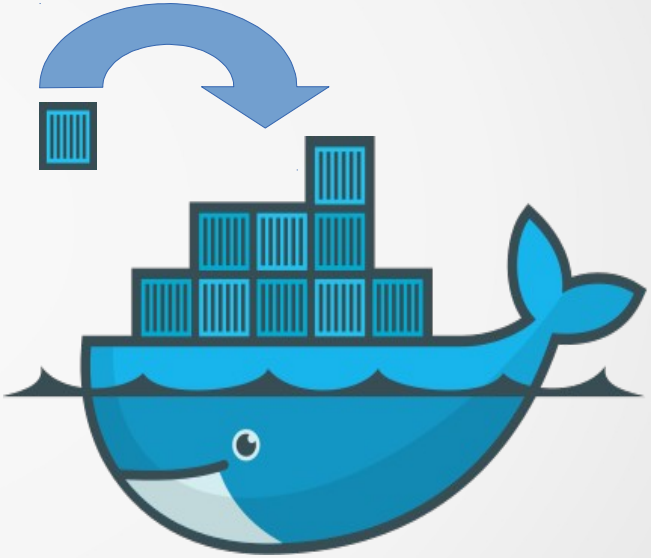
https://openclipart.org/detail/268047/software-developer-character-set
Public Domain

https://openclipart.org/detail/21881/comic-characters-operator
Public Domain

https://openclipart.org/detail/13272/server-1
Public Domain

Delivery im Container

- Entwickler liefert seine Anwendung im Container incl. zusätzlicher Pakete, Libs, …
- Lokale Tests sind einfach möglich
- Einheitlich im Betrieb
- Einheitliche Auslieferung



- Aber: Mehr Verantwortung für den Entwickler
  - Patchmanagement für mitgelieferte Pakete, Libs, ...

Best Practices

- Layered Containers
  - Layer 1: Basis-Linux
  - Layer 2: nginx
  - Layer 3: Python 3
  - Layer 4: Anwendung
- Bauen in einer Build- and Delivery Pipeline
  z. B. Jenkins oder GitLab CI
- Keine Images übergeben, sondern Dockerfiles in git
  - ➔ Image kann jederzeit neu gebaut werden, auch wenn nur Änderungen in den unteren Layers (Patches, Updates, ...)

Agenda

- Warum brauchen wir eine Alternative zur Server-Virtualsierung?
- Hands-on Teil 1: Installieren und ausprobieren
- Hands-on Teil 2: Wir bauen eigene Docker-Images
- Entwickler aufgepasst: Umdenken!
- Docker als einheitliches Format zur Auslieferung von Anwendungen
  - Kleines Hands-on: Container anderer Leute
- Ausblick: Container-Management

Container anderer Leute

- https://hub.docker.com/
- Die Qualität der dort veröffentlichten Container ist sehr unterschiedlich
- Namensschema: <Username>/<Containername>
  - Contributions der Community
- „official“ Containers
  - Basis-Images (ubuntu, centos, …)
  - Empfohlen, um darauf eigene Container aufzubauen
  - Oder Container, die Best Practices zeigen
  - Namensschema: <Containername>

Anwendungsbeispiel

Wir wollen Nextcloud ausprobieren:

- Docker Hub → Search: Nextcloud
- Wir finden nextcloud (official) → Details
  ```
  docker run -d -p 8080:80 --name cloudy nextcloud
  ```
- Nachschauen:
  ```
  docker ps
  ```
- Browser: http://localhost:8080/
- Anfangen mit Nextcloud zu spielen!

Agenda

- Warum brauchen wir eine Alternative zur Server-Virtualsierung?
- Hands-on Teil 1: Installieren und ausprobieren
- Hands-on Teil 2: Wir bauen eigene Docker-Images
- Entwickler aufgepasst: Umdenken!
- Docker als einheitliches Format zur Auslieferung von Anwendungen
  - Kleines Hands-on: Container anderer Leute
- Ausblick: Container-Management
  - Die Geschichte von Phippy

Container-Management

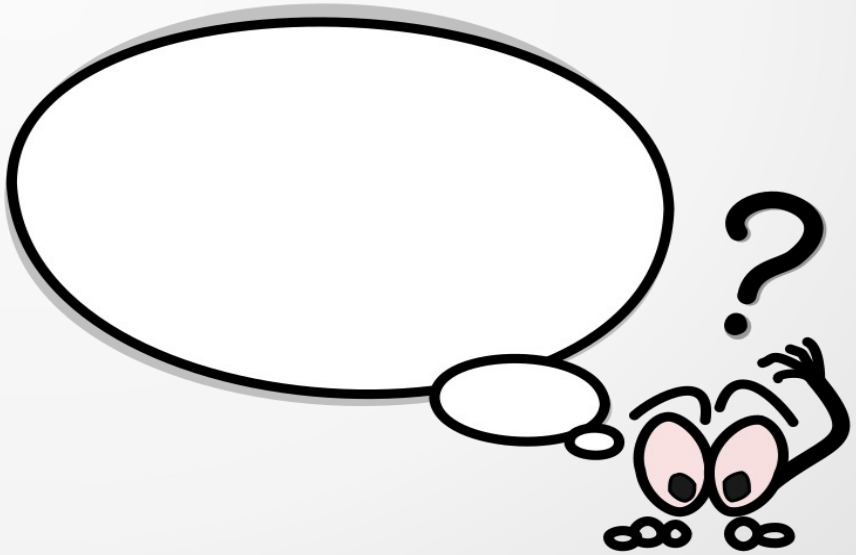
- Einzelne Container sind schön, in der Praxis treten sie aber meist in (größeren) Rudeln auf
- Das Rudel will gemanaged werden
- Container Orchestration Plattformen
  - https://www.openshift.org/
  - https://kubernetes.io/
  - https://www.openstack.org/

Die Geschichte von Phippy

- https://deis.com/blog/2016/kubernetes-illustrated-guide/
- https://youtu.be/4ht22ReBjno

Noch Fragen?

- Jetzt und hier
- Im Anschluß beim Bier
- Bei (fast) jedem Linux-Cafe, Gluga-Stammtisch,...
  http://termine.gluga.de/
- Jederzeit auf der
  Gluga Users
  Mailingliste
  http://mailing.gluga.de/



https://openclipart.org/detail/153/bulle-droite-question
Public Domain