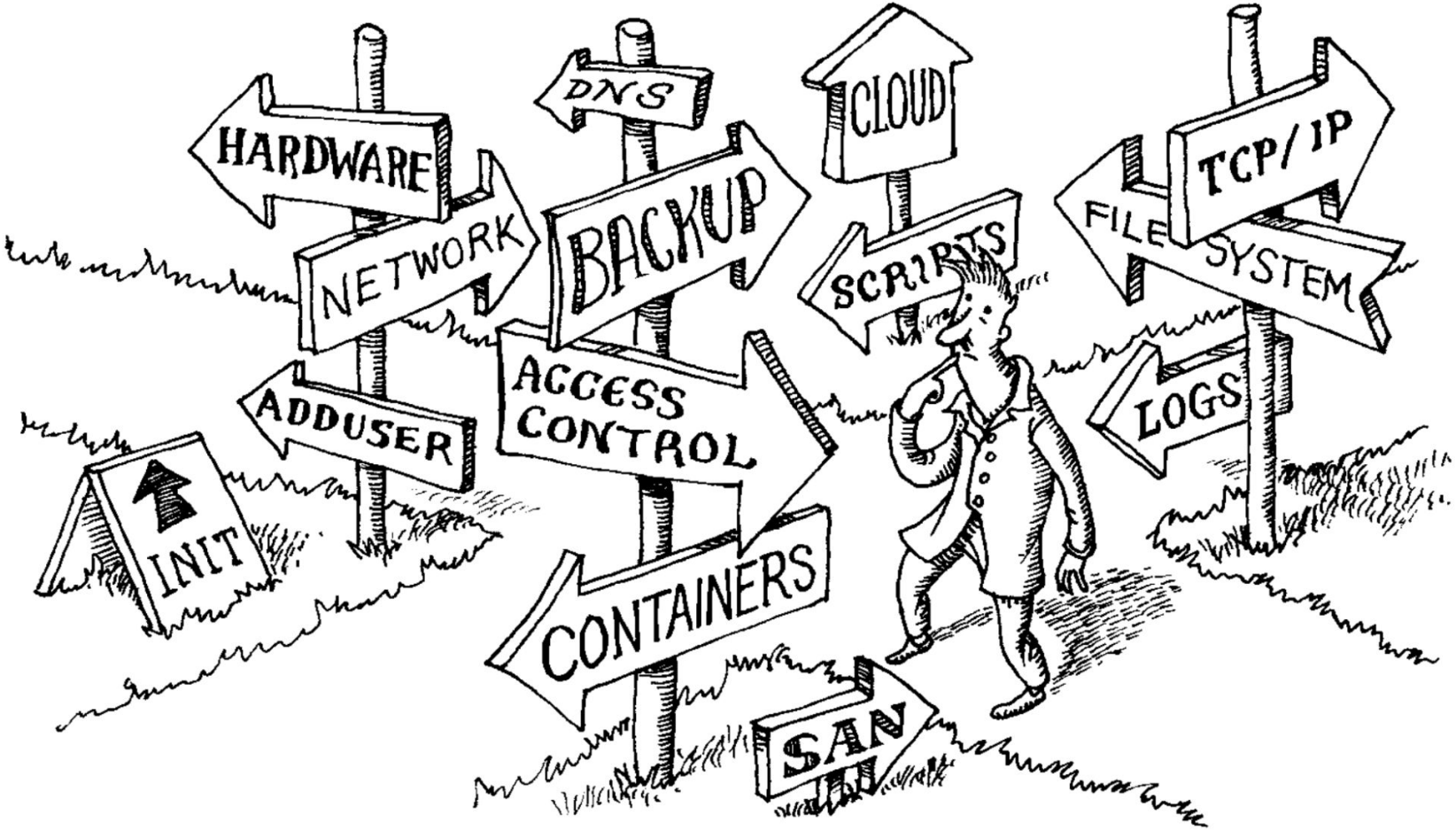
UNIX Systemadministration: Eine Einführung

15 Januar 2024

about://me

- Seit einem halben Jahr Systemadministratorin bei INET
- Viel Legacy Infrastruktur übernommen
- Entsprechend viel gelernt
- Ziel: Möglichst viel Wissen weitergeben


HARDWARE
NETWORK
ADDUSER
INIT
DNS
BACKUP
ACCESS CONTROL
CONTAINERS
SAN
CLOUD
SCRIPTS
TCP/IP
FILE SYSTEM
LOGS

Systemadministration

- Umfasst sehr viele Tasks
  - Software
  - Hardware
  - Security
  - Backups
  - Monitoring
  - Automatisierung
  - Dokumentation
  - Tech-Support
  - …

Die Unix-Philosophie

“Write Programs that do one thing and do it well”

- Einfachheit
- Modularität
- Wiederverwendbarkeit

➔ Aus kleinen Bausteinen komplexe Lösungen zusammenbauen

```
cat file.txt | uniq | sort  vs.  cat file.txt | sort -u
```

Beispiele von Anwendungen mit Unix-Philosophie

- Coreutils
  - cat, echo, cp, mv, rm
- Stream Editor
  - sed, awk
- Vi / Vim
- Curl / Wget
- Bash / zsh / fish
- Mutt
- Nginx
- Thunderbird

Services nach Unix-Philosophie bauen

- Services sind Anwendungen (Software + Hardware), die Kunden nutzen
- Oft über systemd realisiert
- Ziel: Simpelste Lösung, die alle Anforderungen erfüllt
  - Einfach zu maintainen, erweitern und integrieren
  - Nachdem du weg bist, muss irgendwer anders den Service maintainen
- Trennung von Aufgaben
  - Separate Services für z.B. Datenbank
- Die meisten Services sind bereits gebaut - es sollte richtig ausgewählt werden
- Monitoring ist **wichtig**

Unix Service: Email

- Von Email wird oft erwartet, dass es einfach so funktioniert
  - In der Realität ist das meist nicht ganz so einfach
- Der Mail Server besteht aus verschiedenen Komponenten
  - Mail Transfer Agent (MTA): Sendmail, Postfix, Exim
  - Mail Delivery Agent (MDA): Dovecot
  - Mail Access Agent (MAA): Fetchmail, Mutt, Thunderbird
  - Filter Agent (Anti-Spam): Rspamd, Amavis
  - Mailing-Listen (Bonus): Mailman, Aliase
- Vorgefertigte Lösung: Docker-Mailserver
  - Postfix, Dovecot, Rspamd, Amavis, ClamAV, Fail2ban, SASLauthd (LDAP)

Email: SPF, DKIM & DMARC

- Ziel
  - Authentizität des Senders prüfbar sicherstellen
  - Integrität des Nachrichteninhaltes sicherstellen
- SPF: Eintrag im DNS, welcher angibt, welche IP(s) senden dürfen
- DKIM: Kryptographische Signatur der Mail, public key im DNS
- DMARC: Sagt wie der Empfänger mit SPF / DKIM Failures umgehen soll
- Aber auch nicht immer perfekt: Siehe SMTP Smuggling
  - Konnte mit SPF checks von admin@microsoft.com senden
  - https://media.ccc.de/v/37c3-11782-smtp_smuggling_spoofing_e-mails_worldwide

Hardware

Server

- Beschreibt eine physische Maschine
- Sind immer an
- Halten lange
- Haben *viele* Nutzer
- Verschiedene Strategien zur Allokation
  - Riesen-Host: Alle Anwendungen auf einem Server
  - Unikate: Einzigartige Maschinen, welche unterschiedlich konfiguriert sind
  - VM-Cluster: Viele VMs, jede mit einer einzigen Aufgabe

Features von Servern

- Höhere Reliability Garantien
- CPU
  - Mehr Kerne, mehr PCIe
- RAM
  - Mehr, schneller, ECC
- Hardware RAID
  - Am besten RAID5 oder RAID6
  - RAID ist kein Backup!
- Mehrere PSUs
- Remote Management

Server Festplatten

- HDD vs. SSD
- HDDs sind billiger, SSDs sind performanter
- SSDs sind *viel* besser im random-access
  - Z.B. für Mail-Server sehr zu empfehlen
- SATA vs. SAS vs. NVME
  - 6Gb/s vs. 12 Gb/s vs. >32Gb/s
  - SAS Drives sind üblicherweise teurer, dafür mehr reliable
- Festplattenmonitoring mit SMART
  - Festplatten können sehr gut selbst messen wie viel Lebensdauer noch verfügbar ist
  - Scan error count, Reallocation count, Off-line reallocation count, Number of sectors on probation

Partitionierung

- Traditionell: Separate Partitionen für /home, /boot, /opt, /tmp, /var
- Heutzutage *können* die Ordner auch auf der gleichen Partition sein
  - BIOS hatte Limitationen bzgl. Festplattengröße
- Encrypted / ist nur mit separater /boot Partition möglich
- Für Server ist es generell eine sinnvolle Idee separate /boot Partition zu haben
- MBR vs. GPT
  - MBR kommt von 1980 Microsoft, kann keine Platten über 2TB
  - Keinen Grund außer Legacy MBR zu verwenden

Server-Dateisystem

- Was für Features könnte man wollen?
  - Crash-Recovery
  - Filesystem-RAID
  - Error detection & recovery
  - Copy-on-write
  - Snapshots
  - Compression
  - Deduplication

Server-Dateisystem Vergleich

	ext4	XFS	Btrfs	ZFS	NTFS	APFS
Crash Recovery	Journaling	Journaling	Ja	Ja	Journaling	Ja
Filesystem-RAID	Nein	Nein	RAID0 & RAID1	Ja	Nein	Nein
Error recovery	Nein	Nein	Ja	Ja	Nein	Nein
Copy-on-write	Nein	Nein	Ja	Ja	Nein	Ja
Snapshots	Nein	Nein	Ja	Ja	Nein	Ja
Compression	Nein	Nein	Zlib, LZO, Zstd	gzip, lz4, Zstd, ...	LZ77, LZNT1	Ja
Deduplication	Nein	Ja	Ja	Ja	Ja (Server)	Ja

Basierend auf https://en.wikipedia.org/wiki/Comparison_of_file_systems

Netzwerke

Netzwerke

- Verschiedene Netzwerke für verschiedene Aufgaben
  - Backup-Netzwerk
  - Administration von kritischen Systemen (Switches, Firewall, IPMI, …)
  - Drucker
- Vorteile
  - Verschiedene Firewall-Regeln
  - Isolation
  - Bandbreite

Firewall

- Was ist eine Firewall?
  - Software, die alle Paket überwacht und filtert, möglicherweise auch mit State
  - Überwacht Layer 3 & 4 (Ports & Protokoll)
  - Im Kernel möglich mit iptables und nftables
- Warum Firewall?
  - Sicherheit vor exposed ports
  - Um den Uplink in verschiedene Subnetze zu teilen
  - Monitoring
- Application-Firewalls
  - Z.B. für HTTP / HTTPS
  - Müssen TLS-Verbindungen aufbrechen können um HTTPS zu überwachen

Firewall Anwendungen

- Gibt es wie Sand am Meer
- Nftables
  - Die meisten Anwendungen sind nur Frontend für nftables
- Ferm
  - Frontend für iptables, legacy
  - Früher gut, falls falsche iptables Regeln gesetzt wurden
- Firewalld
  - Von RedHat
  - Web GUI
- Ufw
  - “Uncomplicated Firewall”
  - Sehr einfach zu nutzen

Security

Access Control (Filesystem)

- Findet in einer FS-unabhängigen Schicht (VFS) statt
- Jede Datei hat einen Owner und eine Gruppe
- Permissions sind durch 9 Bits konfiguriert (Read, Write, Execute)
- Das VFS kann als sicher angenommen werden
- Aber lieber mehrere Layer an Security! (z.B. Container)

```
-rw-rw-rw- 1 emily emily  26 Jan  1 19:22 all.txt
-rw-rw---- 1 emily admins 40 Jan  1 19:32 group.txt
-rw------- 1 emily emily  22 Jan  1 19:32 owner.txt
```

Security: SSH

- Wie greife ich auf die Kommandozeile meines Servers zu?
  - **Secure** Shell standardmäßig auf Port 22
- Authentifizierung über z.B. Passwort oder Public-Key
  - Besser nur Public-Key erlauben
  - Nachvollziehbarkeit wer sich angemeldet hat
- Keyformat: ed25519 für kürzere Keys
- Keys sollten mit Passwort geschützt sein
- ssh-audit zur Inspektion von Schwachstellen in der sshd_config
- Keine Authentifikationswege, die nicht gebraucht werden!

Root Login – Ja oder Nein?

- Convenience
- Login fürs tty

- Root darf *alles*
- Aus Versehen Dinge kaputt machen: `rm -rf /*`
- Falls kompromittiert, ist der gesamte Server kompromittiert
- Root-Aktionen werden nicht immer gelogged
- Wer hat was gemacht? → Nicht möglich nachzuvollziehen
- Teilen des Accounts ist schwierig
  - Andere Shells, Aliase, …

Security: Passwörter

- An vielen Stellen, werden Passwörter benötigt
  - Service / Admin Accounts, …
- Wie generiere ich starke Passwörter?
  - Random, lang, einzigartig
  - Grundsätzlich reichen auch 16 Zeichen (random) aus
  - Persönlich: `tr -dc qwertuiopasdfghjklxcvbnmQWERTUPASDFGHJKLXCVBNM123456789 </dev/urandom | head -c $(($1-1)) ; echo '.'` mit 64 Stellen
    - Um Größenordnungen mehr als eine UUID-4 (10^111 vs. 10^36)
- Passwörter rotieren nicht empfehlenswert
  - Führt zu weniger guten Passwörtern
- Wie verwalte ich meine Passwörter?
  - Passwort-Manager!

Security: 2FA

- Grundsätzlich empfehlenswert
- Ist aber keine Entschuldigung für schlechte Passwörter!
- Eher ein netter Bonus
  - 16 Stellen random Passwort wird nicht gebruteforced
- Verhindert das ablesen z.B. in der S-Bahn
- SMS vs. TOTP
  - TOTP ist sicherer und verlässlicher
    - Der TU SMS Dienst schmiert regelmäßig zu großen Prüfungen ab
  - Problem: Handy verloren
    - Google Authenticator speichert Codes im Konto

Security: Passwort-Manager

- Pass
  - GPG-Verschlüsselte Dateien in git
  - Vorteil: Durch Filesystem-Struktur ist viel Flexibilität möglich
  - Nachteil: Durch die Filesystem-Struktur können relevante Einträge nicht gebündelt werden
- KeePassXC
  - Verschlüsselte Datei auf Filesystem
  - Einträge mit Username + Passwort
  - SSH-Agent Integration
  - Custom Sync über Geräte (Nextcloud, …)
- VaultWarden
  - Open Source Rust Implementation von Bitwarden-server
  - Dateien auf Server, Postgresql Backend
  - Vaults sind über Geräte repliziert
  - Für Mobile nur die Bitwarden App
  - Hat bald SSO
- Dashlane, 1Password, …
  - Vertrauen in eine andere Entität
  - Kostet Geld
  - SSO, Integration, Features
  - Lastpass als Negativbeispiel
    - Incidents: 2015, 2021, 2022

Security: Passwörter verteilen

- User benötigen ein Passwort um auf $Service zuzugreifen
  - $Service ohne SSO Integration, keine Möglichkeit User Passwort erstellen zu lassen

- Passwort-Manager
  - Beste Option, falls Verfügbar
- Papier Zettel
  - Sehr sicher, falls physischer Zugang sichergestellt
  - Zettel danach schreddern
- Verschlüsselter Messenger
  - Signal, Matrix, Whatsapp
  - Kein Telegram, Mail
  - Benötigt Telefonnummern

- Per ssh auf Server legen
  - Benötigt public keys
- Asymmetrische Kryptographie
  - OpenSSL, GPG, PGP
- Einmal-Link
  - Z.B. VaultWarden hat das Feature
- Einmalpasswort
  - Nur Möglich, falls der Service das auch unterstützt

Backups

Warum machen wir Backups?

- Warum fallen Systeme aus?
  - Hardware Failure
  - Bit-Rot
  - Ransomware
- Welche Daten sichern?
  - Alles, was State ist
- Wenn was passiert ist das Geschrei groß
- “Kein Backup, kein Mitleid”

Backup-Konzept

- Komplett
- Inkrementell
  - Delta zur letzten Inkrementellen Sicherung
  - Reicht für die meisten Use-Cases aus
- Differentiell
  - Delta zur letzten Komplettsicherung Sicherung
  - Fängt an sinnvoll zu werden bei vielen Tapes
- Nicht alle Daten lassen sich auf Filesystem-level backupen
  - Postgresql, mysql

Backup-Schedule

- Komplett
  - Kommt auf die Größe der Daten und Geld an
  - Von 7 Tagen bis 30 / 60 / 90 Tagen alles in Ordnung

- Inkrementell / Differentiell
  - Täglich, nach Office-Zeit

Mögliche Backup-Medien

- HDD
  - Klassisch, Gut, Billig
  - Muss mit RAID gesichert werden
- Tape Storage
  - Große Einmalinvestition
  - Auf viel Speicher gerechnet deutlich billiger
  - Read-Only Schalter
  - Schlechte Random-Access performance
- Cloud
  - Kostet deutlich mehr
  - Verantwortung ist outgesourced
  - Benötigt guten Up- und Downlink
  - GitHub, Google Drive, DropBox, …

Redundanz

- Wie Backupe ich meinen Backup Server?
- Mehr Backup Server!
  - Offsite?
  - Kosten
  - Mehr Netzwerkauslastung
  - Wirtschaftlichkeit abwägen: Ab wann kostet es zu viel?
- Testen! Kann ich die Backups wiederherstellen?

Wie erstelle ich Backups?

Client

- Der Client initiiert das Backup

Vorteile

- Einfach zu implementieren
- Viel gute Software
- SSH-Keys liegen auf den Clients
- Clients haben Kontrolle darüber wann ein Backup erstellt wird
- NAT Traversal

Nachteile

- Lastspitzen
- Jeder Client muss konfiguriert werden
- Möglicherweise inkonsistente Configs

Server

- Der Server initiiert das Backup

Vorteile

- Zentralisierte Verwaltung
- Skalierbar durch wenig Konfiguration
- Einmal konfigurieren, dann nur noch hosts hinzufügen

Nachteile

- Es gibt einen “master” SSH-Key
- Wenig Software mit allen Features
- Single Point of Failure
- Die Clients müssen immer online sein
- Die Clients brauchen eine statische IP

Client Backup Software

- Filesystem Snapshots
  - Btrfs, zfs
  - Hilft nicht gegen Plattenausfall
- Rsync
  - Gut geeignet für viele kleine Dateien
  - Delta Algorithmus
  - Keine Encryption
  - Backup-Skript selbst schreiben
- Tar
  - Gute Auslastung des Netzwerks
  - Encryption, Compression
  - Möglichkeit für inkrementelle Backups
  - Es dauert das Archiv zu erstellen
  - Nicht so effizient wie rsync
- Borg
  - Speed, Encryption, SSH
  - CLI
- Kopia
  - SSH, WebDAV, Cloud
  - Web UI, GUI
  - Content-Addressable Storage
  - Error-Correction gegen Bit-Rot
- Duplicati
  - Cloud: Google Drive, OneDrive, WebDAV
  - Web UI
  - Block-Based Storage
  - Funktioniert mit OAuth
  - Daten sind nicht “easily” accessible

Server Backup Software

- Client Software deployed über Ansible
  - Borg, Burp, …
  - Durch Ansible wird Konsistenz der Konfiguration gewährleistet
- Selbstgeschriebene Rsync Skripte
  - Viel manuelle Skripterei
  - Funktion nicht garantiert
- Rsnapshot
  - Rsync über deklarative Config
  - Staggered File Versioning
  - Durch `backup_script` auch Datenbank Backup möglich
  - Keine Web UI
- UrBackup
  - Benötigt eigene Client-Software
  - Benötigt eigene Ports
  - Duplizierte Dateien von verschiedenen Clients werden nur 1× gespeichert
- BackupPC
  - Frontend für rsync
  - Speichert die Checksummen
  - Mails, Multi-User, Monitoring
  - Kein Staggered File Versioning
- Bacula
  - Sehr enterprisig
  - Hat (in gekauft) alle features
  - Sehr kompliziert
  - Hat 5 verschiedene Docker Container

Fall: Neu im Fachgebiet

Legacy: Bestand erfassen

- Hardware
  - Was für Geräte haben wir?
  - Was für Daten haben die?
  - Wie sind die gegen Ausfall geschützt?
  - Wie alt sind die Geräte? → `uptime`
- Software
  - Was läuft auf diesen Servern?
  - Was für Datenbanken haben wir?
- Netzwerktopologie
  - Wie ist das Netzwerk aufgebaut?
  - Wie sind Geräte miteinander verbunden?
  - Welche Firewall regeln sind vorhanden? Existiert eine Firewall?

Legacy

- Sicherheitsüberprüfung
  - Sind die Paketquellen noch aktuell?
  - Ist die Software noch aktuell?
  - Was für ein Kernel läuft? Hat der Schwachstellen?
- Configs von Software checken
  - PAM
  - SSH: `authorized_keys`, mit `ssh-audit` die `sshd_config` überprüfen
- Backups
  - Haben wir Backups?
  - Lassen die sich einspielen?

Ein neuer Anfang: Hardware

- Damit *irgendwas* laufen kann benötigt es Server
- Cloud ist meist keine Option wegen Bezahlung
  - Die TU Berlin kann nur auf Rechnung mit Zahlungsziel 4 Wochen kaufen
- Existiert bereits legacy Hardware? Kann die weiterverwendet werden?
- Was für Server kaufen?
  - Viele (16 - 32) schnelle Kerne (>3 GHz) mit
  - Dauert sehr lange, bis die da sind
- Wo hinstellen?
  - Die Meisten Fachgebiete haben Rackspace im E-N Rechenzentrum

Ein neuer Anfang: Hardware Setup

- Persönliche Empfehlung
- Besten Server ausfindig machen
  - Für Web-Services eignen sich 8-16 schnelle Kerne, 32-64GB RAM und SSDs
- Neue SSDs kaufen
  - Consumer SSD Speicher ist sehr billig, dürfen gerne 2TB pro Platte sein (~150€)
- NixOS
- ZFS, OpenZFS Guide

- Dokumentiere wie und was du installiert hast!
  - Für den Anfang reicht in GitHub Repo mit Markdown Dateien

Essenzielle Services

- Wiki → Dokumentation
  - Schreib alles auf, was du tust. Deine Nachfolger wird es dir danken
  - Falls noch keine Backups vorhanden, ist GitHub ein guter Speicherort
- Backups
- Monitoring
  - Server Load (RAM, CPU, Speicher, Temperatur), Backups, SMART / RAID Werte, Netzwerkauslastung, Security (`authorized_keys`), Logs / Errors
- SSO
  - OpenLDAP, FreeIPA, Keycloak
- Cloud
  - Nextcloud, TrueNAS, …
- Drucker
- VPN

Wie Dokumentation?

- Wiki
  - Git, Plain-Text (Markdown), Permission Management?
  - Wiki.js (INET), Gollum, MediaWiki (FRunde)
- Kein Riesen-Dokument an Doku, viele kleine Dateien
  - Zeitaufwand minimieren um essentielle Information zu finden
- Kurz, Consise, To the point
- Oft braucht es nur kleine Teile (Ordner) der Doku
  - Einfach zu updaten
  - Einfach zu maintainen

Hardware Dokumentation

- Was an Hardware(-Zuordnung) gibt es zu managen?
  - Server
    - Wo im Rack stehen die?
    - Was haben die für Specs?
    - Was für IP Adressen haben die?
  - Switches, Router, USV, Patch Panel, …
  - VLANs / Netzwerke
  - VPNs
  - Virtuelle Maschinen
  - Verschiedene Standorte

- Eine sehr gute Lösung: Netbox
  - Ausprobieren: demo.netbox.dev

To LDAP or not to LDAP

- Sehr viele Services supporten LDAP
  - Einige supporten auch nur LDAP
    - Unix Accounts
- Sehr Flexibel
  - Kann quasi alles abbilden
- Im Vergleich zu Active Directory sehr schlank
- Kann mit Kerberos kombiniert werden
- Wir werden LDAP nicht los wegen AD

- Komplex & Alt
  - Macht maintenance anstrengend
  - LDIF Format ist gewöhnungsbedürftig
- Kann kein OAuth2
  - Lässt sich mit Keycloak verbinden

(Nicht so) Essenzielle Services

- Email
- DNS
  - Falls eigene, selbst verwaltete, Domain vorhanden
- Helpdesk / Ticket-System
  - YouTrack (Jetbrains)
- Webhosting
- VM-Host
  - Proxmox, Xen, VMWare
- Kommunikationssoftware
- Domain-Management

Proxmox vs. Xen

- Früher: Xenserver von Citrix
- Heute: XCP-ng fork von Xenserver, Community-Driven
- Proxmox kann Container (LXC)
- Xen macht LVM over iSCSI
  - Falls du mal Daten recovern musst ist das ***ein pain in the ass***
  - Keine Sinnvolle Backup-Lösung
- Xen hat ein 2TB limit für VDIs
- Xen Hypervisor vs. KVM
  - Inzwischen nutzen AWS, Telekom, … nutzen KVM statt Xen

Moderne Systemadministration mit NixOS und Docker

Warum NixOS?

- Nix: Funktionaler Paketmanager ohne Seiteneffekte
- Eine Config, die das gesamte System beschreibt (deklarativ)
  - Versioniert durch git, für die Ewigkeit
- Alle Dateien sind unter /nix/store/...
  - Alle benötigten Dateien werden gesymlinked
- Atomische Upgrades & Rollbacks
  - nixos-rebuild switch --rollback
- Reproduzierbarkeit
- ISO-Datei aus Config erstellen (zum installieren)

NixOS: Deklarative Config

```
cfg = {
  services.wiki-js = {
    enable = true;

    settings.db = {
      db = "wiki-js";
      host = "/run/postgresql";
      user = "wiki-js";
    };
  };
};
```

```
cfg.services.hedgedoc = {
  enable = true;
  environmentFile = config.age.secrets.HedgeDoc_EnvironmentFile.path;

  settings = {
    domain = "${SUBDOMAIN}.${config.domainName}";
    allowOrigin = [ "localhost" "${SUBDOMAIN}.${config.domainName}" ];
    host = "0.0.0.0";
    protocolUseSSL = true;

    db = {
      dialect = "postgres";
      host = "/run/postgresql";
    };

    # Users and Permissions
    email = false;
    allowAnonymous = false;
    allowEmailRegister = false;
    allowFreeURL = true;
    requireFreeURLAuthentication = true;
    defaultPermission = "limited";
```

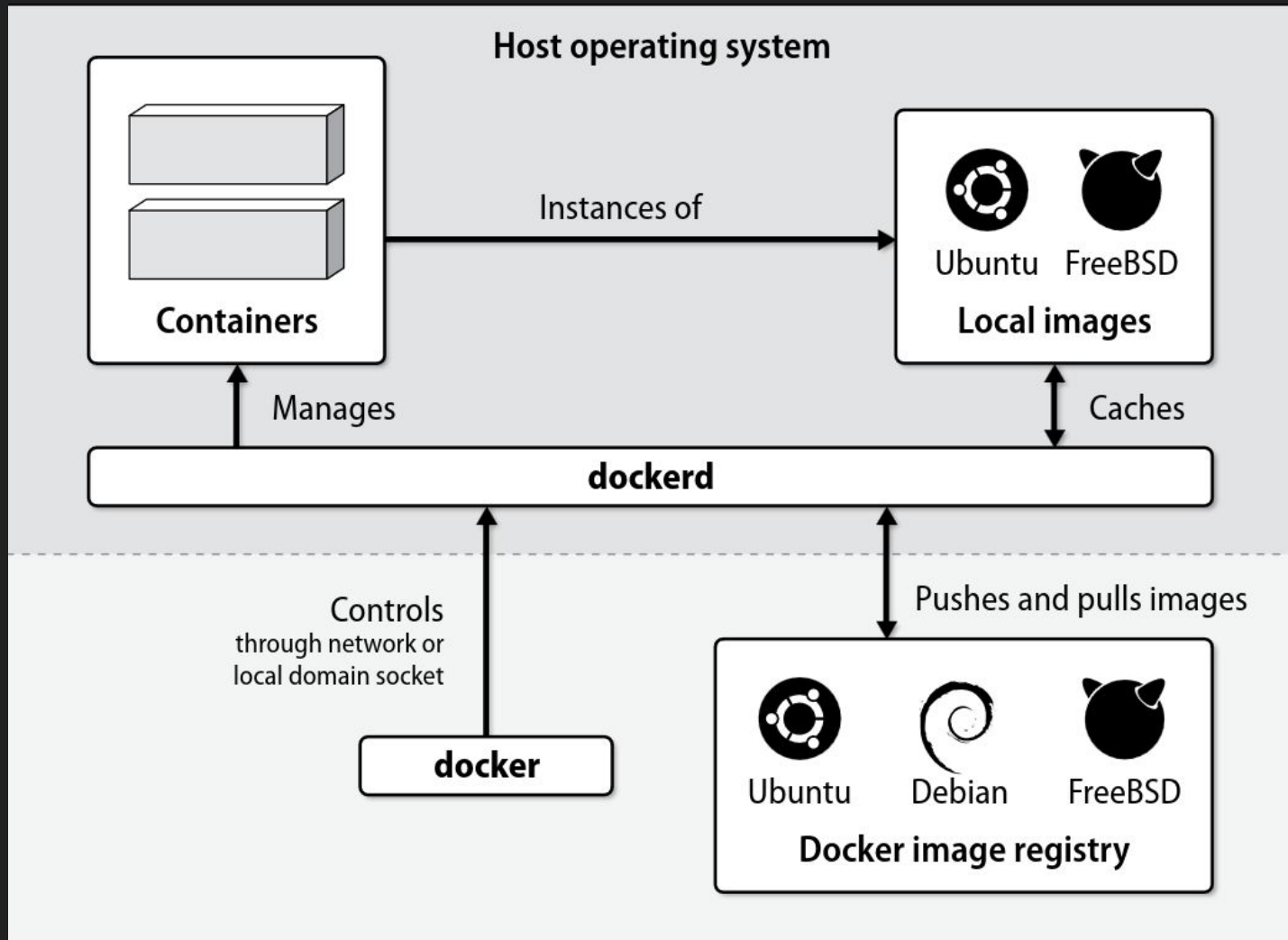
NixOS: Caveats

- Source Code lesen ist meist einzige Option für Dokumentation
- Die Dokumentation ist manchmal veraltet
- Dokumentation ist manchmal nicht vorhanden
- Imperatives System oft schneller aufsetzbar (“quick and dirty”)
  - Über lange Zeit (und viele Installationen) gleicht sich das aus
- Es gibt nicht *jede* Software
  - Docker Container können Abhilfe schaffen
  - Insbesondere bei kommerzieller Software problematisch
- Rebuilden von Configs ist langsam (vor allem auf alten Maschinen)
- Bei Upgrades *können* Dinge immernoch kaputt gehen

(Docker) Container

- Früher: Viele VMs mit z.B. Debian und einem einzigen Service
- Ziel: Gleiches level an Isolierung beibehalten, aber performanter
  - Zusätzlicher Vorteil: Es braucht kein großes Cluster + Hypervisor
- Ein Container ist eine isolierte Gruppe an Prozessen
  - Isolierung durch Namespaces, CGroups & privatem rootfs
- Container teilen sich den Kernel, VMs nicht
- Für gewöhnlich über Environment-Variablen konfiguriert
- Definierte Volumes, welche in den Container gemounted werden
- Bekannt geworden durch Docker, besitzen aber standardisiertes Format

Docker: Architektur



Docker: Security

- Docker-Container sind für gewöhnlich secure
  - Root Zugriff im Container sollte nicht zu Root auf Host führen
  - Sobald ein Container kompromittiert ist, kann dieser mit allen reden
- `dockerd` läuft mit root privileges, falls der exploited wird: problematisch
  - Gab schonmal privilege escalation CVE-2019-5736
  - Gibt aber auch inzwischen rootless Docker
  - Selbst arbitrary Commands können Problematisch sein: `/` unter `/host` im Container
- Docker Daemon Socket = root Zugriff
- Malicious Container sind meist “nur” Kryptominer

Docker vs. Podman

- Daemon: dockerd
- Möglichkeit für rootless
- Monolithisches System

- Kein zentraler Daemon
- Möglichkeit für rootless
- Modularer
  - Benötigt Tools wie Buildah, Skopeo und runC

- Kompatibel mit Docker
  - Registry und CLI

NixOS: Deklarative Container

```
virtualisation.oci-containers.containers.mailman-core = {
  image = "maxking/mailman-core:0.4";

  extraOptions = [ "--ip=10.88.2.1" ];
  ports = [
    "127.0.0.1:8001:8001"
    "127.0.0.1:8024:8024"
  ];

  environment = ENVIRONMENT_CONFIG;
  environmentFiles = [ config.age.secrets.MailMan_EnvironmentFile

  volumes = [
    "/data/MailMan/mailman-core:/opt/mailman"
  ];

};
```

```
virtualisation.oci-containers.containers.mail = {
  image = "mailserver/docker-mailserver:latest";

  ports = [
    "25:25"
    "465:465"
    "587:587"
    "993:993"
    "4190:4190"
    "11334:11334"
  ];

  volumes =
  [
    "/data/Mail/mail-data:/var/mail"
    "/data/Mail/mail-state:/var/mail-state"
    "/data/Mail/mail-logs:/var/log/mail"
    "/data/Mail/config:/tmp/docker-mailserver"
    "/etc/localtime:/etc/localtime"
```

Quellen / Literaturempfehlung

- UNIX and Linux System Administration Handbook (ISBN: 978-0134277554)
- The Practice of System and Network Administration (ISBN: 978-0321919168)