FAIme

Ein Dienst für individuelle Installationsmedien und VM Abbilder

Thomas Lange, Debian Entwickler
Sysadmin an der Universität zu Köln

lange@debian.org

FFG 2019

finger Mrfai@localhost

- whoami
  - Diplominformatiker
  - Systemadministrator an der Uni Köln seit über 25 Jahren
  - SunOS 4.1.1 auf SPARC
  - Solaris Jumpstart
  - 1999 erstes Cluster (16× Dual PII 400MHz)
  - FAI seit über 19 Jahren
  - Debian Entwickler seit 2000
  - Mitglied im Debian Cloud Team, Web Team
  - Vorträge und Tutorials auf zahlreichen Konferenzen:

    Linux Kongress, Linuxtag, DebConf, SLAC, LCA, FOSDEM, CeBit, OSDC, UKUUG, FrOSCon, Grazer Linuxtage, Chemnitzer Linuxtag, GUUG Frühjahrsfachgespräch
  - FAI Schulungen

Motivation

- Debian Installer ist nicht tauglich für Anfänger
- Mind. 24 Fragen, oft nur ENTER drücken
- Einige Fragen sind von einem Anfänger nicht zu beantworten
- Es fehlt ein Modus mit weniger Fragen
- Andere Distributionen machen Installation deutlich einfacher

Die Idee

- Sollte die üblichen Installationen abdecken
- Ignoriere Spezialfälle
- Frage nur die wirklich wichtigen Dinge
- Frage alles am Anfang
- Erstelle ein maßgeschneidertes Installationsmedium
- Boote den Rechner und hole dir einen Kaffe
- Fertig! Arbeite mit dem neu aufgesetzten Rechner

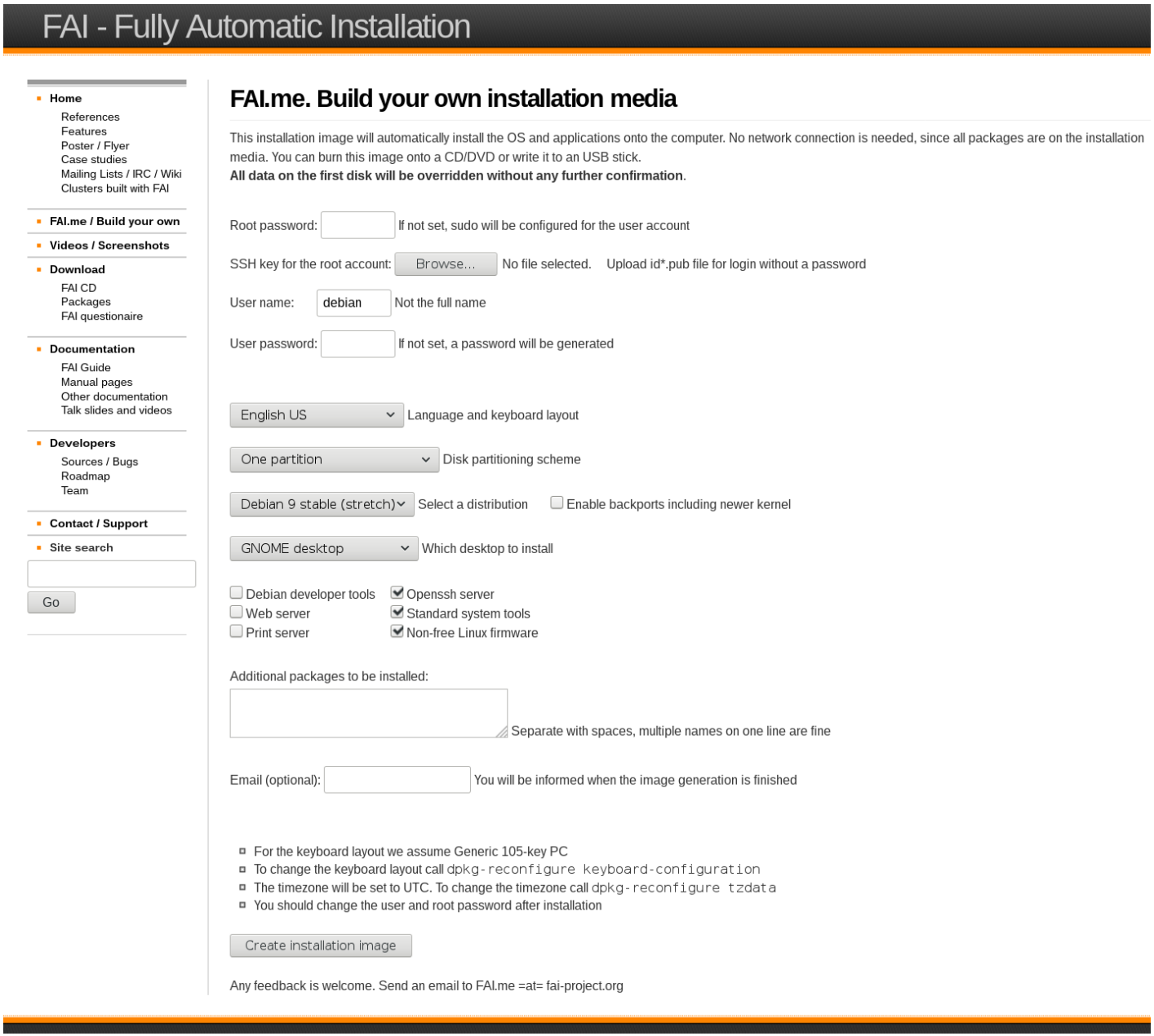
Das klingt ein wenig nach FAI

- FAI = Fully Automatic Installation
- FAI ist ein Tool für erfahrene Syadmins
- FAI ist aber auch nicht tauglich für Anfänger `:-(`
- Wie kann man es nutzbar für Linux User machen?

FAI.me

`https://fai-project.org/FAIme`

FAI.me


FAI - Fully Automatic Installation
Home
References
Features
Poster / Flyer
Case studies
Mailing Lists / IRC / Wiki
Clusters built with FAI
FAI.me / Build your own
Videos / Screenshots
Download
FAI CD
Packages
FAI questionaire
Documentation
FAI Guide
Manual pages
Other documentation
Talk slides and videos
Developers
Sources / Bugs
Roadmap
Team
Contact / Support
Site search
Go
FAI.me. Build your own installation media
This installation image will automatically install the OS and applications onto the computer. No network connection is needed, since all packages are on the installation media. You can burn this image onto a CD/DVD or write it to an USB stick.
All data on the first disk will be overridden without any further confirmation.
Root password:
If not set, sudo will be configured for the user account
SSH key for the root account:
Browse...
No file selected.
Upload id*.pub file for login without a password
User name:
debian
Not the full name
User password:
If not set, a password will be generated
English US
Language and keyboard layout
One partition
Disk partitioning scheme
Debian 9 stable (stretch)
Select a distribution
Enable backports including newer kernel
GNOME desktop
Which desktop to install
Debian developer tools
Web server
Print server
Openssh server
Standard system tools
Non-free Linux firmware
Additional packages to be installed:
Separate with spaces, multiple names on one line are fine
Email (optional):
You will be informed when the image generation is finished
For the keyboard layout we assume Generic 105-key PC
To change the keyboard layout call dpkg-reconfigure keyboard-configuration
The timezone will be set to UTC. To change the timezone call dpkg-reconfigure tzdata
You should change the user and root password after installation
Create installation image
Any feedback is welcome. Send an email to FAI.me =at= fai-project.org

FAI.me Installationsmedium

- Einfache Erstellung eines Installationsmediums
- Einfache Anpassungen möglich machen

- **Sprache, Benutzer + Passwort**, root Passwort, SSH Key
- Alle **Desktops** auswählbar
- Distributionen: **stable, stable+backports, testing**
- **Tools, Non-free Firmware, Web, Print, SSH Server**
- **Eigene Paketliste**
- Varianten für die Partitionierung (`/home`, `LVM`)

Cloud Images

- Keine Installation
- Diskimage kann direkt booten
- Plattengröße
- Format (raw.zst, qcow2, vmdk,vhdx, vdi)
- Hostname
- Immer nur eine Partition

FAI.me weitere Ideen

- FAI kann Diskimages auch cross Architektur bauen
- D.h. arm64 Plattenimage auf einem amd64 bauen
- Andere Distributionen (z.B. Ubuntu, CentOS)
- Einfacher Modus, weniger Boxen
- Generisches FAI.me Installationsmedium, ohne Pakete
- Wer noch mehr will, sollte einen eigenen FAI server aufsetzen

KIWI

OpenSUSE: KIWI, the OS image and appliance builder

- PREPARE (baut die chroot)
  1. Create Target Root Directory
  2. Install Packages
  3. Apply the Overlay Tree
  4. Apply Archives
  5. Execute the User-defined Scripts `config.sh`
  6. Manage The New Root Tree

- CREATE (erzeugt das Image)
  1. Execute the User-defined Script images.sh
  2. Create Requested Image Type

config.kiwi — openSUSE-Leap-42.3-EC2-HVM-Guest

```
image schemaversion="6.2" name="openSUSE-Leap-42.3-EC2-HVM" displayname="openSU
    <description type="system">
        <author>Public Cloud Team</author>
<contact>public-cloud-dev@susecloud.net</contact>
<specification>openSUSE Leap 42.3 guest for Amazon EC2 - HVM</specification>
    </description>
    <preferences>
        <!-- kiwi 7 has no 42.3 boot descriptions, use the old description -->
        <!--suse-leap42.1-->
        <type image="vmx" filesystem="ext4" boot="vmxboot/suse-SLES12" bootload
            kernelcmdline="console=ttyS0 multipath=off net.ifnames=0 memhp_defa
            <size unit="M">10240</size>
        </type>
        <version>0.1.4</version>
        <packagemanager>zypper</packagemanager>
        <locale>en_US</locale>
        <keytable>us.map.gz</keytable>
        <timezone>utc</timezone>
        <hwclock>utc</hwclock>
        <rpm-excludedocs>true</rpm-excludedocs>
    </preferences>
```

config.kiwi

```
users group="root">
        <user password="$1$wYJUgpM5$RXMMeASDc035eX.NbYWFl0" home="/root" name="root"/>
    </users>
    <!-- Get the latest command line tools and initialization code -->
    <repository type="rpm-md">
        <source path="obsrepositories:/"/>
    </repository>
    <drivers>
        <file name="drivers/nvme/*"/>
        <file name="drivers/nvmem/*"/>
    </drivers>
    <packages type="image" patternType="plusRecommended">
        <!-- jeos server -->
        <package name="patterns-openSUSE-minimal_base"/>
        <package name="dhcp-client"/>
        <package name="fontconfig"/>
        <package name="fonts-config"/>
        <package name="grub2"/>
.
.
.
        <package name="cloud-init"/>
        <!-- end framework specific packages -->

        <archive name="kiwi-hooks.tgz" bootinclude="true"/>
    </packages>

    <packages type="bootstrap">
        <package name="openSUSE-release"/>
        <package name="openSUSE-release-dvd"/>
        <package name="filesystem"/>
        <package name="glibc-locale"/>
    </packages>
</image>
```

config.sh

```
# These are all set by YaST but not by KIWI
baseUpdateSysConfig /etc/sysconfig/bootloader LOADER_TYPE grub2
baseUpdateSysConfig /etc/sysconfig/console CONSOLE_ENCODING "UTF-8"
baseUpdateSysConfig /etc/sysconfig/kernel INITRD_MODULES "ext4"
baseUpdateSysConfig /etc/sysconfig/keyboard COMPOSETABLE "clear latin1.add"
baseUpdateSysConfig /etc/sysconfig/language INSTALLED_LANGUAGES ""
baseUpdateSysConfig /etc/sysconfig/language RC_LANG "en_US.UTF-8"
baseUpdateSysConfig /etc/sysconfig/network/config NETCONFIG_MODULES_ORDER "cloud-netconfig dns-resolver dn
baseUpdateSysConfig /etc/sysconfig/network/dhcp DHCLIENT_SET_HOSTNAME no
baseUpdateSysConfig /etc/sysconfig/network/dhcp WRITE_HOSTNAME_TO_HOSTS no
baseUpdateSysConfig /etc/sysconfig/SuSEfirewall2 FW_DEV_EXT "any eth0"
baseUpdateSysConfig /etc/sysconfig/SuSEfirewall2 FW_LOG_DROP_CRIT yes
baseUpdateSysConfig /etc/sysconfig/SuSEfirewall2 FW_LOG_DROP_ALL no
baseUpdateSysConfig /etc/sysconfig/windowmanager X_MOUSE_CURSOR ""
baseUpdateSysConfig /etc/sysconfig/windowmanager DEFAULT_WM ""

echo 'DEFAULT_TIMEZONE="UTC"' >> /etc/sysconfig/clock
egrep -q '^xvc0$' /etc/securetty || echo xvc0 >> /etc/securetty

sed -i 's/#ChallengeResponseAuthentication yes/ChallengeResponseAuthentication no/' /etc/ssh/sshd_config

# Remove the password for root
sed -i 's/$1$wYJUgpM5$RXMMeASDc035eX.NbYWFl0/*/' /etc/shadow

suseInsertService cloud-init-local
suseInsertService cloud-init
suseInsertService sshd
suseRemoveService boot.efivars
suseRemoveService boot.lvm
suseRemoveService kbd
suseRemoveService acpid
```

config.sh

```
function hideSplash
function createInitialDevices {
function setupNFSServices {
function startPlymouth {
function readVolumeSetupAllFree {
function installBootLoader {
function pxeSetupDownloadServer {
function preparePartitionTable {
function createHybridGPT {
function luksResize {
function readVolumeSetupAllFree {
function setupNFSServices {
function deactivateVolumeGroup {
function setupKernelModules {
function setupBootPartitionPXE {

baseCleanMount
baseStripTranslations
baseSetRunlevel 3
baseStripLocales
suseImportBuildKey
suseInsertService sshd
suseRemoveYaST
baseInsertService network
suseRemoveService avahi-daemon
```

config.kiwi

```
<image schemaversion="6.8" name="LimeJeOS-CentOS-07.0">
    <strip type="delete">
        <file name="/usr/lib*/python*"/>
        <file name="/usr/share/zoneinfo"/>
        <file name="/usr/share/i18n"/>
    </strip>
    <preferences>
        <packagemanager>yum</packagemanager>
        <locale>en_US</locale>
        <type image="vmx" primary="true" filesystem="ext3" kernelcmdline="rhgb" bootloader="grub2" firmwar
        <type image="oem" initrd_system="dracut" filesystem="ext3" installiso="true" bootloader="grub2" fi
            <oemconfig>
                <oem-systemsize>2048</oem-systemsize>
                <oem-swap>true</oem-swap>
                <oem-swapsize>200</oem-swapsize>
                <oem-multipath-scan>false</oem-multipath-scan>
            </oemconfig>
        </type>
    </preferences>
    <repository type="rpm-md" alias="centos">
        <source path="http://bo.mirror.garr.it/1/slc/centos/7.2/os/x86_64"/>
    </repository>
    <packages type="image">
        <namedCollection name="core"/>
        <namedCollection name="console-internet"/>
        <package name="syslinux"/>
        <package name="grub2"/>
        <package name="kernel"/>
    </packages>
    <packages type="oem">
        <package name="dracut-kiwi-oem-repart"/>
        <package name="dracut-kiwi-oem-dump"/>
    </packages>
</image>
```

KIWI

- Legacy KIWI war Perl, jetzt kiwi-ng 9.X in Python
- XML Konfig ist einfach nicht gut editierbar
- 157 paketnamen bei SLES 12 based Amazon EC2 Image
- Davon 57 yast2 Pakete (keine Auflösung der Abhänigkeiten?)
- Als Konfig management nur Shell
- Viele Funktionen für config.sh verfügbar
- Doku unübersichtlich: Wie kann ich /etc/fstab.append und /etc/fstab.patch nutzen?
- RELAX NG Schema for the Extension, API Documentation 9.x, Schema Documentation 6.9
- 227 mal wird Python Command.run aufgerufen, cp, mkdir, ln
- Python als Shell wrapper?
- Guter Start: `https://github.com/SUSE/kiwi-descriptions`

PACKER

Packer is a tool for creating identical machine images for multiple platforms from a single source configuration

- Tool in Go von Hashicorp
- Vagrant, Terraform, Vault, Consult, Nomad
- Templates sind Konfiguration in JSON
- Builders: EC2, Azure, Hetzner Cloud, OpenStack, Qemu
- Provisioners: Ansible, Chef, Puppet, Shell, Salt, PowerShell
- Post-processors: z.B. Image komprimieren, Upload machen, Image registrieren
- Post-P: Alicould/Amazon/DigitalOcean/GCE,/Docker Import, Vagrant, vSphere, Shell (local), Compress, Checksum, Docker Push/Save/Tag
- Variablen werden auch im JSON definiert

Packager builder

Der Builder startet die VM und baut daraus ein Image. Braucht immer ein Image oder ISO zum booten. Entweder ein Installationsimage oder ein Base Image, das dann modifiziert wird.

- Läd ISO herunter
- Erzeugt leere VM und bootet von CD
- Autom. installation mit preseed/kickstart
- Preseed/kickstart kommt von packers eingebautem http server
- Packer wartet auf ssh connection
- OS Installer läuft und rebootet
- Packer connected via ssh und führt Provisioners aus
- Packer macht shutdown der VM und führt post-processors aus

Packer Provisioner

```
"provisioners": [{
  "type": "shell",
  "inline": [
    "sleep 30",
    "sudo apt-get update",
    "sudo apt-get install -y redis-server"
  ]
}]
```

Packer

```
{
  "builders":
  [
    {
      "type": "qemu",
      "iso_url": "http://abc.de/centos/6/CentOS-6.9-x86_64-minimal.iso",
      "iso_checksum": "af4a1640c0c6f348c6c41f1ea9e192a2",
      "iso_checksum_type": "md5",
      "output_directory": "output_centos_tdhtest",
      "shutdown_command": "echo 'packer' | sudo -S shutdown -P now",
      "disk_size": 5000,
      "format": "qcow2",
      "accelerator": "kvm",
      "http_directory": "path/to/httpdir",
      "ssh_username": "root",
      "ssh_password": "s0m3password",
      "ssh_timeout": "20m",
      "vm_name": "tdhtest",
      "net_device": "virtio-net",
      "disk_interface": "virtio",
      "boot_wait": "10s",
      "boot_command": [
        "<tab> text ks=http://{{ .HTTPIP }}:/centos6-ks.cfg<enter><wait>"
      ]
    }
```

Packer

```
"builders" : [
    {
        "type" : "amazon-ebs",
        "profile" : "default",
        "region" : "{{user `region`}}",
        "instance_type" : "t2.micro",
        "source_ami" : "ami-1853ac65",
        "ssh_username" : "ec2-user",
        "ami_name" : "docker-17.12.1-ce",
        "ami_description" : "Amazon Linux Image with Docker-CE",
        "run_tags" : {
            "Name" : "packer-builder-docker",
            "Tool" : "Packer",
            "Author" : "mlabouardy"
        }
    }
],
"provisioners" : [
    {
        "type" : "shell",
        "script" : "./setup.sh"
    }
]
```

Packer

```
"provisioners": [
  {
    "type": "shell",
    "inline": [
      "mkdir .ssh",
      "echo '{{user `public_key`}}' >> .ssh/authorized_keys"
    ]
  },
  {
    "type": "shell",
    "execute_command": "echo '{{user `ssh_pass`}}' | {{ .Vars }} sudo -E -S sh
    "inline": [
      "add-apt-repository ppa:rquillo/ansible",
      "apt-get update",
      "apt-get install -y ansible",
      "echo '%sudo    ALL=(ALL)  NOPASSWD:ALL' >> /etc/sudoers"
    ]
  },
  {
    "type": "ansible-local",
    "playbook_file": "site.yml"
  }
]
```

Packer

```
{
  "builders": [
    {
      "type": "amazon-ebs",
      "access_key": "{{user `aws_access_key`}}",
      "secret_key": "{{user `aws_secret_key`}}",
      "region": "us-east-1",
      "source_ami_filter": {
        "filters": {
          "virtualization-type": "hvm",
          "name": "ubuntu/images/*ubuntu-xenial-16.04-amd64-server-*",
          "root-device-type": "ebs"
        },
        "owners": ["099720109477"],
        "most_recent": true
      },
      "instance_type": "t2.micro",
      "ssh_username": "ubuntu",
      "ami_name": "packer-vault-example {{timestamp}}"
    }
  ],
  "provisioners": [
    {
      "type": "shell",
      "inline": [
        "git clone --branch master https://github.com/hashicorp/terraform-aws-vault.git /tmp/terraform",
        "/tmp/terraform/modules/install-vault/install-vault --version 0.10.4"
      ],
      "pause_before": "30s"
    }
  ]
}
```

Packer

- Nachteil/Vorteil: Image bauen passiert in der Ziel VM
- Overhead
- Nutzung des OS Installer ist nicht so flexibel
- Vorhandenes Config Management kann vielleicht genutzt werden
- Viele Builder, Provisioner, Post-Processors vorhanden
- Dokumentation gut
- Plugins sind in Go zu schreiben

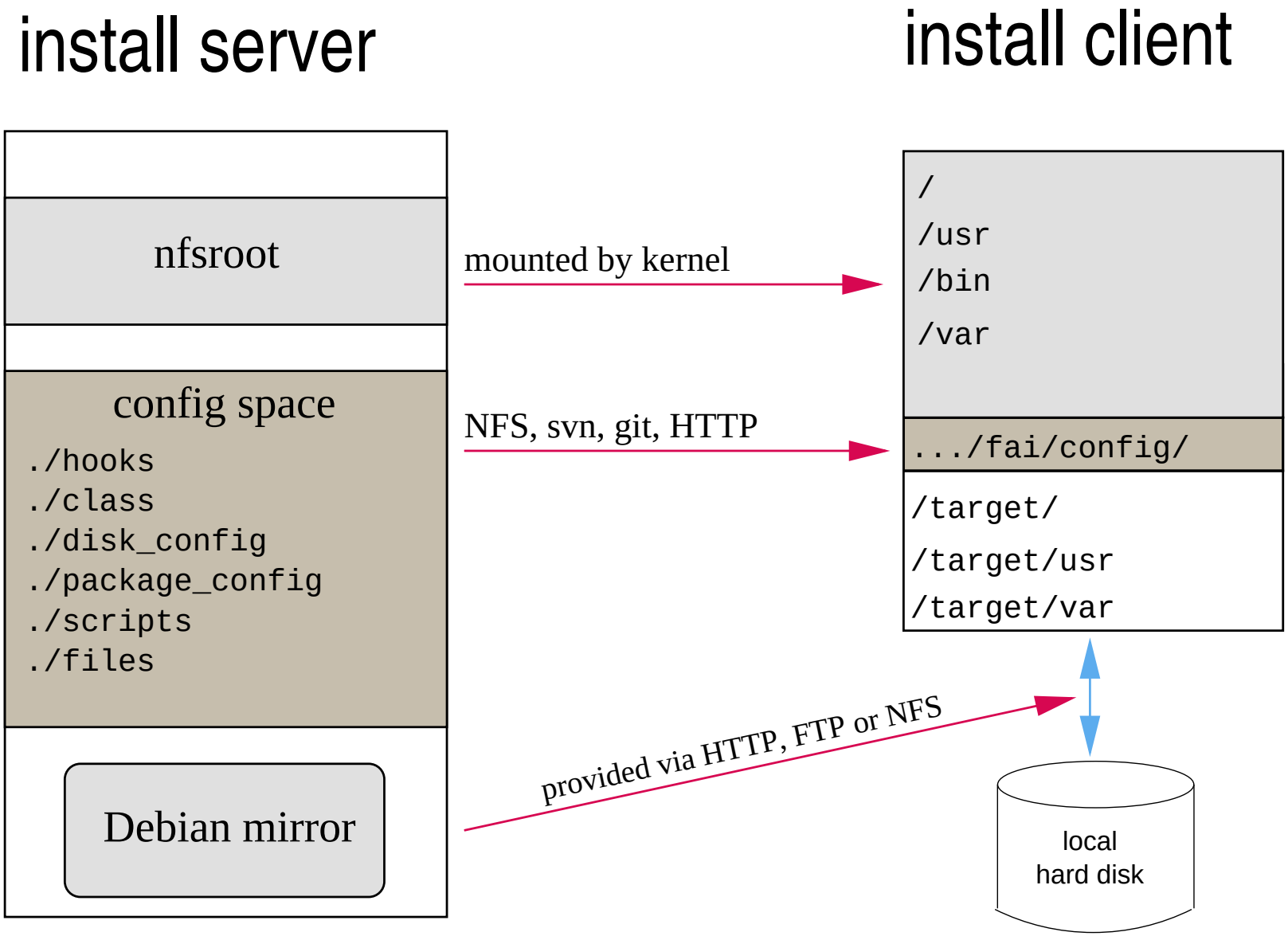
Packer

- `https://github.com/chef/bento`
- Packer templates for building minimal Vagrant baseboxes
- Viele OSes: amazonlinux, centos, freebsd, ubuntu, debian, macos, opensuse, sles, windows, fedora, hardenedbsd, oraclelinux, rhel, solaris
- Provisioner in shell gemacht
- Problem: 6 JSON Varianten f. Ubuntu
- Ca. 310 Zeilen, aber nur 6 Zeilen diff

FAI.me im Hintergrund

- Webserver $\neq$ Buildserver
- Perl CGI das die Eingaben validiert
- Je Auftrag ein Unterverzeichnis
- Schreibt eine config und eine meta Datei
- Status im Web sichtbar (waiting, processing, done, error)
- Shellskript auf dem Build Server verarbeitet neue Jobs
- Neue nfsroot, monitoring, cleanup alter Jobs

FAI Architektur



FAI Architektur

- FAI Config Space
- FAI Klassen
- `AMD64 DEBIAN STRETCH HOME_LVM BACKPORTS SSH_SERVER NONFREE GNOME FAIME`
- Installations ISO
  `fai-mirror -cDEBIAN,GNOME,BACKPORTS,FAIME /tmp/mirror`
  `fai-cd -m /tmp/mirror ABC.iso`
- Cloud Image
  `fai-diskimage -S5G -cDEBIAN,GNOME,FAIME AB12.raw.zst`

Plattenpartitionierung

Example: `.../disk_config/HOME_LVM`:

```
# entire disk with LVM, separate /home
disk_config disk1 fstabkey:uuid align-at:1M

primary /boot   200     ext2    rw,noatime
primary -       4G-     -       -

disk_config lvm
vg vg1  disk1.2
vg1-root     /        3G-50G   ext4    noatime,rw
vg1-swap     swap     200-4G   swap    sw
vg1-home     /home    600-     ext4    noatime,nosuid,nodev,rw
```

- File systems: ext[2,3,4], vfat, xfs, ReiserFS, NTFS, btrfs

Softwareinstallation

Beispiel: .../package_config/DEBIAN:

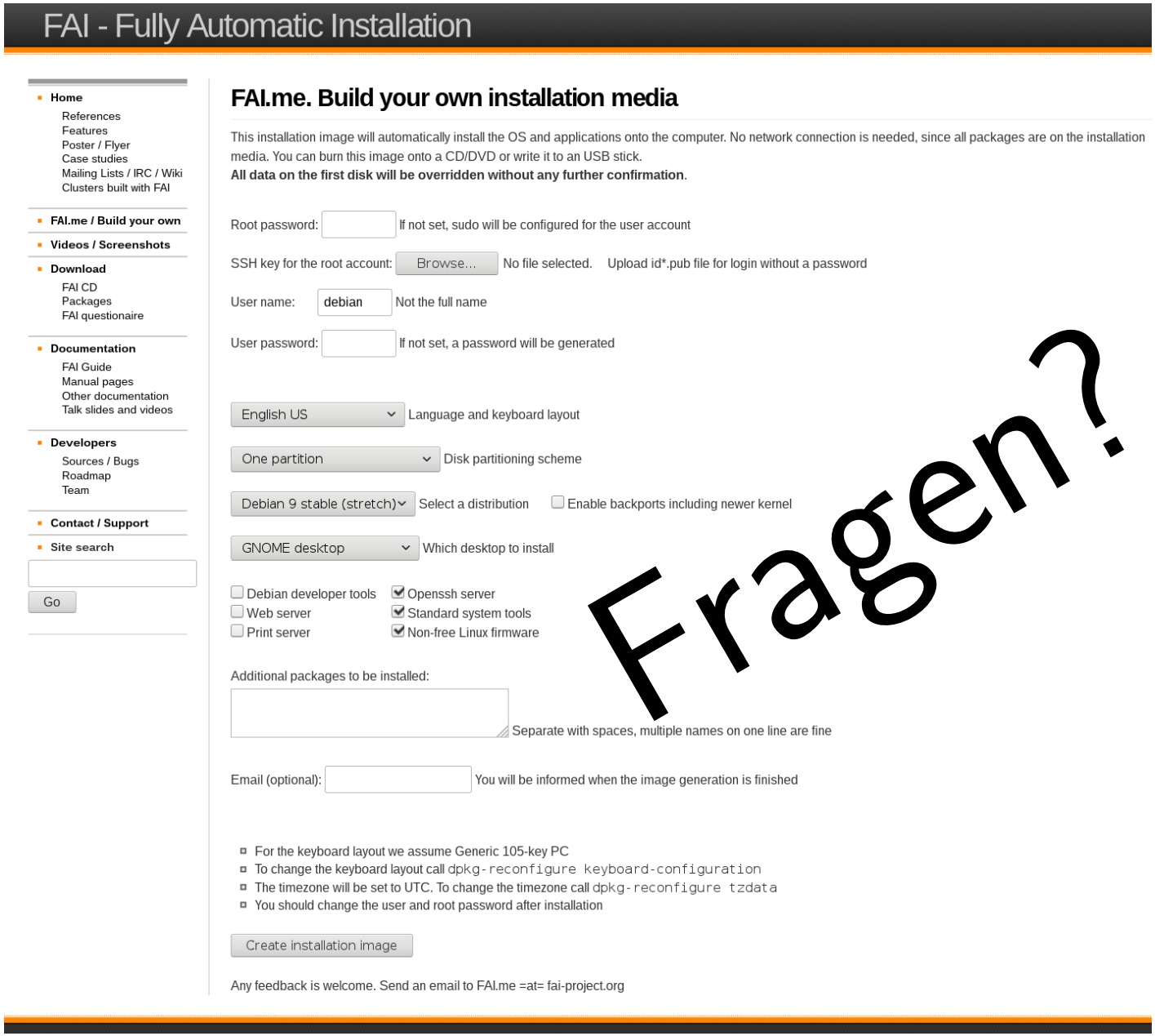
```
PACKAGES install-norec
apt-transport-https # is only needed for stretch
sudo debconf-utils
file less linuxlogo rsync openssh-client
time procinfo
nullmailer
console-setup kbd pciutils usbutils
unattended-upgrades

PACKAGES install NONFREE
firmware-bnx2 firmware-bnx2x firmware-realtek
firmware-linux-nonfree

PACKAGES install AMD64
linux-image-amd64
memtest86+
```

FAI Referenzen

FAI Nutzer

- Anonymous, financial industry, 32.000 hosts
- LVM insurance, 10.000 hosts
- City of Munich, 16.000 hosts
- Albert Einstein Institute, 1725 hosts
- Mobile.de, ∼600 hosts
- StayFriends, 700+ hosts
- XING AG, 300-400 hosts
- Opera Software, ∼300 hosts
- Stanford University, 450 hosts
- MIT Computer science research lab, 200 hosts
- The Welcome Trust Sanger Institute, 540 hosts
- Deutsches Elektronen-Synchrotron, 273 hosts
- Archive.org, 200+ hosts
- Electricité de France (EDF), 1500 hosts
- BUF, digital visual effects company, 1000 hosts
- Zivit, 260 hosts on two IBM z10 EC mainframes
- ETH Zurich, systems group, ∼300 hosts
- Grml, creating eight different ISOs, daily builds

https://fai-project.org/FAIme



Fragen?