ECS Administration Manual

Release 2021-03-28@13:50

ECS Development Team

Mar 28, 2021

Contents

The ecs appliance is a selfservice production setup virtual machine builder and executor. it can be stacked on top of the developer vm, but is independent of it.

1 Requirements

The ecs software is designed to run as an external available internet web-service, reachable by everyone.

1.1 Hosting Requirements

- a Virtual Machine on a supported Hypervisor or Hardware sized according to the *Assessment* Chapter
- a Backup space (for encrypted backup data) accessable by one of 25 supported storage protocols explained under *Backup Storage*
- DNS A Record pointing to a fixed public IPV4 address,a MX Record and a reverse PTR Record as described under *DNS Setup*
- incoming tcp ports 22,25,80,443,465 of a fixed public IPV4 address and IPV4 and internet connectivity for the vm as described under *Internet Connectivity*
- Working HTTPS including HTTP-Client Certificate support. See common challenges described under *Security Products*

Hypervisor

- The base of the virtual machine is a Ubuntu Xenial (16.04 LTS) 64Bit Standard Cloud Image.
- The appliance was tested on the following hypervisors:
  - KVM, XEN, VMware Sphere, HyperV
- Rollouts to Amazon EC2, Google GCE or Openstack Clouds are not tested but meta data gathering from ec2, gce or openstack compatible meta data services is implemented but probably need some tweaking.
- Follow the Assessment Chapter for the right sizing of CPU-cores, memory and harddisk.

Backup Storage

- For storing backup data the appliance needs a storage space accessable via one of the 24 duplicity supported storage protocols.
- In addition to the supported duplicity protocols the appliance has support for **cifs** (windows file sharing attached volumes, including automatic mount and unmount)
- Tested protocols so far: localfile, ftp, ftpssl, ssh/scp, ssh/sftp, http/webdav, cifs
- For Details see: Duplicity Manual - Section URL-Format
- Storage at Rest: All backup data is encrypted before it leaves the machine using gpg (GnuPG) and is saved without any prior decryption on the target space.

- The storage space may be hosted internal or external, using the same provider as the machine hosting or using a different provider. The hosting provider may be a trusted or untrusted third-party.
- Rotation and Retention is automatic, the backup process is unattended.
- See *Assessment* for the correct storage space size.

DNS Setup

- For Web-Access a dns [sub]domain with a A Entry pointing to the public IPv4 Address is needed
- For Email Sending and Receiving the following entries are required:
  - a MX Record pointing to the choosen domain name
  - a reverse PTR Record pointing to the domain name
  - a spf TXT Entry, eg. "v=spf1 a mx ptr ~all"
  - a dkim TXT Entry, see the env.yml configuration file for detailed data

Once you generated a config env.yml, at the top of the file you will find the corresponding DNS Entries (including DKIM TXT key) for pasting into the DNS Server

Warning: For Email functionality the MX Record, reverse PTR Record, SPF TXT Record and the DKIM TXT Record are important to setup. Eg. failing to setup reverse PTR will result in broken email sending, because the target mailserver will think the emails are spam and will bounce messages.

Examples for domains "https://whatever.me" and "https://sub.whatever.me":

```
# Main Domain Example for whatever.me
@  IN A   1.2.3.4
@  IN MX  10 whatever.me
@  IN TXT "v=spf1 a mx ptr ~all"
default._domainkey  IN  TXT  "v=DKIM1; k=rsa; s=email; p=a-long-glibberish-key"
4.3.2.1.in-addr.arpa. IN PTR whatever.me

# Sub domain Example for sub.whatever.me
sub IN A  5.6.7.8
sub IN MX 10 sub.whatever.me
sub IN TXT "v=spf1 a mx ptr ~all"
default._domainkey.sub  IN  TXT  "v=DKIM1; k=rsa; s=email; p=a-long-glibberish-key"
8.7.6.5.in-addr.arpa.  IN PTR sub.whatever.me
```

Internet Connectivity

- permanent internet connectivity with a big (>=100Mbit) upload channel
- an IPv4 address and dns server settings need to be served to the machine by DHCP for automatic configuration
  - this can be **any internal or the public IPv4 Address** if the hosting location needs this, but it must be served by DHCP
  - **Unusable** internal nets are **172.17.X.X and 10.0.3.X**, because these are used by the appliance itself
- a public IPv4-Address or the incoming TCP ports **22,25,80,443,465 and ICMP/ping** of this address forwarded to the machine
  - port 22 - ssh: ssh publickey authentification is used. this connection is used for installation and optional support
  - port 25,465 - smtp: the incoming email server of the appliance will receive emails from answers of forwarded ecs communication. if the hosting locations policy does not permit this, the ecs will work without it, but loses email answering functionality

- port 80 - http: the appliance communicates only over https, but needs http for letsencrypt client certificate renewal and for redirecting to the https site
- port 443 - https: the appliance uses letsencrypt to issue a host certificate and lets internal user administer https client certificates in self-service
    * ICMP/ping - ping: for monitoring the system

Email Challenges

Issue: Emails send from the appliance to a target mailserver that does greylisting will always be delayed.

Resolution: To remove those delays, the greylisting whitelist of the target mailserver has to be extended with the domain of the ecs appliance.

Technical Background: The appliance always uses a new unique email address for each outgoing mail (beside registration) and greylisting always delays the first email from an email address.

"Firewall"/Endpointprotection/"Antivirus" Security Products:

Warning: many security products are known to disturb/break HTTPS Host Certificate and Client Certificate validation and weaken the transport protocols on the wire. See the findings of the Security Impact of HTTPS Interception Paper.

Firewall

The appliance does not need a firewall but works as long as the incoming ports listed are forwarded to the machine and outgoing traffic from the machine is permitted.

Host Certificates

The appliance uses LetsEncrypt to issue https/ssl host certificates and also takes care of the renewal of these host certificates. There is no IT-administration task involved in this process.

Client Certificates

The ecs appliance uses https client certificates for protection of elevated user rights. Https client certificates are issued by the appliance itself and can be done in selfservice as an internal ecs webfrontend user. There is no IT-Administration task involved in this process.

Warning: If the hosting location has some mandatory security product, you probably need some extra configuration in the security product to fix support for Client Certificates.

This may also apply to to the pc's of internal ecs desktops. Some desktop "antivirus" products may need similar extra configuration to have working https client certificate support. So far there have been findings of Kaspersky and McAfee products to need extra configuration.

1.2 Cores, Memory, Harddisk & Backup Space Assessment

All data size units in this document are 1024 based.

- 1 GB = 1024 MB, 1 MB = 1024 KB , 1 KB = 1024 bytes

As a guideline, if you have more memory available, add some extra memory.

Warning: If you deviate from the example values below, be sure to add the needed memory per core if you add more cores, because many of the appliance processes get scaled to the number of cores.

- Minimum CPU/Memory: 2 Cores, 4GB

Core Calculation:

- 20 Studies per Month: 2 Cores
- 40 Studies per Month: 2-4 Cores
- 100-250 Studies per Month: 4-8 Cores

Memory Calculation:

- 2GB Base + 1GB per Core + Disk-size of Database (eg. 2GB)
- eg. 2 Cores = 4 GB + 1 GB Database= 5GB
- eg. 8 Cores = 10 GB + 8 GB Database= 18GB

Storage Calculation (10 Years):

- System: 5GB + Temporary Workspace: 30GB = 35GB
- Database: (Studies-per-Month/ 12,5)GB
- Document-Storage: (120 * Studies-per-Month * 17)MB

Backup Space Calculation (10 Years):

- Data: 2 (Fullbackups) * max(Document-Storage) + max(Database) + 2 Months delta grow

Limits: The current ecs implementation can sustain 250 Studies per Month for 10 years. Larger implementations are possible with some refactoring in certain parts of the ecs.

if using additional storge for data, the root volume needs at least: 30GB (better 50GB) of space

10 Years Calculation

After 10 Years, the values noted will be reached.

100 Studies per Month

Items	Size	Calculation
Base & Temp	50GB	
Database	8GB	4074220 / 59 * 120
Document-Storage	195GB	100684328 / 59 * 120
Disksize	253GB	52428800 + 8286549 + 204781684
Backup	444GB	2 x 204781684+ 52428800+ 204781684/120*2
recommended Cores	6	
recommended Memory	16GB+	2+ 8+ 6

250 Studies per Month

Items	Size
Base & Temp	50GB
Database	20GB
Document-Storage	500GB
Disksize	570GB
Backup	1332GB
recommended Cores	8
recommended Memory	30GB+

40 Studies per Month

Items	Size
Base & Temp	50GB
Database	3,2GB
Document-Storage	78GB
Disksize	131GB
Backup	178GB
recommended Cores	4
recommended Memory	9,5GB+

20 Studies per Month

Items	Size
Base & Temp	50GB
Database	1,6GB
Document-Storage	40GB
Disksize	92GB
Backup	89GB
recommended Cores	2
recommended Memory	6GB+

Test Instance

Items	Size
Disksize	20-40GB
recommended Cores	1-2
recommended Memory	4GB+

Research data

- EC with ~100 Studies per Month
- Runtime: ~5 Years (59 month) (15.1.2012 - 15.12.2016 ~ 4 Years, 11 Months)
- Studies: 5861 in 59 Months, or ~100 (99,339) Studies per Month

Document Storage:

- Current space used: 97GB (100684328KB)
- Directories: 69K Directories
- Files: 296K Files
- Files per Directory: Peak at 4-6 Files per Directory
- average 339KB per File
- average space per study: 17178KB ~ 16,78MB

Postgres Database:

- compressed migrated production dump: 475MB
- diskspace used: ~4GB (4074220 KB, 1GB pg_xlog)

1.3 Client Desktop Requirements

- As a submitter one of the following browsers is needed:
  - Mozilla Firefox
  - Google Chrome
  - Apple Safari
  - Microsoft Edge
- Internal ECS User need:
  - Mozilla Firefox
  - Working HTTPS including HTTP-Client Certificate support. See common challenges described under *Security Products*
- Additional requirements for Signing User:
  - **Java** Version 8/9, to install go to the Download-Page of java.com
  - Installation of the Java Signingsoftware Mocca
  - see Digital signed pdf documents for technical details
  - see User-Manual (German) - Elektronische Signatur

2 Deploy Appliance

The base of the appliance is Ubuntu Xenial (16.04 LTS). Installation is done unattended using ssh to login into the machine.

For a quickstart look at github.com/ecs-org/ecs-appliance/README.md.

2.1 As a Virtual Machine

You either need:

- a standard ubuntu cloud image from Ubunu Xenial Cloud Images
  - a cloud-init cidata iso volume with your public key
    - use the prebuilt cidata iso with the vagrant user and the insecure vagrant ssh key or password
      - vagrant-publickey-growroot.iso
      - vagrant-password-growroot.iso
    - you can build your own seed iso using github.com/ecs-org/cidata-seed
- an already running empty Ubuntu Xenial and a ssh key to login
  - eg. if your rootserver hoster has a default xenial image
- a local development machine with vagrant and a hypervisor for vagrant installed.
  - vagrant will setup the base machine for you

For a virtual machine deployment it is best to stick to the default xenial cloud images layout (flat, first Partition as Root taking all space), unless there is good reason to deviate.

Start the new virtual machine and login via ssh.

2.2 On Hardware or other custom Configurations

Needed:

- an already running empty Ubuntu Xenial and a ssh key to login
  - eg. if your rootserver hoster has a default xenial image

In a typical root server hosting setup there are two harddisks per machine. These should be configured as a raid1 (mirroring) setup with lvm on top of it.

Example:

- Hardware: Hetzner px61nvme Rootserver Setup Config:

```
DRIVE1 /dev/nvme1n1
DRIVE2 /dev/nvme0n1
SWRAID 1
SWRAIDLEVEL 1
BOOTLOADER grub
HOSTNAME Ubuntu-1604-xenial-64-minimal
PART /boot ext3 512M
PART lvm vg0 all
LV vg0 root / ext4 300G
IMAGE /root/.oldroot/nfs/install/../images/Ubuntu-1604-xenial-64-minimal.tar.gz
```

2.3 Optional Network attached Storage

- The appliance supports two external network attached storage volumes, one for permanent data and one for volatile data.
- To have seperate volatile and/or data partitions, change storage:ignore:volatile and/or storage:ignore:data to false.
- The volatile volume must be labeled “ecs-volatile”, the data volume “ecs-data”.
- Setup will add mountpoints for /data and /volatile and mount them on startup.
- Use appliance:extra:states and :packages if storage setup needs additional packages installed.
- The root volume needs at least: 30GB (better 50GB) of space

2.4 Install via ssh to an empty Xenial VM

ssh into target vm:

```
export DEBIAN_FRONTEND=noninteractive
apt-get -y update && apt-get -y install curl
curl https://raw.githubusercontent.com/ecs-org/ecs-appliance/master/bootstrap-
↪appliance.sh > /tmp/bootstrap.sh
cd /tmp; chmod +x /tmp/bootstrap.sh; /tmp/bootstrap.sh --yes  2>&1 | tee /tmp/
↪setup.log
```

2.5 Testinstall using Vagrant

on your local machine:

```
git clone https://github.com/ecs-org/ecs-appliance ~/ecs-appliance
cd ~/ecs-appliance
vagrant up
```

2.6 Upgrade a developer vm

Requirement: you need at least 3gb total memory for the appliance, 4gb minimum if you want metrics active.

on a developer vm:

```
# clone from public repository
git clone https://github.com/ecs-org/ecs-appliance /app/appliance
# install saltstack and start appliance install
curl -o /tmp/bootstrap_salt.sh -L https://bootstrap.saltstack.com
sudo bash -c "mkdir -p /etc/salt; cp /app/appliance/salt/minion /etc/salt/minion; \
    chmod +x /tmp/bootstrap_salt.sh; /tmp/bootstrap_salt.sh -X"
sudo salt-call state.highstate pillar='{"appliance": {"enabled": true}}'
```

if you also want the builder (for building the appliance image) installed:

```
sudo salt-call state.highstate pillar='{"builder": {"enabled": true}, "appliance":
↪{"enabled": true}}'
```

3 Configure Appliance

The Appliance is configured using a yaml configuration file which can be placed in one of the following locations:

- local file at `/app/env.yml` (no-cloud config)
- a attached drive with label cidata (cloud-init: no-cloud)
- a attached drive with label config-2 (cloud-init: openstack)
- aws-ec2 meta-data server (amazon elastic computing cloud)
- gce meta-data server (google compute engine)

It will be copied to `/run/active-env.yml` (ramdisk) on each appliance startup.

warning: Do **not** reuse an already existing configuration for a different domain/instance, **always** create a new configuration, to not leak secrets between domains.

3.1 Create a new Configuration

on an installed but unconfigured appliance:

- enter installed but empty appliance
- make a new env.yml: `env-create.sh *domainname.domain* /app/env.yml`
- edit settings in "/app/env.yml", see comments inside file
- Optional: package env into different formats
  - `env-package.sh --requirements; env-package.sh /app/env.yml`
  - see `env-package.sh` for more options
  - transfer, print out "/app/env.yml.pdf" and store in a safe place.
- save an encrypted copy of env.yml in a safe place.
- **Important**: The **env.yml contains all needed secrets** for a working appliance and is **the only relevant piece of information** if you want to recover from backup in case of a storage failure.
- **Warning**: Only use ascii charset in env.yml, saltstack expects yaml in ascii charset

for offline environment creation, using your local machine:

- have saltstack installed (minion does not need to run)
- `git clone https://github.com/ecs-org/ecs-appliance ~/path-to-project/ecs-appliance`
- use env-create.sh and env-package.sh like explained below, but add ~/path-to-project/ecs-appliance/salt/common/ to the callpath.
- copy *~/domainname.domain*/env.yml to appliance machine at /app/env.yml

```
ssh root@target.vm.ip '/bin/bash -c "mkdir -p /app/"'
scp env.yml root@target.vm.ip:/app/env.yml
```

for a development server:

- run `cp /app/appliance/salt/pillar/default-env.sls /app/env.yml` and edit settings in "/app/env.yml".

3.2 Modify Configuration

Any applied change in the config file will reconfigure the appliance on the next appliance restart run to the new values found in the configuration.

Eg. if you want to change your backup target to a different location, just change the value and restart the appliance, it will detect and configure all needed changes to the environment.

See the comments in the configuration for different possibilities for the appliance configuration.

“gpg_secret”, “base64_secret”, “rsa_secret” are placeholder which will be generated with the corresponding data on environment creation using `env-create.sh`. see `salt/appliance/env-template.yml` for details on the creation procedure.

```
#cloud-config
# XXX keep the "#cloud-config" line first and unchanged, software expects this␣
↪header

ssh_authorized_keys:
  # # you can put your ssh keys here, this is also used by cloud-init
  # - "ssh-rsa and some long glibberish somebody@somewhere"
ssh_deprecated_keys:
  # # you can copy deprecated keys here,
  # # state.highstate will remove these keys from allowed login,
  # # additionaly this section serves as log of past access keys
  # - "ssh-rsa and some long glibberish somebody@somewhere"
disable_root: false
# disable_root set to false for cloud-init compatibility, appliance expects root␣
↪to be usable

appliance:
  # # standby: default false, if set appliance will not activate
  # standby: true
  domain: {{ domain }}
  allowed_hosts: {{ domain }}
  ssl:
    letsencrypt:
      enabled: true
    # # client_certs_mandatory, default false, if true, always need a client␣
↪certificate
    # client_certs_mandatory: true
    client_certs_mandatory: false
    # # key: if set ssl key for https host will be used, default empty
    # key: filename-key.pem
    # # cert: if set ssl key for https host will be used, default empty
    # cert: filename-cert.pem
  # # sentry:dsn set to your sentry url, may be the same as ecs:settings:SENTRY_DSN
  # sentry:
  #   dsn: 'https://url'
  # metric:
  #   exporter: false
  #   server: false
  #   gui: false
  #   pghero: false
  # git:
  #   # default see appliance.include
  #   branch: master
  #   source: git_url
  # extra:  # write out extra files on appliance configure
  #   files:
  #     - path: /path/of/filename
  #       contents: |
  #           # Your content here
```

(continues on next page)

(continued from previous page)

```
  #       owner: user:group
  #       permissions: "0600"
  #   packages: # include extra packages at state.highstate
  #     - qrcode
  #   # states: # string to be executed as a salststack sls at state.highstate
  #   # # warning: syntax and execution errors will break initial deployment and␣
↪appliance-update
  #   states: |
  #       testing:
  #         pkg.installed:
  #           - name: curl
  # # update:oncalendar: # set a different update timer than default: "*-*-*␣
↪06:30:00"
  # # update:automatic: # default to true, to disable automatic update set to false
  # # XXX: do not update in the time between 00:30 and 06:30 because backup runs␣
↪at this time
  # update:
  #   automatic: true
  #   oncalendar: Sun *-*-* 06:30:00
  storage:
    # # setup: optional, will be executed by appliance.storage.setup if volatile␣
↪or data can not be found
    # setup: |
    # proxy_cache: true
    # # default false, if true 10 additional GB diskspace are used
    # # for operating polipo, a http proxy cache
    ignore: # default false, if true: will not look for ecs-volatile or ecs-data␣
↪filesystem
      volatile: true
      data: true
  dkim:
    key: |
{{ rsa_secret()|indent(8,True) }}
  backup:
    url: file:///volatile/ecs-backup-test/
    # options: "string of options directly passed to duplicity"
    # # mount default empty, script will mount & unmount source to target on␣
↪backup run
    # mount:
    #   type: "cifs"
    #   source: "//1.2.3.4/datavolume"
    #   target: "/mnt/appliance-backup-mount"
    #   options: "user=username,pass=password"
    # # options are passed to mount via "-o"
    encrypt: |
{{ gpg_secret('ecs_backup')|indent(8,True) }}

ecs:
  # git: # default see appliance.include
  #   branch: stable
  #   source: git_url
  userswitcher:
    enabled: false
    # set userswitcher to enabled on a test instance to change to different users
  settings: |
      DOMAIN = '{{ domain }}'
      ABSOLUTE_URL_PREFIX = 'https://{}'.format(DOMAIN)
      ALLOWED_HOSTS = [DOMAIN, ]
      PDFAS_SERVICE = ABSOLUTE_URL_PREFIX+ '/pdf-as-web/'
      # use PDFAS_SERVICE = "mock:" for mock signing
      SECURE_PROXY_SSL = True
```

(continues on next page)

(continued from previous page)

```
      ECS_REQUIRE_CLIENT_CERTS = True
      DEBUG = False
      # SENTRY_DSN = 'https://url' # set to sentry url if available
      ETHICS_COMMISSION_UUID = 'ecececececececececececececececec'
      # set ETHICS_COMMISSION_UUID to the desired UUID of the target commission

      SECRET_KEY = '{{ base64_secret() }}'
      REGISTRATION_SECRET = '{{ base64_secret() }}'
      PASSWORD_RESET_SECRET = '{{ base64_secret() }}'

      EMAIL_BACKEND = 'django.core.mail.backends.console.EmailBackend'
      EMAIL_BACKEND_UNFILTERED = 'django.core.mail.backends.smtp.EmailBackend'
      EMAIL_UNFILTERED_DOMAINS = () # = ('example.com',)
      EMAIL_UNFILTERED_INDIVIDUALS = () # = ('ada@example.org', 'tom@example.com')

      SMTPD_CONFIG['listen_addr'] = ('0.0.0.0', 8025)
      SMTPD_CONFIG['domain'] = DOMAIN

      # EMAIL_BACKEND_UNFILTERED will be used for
      #  User registration & invitation, password reset, send client certificate,
      #  and all mails to domains in EMAIL_UNFILTERED_DOMAINS and user
      #  listed in EMAIL_UNFILTERED_INDIVIDUALS will be sent via
      #  EMAIL_BACKEND_UNFILTERED. All other mail will be sent via EMAIL_BACKEND.
      #
      # Backend to use to NOT sent email but log email to console:
      #   django.core.mail.backends.console.EmailBackend
      #
      # Backend to use to send via EMAIL_* smtp settings:
      #   django.core.mail.backends.smtp.EmailBackend
      #

  vault_encrypt: |
{{ gpg_secret('ecs_mediaserver')|indent(6,True) }}

  vault_sign: |
{{ gpg_secret('ecs_authority')|indent(6,True) }}
```

3.3 Selecting the Ethics Commission UUID

```
| 23d805c6b5f14d8b9196a12005fd2961 | Ethikkommission der Medizinischen Universität␣
↪Wien
| 7b51f38bde8a4161a0dc34647fc7e654 | Ethikkommission Krankenhaus Barmh.Brüder -␣
↪Wien
| 85dc386061584fbe8549ce4e4d828fbd | Ethikkommission der Stadt Wien gemäß KAG, AMG␣
↪und MPG
| d6a22c635a584521b107481ac18318f6 | Ethikkommission Forschungsinstitut des Wiener␣
↪Roten Kreuzes
| 55ae93ec9df04d6abfc8d233ec5ccf8e | Nicht verwenden: Ethikkommission Krankenhaus␣
↪Barmh.Schwestern - Wien
| 7cd6d52120b3474ba502931b9f60a5f3 | Ethikkommission Confraternität-Priv.Klin.␣
↪Josefstadt und Priv.Klin. Döbling
| 7df9ebaf15434709b09c3def9a6c8769 | Ethikkommission St.Anna Kinderspital
| f122f144616541d391fde2dcc761aff4 | Ethikkommission Österreichische␣
↪Arbeitsgemeinschaft für Klinische Pharmakologie
| 25b6744780434a3f96a1e43b405d3848 | Ethikkommission Privatkrankenanstalt␣
↪Rudolfinerhaus
| d542994ced34403db841786a1c1ab892 | Ethikkommission Rheuma-Zentrum Wien-Oberlaa
| 5615dfbaf8c8445d960d1e2cd9c00dc3 | Nicht verwenden: Ethikkommission Krankenhaus␣
↪des Göttlichen Heilandes
```

(continues on next page)

(continued from previous page)

```
| 4d3a2d5f138940f293ee87fe6ec1d5b2 | Ethikkommission Evangelisches Krankenhaus
| 8d2950e3a0294f68bde647a54df6d823 | Ethikkommission der Allgemeinen␣
↪Unfallversicherungsanstalt
| 9f6b509e716e413f865d95bdd630e9bc | Ethikkommission Immunologische Tagesklinik
| b17f32f604fa4452b5ff3a2baa9e0704 | Ethikkommission des Landes Niederösterreich
| 6688ce16a3b84d42b1531389e6039891 | Ethikkommission Krankenhaus Elisabethinen
| e4dcd05a31ad475ca72dea7b84ef030e | Ethikkommission der Medizinischen Fakultät␣
↪der JKU
| e269491bb9c040aaad6a5f11df343f38 | Ethikkommission Krankenhaus Barmh.Brüder -␣
↪Linz
| 1cca34032077445d95dabf7802fade28 | Ethikkommission Krankenhaus Barmh.Schwestern -
↪ Linz
| 39cbb589ef044d27bceb6ee5ac796ae7 | Ethikkommission für das Bundesland Salzburg
| 280414583b894c809a9baa8134d7fe4b | Ethikkommission der Medizinischen Universität␣
↪Innsbruck
| 183881da8200493aa7edd8bebeea75b9 | Ethikkommission des Landes Vorarlberg
| 95821eba88f34b2195f96e747d7f6b16 | Ethikkommission des Landes Burgenland gemäß␣
↪KAG, AMG und MPG
| 6e7cfab5f8cd40df83c9de4fac9bb20f | Ethikkommission des Landes Steiermark gemäß␣
↪AMG und MPG
| 75b5a9714f354a5b842aa01029148036 | Ethikkommission Krankenhaus Barmh.Brüder -␣
↪Graz
| 9287fb0878b94f9f9d90b5582d1cefca | Ethikkommission Krankenhaus Barmh.Brüder -␣
↪Eggenberg
| e17ee744e4e840778f316cf3f79de6b4 | Ethikkommission der Medizinischen Universität␣
↪Graz
| ec9f179a52a04a7aa52446232c3fd4bd | Ethikkommission des Landes Kärnten
| c890205dcb7543c8a76bf324512c5f81 | Nicht verwenden: Ethikkommission des␣
↪Krankenhaus St. Josef
| dc1b115d9809461ba3ea9450b079ddd6 | Kommission für Scientific Integrity und Ethik␣
↪der Karl Landsteiner Privatuniversität
| 50dba0126a0746dc8802e6c0e0199dad | Ethik-Kommission der Vinzenz Gruppe Wien
```

3.4 Automatic Update Setup

Updates are scheduled depending appliance:update:oncalendar once per day at 06:30 per default.

Depending the types of updates available the update will take between 1 and 5 minutes in most cases, 5-10 minutes if the ecs container will be rebuild and up to 30 minutes if there are database migrations to be executed.

The following items are updated:

- appliance source will be updated and executed
- all system packages are updated, special precautions are taken for docker, postgresql and the kernel including a reboot if needed
- lets encrypt certificates are updated
- ecs source will be updated and rebuild
  - the ecs-docs source will be updated
  - the corresponding support container will be updated
  - database migrations will be executed (including a dump before doing so)

Warning: Automatic updates are intended to run with Metric and Alert support, so you will get alerts to react and can investigate using the Metric Server to find the root cause. **If you do not make metric recording and alerting, we recommend updating only manual.** To do this, enter "False" under appliance:update:automatic in the file env.yml. For a manual update run call `systemctl start appliance-update`

3.5 Backup Configuration

- Backup is done using duplicity, see Duplicity Manual and duply
- Cycle: Backup is done once per day around 00:30
- Safety Measures for backup:
  - Database must exist
  - Storage-Vault must not be empty
- Contents:
  - /data/ecs-storage-vault: All uploaded and all created documents
  - /data/ecs-pgdump: Current database dump
- Type, Rotation & Retention:
  - Backup will start with a full backup and do incremental backups afterwards
  - 2 Months after the first full backup a second full backup will be created
  - Rotation: Every Backup (full or incremental) will be purged after 4 Months

Extra Configuration needed for scp/sftp backup (ssh)

if appliance:backup:url is using either scp or sftp (preferred):

- add ssh host entry in /root/.ssh/known_hosts for duplicity to connect to the ssh server
- create this entry by executing: `duply /root/.duply/appliance-backup status`
- add this key to env.yml as extra:state:

```
appliance:
  extra:
    states: |
        sftp_server_hostkey_into_known_hosts:
          file.append:
            - name: /root/.ssh/known_hosts
            - makedirs: true
            - text: |
                [1.2.3.4]:12345 ssh-rsa XXXHOSTKEYXXX
```

3.6 Metrics Collection Setup

- if `appliance:metric:exporter` is set, metrics are exported from the subsystems
  - export metrics of: frontend nginx, redis, memcached, uwsgi, cadvisor, process details(uwsgi, postgres, nginx, celery), prometheus node(diskstats, entropy, filefd, filesystem, hwmon, loadavg, mdadm, meminfo, netdev, netstat, stat, textfile, time, uname, vmstat)
  - additional service metrics from: appliance-backup, appliance-update
- if `appliance:metric:server` is set, these exported metrics are collected and stored by a prometheus server and alerts are issued using email to root using the prometheus alert server
  - there are alerts for: NodeRebootsTooOften, NodeFilesystemFree, NodeMemoryUsageHigh, NodeLoadHigh, a.o.
    - for a detailed alert list look at the alert.rules.yml sourcefile
    - to **add custom rules create rules** files in the config yaml using "file:*:contents" with a path of "/app/etc/prometheus-rules.d/*.rules.yml"

  - the prometheus gui is at http://172.17.0.1:9090
  - the prometheus alert gui is at http://172.17.0.1:9093
- if `appliance:metric:gui` is set, a grafana server is started to display the collected metrics
  - grafana is available at http://localhost:3000
- if `appliance:metric:pghero` is set, start a pghero instance for postgres inspection
  - pghero is avaiable at http://localhost:5081

Use ssh port forwarding to access these ports, eg. for 172.17.0.1:9090 use `ssh root@hostname -L 9090:172.17.0.1:9090`

3.7 Alerting Setup

if `ecs:settings["SENTRY_DSN"]` and `appliance:sentry:dsn` are defined, the appliance will report the following items to sentry:

- python exceptions in web, worker, beat, smtpd
- salt-call exceptions and state returns with error states
- systemd service exceptions where appliance-failed or service-failed is triggered
- shell calls to appliance.include: appliance_failed, appliance_exit, sentry_entry
- internal mails to root, eg. prometheus alerts

3.8 First Activation of Instance

on the target vm:

```
# create a empty ecs database
sudo -u postgres createdb ecs -T template0  -l de_DE.utf8

# activate env
chmod 0600 /app/env.yml
cp /app/env.yml /run/active-env.yml

# apply new environment settings and start service
systemctl restart appliance
```

While restarting, the appliance configures itself to the new environment. See the progress of the preperation by browsing to https://*domainname.domain* .

3.9 Create first internal User

After the appliance is ready and shows the login screen, login via ssh to create the first internal office user, with a corresponding client certificate and disable test user.

```
# create first internal office user (f=female, m=male)
create-internal-user.sh useremail@domain.name "First Name" "Second Name" "f"

# create and send matching client certificate
create-client-certificate.sh useremail@domain.name cert_name [daysvalid(default=7)]
# Communicate certificate transport password over a secure channel

# disable test user
cat << EOF | docker exec -i ecs_ecs.web_1 /start run ./manage.py shell
from django.contrib.auth.models import User
```

(continues on next page)

(continued from previous page)

```
User.objects.filter(profile__is_testuser=True).update(is_active=False)
EOF
```

3.10 configure ECS Settings as internal Office User

After completing the First User Setup, import the created Client Certificate into a browser and login as Internal User and continue to configure the ECS as described in the User-Manual, Chapter Commissioning

4 Maintenance

4.1 Reconfigure a running Appliance

- edit /app/env.yml
- optional, build new env package:
  - first time requisites install, call `env-package.sh --requirements`
  - build new env package call `env-package.sh /app/env.yml`
- activate changes into current environment, call `env-update.sh`
- restart and apply new environment: `systemctl restart appliance`

4.2 Start, Stop, Update Appliance

- Start appliance: `systemctl start appliance`
- Stop appliance: `systemctl stop appliance`
- Update Appliance (appliance and ecs): `systemctl start appliance-update`

4.3 Recover from failed state

if the appliance.service enters fail state, it creates a file named "/run/appliance_failed".

After resolving the issue, remove this file using `rm /run/appliance_failed` before restarting the service using `systemctl restart appliance`.

If the issue was within the ecs-appliance sourcecode, re-run an appliance update:

```
rm /run/appliance-failed
touch /app/etc/flags/force.update.appliance
systemctl restart appliance-update
```

4.4 Reviewing the Logs

Container:

- all container log to stdout and stderr
- docker has the logs of every container available
  - look at a log stream using eg. `docker logs ecs_ecs.web_1`
- journald will get the container logs via the appliance.service which calls docker-compose
  - this includes backend nginx, uwsgi, beat, worker, smtpd, redis, memcached, pdfas, mocca

  - to follow use `journalctl -u appliance -f`

Host:

- all logging (except postgres) is going through journald
- follow whole journal: `journalctl -f`
- only follow service, eg. prepare-appliance: `journalctl -u prepare-appliance -f`
- follow frontend nginx: `journalctl -u nginx -f`
- search for salt-call output: `journalctl $(which salt-call)`

4.5 Desaster Recovery from backup

- install a new unconfigured appliance as described in chapter install
- copy old saved env.yml to new target machine at /app/env.yml
- reboot new target machine, appliance will configure but stop because of empty database
- ssh into new target machine, execute `recover-from-backup.sh --yes-i-am-sure`

4.6 Searching for and restoring specified files from backup

- List all files in current backup

```
duply /root/.duply/appliance-backup list
```

- Restore database dump of last backup run to /root

```
# exclude backup source base directory from requested file path
# eg. requested file is "/data/ecs-pgdump/ecs.pgdump.gz"
# filename should be "ecs-pgudump/ecs.pgdump.gz"
duply /root/.duply/appliance-backup fetch ecs-pgdump/ecs.pgdump.gz /root/ecs.
↪pgdump.gz
```

4.7 Maintenance commands in a running ecs container

for most ecs commands it is not important to which instance (web,worker) you connect to, “ecs_ecs.web_1” is used as example.

- image = ecs, mocca, pdfas, memcached, redis
- ecs.startcommand = web, worker, beat, smtpd
- as root `docker exec -it ecs_image[.startcommand]_1 /path/to/command`
  - eg. `docker exec -it ecs_ecs.web_1 /bin/bash`
- shell as app user with activated environment
  - `docker exec -it ecs_ecs.web_1 /start run /bin/bash`
- manualy create a celery task:
  - `docker exec -it ecs_ecs.web_1 /start run celery --serializer=pickle -A ecs call ecs.integration.tasks.clearsessions`
- celery events console
  - `docker exec -it ecs_ecs.web_1 /start run /bin/bash -c "TERM=screen celery -A ecs events"`
- enter a django shell_plus as app user in a running container

  - `docker exec -it ecs_ecs.web_1 /start run ./manage.py shell_plus`

- generate all workflow graphs

```
docker exec -it ecs_ecs.web_1 /start run /bin/bash
./manage.py workflow_dot core.submission | dot -Tpng -osubmission.png
./manage.py workflow_dot notifications.notification | dot -Tpng -onotification.png
./manage.py workflow_dot votes.vote | dot -Tpng -ovote.png
```

- generate ECX-Format Documentation

```
docker exec -it ecs_ecs.web_1 /start run /bin/bash
./manage.py ecx_format -t html -o ecx-format.html
./manage.py ecx_format -t pdf -o ecx-format.pdf
```

4.8 Maintenance commands for the appliance host

All snippets expect root.

- destroy and recreate database:

```
gosu app dropdb ecs
gosu postgres createdb ecs -T template0 -l de_DE.utf8
rm /app/etc/tags/last_running_ecs
systemctl restart appliance
```

- get latest dump from backup to /root/ecs.pgdump.gz:
  - `duply /root/.duply/appliance-backup fetch ecs-pgdump/ecs.pgdump.gz /root/ecs.pgdump.gz`
- quick update appliance code:
  - `cd /app/appliance; gosu app git pull; salt-call state.highstate pillar='{"appliance":{"enabled":true}}' 2>&1; rm /var/www/html/503.html`
- check which system packages are available for update:
  - `/usr/lib/update-notifier/apt-check -p`
- cleanup last activity stamps for unattended upgrades, so unattended-upgrades will do all activity again
  - `touch /app/etc/flags/force.update.system` before `systemctl start appliance-update`
- list active systemd timer: `systemctl list-timers --all`
- display systemd service change: `journalctl -m _PID=1 -f`
- manual run letsencrypt client (do not call as root): `gosu app dehydrated --help`
- display revoked certificates serials:
  - `openssl crl -inform PEM -text -noout -in /app/etc/crl.pem`
- get cummulative cpu,mem,net,disk statistics of container:
  - `docker stats $(docker ps|grep -v "NAMES"|awk '{ print $NF }'|tr "\n" " ")`
- read details of a container in yaml:
  - `docker inspect 1b17069fe3ba | python -c 'import sys, yaml, json; yaml.safe_dump(json.load(sys.stdin), sys.stdout, default_flow_style=False)' | less`
- activate /run/active-env.yml in current shell of appliance vm:

  - `. /usr/local/share/appliance/env.include; ENV_YML=/run/active-env.yml userdata_to_env ecs,appliance`
  - to also set *GIT_SOURCE defaults: `. /usr/local/share/appliance/appliance.include`

- send sentry test entry:
  - using bash and ravencat.py

```
. /usr/local/share/appliance/env.include; . /usr/local/share/appliance/appliance.
↪include; ENV_YML=/run/active-env.yml userdata_to_env ecs,appliance; sentry_entry
↪"test" "Test Message $(hostname -f)"
```

```
+ using a django management command
```

```
docker exec -it ecs_ecs.web_1 /start run /bin/bash
./manage.py raven test
```

- most time spent in state.highstate:
  - `journalctl -u appliance-update | grep -B 5 -E "Duration:  [0-9]{3,5}\."`
  - `journalctl -u appliance-update | grep "ID:" -A6 | grep -E "(ID:|Function:|Duration:)" | sed -r "s/.*(ID:|Function:|Duration)(.*)/\1 \2/g" | paste -s -d ' \n' - | sed -r "s/ID: +([^ ]+) Function:  +([^ ]+) Duration :  ([^ ]+ ms)/\3 \2 \1/g" |sort -n`
- check send emails from postfix
  - `for a in sent deferred bounced; do echo "#### $a"; journalctl -u postfix | grep "status=$a" | awk '{print $7}' | sed 's/to=<//g' | sed 's/>,//g' | sort -n; done`
- check for incoming or outgoing smtp from ecs
  - `journalctl -u appliance --since "2019-07-27" | grep -Ei "ecs.(worker_1|smtpd_1).+(Accepted email|Rejected email|Forward|Forwarding|Not forwarding|email raised exception|Invalid message format|Relay access denied)" | sed -r "s/([^]+ [^]+ [0-9:]+).*ecs.communication.tasks.forward_messages\[[0-9a-f-]+\]:(.*)/\1\2/g" | sed -r "s/([^]+ [^]+ [0-9:]+).+ecs.smtpd_1.+INFO (.*)/\1\2/g"`
- ip adress config
  - `ip -o addr show | grep -Ev "veth[0-9a-f]{7}"; default_iface=$(awk '$2 == 00000000 { print $1 }' /proc/net/route); default_ip=$(ip addr show dev "$default_iface" | awk '$1 == "inet" { sub("/.*", "", $2); print $2 }'); echo "Default Interface:  $default_iface , Default IP: $default_ip`

5 Digital signed pdf documents

The ECS uses Mocca and PDF-AS to create digital signed pdf documents.

Mocca and PDF-AS are developed by EGIZ, the eGovernment Innovation Centre of Austria, a joint initiative of the Federal Chancellery and the Graz University of Technology, at the Institute for Applied Information Processing and Communications Technology (IAIK).

- EGIZ Mocca - a modular, open source citizen card environment
- EGIZ PDF-AS - a java framework for creating digitial signatures on PDF documents

5.1 Mocca

- Current Frozen (onlineBKU) Release: Mocca 1.3.27 (2017-06-23)
- Next Version (localBKU) Release: Mocca 1.4.1 (2019-05-07)
- Mocca WebStart from buergerkarte.at
  - Install
  - CA Install from local running BKU
  - Test
- Security Analysis of the Austrian Citizen Card Environment MOCCA and E-Card
- MOCCA Usability Analyse

Download

- Online BKU: Mocca 1.3.27 - bkuonline-1.3.27.war
- Local BKU: Mocca 1.4.1 - Release Directory

```
sha256=c3a7270294bc3a43790061e9eec059ef6a124eb3bd55d20edfd8a7b6b6b89b23  bkuonline-
↪1.3.27.war
```

- Sourcecode:
  - http://git.egiz.gv.at/mocca

Changes

- MOCCA 1.4.1 - 2019-05-07
  - Update TrustStore and CertStore (The following Certificates were added to MOCCA):
    - D-TRUST Root Class 3 CA 2 EV 2009
    - D-TRUST CA 2-2EV 2016
    - Thwate RSA CA 2018
    - RapidSSL RSA CA 2018
  - The BKU icon was updated
- MOCCA 1.4.0 - 2018-09-25
  - Update of PKI-Module and the CertSore to the latest version
  - Changes to build standalone-installer for Java Version 11 and later
    - Standalone installer for Windows (MSI), MacOS (DMG) and Linux (ZIP) are now available.

  - New MOCCA update mechanism:
    * MOCCA verifies now during startup if there is a newer version available and notifies the user if this is the case.
  - Update of security-relevant libraries
- MOCCA 1.3.34 - 2018-08-30
  - New Certificates in the truststore:
    * DSTRootCAX3
    * DigiCertGlobalRootG2
  - New Certificates in the certstore:
    * GeoTrustRSACA2018
    * ThwateTLSRSACAG1
- MOCCA 1.3.33 - 2018-05-15
  - Bulk Signature Branch merged into Master Branch
  - Small Bug Fix: Error code 2000 when using Java 8
- MOCCA 1.3.32 - 2018-04-20
  - BKUCertificates updated
    * New certificated added to certstore for secure.gesundheit.gv.at
    * Letsencrypt root ca added to truststore
- MOCCA 1.3.31 - 2018-02-07
  - Small bug fix in the Request GetInfoBox
  - Support for Java 9
- MOCCA 1.3.30 - 2018-01-15
  - Certificate update
- MOCCA 1.3.29 - 2017-08-28
  - Library Update
  - Included PAdES signature
  - Certificate update
- MOCCA 1.3.28 - 2017-07-26
  - Library Update
  - Removed bkuonline support
- MOCCA 1.3.27 - 2017-06-23
  - Last bkuonline support
  - Security Bugfix
- MOCCA 1.3.26 - 2017-05-29
  - New card implementation (A-Trust signature card CardOSV5.3)
- MOCCA 1.3.25 - 2017-05-08
  - MOCCA Library Update
  - Bugfix
- MOCCA 1.3.24 - 2016-12-01

  - Fixed problem, which occured when using MOCCA together with MOA-ID in a single webapp container.
  - System-Properties include now the prefix ".jnlp"
- MOCCA 1.3.23 - 2016-03-31
  - First ECS Version

5.2 PDF-AS

- Current Frozen (onlineBKU) Release:
  - PDF-AS 4.1.0 (2017-01-17)
  - PDF-AS defaultConfig 4.1.1 (2017-08-29)
- Next Version (localBKU) Release: PDF-AS 4.1.5 (2018-12-07)
  - Documenation - DE
  - Web Interface Documenation - DE
  - External Service Documentation - DE
  - Profile Documentation - DE
- Goverment Profile Documentation - DE
- PDF-AS Workshop Part 1 - DE
- PDF-AS Workshop Part 2 - DE
- Goverment sponsored Digital Signature Test Site - DE

Download

- Download Site
- Current 4.1.5 pdf-as-web-4.1.5.war

```
sha256=a54f3dd8c77e5d7ef47b05465143b09014dfd802240e557f39aaaf7c9fb2d518  pdf-as-
↪web-4.1.5.war
```

- Current 4.1.5 defaultConfig.zip

```
sha256= 22a73c4a45fb44f6a0a30db9a95a5cb5ef3b5ee4e14ca768e3ebb15f5597813f ␣
↪defaultConfig.zip
```

- old frozen 4.1.0 pdf-as-web-4.1.0.war

```
sha256=d625cd40a0baadf9ccf2a9bb46f6808efb7929e3fbe107b087b8c42f5b6c1971  pdf-as-
↪web-4.1.0.war
```

- old frozen 4.1.1 defaultConfig-4.1.1.zip

```
sha256=392718b4b8e57582ec529061c25437b3858574a90214e429069cbc084333ca8d ␣
↪defaultConfig-4.1.1.zip
```

- Sourcecode:
  - http://git.egiz.gv.at/pdf-as-4/

Changes

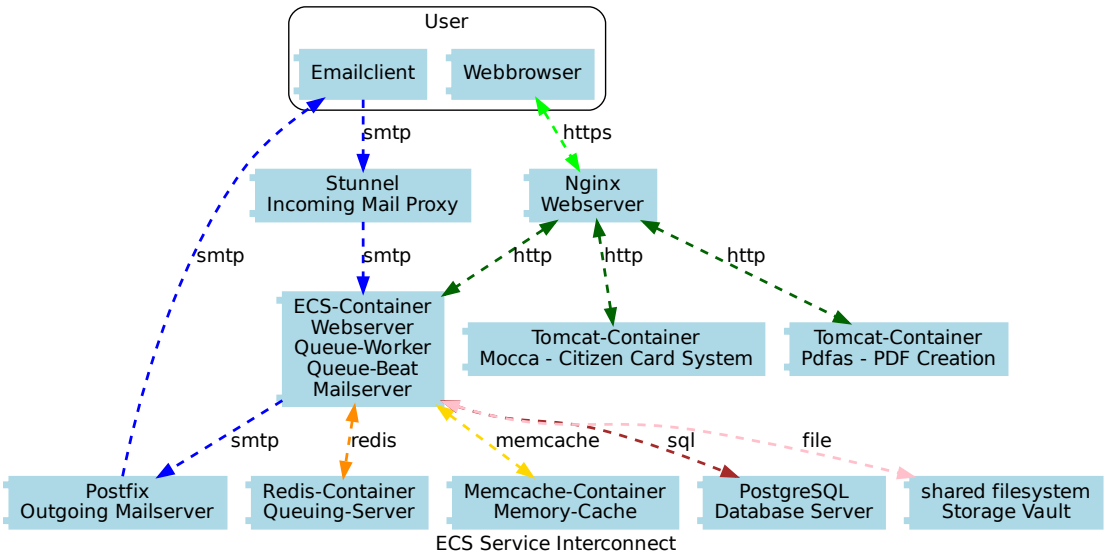
- PDF-AS 4.1.5 - 2018-12-07
  - Beta state of the Security Layer 2.0 functionality has been added
  - Bug fixes regarding positioning feature (positioning with a “footer” paramter, positioning with a “page” parameter)
- PDF-AS 4.1.4 - 2018-06-27
  - PDF-AS Library
    - Color profile for PDFA added
- PDF-AS 4.1.3 - 2018-05-14
  - Minor bug fixes
- PDF-AS 4.1.2 - 2018-03-23
  - Minor bug fixes
  - Additional feature regarding placeholder search
  - Pdf file protection
- PDF-AS 4.1.1 - 2017-08-29
  - Minor bug fixes
  - Changes for PAdES signatures
  - Library Update
  - Removed bkuonline support
- PDF-AS 4.1.0 - 2017-01-17
  - Last bkuonline support
  - PDF-AS Library
    - a new pdf backend is available based on pdfbox 2.0.3, changes that are available with the new backend:
    - unicode characters can now be rendered in the signature block when using TTF fonts
    - when using TTF fonts the new backend includes only a subset of used characters in the resulting document, which reduces the resulting size of the document
    - some minor bugs fixes
  - PDF-AS Web
    - some minor bugs fixes
- PDF-AS 4.0.11 - 2016-07-01
  - First ECS Version

5.3 Mobile Signature (aka “Handy Signatur”) from A-Trust

- Technical Documentation

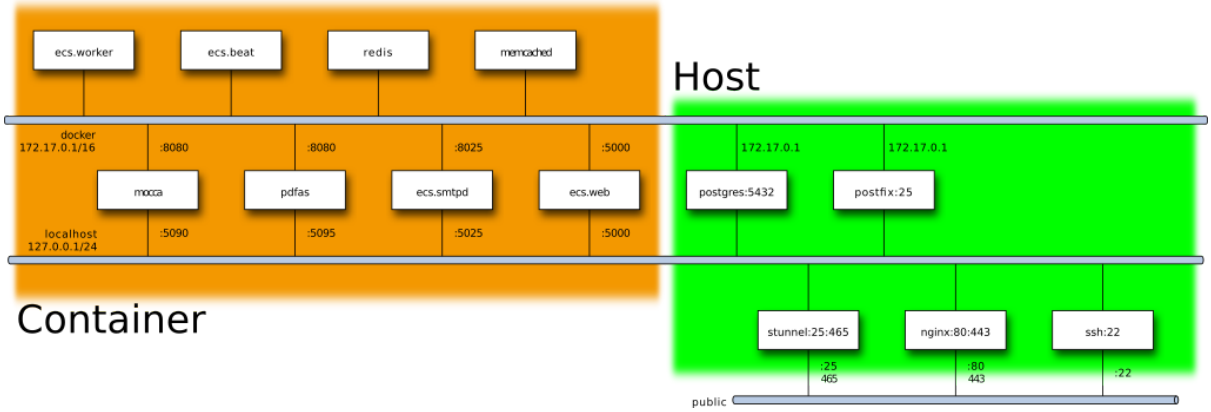
6 Development

6.1 Service Architecture



- Ubuntu Xenial (16.04 LTS) 64Bit Standard Cloud Image
- Saltstack salt is used for system installation
- Letsencrypt is used for ssl host certificates
- Services running on the host system
  - Webserver: NGINX 1.10 - nginx.conf
  - Database: Postgresql 9.5 Database postgres config
  - SMTP/SSL Proxy: Stunnel - stunnel.conf
  - SSH Daemon: openssh ssh config
  - SMTP Outgoing: postfix - main.cf
  - SMTP Incoming: ecs - smtpd.py
- Compliance to rfc3834 Recommendations for Automatic Responses to Electronic Mail
  - set and evaluate headerfield “Auto-Submitted”
  - add headerfield “In-Reply-To” and “References” to emails originated via smtp
- Services running inside docker container
  - The ECS Application (Web, Worker, incoming Mail)
    - Django is served via uwsgi inside the web container
    - Celery is used for asynchronous worker & beat

    - python smtpd is used for incoming Mail processing
  - Redis for Queuing Services
  - Memcache for Caching Services
  - tomcat running PDF/AS for Electronic Signed PDF Generation
  - tomcat running Mocca for accessing the Digital ID-Card used for signed PDF-Generation
- The Appliance uses the default docker network (172.17.0.1/16) for docker container
- Public (Outside) facing Ports
  - NGINX Webserver Ports 80(http) and 443(https)
  - Stunnel Ports 25(smtp) and 465(smtpssl)
  - SSH Daemon 22(ssh)



6.2 Repository Layout

Path	Description
/pillar/*.sls	salt environment
/pillar/top.sls	defines the root of the environment tree
/pillar/default-env.sls	fallback env yaml and example localhost ecs config
/salt/*.sls	salt states (to be executed)
/salt/top.sls	defines the root of the state tree
/salt/common/init.sls	common install
/salt/common/env-create.sh	cli for env generation
/salt/common/env-package.sh	cli for building pdf,iso,tar.gz.gpg out of env
/salt/common/env-update.sh	get env, test conversion and write to /run/active-env.yml
/salt/appliance/env-template.yml	template used to generate a new env.yml
/salt/appliance/init.sls	ecs appliance install
/salt/appliance/scripts/prepare-env.sh	script started first to read environment
/salt/appliance/scripts/prepare-appliance.sh	script started next to setup services
/salt/appliance/scripts/prepare-ecs.sh	script started next to build container
/salt/appliance/update/appliance-update.sh	script triggerd from appliance-update.service
/salt/appliance/ecs/docker-compose.yml	main container group definition
/salt/appliance/systemd/appliance.service	systemd appliance service that ties all together

6.3 Execution Order

```
[on start]
|
|-- prepare-env
|-- prepare-appliance
|   |
|   |-- optional: call salt-call state.sls appliance.storage.setup
|---|
|
|-- prepare-ecs
|-- appliance
|   |
|   |-- docker-compose up
:
:   (post-start)
|-- appliance-cleanup

[on error]
|
|-- appliance-failed

[on update]
|
|-- appliance-update
|   |
|   |-- salt-call state.highstate
|   |-- apt-daily unattended-upgrades
|   |-- letsencrypt update
?   ?-- optional reboot
|   |-- systemctl restart appliance
```

6.4 Runtime Layout

Application:

Path	Description
/app/env.yml	local (nocloud) environment configuration
/app/ecs	ecs repository used for container creation
/app/appliance	ecs-appliance repository active on host
/app/etc	runtime configuration (symlink of /data/etc)
/app/etc/tags	runtime tags
/app/etc/flags	runtime flags
/app/etc/hooks	runtime hooks
/app/ecs-ca	client certificate ca and crl directory (symlink of /data/ecs-ca)
/app/ecs-gpg	storage-vault gpg keys directory (symlink of /data/ecs-gpg)
/app/ecs-cache	temporary storage directory (symlink of /volatile/ecs-cache)
/run/active-env.yml	current activated configuration
/run/appliance-failed	flag that needs to be cleared, before a restart of a failed appliance is possible
/usr/local/share/appliance	scripts from the appliance salt source
/usr/local/[s]bin	user callable programs

Data:

Path	Description
/data	data to keep
/data/ecs-ca	symlink target of /app/ecs-ca
/data/ecs-gpg	symlink target of /app/ecs-gpg
/data/ecs-storage-vault	symlink target of /app/ecs-storage-vault
/data/etc	symlink target of /app/etc
/data/ecs-pgdump	database migration dump and backup dump diretory
/data/postgresql	referenced from moved /var/lib/postgresql

Volatile:

Path	Description
/volatile	data that can get deleted
/volatile/docker	referenced from moved /var/lib/docker
/volatile/ecs-cache	Shared Cache Directory
/volatile/ecs-backup-test	default target directory of unconfigured backup
/volatile/redis	redis container database volume

6.5 Container Volume Mapping

Host-Path	Container	Container-Path
/data/ecs-ca	ecs	/app/ecs-ca
/data/ecs-gpg	ecs	/app/ecs-gpg
/data/ecs-storage-vault	ecs	/app/ecs-storage-vault
/volatile/ecs-cache	ecs	/app/ecs-cache
/app/etc/server.cert.pem	pdfas/mocca	/app/import/server.cert.pem:ro

6.6 Environment Mapping

Types of environments:

- saltstack get the environment as pillar either from /run/active-env.yml or from a default
- shell-scripts and executed programs from these shellscripts get a flattened yaml representation in the environment (see flatyaml.py) usually restricted to ecs,appliance tree of the yaml file

Buildtime Environment:

- the build time call of `salt-call state.highstate` does not need an environment, but will use /run/active-env.yml if available

Runtime Environment:

- prepare-env
  - get a environment yaml from all local and network sources
  - writes the result to /run/active-env.yml
- appliance-update, prepare-appliance, prepare-ecs, appliance.service
  - parse /run/active-env.yml
  - include defaults from appliance.include (GIT_SOURCE*)
- Storage Setup (`salt-call state.sls appliance.storage.setup`) parses /run/active-env.yml
- appliance-update will call `salt-call state.highstate` which will use /run/active-env.yml
- appliance.service calls docker-compose up with active env from /run/active-env.yml

- docker compose passes the following to the ecs/ecs* container
  - service_urls.env, database_url.env
  - ECS_SETTINGS
- docker compose passes the following to the mocca and pdfas container
  - APPLIANCE_DOMAIN as HOSTNAME