CRIMSy Documentation

Release 0.2

Frank Broda, Fabian Mauz

Dec 03, 2021

CONTENTS:

Project resources:

- https://crimsy.org (Project Homepage)
- https://github.com/ipb-halle/CRIMSy (Repository)
- https://crimsy.org/project/ui/site/project-reports.html (Project reports)
- https://crimsy.org/handbook/CRIMSy.pdf (Handbooks in PDF)

CHAPTER

ONE

KONFIGURATION UND INSTALLATION

1.1 Konfiguration und Installation

Die nachfolgenden Abschnitte befassen sich mit der Konfiguration, der Installation und dem Management eines Knotens aus Sicht eines lokalen Administrators. Das Handbuch behandelt folgende Teilaspekte:

1. Vorbetrachtungen / Installationsvoraussetzungen
2. Konfigurationsskript
3. Konfiguration eines externen Proxy (optional)
4. Installation
5. Management des Knotens

Vor der Installation bzw. während des Installationsvorgangs müssen einige Parameter (z.B. Speicherorte, Hostnamen, private Schlüssel oder Zertifikatspasswörter) konfiguriert werden. Einige dieser Einstellungen sind sensitiv. Der gesamte Konfigurations- und Installationsprozess ist so gestaltet, dass sensitive Informationen Ihre Einrichtung nie verlassen.

Warning: Wir haben bei der Entwicklung von CRIMSy Wert auf Sicherheit und Datenschutz gelegt. Die Software darf jedoch keinesfalls zur Speicherung und Verarbeitung sensitiver personenbezogener Daten (z.B. Patientendaten) verwendet werden! Näheres entnehmen Sie bitte dem Handbuch Systembeschreibung (Kapitel Risikoanalyse).

1.1.1 Ablauf

Zunächst prüfen Sie die im folgenden Abschnitt beschriebenen Systemanforderungen und stellen ein entsprechend konfiguriertes System zusammen. Anschließend laden Sie das Konfigurationsskript und unsere Schlüssel herunter. Nachdem Sie die digitale Signatur des Konfigurationssskripts verifiziert haben, führen Sie dieses aus und senden uns die vom Skript erstellte Konfigurationsdatei zu. Anhand dieser Konfigurationsdatei wird von uns ein Installationspaket betriebsfertig konfektioniert, an Sie verteilt und schließlich von Ihnen in Betrieb genommen.

Note: Für die Konfiguration und die Installation benutzen Sie bitte einen einzigen dedizierten unprivilegierten Account.

Falls Sie die Konfiguration über PuTTY ausführen wollen, stellen Sie bitte den Zeichensatz auf UTF-8 um.

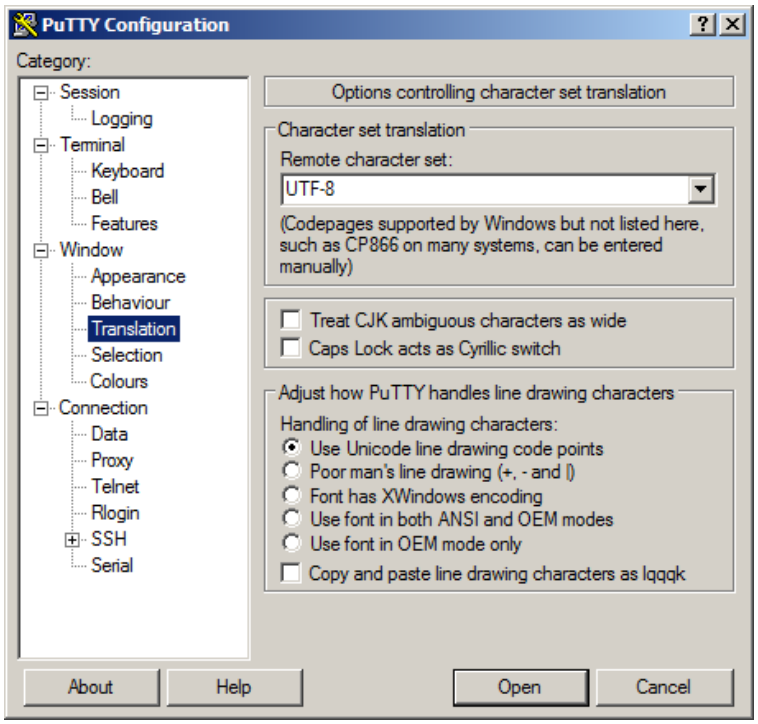


Fig. 1: PuTTY Terminal Konfiguration

1.1.2 Systemanforderungen

In der Einstiegsphase sind zum Betrieb eines Cloud-Knotens 1 - 2 CPU-Kerne, 2 GByte RAM und ca. 20 GByte Plattenspeicher sowie eine Netzwerkanbindung ab ca. 10 Mbit/s ausreichend. Die Entwicklung erfolgt auf virtualisierter x86_64-Hardware (KVM). Der Cloud-Knoten kann also als virtuelle Maschine betrieben werden – ein dedizierter physischer Server ist nicht notwendig. Sofern eine Java Runtime zur Verfügung steht, sollte der Betrieb auch auf jeder anderen Architektur (z.B. ARM) möglich sein. Eine Gewähr wird hierfür jedoch nicht übernommen.

1.1.3 Speicher

Ein Teil des Plattenspeichers wird für die permanente Speicherung der lokalen Cloud-Daten verwendet. In der Standardeinstellung wird hierfür das Verzeichnis Home-Verzeichnis des Nutzers verwendet, der das Konfigurationsskript aufruft. Diese Einstellung ist jedoch konfigurierbar. Diese Daten sollten durch ein regelmäßiges Backup vor Datenverlust geschützt werden. Wichtig: für das Datenverzeichnis soll kein Netzwerkdateisystem (z.B. NFS) verwendet werden.

Note: Es gibt Szenarien, in denen der Einsatz von NFS unvermeidbar ist, z.B. wenn ein ESX-Cluster seine Volumes per NFS von einem NetApp Filer bezieht. Innerhalb des Knotens bzw. aus Knotensicht handelt es sich hier jedoch nicht um den Einsatz von NFS, da dies dem Knoten vollständig verborgen ist.

1.1.4 Betriebssystem

Die Entwicklung erfolgt unter OpenSUSE (aktuell OpenSUSE 15.1 LEAP). Ein Knoten der Cloud setzt sich dabei aus mehreren Docker Containern zusammen. Die Verwendung anderer Linux-Distributionen ist möglich, getestet wurde unter anderem Ubuntu. Ein direkter Betrieb unter Windows wird nicht empfohlen, da während Installation und

Betrieb zahlreiche Linux/Unix-Werkzeuge zum Einsatz kommen, die bei einem Betrieb unter Windows zahlreiche Anpassungen erforderlich machen würden.

1.1.5 Software

Zusätzlich zu einem minimalen Linux-System erfordern Installation und Betrieb eines Cloud-Knotens folgende Software:

- Docker und Docker Compose (ab Docker Version 1.12)
- Dialog (NCurses-Dialoge für Shell-Skripte)
- GnuZip
- m4 Makroprozessor
- ed (Unix Line Editor)
- OpenSSL
- ssh (nur falls Remote-Administration gewünscht wird)
- sudo
- tar
- sharutils (uuencode / uudecode)
- uuidgen
- curl

1.1.6 Benutzer

Für die Cloud sollte auf dem Knoten ein unprivilegierter Benutzer (z.B. lbac) eingerichtet werden. Dieser Benutzer benötigt Zugriff auf die verwendeten Schlüssel und Zertifikate (s.u.). Für bestimmte Zwecke (z.B. Einrichten von Cron-Jobs, Starten von Docker-Containern) benötigt dieser Nutzer auch Zugriff auf eine administrative Shell (sudo). Während der Installation konfiguriert sich das Setup-Skript einen entsprechenden Zugang für den unprivilegierten Benutzer, wofür einmalig das Root-Passwort benötigt wird.

Note: Zur Erhöhung der Sicherheit kann die sudo-Berechtigung nach Beendigung des Installationsskripts wieder entzogen werden. Dazu muss die Datei */etc/sudoers.d/lbac* gelöscht werden.

Falls der Speicherort nicht anderweitig konfiguriert wurde, sollte das Homeverzeichnis des Nutzers ausreichend freien Platz für die Daten des Knotens haben (aktuell 10 GByte).

1.1.7 Zertifikate

Für die Absicherung der Maschine-zu-Maschine-Kommunikation der Knoten untereinander und für die Softwareverteilung werden Zertifikate eingesetzt. Für jede Cloud gibt es eine eigene Zertifizierungsstelle (CA - Certificate Authority), die diese Zertifikate ausstellt. Die Authentizität einer Zertifizierungsstelle kann anhand der sha256-Hashwerte der Zertifikatskette überprüft werden. Wichtig: Die Authentizität der Cloud-Installation hängt von einer sorgfältigen Prüfung dieser Hash-Werte ab. Folgende Zertifizierungsstellen sind momentan bekannt:

Hash-Wert der Zertifikatskette	Cloud
acb7b11ec12d21a83da6a47cc8b0ee89.. ..0a73e33f3a9dc23b7cafcc21f2343098	Leibniz Bioactives Cloud (nur Zertifikat: *cacert.pem*)
725ed0ea27e7f91bc9248ae19deb6aac.. ..47f3a5597ee2d2dd96e9f8f131c0365f	Leibniz Bioactives Cloud (Zertifikatskette: *chain.txt*)

Note: Die Leibniz Bioactives Cloud stellt insofern eine Besonderheit dar, als dass zunächst nur das Zertifikat verbreitet wurde. Ab Version CRIMSy 1.3.x gibt es auch für die Leibniz Bioactives Cloud eine 'Zertifikatskette', die allerdings auch nur das selbe CA-Zertifikat enthält.

Die Zertifikatsketten sowie zugehörige Zertifikatssperrlisten usw. können von den jeweiligen Distributionsseiten heruntergeladen werden. Die URL und etwaige Credentials können beim Verwalter der Cloud erfragt werden. Die vom Verwalter übermittelten Informatioenn sollten auch den Hashwert der Zertifikatskette beinhalten, um durch Übermittlung auf einem unabhängigen Kanal die Sicherheit zu erhöhen.

Note: Die Webadressen, Prüfsummen usw. dieses Handbuchs beziehen sich auf die von den CRIMSy-Entwicklern verwalteten Clouds (z.B. die Leibniz Bioactives Cloud). Falls Sie eine separate Distribution betreiben (vielleicht für Maschinenbauer oder Germanisten), werden die Webadressen, Prüfsummen usw. abweichen. Der Verantwortliche einer separaten Distribution sollte aber die unabhängige Prüfung der Authentizität der Zertifikate, Konfigurations- und Installationsskripte ermöglichen (z.B. über eine zusätzliche Email-Information).

Das Zertifikat für Ihren Knoten wird bei der Zusammenstellung des Installationspakets durch uns erzeugt und basiert auf dem während der Konfiguration erstellten Zertifikatsrequest. Für die Interaktion mit dem Nutzer sollte ein offizielles (d.h. ein von einer allgemein akzeptierten CA herausgegebenes) Zertifikat verwendet werden, um Fehlermeldungen im Browser des Nutzers zu vermeiden. Als Fallback-Lösung kann jedoch auch das Zertifikat Ihrer Cloud-CA benutzt werden.

Die privaten Schlüssel Ihrer Zertifikate (sowie die zugehörigen Passwörter) verlassen Ihren Knoten niemals.

1.2 Konfigurationsskript

Das digital signierte Konfigurationsskript steht auf der Distributions-Webseite der Cloud zum Herunterladen zur Verfügung. Die URL und die Credentials für den Zugriff werden vom Verwalter per Email mitgeteilt. Die Beispielwerte in den hier dargestellten Code-Schnipseln sind entsprechend zu ersetzen; zur Vereinfachung enthält die Distributionswebseite entsprechende Vorlagen für Copy & Paste.

```
curl -o conimage.sh.sig \
  https://test.crimsy.invalid/TEST/configure.sh.sig
curl -o chain.txt \
  https://test.crimsy.invalid/TEST/chain.txt
curl -o devcert.pem \
  https://test.crimsy.invalid/TEST/devcert.pem
sha256sum chain.txt
openssl verify -CAfile chain.txt devcert.pem
openssl smime -verify -in configure.sh.sig -certfile \
    devcert.pem  -CAfile chain.txt -out configure.sh
```

Zur Prüfung der Authentizität vergleichen Sie bitte die Ausgabe in Ihrem Terminal mit den kursiv markierten Zeilen in folgendem Block, wobei der Hash-Code cloudspezifisch ist:

```
[...]

~> sha256sum chain.txt
725 ...      Hash Code der Cloud-CA-Zertifikatskette     ... 65f chain.txt
~> openssl verify -CAfile chain.txt devcert.pem
devcert.pem: OK
~> openssl smime -verify -in configure.sh.sig -certfile \
    devcert.pem -CAfile chain.txt -out configure.sh
Verification successful
```

Bitte benachrichtigen Sie uns (bzw. Ihren Cloud-Verwalter) unbedingt, falls bei der Prüfung Ungereimtheiten auftreten. Falls die Signaturprüfung des Konfigurationsskripts erfolgreich war, können Sie das Skript anschließend aufrufen:

```
chmod +x configure.sh
./configure.sh
```

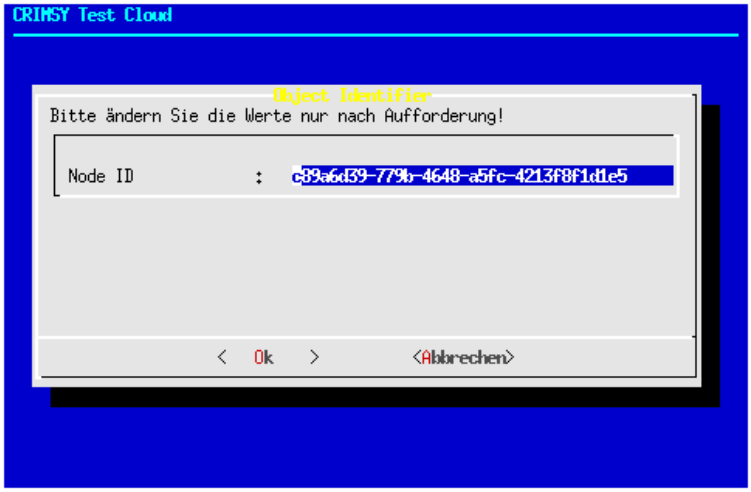
Das Konfigurationsskript beginnt mit einer Informationsseite:

Note: Da unsere Ressourcen sehr begrenzt sind, kann das Konfigurationsskript nur eine rudimentäre Prüfung der Eingabeparameter vornehmen. Durch entsprechend bösartige Eingaben (Backquotes “`”, $-Zeichen usw.) können wahrscheinlich Datenverluste und möglicherweise auch anderweitige Schäden provoziert werden. Bitte kontaktieren Sie uns, wenn Sie eines der Zeichen “`”$ verwenden müssen.

Als nächstes wird der Speicherort für die Daten der Cloud festgelegt:

Im einfachsten Fall können sie die Default-Einstellung übernehmen. Die größtmögliche Ausbaufähigkeit sichern Sie sich, wenn sie das Dateisystem als "logical volume" (LVM, ZFS, ...) anlegen. Im weiteren Verlauf wird dieser Speicherort in der Variable *$LBAC_DATASTORE* gespeichert. Auch in der Dokumentation wird diese Variable entsprechend verwendet. Falls das Konfigurationsskript im angegebenen Speicherort eine Konfigurationsdatei aus früheren Läufen des Skripts finden kann, werden Sie gefragt, ob Sie diese einlesen wollen:

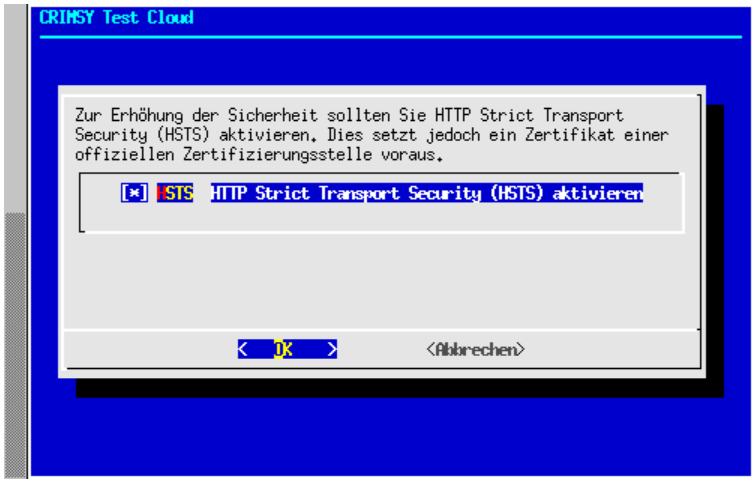
In den allermeisten Fällen sollten Sie die Datei einlesen lassen, Sie sparen dadurch Tipparbeit. Eine kritische Prüfung aller Eingaben bleibt dennoch unerläßlich. Das Skript zeigt als nächstes den Unique Identifier des Knotens an:



An diesem Wert sollten Sie nur Änderungen vornehmen, wenn Sie von uns dazu aufgefordert werden. Die nächste Frage ist eine einfache Kontrollfrage, ob Ihr System über ausreichende Ressourcen verfügt:

Wir hoffen, Sie können diese Frage mit "Ja" beantworten.

Im darauffolgenden Formular können Sie entscheiden, ob Ihr Knoten durch HTTP Strict Transport Security (HSTS) abgesichert werden soll. Für Installationen im produktiven Einsatz wird dies klar empfohlen. Die Einstellung setzt jedoch ein Zertifikat einer offiziellen Zertifizierungsstelle voraus. Für Test-Setups, bei denen Zertifikate einer CRIMSy-CA genutzt werden, sollte HSTS entsprechend nicht aktiviert werden.

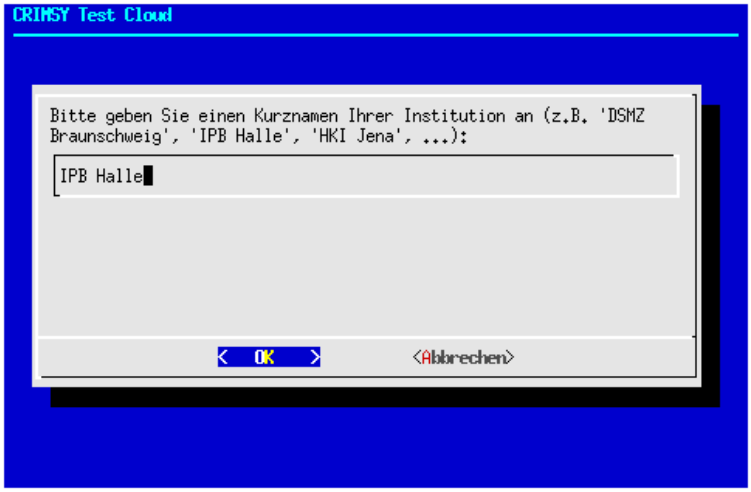


Mittlerweile verwenden viele Linux-Distributionen SystemD als Init-System. Im Rahmen des Konfigurationsskripts wird abgefragt, für welches System der Knoten konfiguriert werden soll, damit CRIMSy nach einem Neustart des Knotens automatisch starten kann. SystemV-Init-Skripte existieren für alle anderen Distributionen und als Fallback-Lösung; es finden allerdings keine Tests statt.

Ebenso ist für die Verwaltung des Knotens wichtig, ob auf dem Knoten noch andere Docker-Container ausgeführt werden. Die parallele Ausführung weiterer Container bedeutet, das beim Aufräumen von Docker-Containern, -Images und Volumes weniger aggressiv vorgegangen werden muss. Deshalb wird von der parallelen Ausführung weiterer Container auf dem Knoten dringend abgeraten.

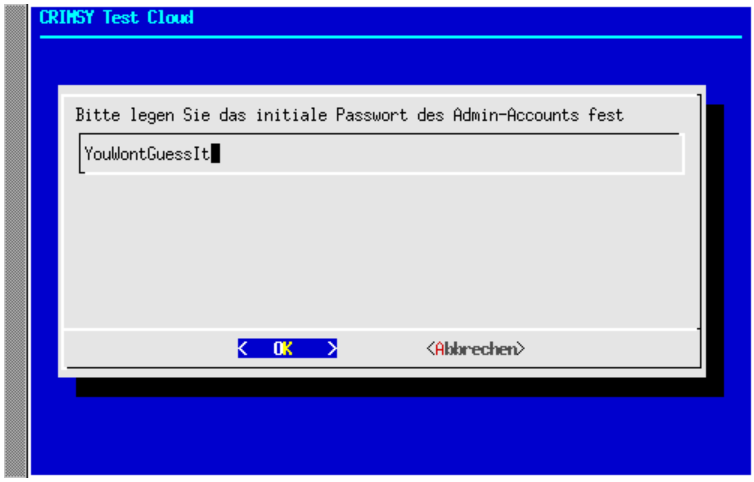
Als nächstes wird der Name Ihrer Einrichtung (Ihres Instituts) abgefragt. Geben Sie bitte die vollständige offizielle deutsche Bezeichnung an (ohne Adresse). Der hier abgefragte Name wird den Nutzern der Cloud angezeigt. Der Wert dieses Feldes wird außerdem in das Formular für den Zertifikatsantrag übertragen, kann dort aber angepasst werden, wenn für den Zertifikatsantrag eine andere Schreibung (z.B. wegen Umlauten) gewünscht wird.

Da der vollständige Name des Instituts manchmal unhandlich lang sein kann, wird im folgenden Formular eine Kurzbezeichnung abgefragt. Die Kurzbezeichnung sollte sich üblicherweise aus einer Abkürzung des Institutsnamens und dem Ort des Hauptsitzes zusammensetzen:

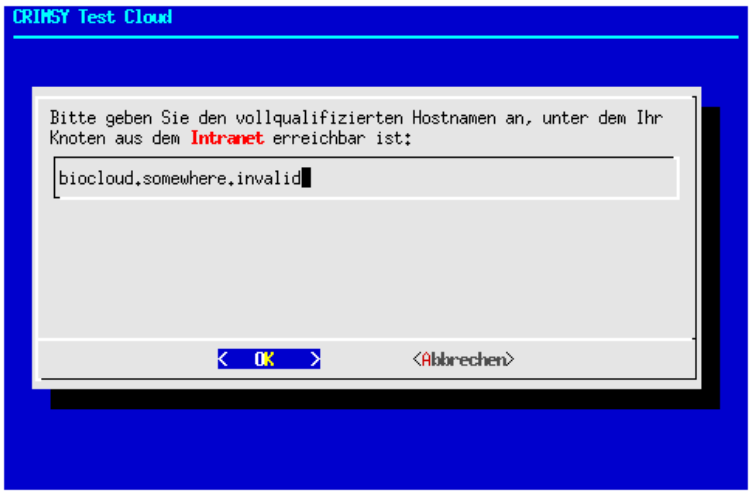


Eine weitere wichtige Information ist die Emailadresse einer Kontaktperson, die für die Administration des Knotens zuständig ist. Selbstverständlich ist auch eine Sammeladresse (z.B. helpdesk@somewhere.invalid) möglich.

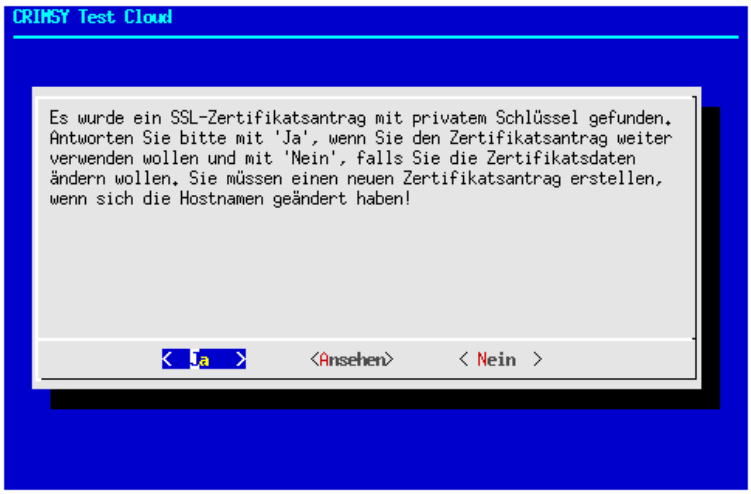
Zur Erhöhung der Sicherheit wird CRIMSy nicht mit einem fest vorgegebenen Standardpasswort für den Administrator-Account installiert. Stattdessen wird im Rahmen der Konfiguration ein Passwort abgefragt, mit dem der Administrator-Account initialisiert wird. Das Passwort wird im Klartext auf der lokalen Platte des Knotens gespeichert. Daher sollte das Admin-Passwort nach der Installation umgehend geändert werden.



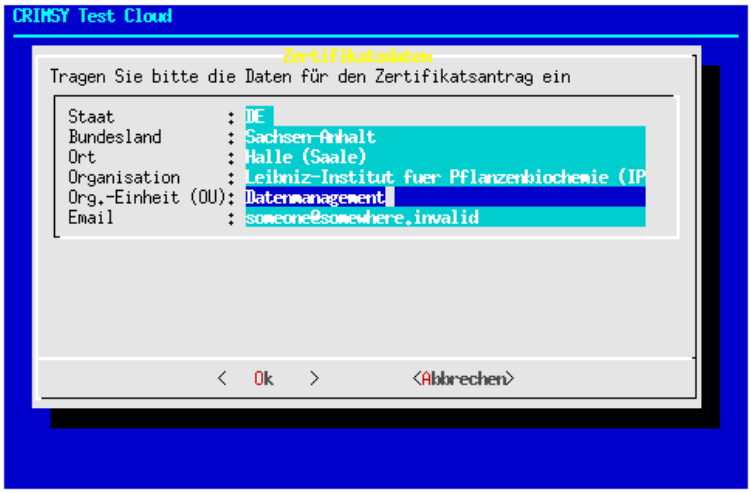
Sowohl für die Erstellung des Zertifikats (Subject Alternate Names) als auch für die Konfiguration des Proxy werden die Hostnamen Ihres Knotens benötigt. Dabei kann sich der Hostname im Intranet vom Hostnamen im Internet unterscheiden, weshalb beide abgefragt werden. Die Portnummer für die Kommunikation aus dem Internet ist auf 8443 festgelegt und kann nicht geändert werden.



In den folgenden Schritten muss für den Knoten ein Zertifkatsrequest erstellt werden. Falls jedoch bereits ein Zertifikatsrequest ausgestellt wurde, kann dieser geprüft und gegebenenfalls wiederverwendet werden (hier nicht gezeigt).

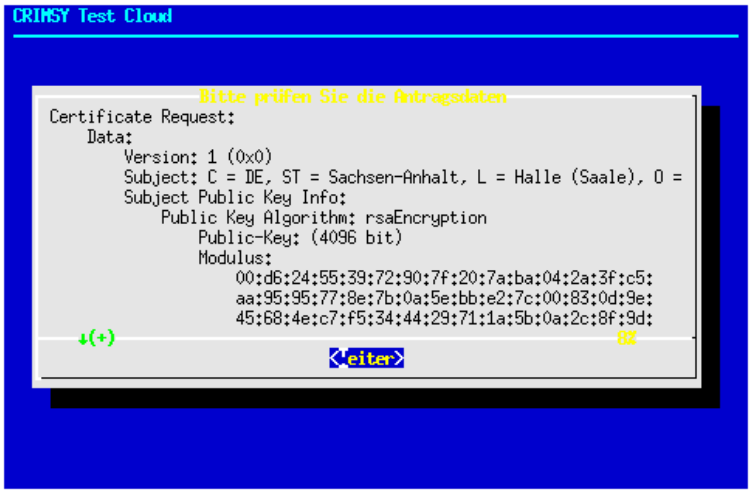


Das Konfigurationsskript übernimmt dabei die Felder “Organisation” und “Email” aus den bislang erfassten Daten; es sind aber Änderungen möglich. Zusätzlich müssen der Staat (Vorauswahl “DE”), das Bundesland, der Ort und die Organisationseinheit (Vorauswahl ist “Verwaltung”) eingegeben werden.



Nach Erfassung der Daten wird mit OpenSSL ein Zertifikatsrequest erzeugt. Auf Ihrem Terminal werden vorübergehend einige Ausgaben des Programms sichtbar sein. Im nächsten Formular sind Sie aufgefordert, die Daten des

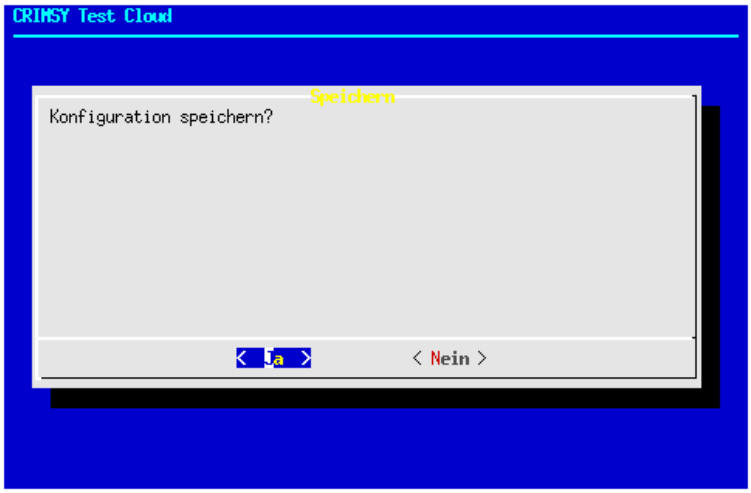
Zertifikatsrequests zu prüfen. Mit den Pfeiltasten bzw. PageUp und PageDown können Sie den Text scrollen.



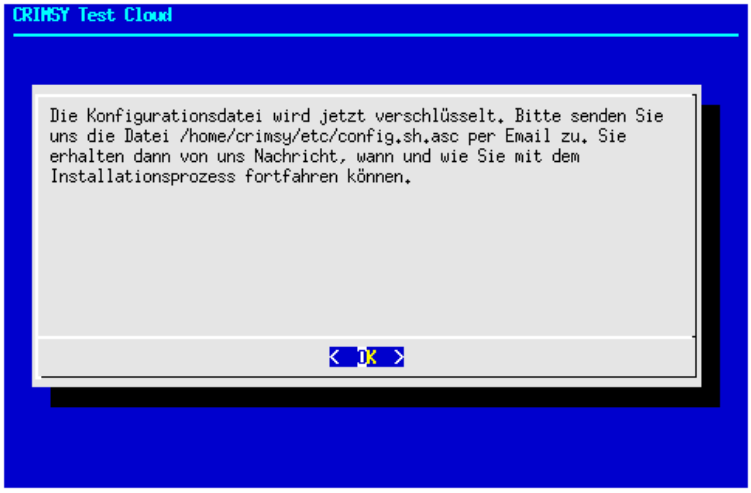
Falls alle Daten (auch die Hostnamen) richtig sind, können Sie dies im nächsten Formular bestätigen:

Falls Sie korrigieren möchten, kehrt das Skript zur Erfassung der (vollqualifizierten) Hostnamen zurück. Ansonsten erscheint eine Informationsseite zum Zertifikatsantrag und zur Verwendung eines "offiziellen" Zertifikats für die Nutzerkommunikation:

Die Konfiguration ist damit abgeschlossen und das Ergebnis sollte noch abgespeichert werden:



Die abgespeicherte Konfiguration wird anschließend verschlüsselt:



… und das Konfigurationsskript verabschiedet sich mit der Bitte, uns die verschlüsselte Konfiguration zukommen zu lassen (z.B. per Email).

Direkt nach Abschluß des Konfigurationsskripts sollte auch das offizielle Zertifikat für den Knoten beantragt werden, das für die Kommunikation mit den Browsern der Nutzer verwendet wird. Normalerweise sollte es ausreichend sein, den unter *$LBAC_DATASTORE/etc/lbac_cert.req* gespeicherten Zertifikatsrequest des Konfigurationsskripts bei einer CA (z.B. der DFN-PKI) einzureichen. Falls die CA nicht bereit ist, ein Zertifikat für diesen Request auszustellen, müssen Sie selbst einen passenden Request erzeugen.

Das weitere Vorgehen zur Installation des offiziellen Zertifikats richtet sich danach, ob Sie (1) das Zertifikat mit dem Zertifikatsrequest des Konfigurationsskripts beantragt haben oder (2) einen unabhängigen Request erzeugt haben. Im zweiten Fall wurde nämlich ein neuer privater Schlüssel erzeugt, während im ersten Fall die privaten Schlüssel identisch sind. Entsprechendes gilt auch für die Passworte der privaten Schlüssel.

Note: Das offizielle Zertifikat (und ggf. der private Schlüssel und das Passwort) müssen momentan nach der Konfiguration manuell in das Konfigurationsverzeichnis *$LBAC_DATASTORE/etc/* kopiert werden. Die Dateinamen sind festgelegt und lauten:

Datei	Beschreibung
official_cert.pem	Das offizielle Zertifikat im PEM-Format
official_cert.key	Der private Schlüssel des offiziellen Zertifikats. Falls das offizielle Zertifikat mit dem vom Konfigurationsskript erzeugten Zertifikatsrequest beantragt wurde, muss der private Schlüssel *lbac_cert.key* kopiert werden. Ansonsten verwenden Sie bitte den privaten Schlüssel Ihres unabhängigen Zertifikatsrequests.
official_cert.passwd	Das Passwort für den privaten Schlüssel des offiziellen Zertifikats. Falls der private Schlüssel *lbac_cert.key* dupliziert wurde (s.o.), muss auch die Passwortdatei *lbac_cert.passwd* kopiert werden. Ansonsten hinterlegen Sie bitte das Passwort für den privaten Schlüssel in dieser Datei.

1.3 Externer Proxy

Neben der Authentifizierung des Datenverkehrs erfüllt der Proxy noch eine weitere Funktion, indem er statischen Content, wie z.B. JavaScript-Bibliotheken von Browser-Plugins zur Verfügung stellt. Es wird daher empfohlen, den vorkonfigurierten Proxy-Container des Systems zu verwenden. Der nachfolgende Abschnitt kann aber als Referenz dienen, wenn ein dedizierter eigener Proxy-Server verwendet werden soll. Ebenso kann das Github-Projekt CRIMSy-CI nützliche Hinweise liefern. Notwendig ist auf jeden Fall die manuelle Anpassung der Docker Compose-Datei.

Warning: Die Nutzung eines externen Proxy-Servers wird nicht mehr offiziell unterstützt, die hier gegebenen Hinweise können zum Teil veraltet sein.

Der interne Proxy der Cloud basiert auf Apache httpd 2.4.x. Die Übernahme der Konfiguration in einen externen Proxy mit Apache httpd 2.4.x dürfte sich daher trivial gestalten. Beim Einsatz einer anderen Software wird der Aufwand naturgemäß höher ausfallen - eine Abschätzung können wir an dieser Stelle leider nicht liefern. Da der Apache httpd als (Reverse) Proxy mit Verschlüsselung betrieben wird, müssen zunächst die Module *mod_proxy*, *mod_proxy_http* und *mod_ssl* geladen sein. Das Modul *mod_proxy_wstunnel* ist mittlerweile entbehrlich, weil Websockets nicht mehr verwendet werden.

Als nächstes muss der Proxy Zugriff auf die Zertifikate des Knotens erhalten. Da ein Knoten Mitglied in mehreren Clouds sein kann, befinden sich die Dateien für jede Cloud in einem entsprechenden Unterordner unterhalb von *$LBAC_DATASTORE/dist/etc/*. Für eine Cloud *TEST* befinden sich das Zertifikat des Knotens und der dazugehörige private Schlüssel in den den Dateien *TEST.cert* und *TEST.key* im Ordner *$LBAC_DATASTORE/dist/etc/TEST/*. Die URLS für den Download der Zertifikate der Zertifikatskette sowie der zugehörigen CRLs befinden sich in der Datei *addresses.txt* (jeweils im Unterordner der Cloud). Für offizielle Zertifikate gibt es bei Verwendung externer Proxies keinen vorgeschriebenen Speicherort.

Schließlich müssen zwei VHosts für den internen und externen Datenverkehr konfiguriert werden. Für den Verkehr mit dem Intranet kommt folgendes Template für die Konfiguration des VHosts zum Einsatz:

```
<VirtualHost _default_:443>
    # General setup for the virtual host
    ServerName LBAC_INTRANET_FQHN:443
    ServerAdmin LBAC_MANAGER_EMAIL
    DocumentRoot "/usr/local/apache2/htdocs"

    SSLEngine on
```

(continues on next page)

(continued from previous page)

```
    SSLCertificateFile "/usr/local/apache2/conf/official_cert.pem"
    SSLCertificateKeyFile "/usr/local/apache2/conf/official_cert.key"
    SSLCACertificateFile "/usr/local/apache2/conf/official_cacert.pem"

    <IfModule mod_proxy.c>
        ProxyPass         /ui http://DOCKER_HOST:8080/ui
        ProxyPassReverse  /ui http://DOCKER_HOST:8080/ui
    </IfModule>
</VirtualHost>
```

Für die Kommunikation mit dem Internet übernimmt der Proxy die zertifikatsbasierte Authentifizierung der Clients. Der Bereich erlaubter URLs ist zudem gegenüber dem Intranet eingeschränkt. Das Template für die Konfiguration des Internet-VHosts lautet daher:

```
<VirtualHost _default_:8443>
    # General setup for the virtual host
    ServerName LBAC_INTERNET_FQHN:8443
    ServerAdmin LBAC_MANAGER_EMAIL
    DocumentRoot "/usr/local/apache2/htdocs"

    SSLEngine on

    SSLCertificateFile "/usr/local/apache2/conf/TEST.pem"
    SSLCertificateKeyFile "/usr/local/apache2/conf/TEST.key"

    # Certificate Revocation Lists (CRL):
    SSLCARevocationCheck chain
    SSLCARevocationPath "/usr/local/apache2/conf/crl/"

    SSLVerifyClient require
    SSLVerifyDepth 3
    SSLCACertificatePath "/usr/local/apache2/conf/crt/"

    <IfModule mod_proxy.c>
        ProxyPass         /ui/rest  http://DOCKER_HOST:8080/ui/rest
        ProxyPassReverse  /ui/rest  http://DOCKER_HOST:8080/ui/rest
        ProxyPass         /ui/servlet/document http://ui:8080/ui/servlet/document
        ProxyPassReverse  /ui/servlet/document http://ui:8080/ui/servlet/document
    </IfModule>
</VirtualHost>
```

Die Zertifikate aller Cloud-CAs eines Knotens werden im Verzeichnis */usr/local/apache2/conf/crt/* abgelegt. Analog müssen die Sperrlisten aller beteiligten CAs im Verzeichnis */usr/local/apache2/conf/crl/* abgelegt werden. Apache erwartet, dass die Zertifikate und Zertifikatssperrlisten über ihre *subject hashes* gefunden werden können (siehe Apache Dokumentation und *c_rehash*). Wichtig ist, die Zertifikatssperrliste täglich zu aktualisieren.

Optional (hier nicht gezeigt) kann ein dritter VHost zum Einsatz kommen, um bei unverschlüsseltem Zugriff aus dem Intranet auf die verschlüsselte Seite(n) umzuleiten.

1.4 Installation

An die Konfiguration schließt sich die Installation an. Die Datei *$LBAC_DATASTORE/bin/install.sh* ist ein kurzes Shellskript, welches bereits bei der Konfiguration angelegt wurde. Dieses Skript lädt das eigentliche Installationspaket herunter, entschlüsselt es und überprüft die digitale Signatur, bevor die Installation ausgeführt wird. Das

Installationspaket ist ein Shell-Skript mit eingebettetem, base64-codiertem tar.gz-Archiv, das alle notwendigen Bestandteile für die Installation enthält. Das Installationspaket wird für jede Einrichtung separat erzeugt und enthält neben der eigentlichen Software (WAR-Datei, Java-Bibliotheken, Docker-Konfiguration) auch Informationen über den Master-Knoten und das Zertifikat für die Cloud-interne Kommunikation. Der Dateiname des Installationspakets ergibt sich aus dem MD5-Hash des Namens der Einrichtung, z.B. *Leibniz-Institut fuer Pflanzenbiochemie (IPB)* → *10e8bcf6014cdd7ccc82b54ddaffdb00.asc.sig*.

Das Installationsskript kann mit verschiedenen Kommandozeilenoptionen aufgerufen werden, die an das Installationspaket durchgereicht werden. Momentan kann der Installationsprozess mit folgenden Optionen modifiziert werden:

Option	**Erläuterung**
–help	Zeigt einen Hilfetext an und beendet das Programm ohne weitere Aktionen
–debug	Öffnet die Ports der Datenbank (5432) und TomEE (8080) für direkte Zugriffe auf die Docker-Container.
–skip-snapshot	Überspringt die Prä-Installationsroutinen, in denen z.B. Backups der bisherigen Installation angelegt werden.
–standalone	Führt eine Erstinstallation so aus, dass keine Referenz auf den Master-Knoten eingetragen wird, so dass der Knoten selbst als Master operiert. Diese Option ist hauptsächlich "zum Ausprobieren" gedacht. Ein Standalone-Knoten kann nachträglich nicht in einen normalen Knoten umgewandelt werden - hierzu ist eine Neuinstallation notwendig. Insbesondere müssen die Verzeichnisse *data/* und *dist/* gelöscht werden. Bei späteren Installationen (Updates, . . .) hat diese Option keine Wirkung.

Der größte Teil des Installationsskripts wird mit einem unprivilegierten Account ausgeführt. Für eine Reihe von Installationsschritten sind jedoch administrative Privilegien notwendig. Hierunter fallen

- die Installation der Init-Skripte
- die Änderung der Ordner-Eigentümer im Verzeichnis data/
- die Erstellung von Backups
- die Erstellung von Cron-Jobs
- die Erzeugung von Docker-Images und -Containern

Hierfür wird auf das Kommando *sudo* zurückgegriffen, dessen Benutzung einmalig autorisiert werden muss. Mit Ausnahme der Passworteingabe sind für die Installation keine weiteren Nutzer-Interaktionen notwendig.

Sofern das Installationsskript beim Aufruf kein offizielles Zertifikat findet, wird das Zertifikat der Leibniz Bioactives Cloud CA als Fallback-Lösung verwendet. In diesem Fall kann einfach das Installationsskript ein zweites Mal aufgerufen werden, nachdem das Zertifikat nachträglich in das Konfigurationsverzeichnis kopiert wurde.

In der Regel ist der Knoten nach Durchlaufen aller Installationsschritte betriebsbereit. Eine Überprüfung kann mit dem Befehl *sudo docker ps -a* erfolgen und sollte in etwa folgendes Ergebnis liefern :

```
sudo docker ps -a
 CONTAINER ID    IMAGE          COMMAND                  CREATED       STATUS          PORTS                       NAMES
 69cdf443ee22    dist_proxy     "httpd-foreground"       1 hour ago    Up 2 minutes    0.0.0.0:80->80/tcp, [...]   dist_proxy_1
 cfc3b9444175    dist_ui        "su - tomee /usr/l..."   1 hour ago    Up 3 minutes    8080/tcp                    dist_ui_1
 82579971b2ed    dist_db        "docker-entrypoint..."   1 hour ago    Up 4 minutes    5432/tcp                    dist_db_1
```

Fig. 2: Example output of docker ps

Sofern alle Container ordnungsgemäß gestartet wurden, kann der Knoten mit einem Webbrowser benutzt und weiter konfiguriert werden (s.u.).

Warning: Insbesondere sollte als erstes das beim Konfigurationsprozess vergebene initiale Passwort des lokalen Administrator-Accounts (admin) geändert werden!

Durch die bei der Installation hinterlegte Adresse des Master-Knotens werden die übrigen Knoten der Cloud dem neuen Knoten in den ersten Minuten seines Betriebs bekannt. Hierzu ist ungehinderte HTTPS-Kommunikation mit externen Knoten auf Port 8443 notwendig.

Leider kommt es gelegentlich vor, dass ein oder mehrere Container nicht ordentlich starten. In einigen Fällen hilft es, die beteiligten Container und Images vom Init-Script neu erstellen zu lassen:

```
sudo bash
systemctl stop lbac.service
# pause for some time
systemctl start lbac.service
```

Bei unseren Tests kam es aber auch vor, dass wir beschädigte Container, Images und Volumes mit den Kommandozeilenwerkzeugen *docker-compose* bzw. *docker* beseitigen bzw. reparieren mussten. In Einzelfällen wurde der Docker-Daemon in Mitleidenschaft gezogen, so dass wir unseren Docker-Host durchbooten mussten.

Note: Eine vollständige Dokumentation aller Eventualitäten oder gar deren automatische Behebung liegen leider jenseits unserer Möglichkeiten. Bitte kontaktieren Sie uns im Falle eines Falles.

1.5 Management des Knotens

Warning: Das initiale Passwort des Administrators (bei der Konfiguration vergeben) sollte umgehend nach der Installation geändert werden!

Einige Einstellungen des Knotens können zur Laufzeit mit dem Webbrowser konfiguriert werden. Dazu gehören

- **Nutzer- und Gruppenverwaltung**
  - Passwortänderungen für lokale Accounts (z.B. lokaler Administrator)
  - Hinzufügen bzw. Deaktivierung von lokalen Nutzern und Gruppen
  - Ändern von Gruppenmitgliedschaften
  - Anbindung an LDAP-Verzeichnisse (z.B. Active Directory)
- Ändern von Systemeinstellungen
- Konfiguration eines externen Job-Schedulers
- Verwaltung von Barcode-Druckern
- Verwaltung von Collections (Anlegen, Ändern, Löschen, . . .)

1.5.1 Firewall

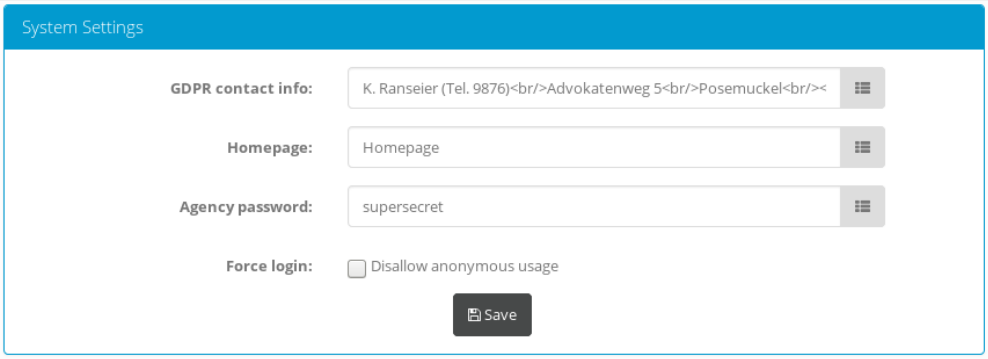
Sofern CRIMSy nicht in einer demilitarisierten Zone (DMZ) betrieben wird, sollte mindestens ein Paketfilter installiert werden, um den Zugang zu den einzelnen Diensten zu regulieren. Während Port 8443 öffentlich erreichbar sein muss, dürfen die Ports 80 und 443 (HTTP und HTTPS) nur aus dem internen Netz erreichbar sein. Für Ubuntu- und Debian-Systeme mit installierter 'Uncomplicated Firewall' (ufw) könnten die nachfolgend wiedergegebenen Regeln einen Basis-Schutz darstellen. Dazu müssen die Regeln nach lokaler Anpassung an die Datei */etc/ufw/after.rules* angefügt werden.:

```
###
### Block worldwide access to CRIMSy user interface
### Filter rules according to: https://github.com/chaifeng/ufw-docker
###
###
*filter
:ufw-user-forward - [0:0]
:ufw-docker-logging-deny - [0:0]
:DOCKER-USER - [0:0]
-A DOCKER-USER -j ufw-user-forward

#
#=====================================================================
#
# - global access to port 8443
# - unlimited traffic among docker containers
#
-A DOCKER-USER -j RETURN -p tcp --dport 8443
-A DOCKER-USER -j RETURN -s 172.16.0.0/12

#
#=====================================================================
#
# repeat the following two lines for each subnet to allow internal
# HTTP / HTTPS traffic (HTTP will be redirected to HTTPS by proxy container)
# Replace '192.168.1.80/28' by the actual subnet address.
#
-A DOCKER-USER -j RETURN -p tcp -s 192.168.1.80/28 --dport 80
-A DOCKER-USER -j RETURN -p tcp -s 192.168.1.80/28 --dport 443

#
#=====================================================================
#
# reject or log malformed traffic
#
-A DOCKER-USER -p udp -m udp --sport 53 --dport 1024:65535 -j RETURN

-A DOCKER-USER -j ufw-docker-logging-deny -p tcp -m tcp --tcp-flags FIN,SYN,RST,ACK␣
↪SYN -d 172.16.0.0/12
-A DOCKER-USER -j ufw-docker-logging-deny -p udp -m udp --dport 0:32767 -d 172.16.0.0/
↪12

-A DOCKER-USER -j RETURN

-A ufw-docker-logging-deny -m limit --limit 3/min --limit-burst 10 -j LOG --log-
↪prefix "[UFW DOCKER BLOCK] "
-A ufw-docker-logging-deny -j DROP

COMMIT
```

Bislang liegen nur Anforderungen und Betriebserfahrungen mit dem IPv4-Protokoll vor. Der Betrieb in IPv6-Netzen sollte prinzipiell möglich sein. Mangels Testmöglichkeiten und aufgrund der zusätzlichen Komplexität wurde bislang von Versuchen in diese Richtung abgesehen.

Warning: Administratoren müssen durch Einführung entsprechender Regeln sicherstellen, dass sich durch das IPv6-Protokoll keine Angriffsvektoren (z.B. bezüglich unkontrollierter Datenabflüsse) ergeben.

1.5.2 Systemeinstellungen



In den Systemeinstellungen können momentan 4 Einstellungen verwaltet werden:

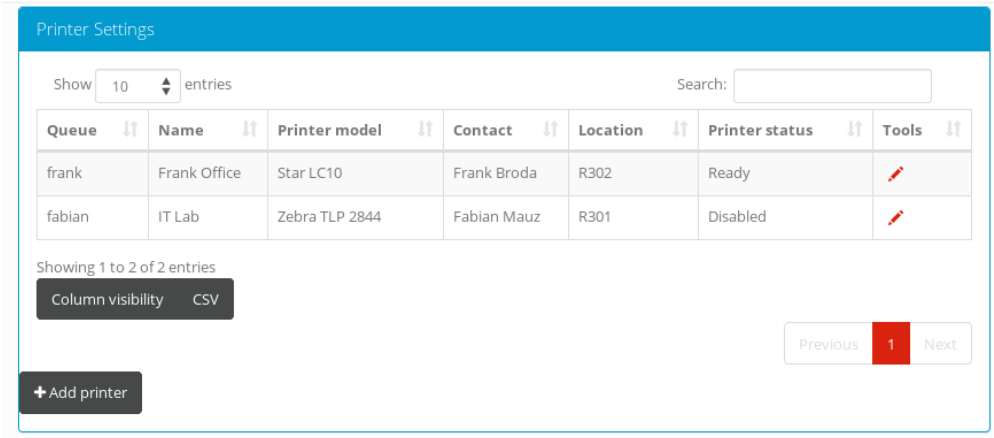
- Die Kontaktinformationen des Datenschutzbeauftragten (HTML-formatiert), wie sie in der Datenschutzerklärung des Knotens angezeigt werden sollen
- Die Homepage der den Knoten betreibenden Institution
- Das *shared secret* für die Jobverwaltung (**Achtung:** wird im Klartext angezeigt)
- Ein Flag, ob die anonyme Nutzung des Knotens möglich ist oder ob die Anmeldung obligatorisch ist

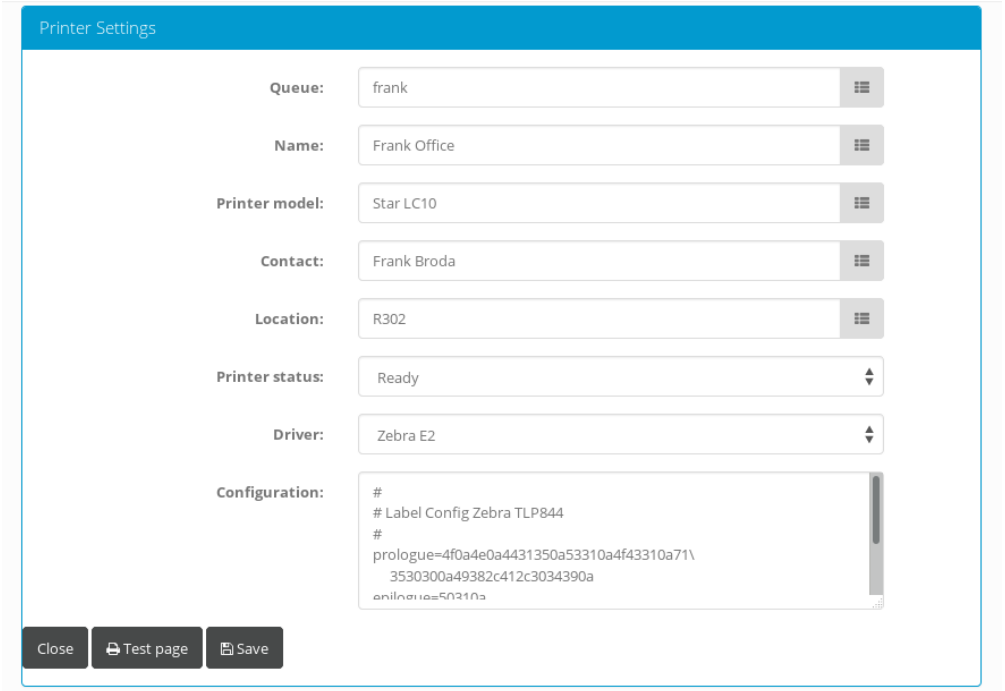
Die Änderung von Systemeinstellungen ist den Mitgliedern der lokalen Administratoren-Gruppe vorbehalten. Da einige Eingaben in den Systemeinstellungen im Kontext des Nutzerbrowsers interpretiert werden, müssen die Mitgliedschaften in der Administratorengruppe restriktiv gehandhabt werden.

1.5.3 Barcode-Drucker

Für klassisches Reporting setzt CRIMSy auf die Erzeugung von PDF-Dateien, die im Nutzerbrowser angezeigt werden können und die der Nutzer herunterladen oder über die ihm zugänglichen Drucker ausdrucken kann. Barcode-Drucker sind jedoch eine spezielle Klasse von Geräten, die sich unter anderem dadurch auszeichnen, dass sie auf ungewöhnliche Papierformate drucken und zum Teil recht komplex konfiguriert werden müssen. Beispielsweise müssen unterschiedliche Label-Formate, Druckgeschwindigkeiten, das Etikettenmaterial und der Schneidemechanismus für Etiketten konfiguriert werden. Die Installation und Konfiguration spezieller Treiber auf jedem einzelnen Nutzer-PC ist daher mit hohem Aufwand verbunden. Außerdem würde diese Herangehensweise die Nutzer mit unnötiger Komplexität konfrontieren. Bei der Entwicklung von CRIMSy wurde daher ein anderer Ansatz gewählt: Barcode-Drucker werden von CRIMSy über eigene Treiber direkt angesteuert. Der Systemadministrator konfiguriert die Drucker einmalig zentral. Nutzer können den für sie passenden Drucker aus einem Drop-Down-Menü auswählen.

Da CRIMSy üblicherweise in einer DMZ betrieben wird, können die Drucker nicht direkt angesteuert werden. Alle Druckjobs landen daher in einer Warteschlange, die von einem externen Dienst (CRIMSy Agency) abgearbeitet wird. CRIMSy Agency fragt den Knoten dabei regelmäßig ab, ob neue Druckaufträge vorliegen und leitet diese dann an einen Druckspooler (z.B. CUPS) weiter. Die Zuordnung erfolgt dabei über den Queue-Namen.





Die Druckertreiber sind recht flexibel gehalten und können konfiguriert werden. Dadurch ist es möglich, das Layout lokaen Wünschen anzupassen. Die Konfiguration erfolgt durch *Schlüssel=Wert*-Paare, wobei die Werte Hexadezimal eingegeben werden. Zeilen die mit einem Doppelkreuz als erstem Zeichen beginnen, werden ignoriert; ein Backslash als letztes Zeichen auf einer Zeile signalisiert, dass weitere Hexwerte auf der nächsten Zeile folgen (siehe Abbildung).

CHAPTER

TWO

SYSTEMARCHITEKTUR HANDBUCH

2.1 Vorwort

Dieses Handbuch unternimmt den Versuch die Architektur und die Gründe für diverse Architekturentscheidungen zu beschreiben. Das Struktur des Handbuchs versucht, sich am Arc42 Template zu orientieren.

Disclaimer: In dieser Dokumentation werden Marken, Produkt- und Warenzeichen ohne besondere Kennzeichnung benutzt.

Logo

CRIMSy (Cloud Resource & Information Management System) war und ist gewissermaßen das 'Betriebssystem der Leibniz Bioactives Cloud'. Großzügig durch den Leibniz-Forschungsverbund "Wirkstoffe und Biotechnologie" gefördert, startete das Projekt als "Leibniz Bioactives Cloud" (daher an vielen Stellen die Abkürzung "LBAC"). Ein neuer Name wurde notwendig, weil auch andere Verbünde bzw. Allianzen Interesse an der Nutzung und Weiterentwicklung der Software angemeldet haben. Stand März 2020 fördern der *Leibniz-Forschungsverbund "Wirkstoffe und Biotechnologie"* und der *Interdisziplinäre Verbund "Autonomie im Alter"* die Weiterentwicklung der Software.

2.2 Einführung

Das Projekt "Leibniz Bioactives Cloud" wurde vom Leibniz-Forschungsverbunds "Wirkstoffe und Biotechnologie" initiiert, um den Wissenschaftlern der beteiligten Institute ein "Smart Data Warehouse" als Forschungswerkzeug zur Verfügung zu stellen. Das "Smart Data Warehouse" soll als dezentral organisierte Web-Anwendung entwickelt werden. Gründe für die dezentrale Organisation sind nicht zuletzt verschiedene rechtliche Fragen (unter anderem Haftung und Finanzierung), deren Beantwortung in einem zentralistischen Ansatz für das Projektteam als unlösbar eingestuft wurden. Die unterschiedlichen Rechtsformen der beteiligten Institute haben hierbei durchaus eine Rolle gespielt. Als "Smart Data Warehouse" erschließt die Leibniz Bioactives Cloud den Forschern Dokumente (z.B. Jahresberichte,

Masterarbeiten und sonstige graue Literatur), auf die sie sonst keinen oder nur umständlichen Zugriff haben. Das Attribut "Smart" impliziert dabei, dass die Forscher auch mit unscharfen Begriffen operieren können und das System selbständig Zusammenhänge herstellt. Beispielsweise soll das System bei einer Suche mit dem Begriff "Brassicaceae" (die Pflanzenfamilie Kreuzblütler) auch Dokumente finden, in denen der Begriff "Arabidopsis" (die zu den Kreuzblütlern gehörende Gattung Schaumkressen) vorkommt. Stichworte in diesem Zusammenhang sind "semantische Annotation", "Ontologien" und "Natural Language Processing". Weiterhin ist geplant, den Forschern eine Plattform zur Verfügung zu stellen, in der sie Know-How oder Services anbieten oder nachfragen können (Marktplatz / Schwarzes Brett). Schließlich soll die Leibniz Bioactives Cloud die Möglichkeit zum Austausch von Datensätzen, beispielsweise Substanzdatenbanken oder Assay-Ergebnisse, bieten.

Die Software muss dabei strenge Qualitätsmaßstäbe erfüllen:

- An erster Stelle steht eine durchgehende Berücksichtigung von Sicherheitsaspekten beginnend beim Entwurf über die Implementierung bis zum Betrieb, um das Betriebsrisiko für die teilnehmenden Institute zu minimieren. Dies bedeutet, nur sichere Verfahren und Algorithmen einzusetzen, sicherheitsrelevante Entscheidungen zu dokumentieren und die Notwendigkeit von Sicherheitsupdates von Anfang an einzukalkulieren.
- Die Software soll ihren Nutzern durch innovative Funktionen Mehrwert bieten
- Die Software soll durch zeitgemäßes Design und Nutzerfreundlichkeit Lust auf die Arbeit mit ihr machen

2.2.1 Stakeholder

Die Entwicklung orientiert sich zunächst stark am Projektantrag für den Leibniz Forschungsverbund "Wirkstoffe und Biotechnologie". Als Auftraggeber bestimmt die Mitgliederversammlung des Forschungsverbunds die Richtung der Entwicklung. Der Sprecher und die Koordinatoren des Forschungsverbunds stehen ebenfalls beratend zur Seite. Für technische Fragen im Zusammenhang mit dem Roll-Out stehen wir in Kontakt mit den IT-Verantwortlichen in den Mitgliedsinstituten. Die einzelnen Beteiligten nehmen dabei verschieden Rollen wahr:

- *Wissenschaftler* sind die primäre Zielgruppe des Projekts. Als Anwender haben sie maßgeblichen Einfluss auf die Richtung der Entwicklung. Diese Gruppe beinhaltet sowohl "normale" Wissenschaftler als auch die Institutsdirektoren bzw. Sprecher des Forschungsverbunds.
- *Systemadministratoren* betreiben in ihrer Gesamtheit die verteilte Infrastruktur und sind mitverantwortlich (jeweils an ihrem Institut) für die Absicherung der Cloud (z.B. durch Firewalls, Backup, . . .).
- *Koordinatoren* unterstützen die Arbeit durch Beratung und organisatorische Hilfen.
- *Entwickler* Führen alle wesentlichen Schritte der Softwareentwicklung (Konzeption, Umsetzung, Tests, Dokumentation, Auslieferung) durch.

Die Kontaktdaten der beteiligten Personen können aus Datenschutzgründen nicht in diesem Dokument aufgelistet werden.

2.3 Aufgabenstellung

Die Umsetzung des Gesamtprojekts gliedert sich in diverse Einzelaufgaben, die in der Abfolge der Projektphasen zunehmend komplexer miteinander wechselwirken.

- **Dokumente indexieren:** Das System bietet eine Möglichkeit zur Volltext-Indexierung von Dokumenten. Verschiedene Produkte sollen auf ihre Eignung für diesen Zweck geprüft werden
- **Dokumente suchen:** Das System bietet eine Möglichkeit, Dokumente im Volltext-Index zu suchen (und zu finden)

- **Dokumente hochladen:** Das System bietet die Möglichkeit, Dokumente über ein geeignetes Interface hochzuladen. Die hochgeladenen Dokumente werden automatisch indexiert. Die Möglichkeit, Dokumente zu aktualisieren oder zu löschen muss geschaffen werden.
- **Dokumente herunterladen:** Gefundene Dokumente sollen von Benutzern heruntergeladen werden können.
- **Verteilte Operation:** Die Suche nach Dokumenten findet über mehrere Knoten verteilt statt, so dass mit einer Abfrage Dokumente aus mehreren beteiligten Instituten gefunden werden können
- **Verwaltung von Collections:** Dokumente werden in Collections (Kollektionen / Sammlungen / . . .) organisiert und gemeinsam verwaltet, die jeweils in einem Institut lokalisiert sind. Damit übt das jeweils beherbergende Institut die Gewalt über sie aus. Üblicherweise enthalten die Collections nur Dokumente aus dem eigenen Institut. Falls im späteren Projektverlauf die Möglichkeit eingeräumt wird, auch von anderen Instituten in eine Collection hochzuladen, so hat die beherbergende Institution insoweit auch die Gewalt über diese Dokumente inne. Das System bietet die Möglichkeit Collections anzulegen und zu löschen.
- **Verwaltung von Nutzern:** Das System bietet die Möglichkeit, lokale Nutzer zu authentifizieren. Dies wird über eine lokale Nutzerdatenbank realisiert. Weiterhin gibt es die Möglichkeit zur Anbindung an externe Verzeichnisdienste (LDAP, AD). Die Nutzung von Diensten wie Shibboleth wird gelegentlich geprüft, ist aber momentan nicht geplant. Die Authentifizierung von externen Benutzern soll stattdessen immer durch ihr Heimatsystem erfolgen. Gegenüber einem Remote-System weisen sie sich ggf. durch ein Token aus.
- **Verwaltung von Nutzergruppen:** Zur Vereinfachung der Verwaltung können Nutzer zu Gruppen zusammengefasst werden. Außerdem soll das Verschachteln von Gruppen ermöglicht werden. Es muss sichergestellt werden, dass trotz der verschiedenen Kontexte von Nutzern und Gruppen (lokale Nutzerdatenbank, Verzeichnisdienst, remote) keine Inkonsistenzen verusacht werden können.
- **Verwaltung von Berechtigungen:** Es findet eine feingranulare Berechtigungsprüfung (Access Control Lists) für alle Ressourcen des Systems statt.
- **Installation, Betrieb, Update:** Das System unterstützt die Systemadministratoren bei der Konfiguration, der Installation, beim Betrieb und beim Update von Knoten der Leibniz Bioactives Cloud
- **Sicherheit** Das System ist durch aktuelle Software und aktuelle Protokolle gegen Angriffe und Manipulation geschützt. Alle Kommunikation mit externen Systemen findet verschlüsselt statt.
- **Semantische Annotation:** . . .

2.4 Randbedingungen

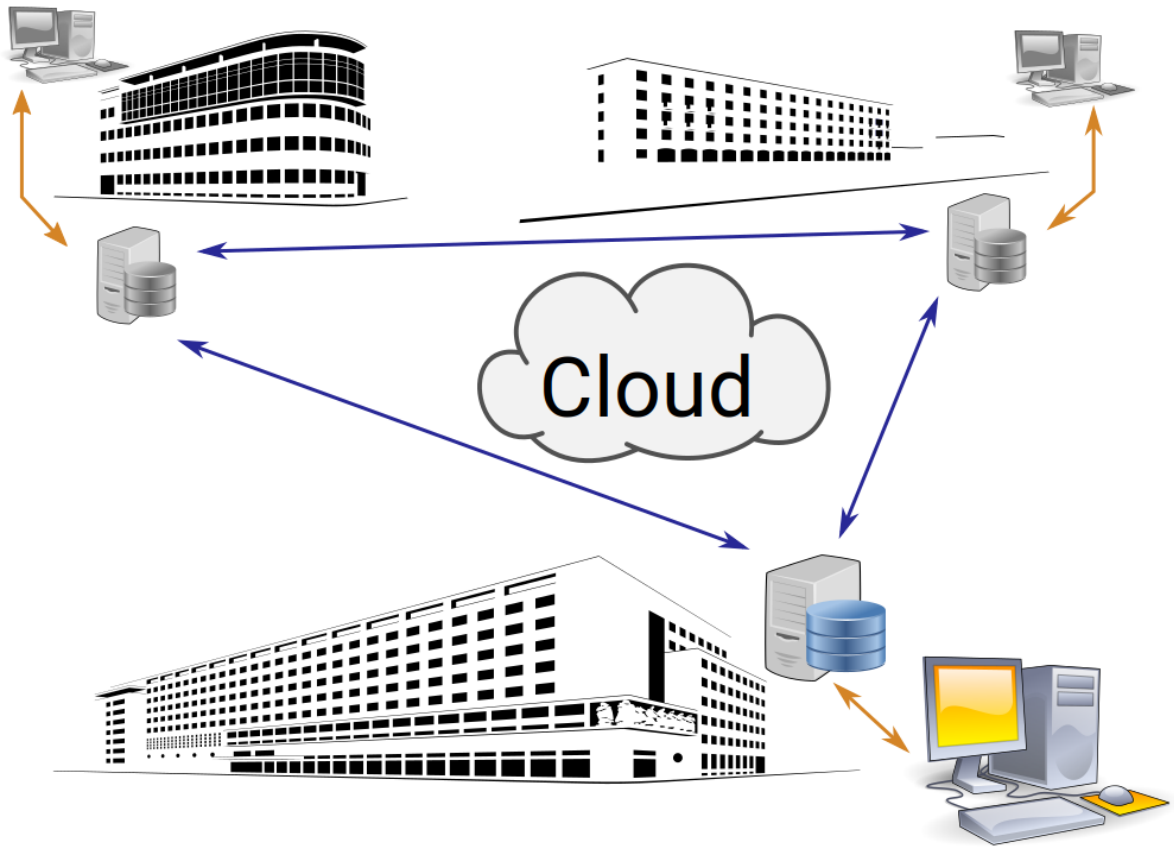
Das Projekt unterliegt verschiedenen Randbedingungen, die unmittelbaren Einfluß auf die Architektur und Implementierung des Projekts ausüben. Dieses Kapitel versucht diese Randbedingungen möglichst vollständig zu erfassen geht auf ihre Implikationen ein:

- **Budget:** Das Budget für das Projekt Leibniz Bioactives Cloud beträgt ungefähr 3 Mannjahre. Dem Projekt stehen 1 Entwickler in Vollzeit und ein weiterer Entwickler mit zusätzlichen anderen Aufgaben zur Verfügung. Damit sind die Möglichkeiten des Projekts sehr begrenzt und es kommt darauf an, die knappen Ressourcen möglichst effizient zu nutzen, indem weitgehend auf vorgefertigte Open Source Komponenten zurückgegriffen wird. Die Entwicklung soll weitgehend agil erfolgen - viele Vorgehensmodelle (bspw. Scrum) setzen jedoch größere Teams voraus, so dass eine Anpassung stattfinden muss.
- **Programmiersprache:** Prinzipiell gibt es eine Reihe von Frameworks in verschiedenen Programmiersprachen (Java, PHP, Python, Ruby, . . .), mit denen man die geplante Webanwendung umsetzen könnte. Ausschlaggebend für die Entscheidung für Java waren folgende Gesichtspunkte:
  - die bereits vorliegenden Erfahrungen mit der Entwicklung von Web-Anwendungen
  - das Vorhandensein einer breiten Auswahl allgemeiner (Nutzerinterfaces, Datenbankzugriffe, . . .) und fachspezifischer (Chemie, Natural Language Processing, Ontologien, . . .) Bibliotheken

  - aktiver Support, große Community
  - Möglichkeit der Realisierung in einer einzigen Programmiersprache (d.h. Vermeidung von Sprachmix)

- **Dezentralisierung:** Die Rechtsformen der am Leibniz Forschungsverbund "Wirkstoffe und Biotechnologie" beteiligten Institute sind sehr unterschiedlich: eingetragene Vereine (e.V.), Gesellschaften mit beschränkter Haftung (GmbH), Stiftungen öffentlichen Rechts. Die Institute sind zudem in unterschiedlichen Bundesländern angesiedelt, so dass die Rechtslage zu einzelnen Fragen unterschiedlich sein dürfte. Ein zentraler Ansatz würde bedeuten, dass entweder eine eigene Rechtspersönlichkeit gegründet werden müsste, die die Haftung und Finanzierung der Leibniz Bioactives Cloud übernimmt oder dass eines der beteiligten Institute diese Verbindlichkeiten schultert. Mit einem dezentralen Ansatz behält jedes Institut die Hoheit über seine Daten (präziser: über die bei ihm liegenden Daten) und ist selbst verantwortlich für die Sicherheit seiner Daten und der Infrastruktur sowie deren Finanzierung. Der im Rahmen des Projekts verfolgte dezentrale Ansatz hat zudem den Charme, dass weitere Interessenten eigene verteilte Clouds betreiben können. Jeder Teilnehmer kann Mitglied in einer oder mehreren solcher Clouds sein und muss dafür nur eine lokale Instanz betreiben.
- **Übertragungsprotokolle:** Der Austausch mit den IT-Verantwortlichen hat gezeigt, dass der Definition von Schnittstellen und Übertragungsprotokollen eine besondere Bedeutung zukommt. Als Beispiele für auftretende Hürden seinen genannt:
  - Am HKI gibt es starke Vorbehalte gegenüber dem CIFS-Protokoll, das in einem frühen Prototypen für das Hochladen von Dokumenten genutzt wurde.
  - Das DSMZ besitzt keine Hoheit über seine DMZ; alle Dienste müssen eine Prüfung durch das Helmholtz-Zentrum für Infektionsforschung (HZI) durchlaufen und die verwendeten Schnittstellen, Protokolle und die Art der übertragenen Daten deklarieren
  - Das Leibniz-Institut für Lebensmittelsystembiologie an der TU München (Leibniz LSB@TUM) verfügt nicht über eine demilitarisierte Zone für den Betrieb seines Knotens. Die Firewall (Paketfilter) muss entsprechend manuell für das System konfiguriert werden.

Dieses Kapitel listet alle beteiligten Kommunikationspartner (Nutzer und IT-Systeme) auf und beschreibt die verwendeten Ein- und Ausgabedaten sowie die verwendeten Protokolle. Die Darstellung beginnt mit dem Blick auf das Gesamtsystem geht im weiteren Verlauf auf die Details ein.



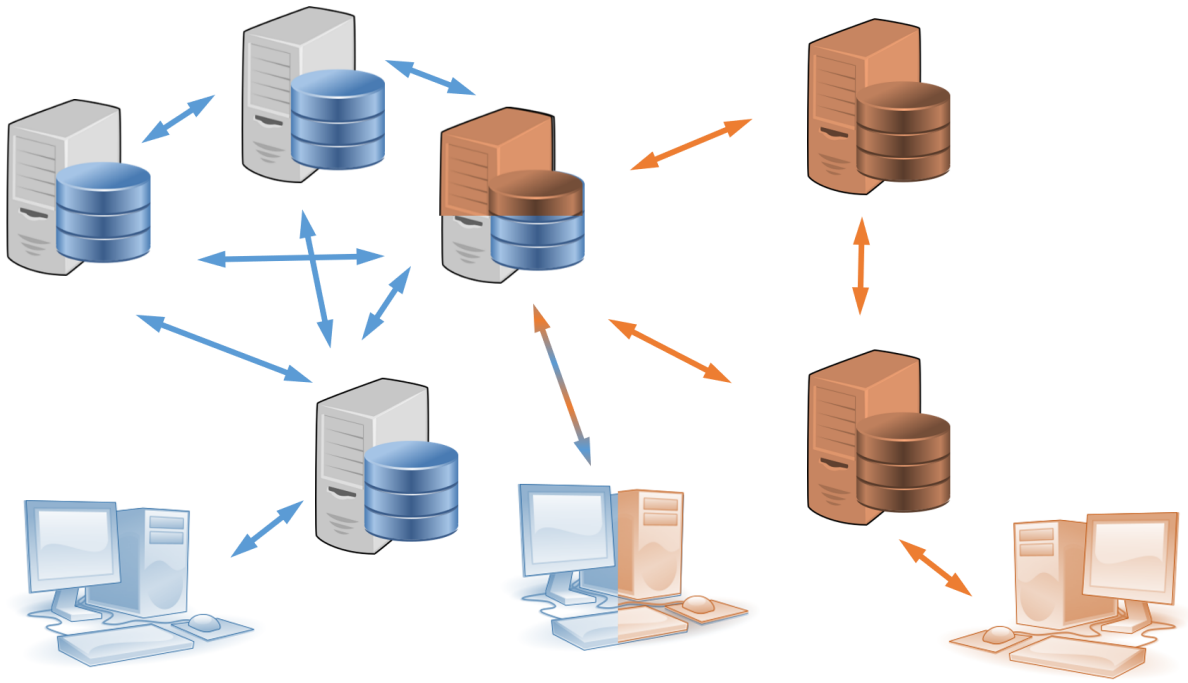
Cloud

Architecture Der ursprüngliche Ansatz, die Leibniz Bioactives Cloud, zeichnet sich dadurch aus, dass die Cloud durch mehrere Knoten gebildet wird, wobei von jeder beteiligten Institution ein eigener Knoten beigesteuert wird. Dieser Knoten kann von den Nutzern des jeweiligen Instituts per Internet-Browser kontaktiert werden (orange Pfeile) und steht mit allen Knoten der übrigen Institute über eine verschlüsselte Verbindung (blaue Pfeile) in Verbindung. Die Knoten sind dabei weitgehend gleichberechtigt. Die einzige Ausnahme ist, dass ein Server als Master-Knoten fungiert, indem er ein Verzeichnis aller Knoten zur Verfügung stellt. Die Master-Rolle kann jedoch prinzipiell von jedem Knoten übernommen werden.

Als Verbindungsprotokoll kommt ausschließlich HTTP mit TLSv1.2-Verschlüsselung (ggf. auch aktuellere Versionen) zum Einsatz. Die Zertifikate für Maschine-zu-Maschine-Kommunikation (also der Knoten untereinander) werden von einer eigenen CA (z.B. der Leibniz Bioactives Cloud CA) ausgestellt, während für die Kommunikation mit den Browsern der Nutzer Zertifikate weithin anerkannter CAs (z.B. DFN PKI) verwendet werden sollen. Als weitere Sicherung erfolgt die Maschine-zu-Maschine-Kommunikation ausschließlich mit gegenseitiger zertifikatsbasierter Authentifizierung.

Bei den ausgetauschten Daten handelt es sich einerseits um Suchanfragen, entsprechende Antworten und ggf. um die entsprechenden Dokumente. Andererseits tauschen die Systeme technische Informtionen, wie z.B. Adressen der verfügbaren Knoten, die durchsuchbaren Collections oder für die Berechtigungsermittlung notwendige Informationen (Name, Emailadresse, Session-Token usw.) aus.

Im Schema nicht gezeigt sind Verbindungen zu öffentlichen Repositories, die die Knoten für Updates herstellen.

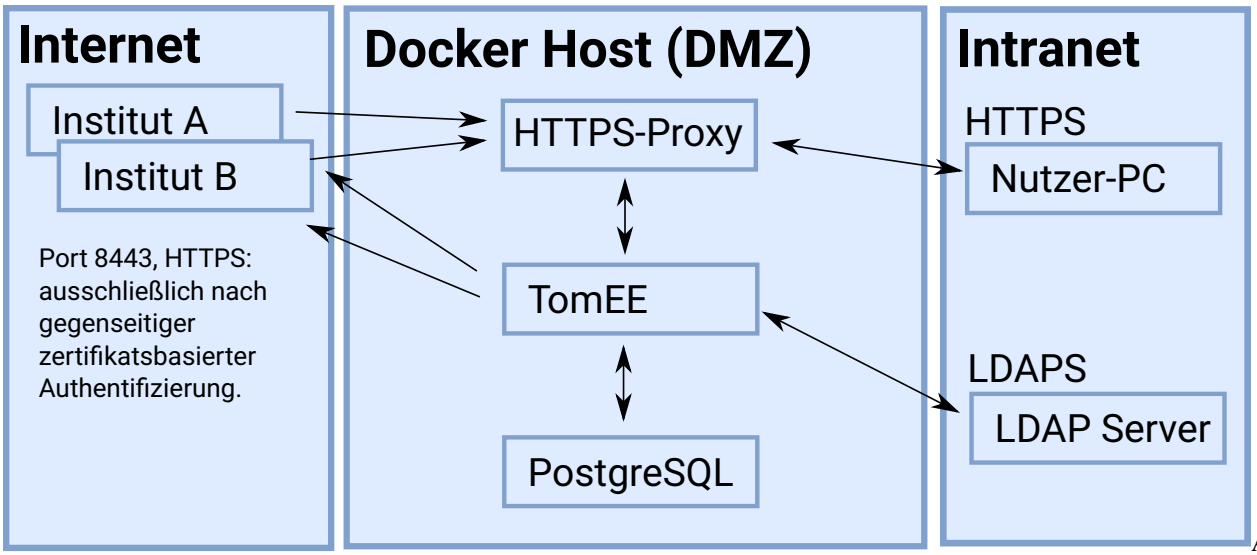


MultiCloud Architecture Im neueren "MultiCloud"-Ansatz hat kann ein Knoten Mitglied in mehreren Clouds sein. Er erhält dazu von der für die jeweilige Cloud zuständigen CA ein Zertifkat. Mit diesem kann er sich gegenüber den übrigen Knoten der Cloud ausweisen. Vom Master-Knoten der jeweiligen Cloud erhält er eine Liste der jeweils in der Cloud bekannten Knoten. Durch die verschiedenen Zertifikate ist sichergestellt, dass Knoten die unterschiedlichen Clouds angehören, nicht miteinander kommunizieren können.

> **Info:** *Seit Version 1.2.4 können die Cloud-CAs von zwei unterschiedlichen Clouds zudem von einer gemeinsamen Root-CA zertifiziert werden. Diese gemeinsame Root-CA ist jedoch nicht zwingend und die Leibniz Bioactives Cloud wird weiterhin mit ihrer ursprünglichen CA betrieben. Ein Grund hierfür war, dass eine Änderung der CA den in der Dokumentation angegebenen Hash des CA-Zertifikats ungültig gemacht hätte.*

2.5 Knotensicht

Jeder Knoten seinerseits besteht aus einem Docker-Host (Abbildung Mitte), der die verschiedenen Komponenten in sich vereint.



Architektur

> **Info:** *Die ursprüngliche Trennung der TomEE-Applikationen in Backend und UI wurde aufgegeben, da dieser Architekturansatz bei der Speicherung und Auslieferung von Dokumenten unpraktikabel ist. Die weitgehende Trennung der Nutzerkommunikation von der Maschine-zu-Maschine-Kommunikation wird jetzt durch die Komponente HTTPS-Proxy erreicht.*

Bevor auf die einzelnen Komponenten eingegangen wird, werden zunächst die verschiedenen Kommunikationspfade analysiert. Bei der Betrachtung der Kommunikation müssen zunächst folgende grundsätzliche Szenarien unterschieden werden:

- die Kommunikation innerhalb eines Cloud-Knotens
- die Kommunikation der Nutzer mit ihrem Cloud-Knoten
- die Kommunikation des Knotens mit lokalen Ressourcen (LDAP, DNS, NTP, . . .) des Instituts
- die Kommunikation der Cloud-Knoten untereinander
- die Kommunikation mit Software-Repositories für Update-Zwecke (nicht gezeigt)

Mit Ausnahme der Kommunikation innerhalb eines Knotens soll jegliche Kommunikation der Knoten verschlüsselt erfolgen. Es wird zudem dringend empfohlen, den Knoten durch eine externe Firewall abzusichern.

2.6 Intra-Node Kommunikation

Die Kommunikation innerhalb eines Cloud-Knotens kann nach aktueller Einschätzung unverschlüsselt erfolgen, da Vertraulichkeit und Integrität wirksam durch die Container-Umgebung geschützt werden. Eine zusätzliche Verschlüsselung oder Authentifizierung erhöht die Sicherheit in diesem Szenario nur minimal und geht vermutlich mit meßbarem Ressourcenmehrverbrauch einher.

Die Kommunikation innerhalb eines Knotens umfasst folgende Anwendungsfälle:

- JDBC-Datenbankverbindung zu einer PostgreSQL-Instanz
- HTTP-Kommunikation der TomEE-Webapplikation mit dem Reverse HTTPS-Proxy

2.7 Nutzer-Kommunikation

Die Kommunikation der Nutzer mit einem Cloud-Knoten erfolgt ausschließlich über das HTTPS-Protokoll (Port 443); Zugriffe über HTTP (Port 80) werden durch Redirection automatisch auf Port 443 weitergeleitet. Damit die Browser der Nutzer die Sicherheit der Verbindung erfolgreich überprüfen können, sollten bevorzugt öffentliche Zertifikate (z.B. von Zertifizierungsstellen des DFN-Vereins) zum Einsatz kommen. Die Verwendung der Leibniz Bioactives Cloud CA-Zertifikate wird lediglich als Fallback angeboten. Die Kommunikation der Nutzer umfaßt dabei die normale Interaktion von Nutzern mit einer Webseite und den Upload und Download von Dokumenten.

Durch Firewall-Regeln soll sichergestellt werden, dass der Zugriff auf die HTTP(S)-Schnittstelle (Ports 80 und 443) des Knotens nur aus dem Intranet des Instituts möglich ist.

2.8 Lokale Ressourcen

Für die Nutzerauthentifizierung ist die Anbindung an einen lokalen LDAP-Server geplant. Die Kommunikation mit dem LDAP-Server sollte verschlüsselt (SSL, TLS, . . .) erfolgen. Die Kommunikation mit weiteren Diensten (DNS, DHCP, NTP) erfolgt nach derzeitiger Planung unverschlüsselt. Für Wartungszwecke kann zudem auf dem Knoten ein SSH-Server installiert sein.

2.9 Maschine-zu-Maschine-Kommunikation

Die Maschine-zu-Maschine-Kommunikation erfolgt nur verschlüsselt und nur nach gegenseitiger zertifikatsbasierter Authentifizierung. Die Zertifikate hierfür werden durch die Leibniz Bioactives Cloud CA ausgestellt, die auch eine Zertifikatssperrliste (CRL) pflegt. Die Kommunikation zwischen den Knoten erfolgt über Port 8443 und nutzt das HTTPS-Protokoll. Die Zugriffsmöglichkeiten sind auf definierte Schnittstellen (i.d.R. REST-API) beschränkt. Beispiele für ausgetauschte Inhalte sind:

- die Liste der an der Cloud beteiligten Knoten
- die Liste der verfügbaren Collections
- Authentifizierungsinformationen für einen angemeldeten Nutzer (nur Token, keine Passwörter!)
- Suchanfragen
- Dokumentenauslieferung (z.B. nach einer Suchanfrage)

2.10 Software-Repositories

Für Updates der Software sind Kommunikationsverbindungen zu öffentlichen und nichtöffentlichen Repositories notwendig. Diese Verbindungen werden vom Docker-Host initiiert und durchweg über http / https abgewickelt. Die über diese Verbindungen übertragenen Daten sind üblicherweise durch digitale Signaturen vor Manipulation geschützt. Eine vollständige Liste der Endpunkte kann momentan nicht angegeben werden.

2.11 Komponenten-Sicht

Wie bereits in der Knoten-Sicht erkennbar besteht ein Knoten aus in Docker-Containern verpackten Komponenten:

- **Apache HTTP Server** Der Apache HTTP Server nimmt als Proxy (mit Ausnahme einiger weniger dokumentierter Fälle) alle eingehenden Verbindungsanfragen entgegen.

- **TomEE Web und EE Application Server** Im TomEE Server verbirgt sich der Hauptteil der Cloudlogik. Die Webanwendung steuert und organisiert im Verbund mit den anderen Knoten die Cloud und verarbeitet Suchanfragen, Upload- und Download-Requests und alle sonstigen Nutzeranforderungen. Sie arbeitet dazu mit den übrigen Komponenten (v.a. s.u.) zusammen.
- **PostgreSQL** Stellt eine fortgeschrittene relationale Datenbank zur Verfügung, auf die von der Webanwendung mittels JDBC-Schnittstelle (bzw. JPA / JPA2) zugegriffen wird.

und last but not least

- **Docker-Host** Stellt Speicher, Rechenleistung und Netzwerkverbindung bereit und orchestriert (mittels Docker-Compose) die Container.

2.12 Technischer Kontext

Der Knoten wird als einzelner Docker-Host realisiert, der in eine DMZ eingebettet ist. Ein Scale-Out (z.B. mit Kubernetes oder Docker Swarm) ist momentan nicht geplant. Wie bei Docker üblich stellt der Host Speicherplatz zur Verfügung und sorgt für die (Netzwerk-)Konnektivität der Komponenten. Dabei ist wichtig, dass der Speicherplatz nicht über Protokolle wie NFS oder CIFS angebunden ist, sondern als interner Plattenspeicher, DAS oder SAN-Speicher zur Verfügung gestellt wird. Das System kann als physischer Server oder virtuelle Maschine realisiert sein. Sämtliche Kommunikation findet über die Ethernet-Schnittstelle(n) des Hosts statt. Die Unterscheidung zwischen institutsinterner und externer Kommunikation wird dabei durch die Firewalls der DMZ getroffen. Eine Unterscheidung durch mehrere Schnittstellen des Hosts ist im Docker-Szenario nicht vorgesehen und müsste manuell durch einen Paketfilter (Firewall) im Docker-Host realisiert werden.

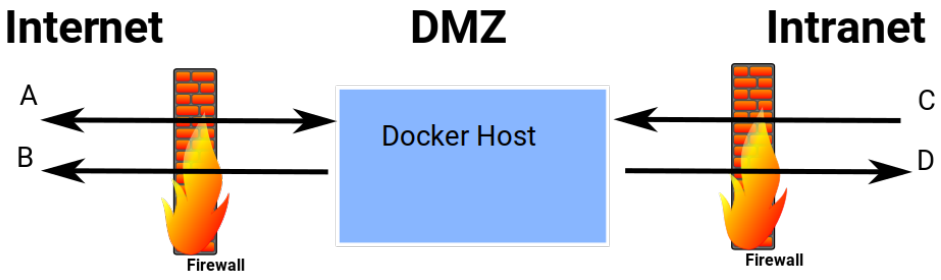


Fig. 1: DMZ: Netzwerkverkehr

Die Netzwerkverbindungen des Knotens (Docker Host) lassen sich wie folgt charakterisieren:

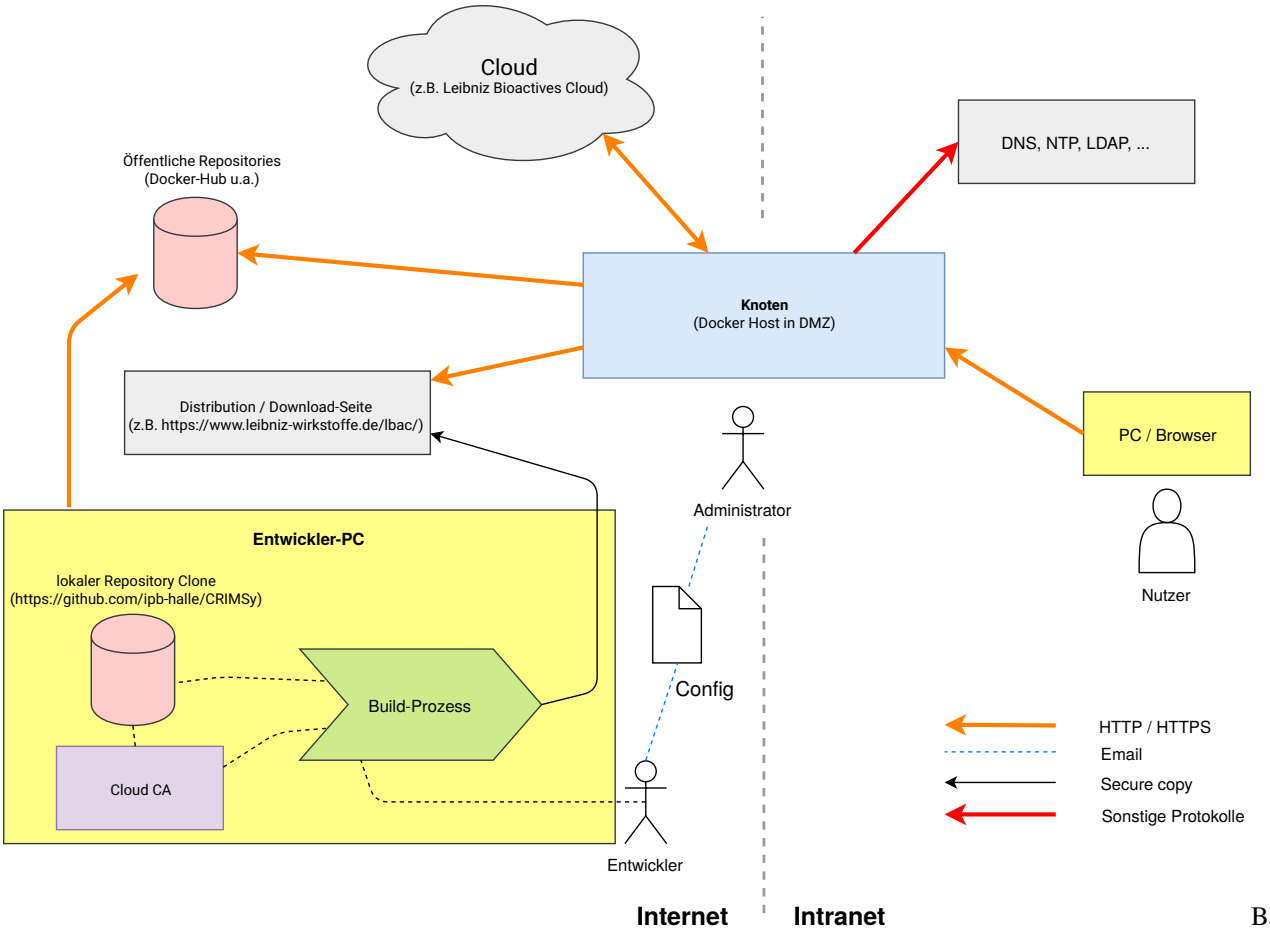
	Erläuterung
A	Der Knoten initiiert und empfängt HTTPS-Verbindungen zu bzw. auf Port 8443 für die Maschine-zu-Maschine-Kommunikation. Die Verbindungen sind durch gegenseitige zertifikatsbasierte Authentifizierung abgesichert.
B	Der Knoten initiiert HTTP- und HTTPS-Verbindungen zu mehreren Software-Repositories.
C	Der Knoten empfängt HTTPS-Verbindungen auf Port 443 von den Nutzern seines Instituts. Anfragen auf Port 80 (HTTP) werden auf Port 443 (HTTPS) umgeleitet. Sofern vom IT-Verantwortlichen ein "offizielles" Zertifikat zur Verfügung gestellt wurde, wird dieses verwendet. Ansonsten ist eine Fallback-Lösung implementiert, die allerdings zu Warnungen in den Browsern der Nutzer (bei einigen Browsern auch Unbenutzbarkeit) führt. Die Auswirkungen des Fallbacks sind auf das jeweilige Institut beschränkt. Unter Umständen fallen in die Kategorie C auch SSH-Verbindungen, die von IT-Verantwortlichen initiiert werden.
D	Diverse, zum Teil institutsspezifische Verbindungen (DNS, NTP, LDAP, DHCP), die nur teilweise durch Verschlüsselung abgesichert sind.

Abhängig von den lokalen Gegebenheiten können noch weitere Verbindungen ins Intranet existieren, die hier nicht betrachtet werden und nur beispielhaft und stichpunktartig aufgeführt werden:

- SNMP-Agenten für das Monitoring
- Backup-Agenten
- SAN-Verbindungen (iSCSI, . . .)

2.13 Bausteinsicht

Dieses Kapitel beschreibt das System als eine Sammlung von Bausteinen (Module, Komponenten, Klassen, Intefaces, usw.) und ihre Beziehungen, beginnend mit der obersten Ebene (Top-Down).



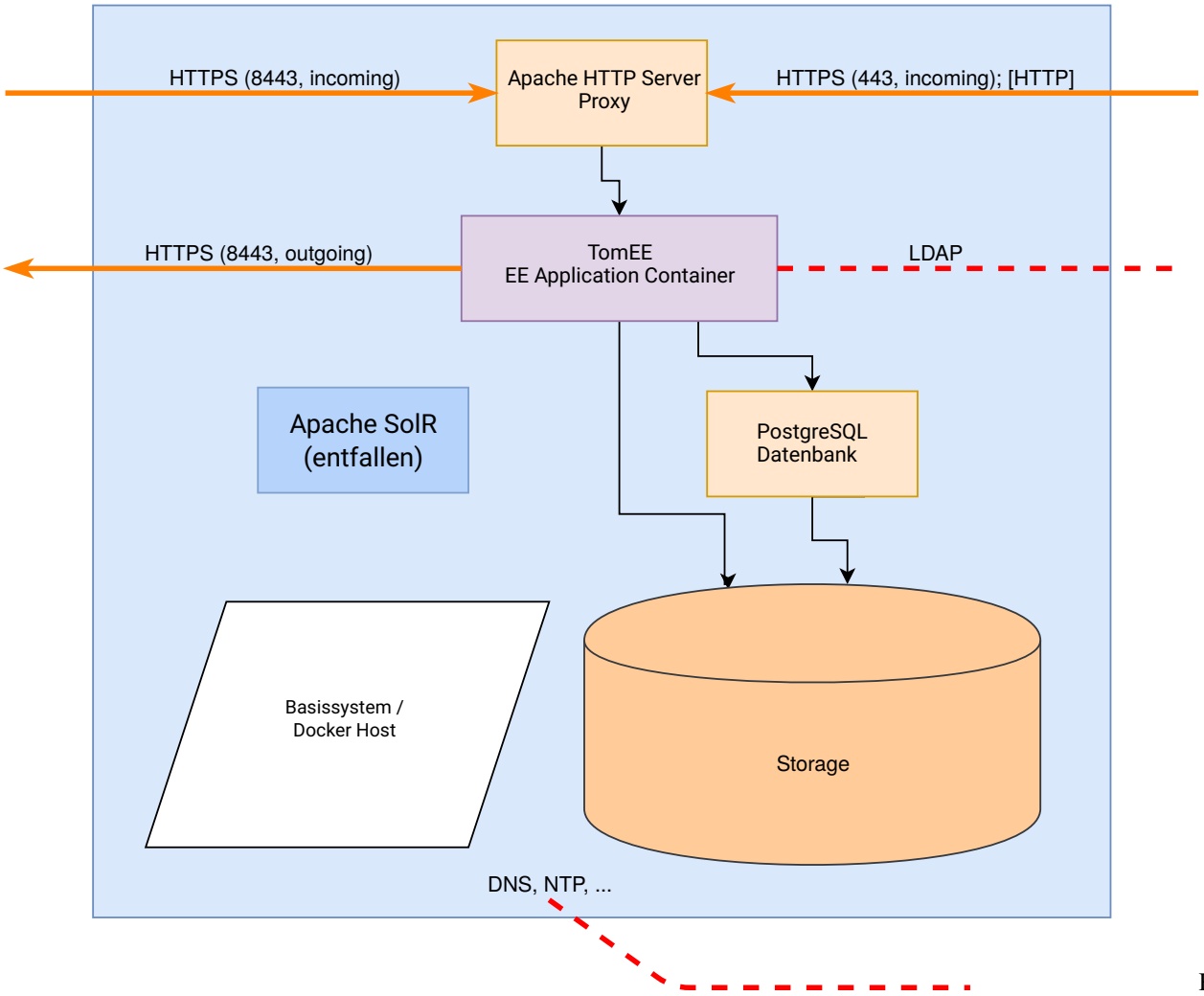
Bausteine

Die vorstehende Abbildung zeigt das vollständige Ökosystem der Leibniz Bioactives Cloud aus der Sicht eines Knotens. Die Pfeilrichtung zeigt immer an, aus von welchem Partner eine Verbindung aufgebaut wird. Insbesondere schließt die Darstellung die für die Initialisierung benötigten Komponenten mit ein. Bei den meisten auf dieser Ebene gezeigten Bausteinen handelt es sich um Standardkomponenten (z.B. Nutzer-PCs, normale Webserver, Software-Repositories usw.), die keiner weiteren Erläuterung oder Konfiguration bedürfen. In den folgenden Abschnitten sind daher nur die Bausteine

- **Knoten** Die Knoten sind die dezentralen Elemente der Leibniz Bioactives Cloud und stellen die Funktionen der Cloud zur Verfügung. Dieser Abschnitt vermittelt eine Übersicht, welche Komponenten in einem Knoten zusammenarbeiten. Auf die eigentliche Webanwendung wird aufgrund der besseren Übersicht in einem separaten Abschnitt eingegangen.
- **Entwickler-PC** Der Entwickler-PC erfüllt die Aufgabe, aus dem Quellcode im Source-Code-Repository und der vom Administrator bereitgestellten Konfiguration ein individualisieres Installationspaket zu erzeugen. Ausserdem ist auf diesem Gerät die Leibniz-Bioactives Cloud-CA lokalisiert, die eine zentrale Rolle bei der Absicherung der Cloud spielt.
- **Webanwendung** Die Webanwendung ist eine Java Enterprise Edition Webanwendung, die letztendlich das Zusammenspiel der Knoten in der Cloud als auch der Komponenten innerhalb des Knotens steuert und die den Anwendern die Services zur Verfügung stellt. Während die übrigen Komponenten weitgehend fremdgefertigt sind und daher als Blackbox betrachtet werden können, steckt der größte Teil der Entwicklungsarbeit in dieser Komponente.

2.13.1 Bausteinsicht: Knoten

Die Cloud-Knoten in ihrer Gesamtheit bilden die Cloud. Jeder Knoten setzt sich aus verschiedenen Bestandteilen zusammen, um die benötigten Funktionen ausführen zu können. Die Cloud-Knoten sind weitgehend gleichberechtigt; ein Knoten fungiert jedoch als Master-Knoten und verteilt die Liste der bekannten Knoten.



Knotenübersicht

Die Grafik zeigt eine schematische Darstellung eines Knotens mit den Beziehungen der Bestandteile untereinander und den Verbindungen nach außen. Verbindungen (insbesondere des Basissystems zu öffentlichen Repositorien), die nur einmalig während der Installation bzw. für Updates notwendig sind, sind nicht dargestellt. Die einzelnen Bestandteile werden nachfolgend beschrieben:

- Basissystem
- Storage
- PostgreSQL-Datenbank
- Proxy (Apache HTTP Server)
- TomEE (Web Application Server)

Darüberhinaus wird der im TomEE Web Application Server laufenden Java-Web-Anwendung ein eigener Abschnitt gewidmet.

Basissystem

Das Basissystem besteht aus einem Linux-Server in Minimalkonfiguration, der um folgende Softwarepakete (zuzüglich eventueller Abhängigkeiten) ergänzt wurde:

- Docker und Docker Compose (ab Docker Version 1.12)
- Dialog (NCurses-Dialoge für Shell-Skripte)
- cron
- ed (Zeileneditor)
- GnuZip
- M4 Makroprozessor
- OpenSSL
- sshd (nur falls Remote-Administration gewünscht wird)
- sudo
- tar
- sharutils (uuencode / uudecode)
- curl

Java ist auf dem Basissystem nicht erforderlich.

Ein Teil der Entwicklung der Leibniz Bioactives Cloud erfolgt unter OpenSUSE Leap 15.1 (Stand Mrz. 2020). Die Architektur ist jedoch weitgehend distributionsunabhängig; die Auswahl einer Distribution wird dem lokalen Administrator überlassen.

Init-System

Momentan unterstützt das Installationsskript vorrangig Systeme mit Systemd-Initsystem. Das traditionelle SystemV-Init kann ebenfalls noch ausgewählt werden, die Testabdeckung ist jedoch schlechter.

Härtung

Es wäre wünschenswert, das System durch Erweiterungen wie beispielsweise SELinux gegen Angriffe zur härten. Bislang wurden noch keine Aktionen in diese Richtung unternommen.

Performance

Die Docker-Performance und insbesondere der Speicherplatzbedarf sind von verwendeten Docker-Storage-Subsystem abhängig. Siehe nachfolgenden Abschnitt: Storage.

Storage

Beim Speichersubsystem des Knotens handelt es sich um ein einfaches Verzeichnis in einem normalen Linux-Dateisystem (XFS, Ext4, ...). Das PostgreSQL-Handbuch warnt ausdrücklich davor, den Speicher als Netzwerkdateisystem (z. B. NFS oder CIFS) zur Verfügung zu stellen. Unsere Empfehlung lautet daher, den Speicher als internen Plattenspeicher, DAS oder SAN-Speicher zur Verfügung zu stellen.

> **Tip:** *Um spätere Erweiterungen des Speicherplatzes zu erleichtern, sollte das Verzeichnis für die Cloud-Daten in einem separaten Dateisystem mit darunterliegendem Volume-Management angelegt werden. Über die Natur dieses Managements (LVM, ZFS, Feature des SAN-Systems) werden keine Annahmen getroffen. Für den Anfang kann zunächst auch in der Root-Partition des Knotens gearbeitet werden.*

> **Tip:** *Sollte NFS die einzig mögliche Speicheroption sein, so empfehlen wir dringend, das NFS-Volume mit den Optionen sync und hard zu mounten, um möglichst weitreichende Konsistenzgarantien zu erhalten.*

Der Speicherort der Cloud-Instanz wird während der Konfiguration festgelegt und in der Datei `~/.lbac` gespeichert. Während der eigentlichen Installation wird eine gleiche Datei (`/root/.lbac`) für den Nutzer root erzeugt, auf die durch die Init-Scripte zugegriffen wird. Durch die Datei wird die Variable `$LBAC_DATASTORE` definiert, die den absoluten Pfad zum eigentlichen Speicherort der Cloud-Instanz definiert.

On Disk Layout

Während der Konfiguration und anschließenden Installation werden in dem von der Variable `$LBAC_DATASTORE` bestimmten Speicherort folgende Verzeichnisse erzeugt:

- `backup/` Backups und Datenbank-Dumps
- `bin/` Für diverse Skripte (Installation, Backup, . . .).
- `data/` Die Daten der Cloud (Dokumente, PostgreSQL-Datenbanken usw.); unterhalb von /data werden entsprechend Unterverzeichnisse erzeugt:
  - `data/db/`
  - `data/ui/`
- `dist/` Die Softwaredistribution (Bibliotheken, WAR-Dateien, Konfiguration der Docker Container, usw.). Aus diesem Verzeichnis werden die Docker-Container instantiiert.
- `etc/` Das Verzeichnis `etc/` enthält Konfigurationsdaten, Zertifikate, Credentials usw. die bereits mit dem Konfigurationsskript erfasst bzw. erzeugt werden. Im Verlauf der Installation wird ein Großteil dieser Daten in das Verzeichnis `dist/etc/` kopiert.
- `tmp/` Das Verzeichnis tmp/ steht für temporäre Speicherung zur Verfügung. Während der Installation bzw. bei Updates wird dort das Installationspaket gespeichert. Ebenso speichert das Installationsskript das Logfile in diesem Verzeichnis.

Docker Storage

Die Standardkonfiguration des Docker-Storage-Subsystems (Verwendung eines Loop-Device) ist extrem ineffizient: Erstens wird der einmal von der Loop-Datei allozierte Speicher nicht mehr freigegeben, auch wenn die enthaltenen Volumes, Container und Images gelöscht werden. Dadurch können substanzielle Bereiche des Plattenspeichers dauerhaft blockiert werden. Zweitens ist die Verwendung eines Loop-Device die langsamste Variante. Es empfiehlt sich daher, die Standard-Einstellung des Docker-Storage-Treibers zu ändern. Am IPB liegen Erfahrungen mit dem overlay2-Treiber vor, bei dem Teile des Host-Dateisystems per bind-mount in die Container eingeblendet werden. Dazu muss der Docker-Daemon mit der zusätzlichen Kommandozeilenoption "-s overlay2" gestartet werden. Das IPB verwendet diese Einstellung in seinem Produktivsystem. Das darunterliegende Dateisystem ist XFS.

Einige unserer Testsysteme nutzen ZFS, das sich bislang ebenfalls als stabil und zuverlässig erwiesen hat. Erfahrungen mit BTRFS oder einem LVM thin pool im Zusammenhang mit Docker liegen am IPB bislang nicht vor. Im Übrigen wird auf die Docker-Dokumentation verwiesen.

PostgreSQL

Die Konfiguration des PostgreSQL-Containers ergänzt das offizielle Docker-Image um das Plugin pgChem:Tigress (zukünftig ist die zusätzliche Integration von RDKit geplant). Die Datenbankdateien werden im Datenverzeichnis des Knotens unterhalb von `$LBAC_DATASTORE/data/db` gespeichert. Sofern das Datenverzeichnis noch nicht initialisiert wurde, erfolgt die Initialisierung durch SQL-Skripte, die in der Build-Phase unterhalb von /docker-entrypoint-initdb.d abgelegt werden und die Initialisierung der Datenbank beim Hochfahren des Containers übernehmen. Die Initialisierungsdateien sind numerisch geordnet, so dass im Verlauf der Entwicklung anfallende Änderungen am Datenbankschema ggf. nachvollzogen werden können. Die Datenbank selbst speichert die Schema-Version in der Tabelle `info`, so dass bei Versions-Updates fehlende Schemaänderungen erkannt werden.

Aus der Webanwendung wird über Hibernate auf die Datenbank zugegriffen. Es hat sich gezeigt, dass aufgrund der Komplexität der Datenstrukturen und der vielfältigen Verknüpfungen (lokal und remote), innerhalb der Anwendung eine Zwischenschicht, nämlich Datatransferobjekte, eingeführt werden mussten.

Das Backup der Datenbank sowie Datenbank-Dumps vor Softwareupdates müssen extern organisiert werden. Beide Funktionen werden vom Skript `$LBAC_DATASTORE/bin/backup.sh` wahrgenommen, dass auch die Sicherung der übrigen Datenquellen (ui) übernimmt. Der Aufruf erfolgt entweder durch CRON oder während der Installation durch das Setup-Skript. Die Sicherung erfolgt in das Verzeichnis `$LBAC_DATASTORE/backup`.

Die Datenbank wird momentan nicht im ArchiveLog-Modus betrieben.

Debugging

Normalerweise ist die Datenbank nur innerhalb des privaten Netzes (`lbac_private`) des Container-Hosts erreichbar. Durch die Installations-Option "`--debug`" kann man erreichen, dass die Datenbank auch von außerhalb erreichbar ist. Dies stellt ein Sicherheitsrisiko dar und sollte im Produktivbetrieb daher unterbleiben.

Proxy

Der Baustein "Proxy" empfängt alle eingehenden Anfragen, übernimmt die Entschlüsselung und Zertifikatsprüfung sowie eine erste Zulässigkeitsprüfung der angefragten URL. In den Standardeinstellungen kommt ein Container mit Apache httpd 2.4 für diesen Zweck zum Einsatz. Externe Zugriffe werden vom Proxy auf folgende URLs beschränkt:

- `https://$LBAC_INTERNET_FQHN:8443/ui/rest/*`
- `https://$LBAC_INTERNET_FQHN:8443/ui/servlet/document`

Der Wert der Variable `$LBAC_INTERNET_FQHN` ergibt sich aus der Konfigurationsdatei `$LBAC_DATASTORE/etc/config.sh`.

Zertifikatsprüfung

Externe Verbindungen werden durch gegenseitige zertifikatsbasierte Authentifizierung abgesichert. Die Prüfung der Zertifikate der Gegenstellen erfolgen mittels einer Zertifikatssperrliste (Certificate Revocation List; CRL), die durch einen Cron-Job auf dem Docker-Host regelmäßig aktualisiert wird. Die Einrichtung des Cron-Jobs erfolgt während der Installation. Für interne Verbindungen findet auf dem Proxy keine solche gegenseitige Authentifizierung statt.

Externer Proxy

Optional kann anstelle des Proxy-Containers ein externer Proxy eingesetzt werden. Der Vorteil eines externen Proxies kann in erweiterten Sicherheitsfunktionen (z.B. Deep Packet Inspection) liegen, die z.B. die Verbreitung von Malware

oder den Einbruch in das System verhindern. Die Entscheidung für oder gegen einen externen Proxy wird normalerweise durch das Konfigurationsskript abgefragt. Es ist jedoch möglich, bei der Installation von dieser Voreinstellung abzuweichen. Das Installationsskript akzeptiert hierzu folgende Optionen:

- `--proxy`: Unabhängig von der Konfigurationseinstellung wird ein Proxy-Container eingerichtet
- `--noproxy`: Unabhängig von der Konfigurationseinstellung muß ein externer Proxy konfiguriert sein

Der externe Proxy muss jedoch manuell konfiguriert werden; das Installationsskript kann diese Aufgabe nicht erfüllen. Da die unterschiedlichsten Systeme zum Einsatz kommen können, kann das Handbuch keine konkrete Hilfe anbieten. Die nachfolgenden Ausschnitte aus den Konfigurationsdateien des Apache httpd 2.4 mögen als Orientierungshilfe dienen:

Interne Verbindungen

```
<VirtualHost _default_:443>
    ServerName LBAC_INTRANET_FQHN:443
    ServerAdmin LBAC_MANAGER_EMAIL
    DocumentRoot "/usr/local/apache2/htdocs"

    SSLEngine on
    SSLCertificateFile "/usr/local/apache2/conf/official_cert.pem"
    SSLCertificateKeyFile "/usr/local/apache2/conf/official_cert.key"

    <IfModule mod_proxy.c>
        ProxyPass         /ui/websocket/search  ws://ui:8080/ui/websocket/search
        ProxyPass         /ui                   http://ui:8080/ui
        ProxyPassReverse  /ui                   http://ui:8080/ui
    </IfModule>
</VirtualHost>
```

Externe Verbindungen

```
<VirtualHost _default_:8443>
    ServerName LBAC_INTERNET_FQHN:8443
    ServerAdmin LBAC_MANAGER_EMAIL
    DocumentRoot "/usr/local/apache2/htdocs"

    SSLEngine on
    SSLCertificateFile "/usr/local/apache2/conf/lbac_cert.pem"
    SSLCertificateKeyFile "/usr/local/apache2/conf/lbac_cert.key"

    # Certificate Revocation Lists (CRL):
    SSLCARevocationCheck chain
    SSLCARevocationPath "/usr/local/apache2/conf/crl/"

    SSLVerifyClient require
    SSLVerifyDepth 3

    SSLCACertificatePath "/usr/local/apache2/conf/crt/"

    <IfModule mod_proxy.c>
        ProxyPass         /ui/rest              http://ui:8080/ui/rest
        ProxyPassReverse  /ui/rest              http://ui:8080/ui/rest
        ProxyPass         /ui/servlet/document  http://ui:8080/ui/servlet/document
        ProxyPassReverse  /ui/servlet/document  http://ui:8080/ui/servlet/document
    </IfModule>
</VirtualHost>
```

Die vom Proxy benötigten Zertifikate und Schlüssel werden bei der Installation im Verzeichnis

`$LBAC_DATASTORE/dist/etc` abgelegt.

> **Achtung:** *Die Gültigkeit der Zertifikatssperrliste ist auf wenige Tage beschränkt. Die Zertifikatssperrliste (CRL) muss deshalb täglich aktualisiert werden. Die aktuelle Zertifkatssperrliste kann jeweils unter der URL `$LBAC_DISTRIBUTION_POINT/crl.pem` heruntergeladen werden. Der Wert der Variable `$LBAC_DISTRIBUTION_POINT` kann der Konfigurationsdatei `$LBAC_DATASTORE/etc/config.sh` entnommen werden.*

TomEE

Die spezifische Anwendungslogik der Leibniz Bioactives Cloud befindet sich größtenteils in der Webanwendung ui, die vom Baustein "TomEE Web Application Server" gehostet wird. Beim TomEE Web Application Server handelt sich um einen Docker-Container mit "Apache TomEE Plume", der seinerseits als JavaEE Container für die Webanwendung fungiert. Die Konfiguration des Docker-Containers stützt sich auf das offizielle TomEE-Plume Image (Stand 2020-03-12: `tomee:8-jre-7.1.0-plume`) und modifiziert dieses wie folgt:

- Hinzufügen einiger zusätzlicher Java-Bibliotheken, v.a. PostgreSQL-JDBC und Hibernate
- Löschen der Default-Webapps (manager, docs, . . .)
- Installation bzw. Ersetzen von tomee.xml und tomcat-users.xml
- Installation der Keystores und Truststores für zertifikatsbasierte Authentifizierung und verschlüsselte Kommunikation
- Erstellen eines unpriviligierten Nutzers tomee mit uid 8080 und einer Gruppe tomee mit gid 8080
- Installation der Leibniz Bioactives Cloud Webapp (ui.war)

Als Enterprise Java Web Application Server stellt TomEE mithilfe von Hibernate die transaktionskontrollierte Persistierung der Java-Objekte sicher. Im Laufe der Entwicklung hat sich gezeigt, dass die Datenbankcredentials dafür am besten in der globalen Konfigurationsdatei tomee.xml hinterlegt werden. Die DataSource-Konfiguration für Hibernate innerhalb der Webanwendung (über hibernate.cfg.xml) hat sich demgegenüber nicht bewährt.

2.14 Entwickler-PC

Auf dem Entwickler-PC sind mehrere Komponenten vereint:

- ein Clon des CRIMSy Git-Repository
- ein Entwicklungssystem (mindestens Maven und JDK)
- OpenSSL und das camgr.sh-Skript aus dem Repository zur Verwaltung der CA
- etwas Speicher zur persistenten Speicherung der Konfiguration und der Zertifikate

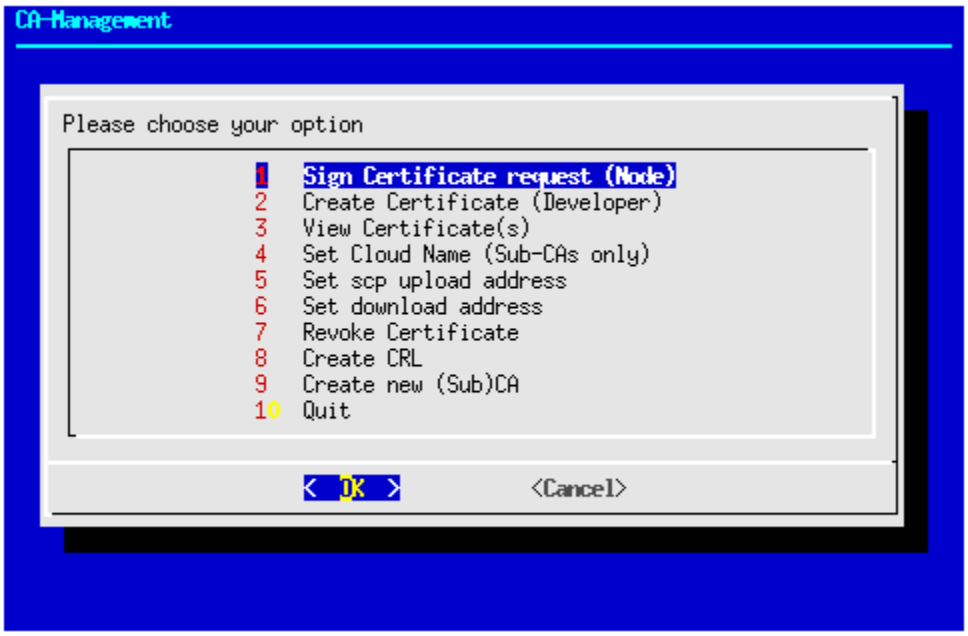
2.14.1 camgr.sh

Viele Funktionen der Cloud werden über Zertifikate abgesichert. Mit dem Skript `util/bin/camgr.sh` kann diese Public Key Infrastruktur (PKI) verwaltet werden. Die funktionen des Skripts sind im einzelnen:

- die Erzeugung der Root-CA
- die Erzeugung und Ausstellung von Zertifikaten für Sub-CAs
- die Ausstellung von Zertifikaten für die Maschine-zu-Maschine-Kommunikation
- die Ausstellung von Entwicklerzertifikaten für Codesigning und (Email-)Verschlüsselung

- die Anzeige der momentan gültigen Zertifikate
- der Widerruf von Zertifikaten und die Erstellung einer Certificate Revocation List
- die Festlegung des Code Distribution Point (d.h. Download-URL für die Installationspakete und Updates)
- die Erstellung von Java-Truststores mittels des keytool-Utility der JRE

Einige Funktionen des Scripts können interaktiv über Menü genutzt werden:



Hauptmenü Einige Funktionen (z.B. die Erzeugung von Zertifikaten für die Maschine-zu-Maschine-Kommunikation) sind auch über Kommandozeile zugänglich. Das Skript stellt dabei im wesentlichen einen Wrapper für OpenSSL dar und legt bestimmte Konfigurationsparameter für OpenSSL fest (z.B. Schlüssellänge, Digest-Algorithmus, Gültigkeitszeitraum, Key Usage usw.).

On Disk Layout

Das Skript `camgr.sh` erwartet seine Konfigurationsdaten im Verzeichnis `config/CA/` (Root-CA) bzw. `config/CLOUD_NAME/CA` (Cloud-CA), relativ zu einem Checkout des CRIMSy Git-Repositories:

```
+ config/
  + CA/
    + certs/
    + crl/
    + devcert/
    + lbac/
    + req/
    + ca.cfg
    + cloud.cfg
    + index.cloud
    + index.txt
    + cacert.key
    + cacert.passwd
    + cacert.pem
    + serial.txt
    + [...]
```

(continues on next page)

(continued from previous page)

```
+ util/
  + bin/
    + camgr.sh
```

Kommandozeile

Das Skript kann mit Kommandozeilenoptionen aufgerufen werden, um:

- eine Certificate Revocation List (CRL) zu erzeugen und sie zum Distributions-Server hochzuladen
- einen Certificate Request zu signieren
- einen Truststore zu erzeugen

Eine Übersicht über die verfügbaren Kommandozeilenparameter erhält man mit der Option `--help`.

2.15 Webanwendung

Die Webanwendung im TomEE-Container ihrerseits läßt sich wiederum in folgende Bestandteile gliedern:

- Konfigurationsdaten der Webanwendung (Verzeichnisse WEB-INF, META-INF, . . .)
- statische Ressourcen (Icons, CSS-Dateien, Javascript-Bibliotheken, . . .)
- Templates für das Rendering der (X)HTML-Seiten
- Geschäftslogik in Form von Java-Klassen

Zur Buildtime kommen hierzu noch Unit-Tests zur Qualitätssicherung. Alternativ besteht die Webanwendung in folgende Funktionskomponenten:

- Nutzer- und Berechtigungsverwaltung
- Verwaltung lokaler Ressourcen (Collections, . . .)
- Cloud-weite Synchronisierung der Ressourcen
- Daten-Upload
- Verteilte Suche (Simple Search, Word Cloud Search) und Document Delivery (Downloads)

Während einige Inhalte der Leibniz Bioactives Cloud innerhalb der teilnehmenden Institutionen frei zugänglich sein sollen, ist für andere Ressourcen (Dokumente, Daten, Services) eine Zugriffsbeschränkung notwendig. In jedem Fall darf die Änderung administrativer Einstellungen oder das Hochladen von Dokumenten nur für autorisierte Benutzer möglich sein. Um einerseits eine feingranulare Steuerung und andererseits eine bequeme Verwaltung zu ermöglichen, gibt es im System folgende Objekte bzw. Interfaces (für Details siehe Java-Dokumentation):

- Nutzer (*User*)
- Gruppen (*Group*); Gruppen können sowohl Nutzer als auch Gruppen als Mitglieder (*Membership*) haben.
- Access Controlled Objects (Interface *ACObject*); alle Objekte die Zugriffsbeschränkungen erweitern diese abstrakte Klasse
- Permissions (*enum ACPermission*) regeln die Art des erlaubten Zugriffs (Lesen, Anlegen, Ändern, Löschen, Eigentum Übertragen, . . .)
- Access Control Entries (*ACEntry*); legen die Zugriffsberechtigung für einen Nutzer bzw. eine Gruppe fest
- Access Control Lists (*ACList*); sammeln alle Access Control Entries für eine Access Control List.

Ein Großteil der Programm-Logik zu diesem Thema ist in der Java-Package *de.ipb_halle.lbac.admission* gebündelt.

2.15.1 Verwaltung von Nutzern und Gruppen

User und *Group* sind konkrete Implementierungen der abstrakten Klasse *Member*. Diese Struktur wurde gewählt, damit bei der Berechtigungsprüfung nicht zwischen Berechtigungen für Nutzer und Berechtigungen für Gruppen unterschieden werden muss. Außerdem war ein Ziel, verschachtelte Mitgliedschaften (*nested group memberships*) zu ermöglichen, wie sie von LDAP / Active Directory bekannt sind. Dies bedeutet, dass Gruppen ihrerseits Mitglieder von Gruppen sein können; die Mitgliedschaften einer Gruppe vererben sich auf ihre Mitglieder. Da die effektiven Mitgliedschaften eines Nutzers bei jeder Berechtigungsprüfung benötigt werden, muss die Bestimmung der effektiven Mitgliedschaften effizient sein. In komplexeren Szenarien kann eine verschachtelte Mitgliedschaft zudem auf verschiedenen Pfaden zustande kommen, bspw. wenn zwei Gruppen A und B Mitglied in einer Gruppe C sind. Ein Nutzer der sowohl Mitglied in Gruppe A als auch B ist, erbt die Mitgliedschaft in Gruppe C also doppelt. Eine genaue Buchführung über die effektiven Mitgliedschaften ist in solchen Fällen also unerläßlich.

Bei den Nutzern und Gruppen werden dabei aktuell vier verschiedene Typen unterschieden:

Typ	Beschreibung
BUILTIN	vordefinierte Nutzer und Gruppen, z.B. *Public Account*; keine Änderungen möglich
LOCAL	lokale Nutzer und Gruppen des Knotens
LDAP	lokale Nutzer und Gruppen aus einem LDAP-Verzeichnis
REMOTE	remote Nutzer, d.h. Nutzer eines anderen Knotens

Schwierigkeiten ergeben sich daraus, dass die Verwaltung dieser Objekte nicht nur auf dem lokalen Knoten erfolgen muss, sondern dass insbesondere die Information über Nutzer und Gruppen(mitgliedschaften) auch auf anderen Knoten der Cloud vorliegen muss. Zur Sicherstellung der Konsistenz gibt es daher Regeln, zwischen welchen Nutzer- bzw. Gruppentypen Mitgliedschaften eingerichtet werden können:

X Mitglied in Y?	Y='BUILTIN'	Y='LOCAL'	Y='LDAP'	Y='REMOTE'
X = *BUILTIN*	Auto	Nein	Nein	Nein
X = *LOCAL*	Auto	Ja	Nein	Nein
X = *LDAP*	Auto	Ja	Auto	Nein
X = *REMOTE*	Auto	Ja	Nein	Auto

Note: *Es ergibt z.B. keinen Sinn, einer LDAP-Gruppe weitere Mitglieder hinzuzufügen, da Mitgliedschaften dieser Gruppe ausschließlich im LDAP verwaltet werden sollen.*

Ein Administrator kann also nur zu Gruppen vom Typ LOCAL Mitglieder hinzufügen. Andere Mitgliedschaften werden entweder automatisch verwaltet oder sind prinzipiell ausgeschlossen.

Zur weiteren Vereinfachung ist die Authentifizierung von Nutzern auf den lokalen Knoten beschränkt. Ein Nutzer (egal ob knoteneigener Nutzer oder LDAP) kann sich nur an seinem Heimatknoten anmelden. Die Anmeldung von Nutzern externer Knoten wird nicht unterstützt. Falls in den Systemeinstellungen nichts anderes bestimmt wurde, kann der Knoten auch ohne Anmeldung genutzt werden. In diesem Fall werden alle Aktionen mit dem *Public Account* durchgeführt. Mit der Anmeldung eines Nutzers erfolgt die (asynchrone) Übertragung der Nutzer- und Gruppeninformationen zu den übrigen Knoten. Knoten, von denen mehrere Cloud-Mitgliedschaften bekannt sind, werden nur einmal kontaktiert. Gruppenmitgliedschaften werden dabei "flach", d. h. ohne Verschachtelung, übertragen. Soweit auf dem empfangenden Knoten weitere Mitgliedschaften bestehen, kann es jedoch wieder zu einer Verschachtelung kommen. Der sendende Knoten stellt sicher, dass keine sensitiven Informationen (Passworthashes) gesendet werden, der empfangende Knoten stellt sicher, dass keine lokal gemanagten Entitäten überschrieben werden. Die entsprechenden

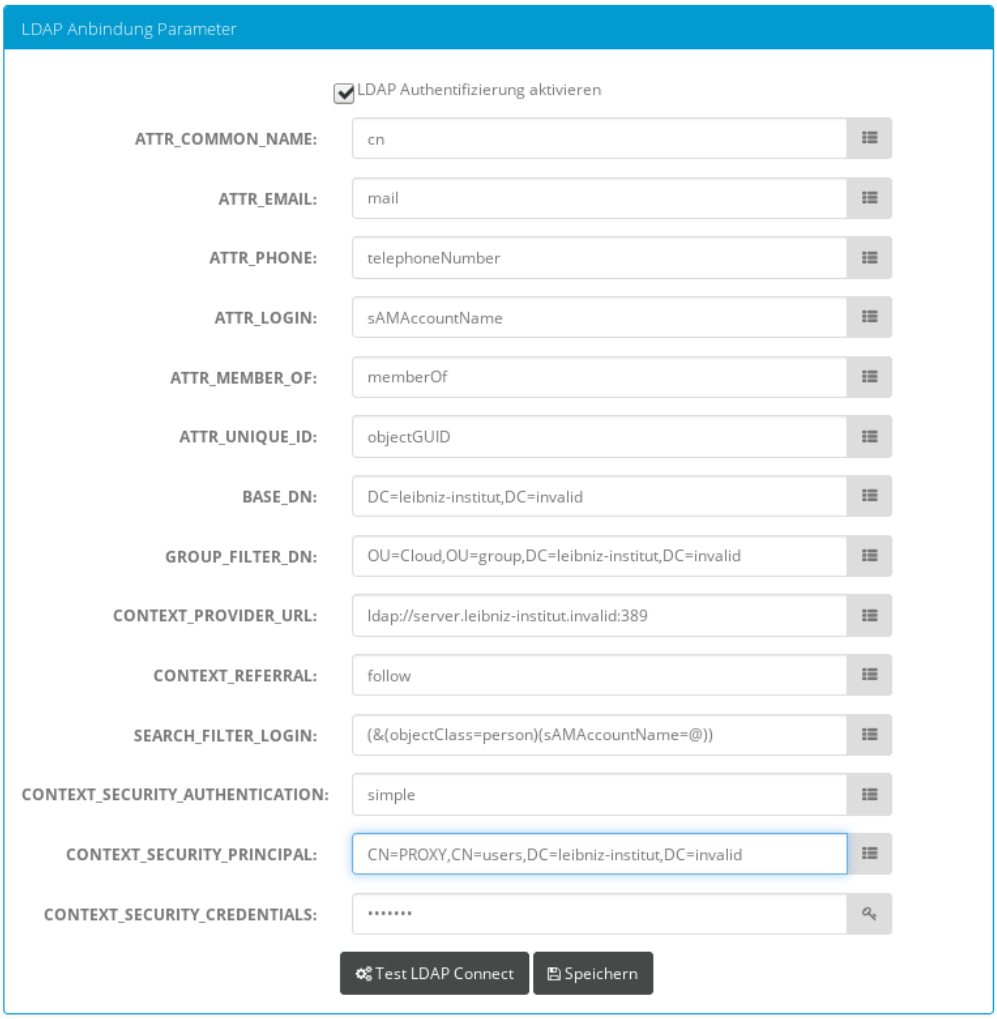
Entitäten implementieren hierzu das Interface *Obfuscatable*. Außerdem führt der empfangende Knoten eine Sicherheitsfilterung durch und akzeptiert nur Nutzer- oder Gruppenobjekte vom sendenden Knoten. Dadurch werden einerseits die Mitgliedschaftsregeln nach obiger Tabelle sichergestellt und andererseits unterbunden, dass Objekte von Drittknoten beeinflusst werden und die Komplexität der Nutzerverwaltung unbeherrschbar wird.

Ein schwerwiegendes Problem stellt das Löschen von Nutzern oder Gruppen dar, da andere Objekte - auch auf entfernten Knoten - direkt oder indirekt davon abhängen (referentielle Integrität). Zudem könnte ein entfernter Knoten zum Löschzeitpunkt vorübergehend nicht erreichbar sein. Aus diesem Grund werden Objekte nicht gelöscht sondern lediglich deaktiviert und unsichtbar geschaltet. Evtl. könnte dies über ein Ablaufdatum erfolgen.

Note: *Die 'obfuscate()'-Methode könnte entweder durch eine Annotation ersetzt werden oder als Implementierung eines Interfaces formalisiert werden. Die Nutzung der '@Transient'-Annotation bzw. des 'transient'-Keywords ist leider nicht zielführend. Das Problem liegt darin, dass die sensitiven Daten lokal persistiert und ggf. auch für lokale Browser / Administratoren serialisiert werden müssen, andererseits die lokale Einrichtung jedoch nicht in serialisierter Form verlassen dürfen.*

Anbindung von LDAP-Verzeichnissen

Ein Knoten der Cloud soll sich möglichst reibungslos in die vorhandene IT-Infrastruktur einer Betreiberinstitution integrieren. Dazu gehört die Anbindung an vorhandene Nutzerverzeichnisse (Active Directory oder andere LDAP-Server). Den lokalen Administratoren soll damit die erspart werden, eine weitere Datenbank mit Nutzerinformationen zu pflegen.



Die Verbindungsparameter werden über den gezeigten Dialog eingestellt. Für die Eingabefelder existieren kleine Tooltips, die kurze Hinweise zum jeweiligen Feld geben. Besonders erwähnenswert sind vor allem die Parameter *ATTR_UNIQUE_ID* und *GROUP_FILTER_DN*. Mit ersterem wird die sichere Zuordnung von Nutzern zu Ihrem Konto gewährleistet, auch wenn sich zwischenzeitlich das Login-Kennzeichen (z.B. wegen Namensänderung infolge Heirat, Scheidung, ...) ändert. Der zweite Parameter sorgt dafür, dass nur ein Teil der Gruppen aus dem LDAP-Verzeichnis in der Cloud sichtbar sind, nämlich jene, welche sich unterhalb der *GROUP_FILTER_DN* befinden. Dies ist aus Sicherheits- und Datenschutzgründen erforderlich, da den Cloud-Teilnehmern ansonsten die komplette Gruppenstruktur offenbart würde.

In Fällen, in denen das LDAP-Verzeichnis keine Information über Gruppenzugehörigkeiten zur Verfügung stellt, müssen die Parameter *ATTR_MEMBER_OF* und entsprechend auch *GROUP_FILTER_DN* leer bleiben. Die Verwaltung von Gruppenzugehörigkeiten von LDAP-Nutzern muss dann ausschließlich innerhalb von CRIMSy erfolgen.

2.15.2 Verwaltung von Berechtigungen

Access Controlled Objects implementieren das Interface *ACObject*, das diesen Objekten einen Besitzer und eine Access Control List zuweist. Access Controlled Objects (z.B. Collections) werden nur zur Laufzeit ausgetauscht und nicht in der Datenbank des Systems persistiert (siehe auch Kapitel "Laufzeitsicht"). Sie stehen dem jeweils angemeldeten Nutzer im Rahmen seiner Session zur Verfügung. Damit wird die Menge der zu synchronisierenden Daten begrenzt. Eine Persistierung würde ansonsten die Synchronisierung aller verbundenen Objekte (ACLs, Nutzer, Gruppen, ggf. Mitgliedschaften) erfordern. Dies würde die Komplexität massiv erhöhen und ist deswegen nicht erwünscht. Der

Abruf der Access Controlled Objects erfolgt, nachdem die Nutzerdaten übertragen wurden. Damit ist sichergestellt, dass die für den Nutzer zum aktuellen Zeitpunkt effektiven Berechtiungen angewendet werden. Im Übrigen verfolgt die Leibniz Bioactives Cloud das Konzept der eventuellen Konsistenz, d. h. Berechtigungsänderungen die während der Laufzeit einer Session vorgenommen werden, werden erst bei der nächsten Anmeldung berücksichtigt. Dies schließt ein, dass ein Nutzer Zugriff auf Ressourcen erlangt, für die Ihm der Zugriff kürzlich entzogen wurde.

Access Control Lists (ACLs, *ACList*) werden von Access Controlled Objects (*interface ACObject*) über ihre Id referenziert. Anders als bei Dateisystem-ACLs können mehrere Objekte eine ACL referenzieren. Das System stellt sicher, dass identische ACLs nicht mehrfach im System vorkommen. Um diese Aufgabe effizient zu erledigen, werden in Analogie zu *hashCode()* für jede ACL *permCode's gepflegt, um den Vergleichsaufwand zu minimieren. Jede ACL besteht aus einem oder mehreren Access Control Entries ('ACEntry*), die für eine Gruppe oder einen Nutzer die Berechtigungen festlegen. Es werden lediglich positive Berechtigungen verwendet, die Berechtigungen der einzelnen ACEntries wirken additiv. Eine Besonderheit ist ein *ACEntry* für den *BUILTIN*-Benutzer *Owner Account*: dieser gibt die Berechtigungen für den Besitzer eines Objekts an.

2.16 Job-Verarbeitung

Das Sicherheitskonzept von CRIMSy sieht vor, dass ein Knoten in einer demilitarisierten Zone betrieben wird. Ein Knoten kann

- Druckjobs für Barcode-Drucker, die aufgrund besonderer Anforderungen an Treiber und Label-Formaten nicht über den gewöhnlichen Weg als PDF-Datei abgewickelt werden können
- Compute-Jobs für aufwendige Berechnungen (z.B. Quantenchemie, künstliche Intelligenz, ...), die die Ressourcen des lokalen Knotens hinsichtlich Rechenaufwand oder Speicherbedarf sprengen

CRIMSy sieht in diesem Falle vor, dass ein zusätzlicher Dienst (*CRIMSy Agency*) regelmäßig beim Knoten anfragt, ob neue Aufträge vorhanden sind. CRIMSy stellt hierzu eine REST-Schnittstelle bereit, die sich jedoch in einem Punkt von den übrigen im System verwendeten REST-Schnittstellen unterscheidet. Normalerweise werden die REST-Schnittstellen über die CRIMSy-PKI abgesichert, d.h. die Gültigkeit der Zugriffe wird mittels der privaten und öffentlichen Schlüssel der Knoten überprüft. Im Kontext der Jobverarbeitung verbietet sich die Benutzung des privaten Schlüssels eines Knotens, da dies der unkontrollierte Verbreitung des privaten Schlüssels Vorschub leisten würde. Die Nutzung zusätzlicher Schlüsselpaare wiederum würde massiv die Komplexität des Vorhabens erhöhen. Da davon ausgegangen wird, dass der Dienst CRIMSy Agency in einer Umgebung mit moderatem Risiko betrieben wird, findet die Absicherung dieser Schnittstelle daher über ein *shared secret* statt. In CRIMSy kann das *shared secret* im Menü **Systemeinstellungen** konfiguriert werden.

CRIMSy Agency selbst ist sehr einfach gehalten, um eine Anpassung an örtliche Gegebenheiten vornehmen zu können. Vorgesehen ist beispielsweise die Anbindung an CUPS für Druckaufträge. Das Programm soll als Dämon-Prozess ausgeführt werden und fragt den Server in regelmäßigen Abständen ab (derzeit alle 5 Sekunden). Für jeden Job wird ein Shellskript aufgerufen. Das Shellskript erhält den Jobtyp (*PRINT* oder *COMPUTE*), dem Namen der Queue und die Job-Id als Kommandozeilenparameter. Die eigentlichen Daten werden dem Shellskript über Standardeingabe übermittelt. Die Ergebnisse bzw. Ausgaben eines Jobs sollen vom Shellskripts über Standardausgabe an CRIMSy Agency übermittelt werden. Die eigentliche Anpassung an lokale Gegebenheiten findet im Shellskript statt. Die mitgelieferte Vorlage kann nach belieben angepasst und modifiziert werden.

2.17 Laufzeitsicht

Die dezentrale Organisation der Leibniz Bioactives Cloud bedingt eine hohe Nebenläufigkeit und damit auch Komplexität der ablaufenden Prozesse. Die einzelnen Prozesse können dabei nach folgenden Kriterien kategorisiert werden:

- Zeitpunkt der Ausführung: dauerhaft oder nur während Installation bzw. Update
- Ort der Ausführung: lokaler Knoten, Remote-Knoten oder andere

- automatische Ausführung oder auf Nutzeranforderung

2.17.1 Cookies und lokaler Speicher

Die Interaktion des Nutzers mit "seinem" Knoten erfordert regelmäßig die Zusammenfassung seiner Aktionen zu einer Session. Da HTTP / HTTPS zustandslose Protokolle sind, muss diese Zuordnung über Session-Cookies realisiert werden. Darüberhinaus werden Cookies bzw. lokaler Speicher dazu verwendet, die Konfiguration des Nutzerinterface (speziell Tabellen-Layout und -Sortierung) zu speichern. Gemäß https://ec.europa.eu/ipg/basics/legal/cookies/index_en.htm ist für diese technischen Cookies keine Einwilligung der Nutzer notwendig. Im Einzelnen handelt es sich um folgende Cookies:

Domain	Cookie / Speicherschlüssel	Zweck
Cloud Host	*JSESSIONID*	Sessionverwaltung
Cloud Host	*serverTime*	Sessionverwaltung
Cloud Host	*sessionExpiry*	Sessionverwaltung

2.17.2 Prozessübersicht

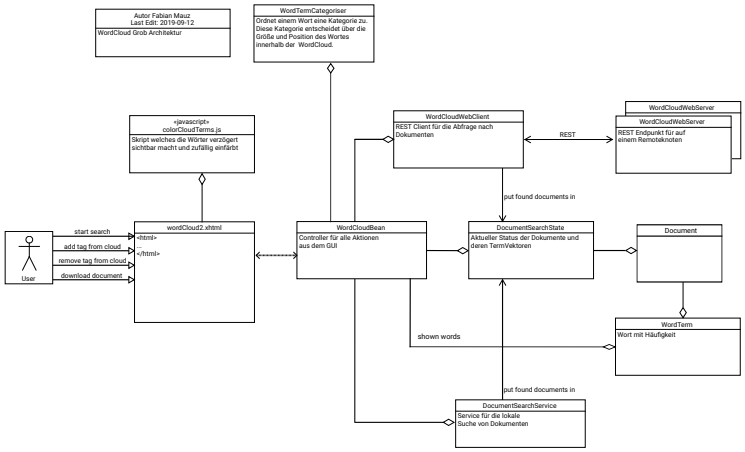
Die nachfolgende Liste gibt eine Übersicht über die zur Laufzeit ablaufenden Prozesse. Zu jedem Prozess ist jeweils eine Kurzbeschreibung gelistet. Ausführliche Beschreibungen folgen im weiteren Verlauf in separaten Abschnitten.

- **Konfiguration:** Die Konfiguration dient der Erfassung von Informationen, die für die Installation und den Betrieb der Cloud notwendig sind. Für eine ausführliche Beschreibung des Konfigurationsprozess aus Administrator-Sicht wird auf das Handbuch "Konfiguration und Installation" verwiesen.
- **Installation:** Im Anschluss an die Konfiguration eines Knotens ist die Installation ein (i.d.R.) automatisch ablaufender Prozess, der lediglich vom Administrator angestoßen wird. Die Auslieferung von Softwareupdates erfolgt ebenfalls über diesen Prozess.
- **Backup:** Durch das Backup wird der lokale Knoten gegen Datenverlust abgesichert. Außerdem wird das Backup während des Updateprozess benötigt, falls es bei einem der Bausteine (inbes. PostgreSQL-Datenbank) zu einem Releasewechsel kommt. Sicherungskopien innerhalb des Knotens werden automatisch von einem Backup-Prozess angefertigt, der während der Installation eingerichtet wird. Die Einrichtung eines externen Backups ist die Aufgabe des lokalen Administrators.
- **Suchanfragen:** Die Inhalte der vorhandenen Dokumente lassen sich durch Suchabfragen duchsuchen. Dazu wird auf jedem Node ein Index der lokalen Dokumente erzeugt.
- **Wordcloudsuche:** Die Wordcloudsuche unterscheidet sich von einer normalen Suche durch die Visualisierung der prominentesten Stichworte, die in den Ergebnisdokumenten vorkommen.
- **Upload von Dokumenten:** Der Upload von Dateien wird durch eine Javascript - Komponente realisiert
- **Download von Dokumenten:** Eine erfolgreiche Suchanfrage liefert eine Liste von Dokumenten, die auf verschiedenen Knoten der Cloud beheimatet sein können. Über einen Download-Link und entsprechende serverseitige Schnittstellen können einzelne Dokumente heruntergeladen werden.
- **Synchronisierung von Netzwerkressourcen:** Da es sich bei der Cloud um eine verteilte Infrastruktur handelt, müssen die Knoten Informationen über sich teilen und verbreiten.
- **Authentifizierung, Nutzerverwaltung:** Diese Prozesse ermöglichen die Identifizierung von Nutzern und bilden die Basis für die feingranulare Verwaltung und Gewährung von Zugriffsrechten.
- **Berechtigungsverwaltung:**
- **Collection-Verwaltung:** Das Collection Management ermöglicht das Anlegen, Ändern und Löschen von Collections und die Verwaltung der Zugriffsrechte.

Suchanfragen

Eine Suchanfrage wird parallel an alle verfügbaren Nodes der Cloud verteilt und ausgeführt. Jeder Node liefert als Ergebnis eine Liste mit Dokumenten, die im Browser Frontend zusammengeführt und dargestellt wird. Die Liste lässt sich anschliessend nach Spalten sortieren.

Wordcloud-Suche

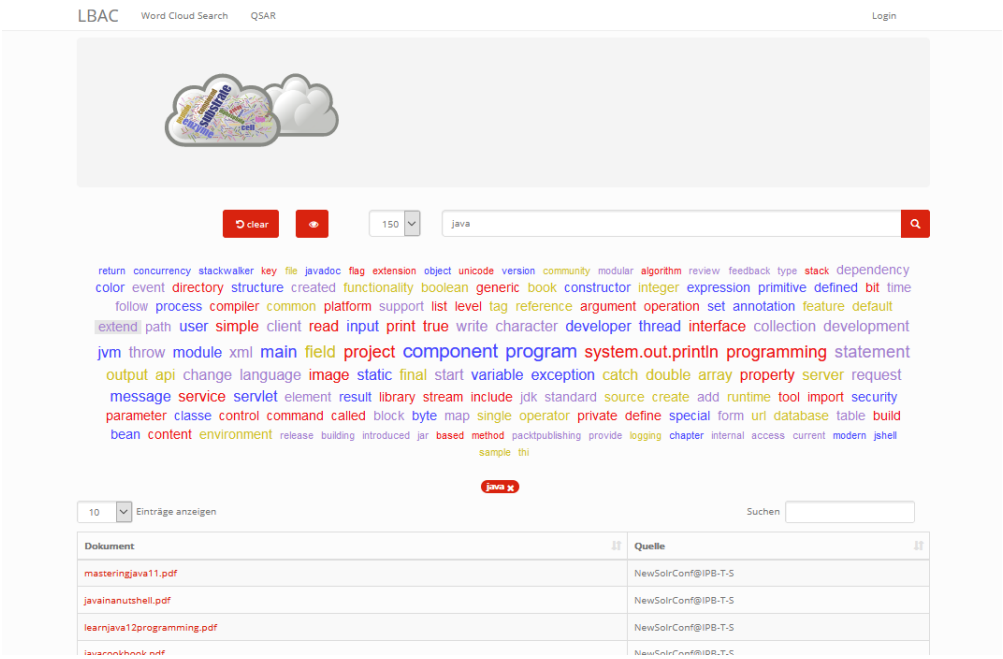


Eine Word Cloud ist eine Methode zur Informationsvisualisierung, bei der eine Liste aus Schlagwörtern, oft alphabetisch sortiert, flächig angezeigt wird, wobei einzelne unterschiedlich gewichtete Wörter größer oder auf andere Weise hervorgehoben dargestellt werden.

Dokumente enthalten je nach Thema fach- und themen-spezifische Begriffe. Diese Begriffe tauchen in einer gewissen Häufigkeit, auch in Kombination mit anderen Begriffen, im Dokument auf. Daraus lassen sich im gewissem Maße Rückschlüsse auf den Inhalt der Dokumente ziehen. Dokumente mit ähnlichen Begriffen und ähnlicher Häufigkeit behandeln eventuell gleichartigen Themen oder überschneiden sich wesentlich. Das soll als Anregung für den Nutzer dienen, um interessante Dokumente für sich zu finden.

Im Projekt LBAC soll eine *dynamisch* erzeugte Word Cloud als intuitive Möglichkeit dienen, um durch eine beliebige Menge von Dokumenten mit Hilfe von Schlagworten quasi zu navigieren. Jedes ausgewählte Schlagwort verfeinert/verändert die Ergebnismenge der Dokumente und liefert eine neue Word Cloud. Die Begriffe werden dabei logisch UND verknüpft. Ausgewählte Begriffe werden aus der Ergebnismenge entfernt (dive in). Der Nutzer hat die Möglichkeit bereits gewählte Suchbegriffe wieder zu löschen und sich somit rückwärts zu bewegen (dive out).

Frontend (View)



Zur Darstellung der dynamischen Wordcloud wird die Komponente <p:tagCloud> aus der PrimeFaces Bibliothek genutzt. Aktuell werden bei einer initialen Suche die lokalen als auch alle remote Anfragen synchron durchgeführt und anschliessend die Cloud gerendert. Eine asynchrone Variante analog zu Dokumetensuche ist derzeit nicht implementiert.

Jedes Schlagwort ist als klickbarer Link dargestellt. Ein Klick auf das Schlagwort erzeugt eine neue Suchanfrage. (dive in) Die Reihenfolge der geklickten Wörter wird in einer Tag-Liste dokumentiert. Die Tag-Liste ist klickbar. Ein beliebiges Tag kann hiermit gelöscht werden. (dive out). Die Word Cloud wird nach jeder Interaktion mit der neuen Ergebnismenge gezeichnet. Die Anzahl der Wörter wird begrenzt (top words) und kann über eine Auswahl (50,100,150) gewählt werden.

Controller (Klasse WordCloudBean)

Bei der Klasse handelt es sich um den Controller zum GUI. Folgende Methoden können vom Client angestoßen werden:

- *startSearch( )* - Startet die Suche mit den aktuellen Wörter in der Suchleiste
- *onSelect(SelectEvent event)* - Zeichnet eine neue Cloud mit den aktualisierten Terme und filtert nicht mehr relevante Dokumente aus
- *clearCloudState ( )* - Löscht alle Terme, die aktuelle Cloud sowie alle angezeigten Dokumente in der View
- *toggleWordCloudVisibility ( )* - Macht die Cloud sichtbar/unsichtbar
- *removeTag(String tag)* - Entfernt den übergebenen Tag, zeichnet die Cloud neu und aktualisiert die Dokumente

Darstellung von Wörtern in der Cloud

Die Größe eines Wortes in der Cloud wird über dessen Kategorie bestimmt. 5 Kategorien existieren aktuell: *HIGHEST*, *HIGH*, *MEDIUM*, *LOW*, *LOWEST*. Die Kategorie eines Wortes wird über einen Algorithmus bestimmt. Die

Einfärbung geschieht per Zufall aus 3 Farben.

WordTermCategoriserComplex

Bei diesem Algorithmus wird werden Wörter in hohe Kategorien einsortiert, welche die Anzahl der Dokumente nach Auswahl des Wortes möglichst halbieren und dennoch eine hoche Frequenz aufweisen.

$$score_T = (0.5 - | {D_T over D} - 0.5|) * F_T$$

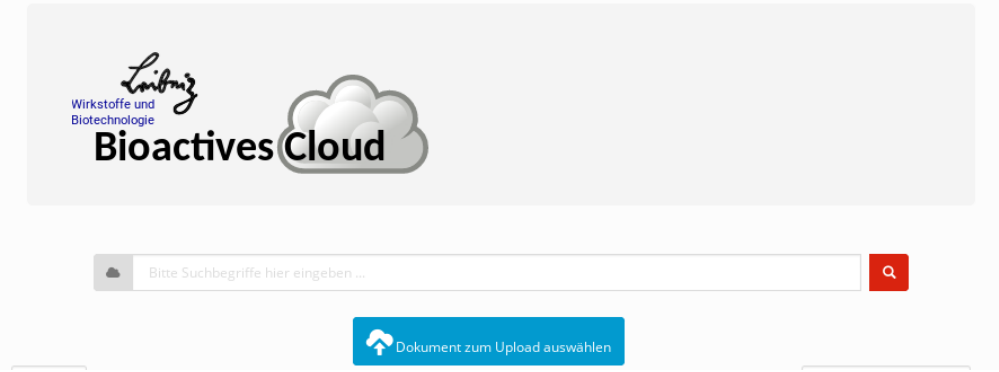
Variable	Bedeutung
D_T	Dokumente mit dem Begriffsterm T
F_T	Anzahl des Begriffs T in allen Dokumenten
D	Gesamtanzahl der Dokumente

Ausblick

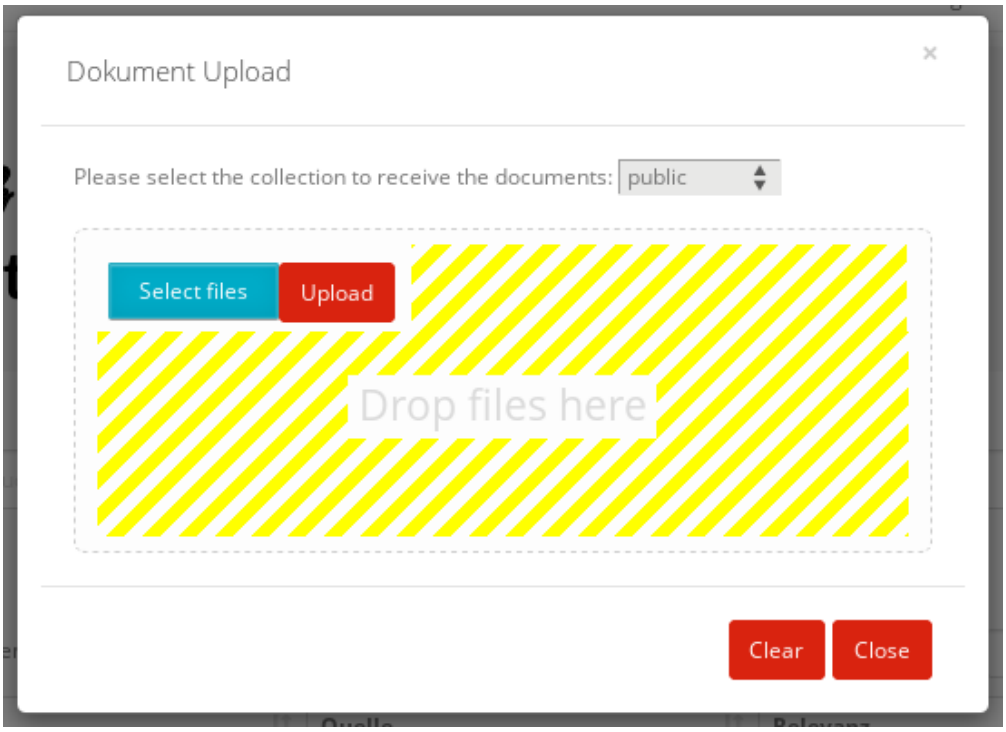
Natural Language Prozessing (NLP): Im September 2020 ist die Komponente Apache Solr entfallen, da die Dokumentenindexierung auf anderem Weg effizienter gestaltet werden konnte. Trotzdem ist die Ergänzung und Erweiterung der Analysepipeline weiterhin Teil der Roadmap (zugegebenermaßen mit niedriger Priorität). Sobald ausreichende Entwicklungskapazitäten zur Verfügung stehen, würden wir unser Projekt gern um die Anwendung von Ontologien, die Analyse von Tabellen und chemischen Formeln, den Einsatz künstlicher neuronaler Netze zur semantischen Textanalyse usw. erweitern.

Upload von Dokumenten

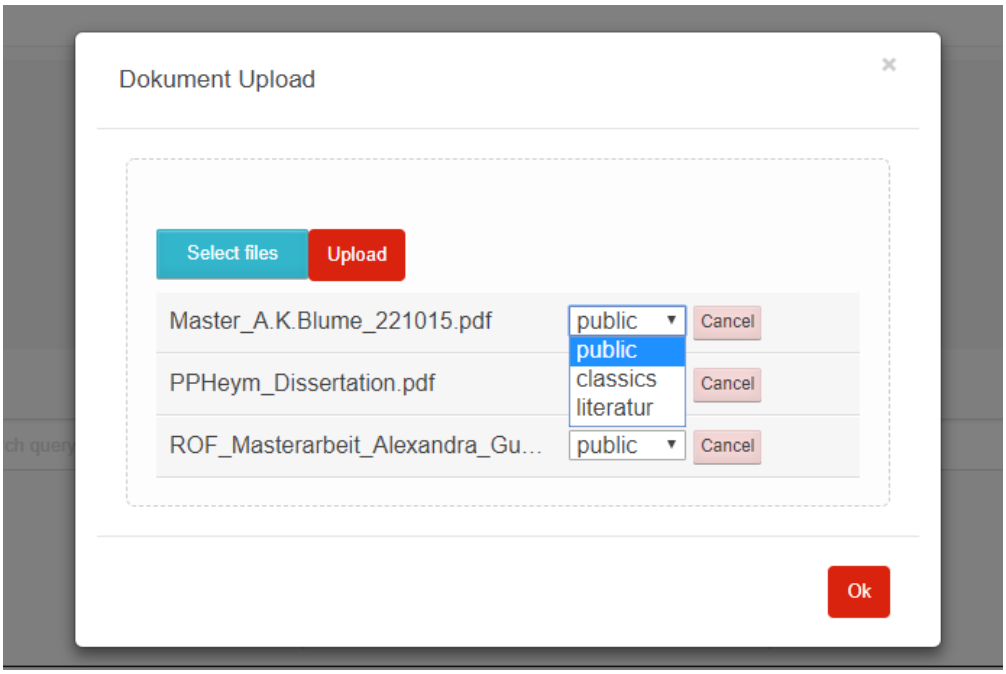
Der Upload von Dokumenten erfolgt durch Einsatz der Javascript-Bibilothek *Fine Uploader*. Diese ermöglicht ein intuitivies Hochladen von Dokumenten mit einer grafischen Oberfläche. Beim Upload mehrerer Dateien erfolgt dies parallel mit max. 3 Dateien. Ein anonymes Hochladen von Dokumenten (d.h. als *Public User*) ist nicht möglich.



Das Hochladen von Dokumenten kann nur in lokale Kollektionen erfolgen und setzt Schreibrechte (*PERM_CREATE*) für mindestens eine lokale Kollektion voraus. Ist Schreibberechtigung für mindestens eine Kollektion gegeben, so erscheint der oben abgebildete blaue Button auf dem Hauptformular.



Die Auswahl der zum Hochladen vorgesehenen Dokumente wird im oben dargestellten modalen Dialog getroffen. Dabei gibt es zwei Möglichkeiten, Dateien zum Hochladen auszuwählen: Die klassische Variante über den blauen *Select files*-Button und einen Dateisystemdialog des Browsers oder per Drag&Drop aus einem Datei-Explorer (klicken und ziehen mit der Maus) in die Dialogfläche. Zur besseren Illustration ist die Zielfläche in der obigen Abbildung gelb schraffiert (weiß oder hellgrau im Original). Es ist möglich und sinnvoll, in gleichzeitig mehrere Dateien zum Hochladen auszuwählen.



Für jede der ausgewählten Dateien kann die Zielkollektion bestimmt werden. Außerdem kann der Vorgang für einzelne Dateien mit *Cancel* abgebrochen werden. Ist die Liste vollständig, kann das Hochladen durch klicken des *Upload*-Buttons gestartet werden. Der Fortschritt wird durch grüne Balken und Prozentwerte angezeigt. Es werden max. 3

Dateien parallel hochgeladen.

Der Upload einer Datei ist vollständig, wenn der Hintergrund grün dargestellt wird und ein *Delete*-Button angezeigt wird. Evtl. auftretende Upload-Fehler werden rot markiert und mit einer Fehlermeldung versehen.

Die Drag-and-Drop Funktion ermöglicht einen Massen-Upload (first upload) von Dokumenten. Man kann z.B. die Windows Datei-Suche nutzen, um z.B. pdf-Dokumente in einer Verzeichnis-Struktur zu finden. Die Ergebnismenge der Suche läßt sich per Drag adn Drop direkt in den Upload übertragen. Das wurde mit ca. 4000 Dokumenten erfolgreich getestet.

Auch nach dem vollständigen Upload besteht noch die Möglichkeit, eine Datei noch nachträglich zu löschen. Dazu wird hinter der Datei ein *Delete*-Button angezeigt. Diese Möglichkeit besteht nur solange der Dialog nicht geschlossen wird.

Download von Dokumenten

Als Ergebnisse ihrer Suche erhalten die Nutzer eine Liste von Links, mit denen sie Zugang zu den einzelnen Dokumenten erhalten können. Beim Download von Dokumenten (vom lokalen Knoten und aus der Cloud) muss das System die Einhaltung verschiedener Randbedingungen garantieren:

- Jeder Nutzer erhält nur Zugriffsrechte auf Dokumente, für die er auch Leseberechtigung besitzt
- Ein Massendownload (das Abgreifen der kompletten Daten eines Knotens oder einer Collection) sind nicht vorgesehen
- Der Download-Link zeigt immer auf den lokalen Knoten, auch wenn das Dokument auf einem entfernten Knoten beheimatet ist. Niemals wird eine direkte Verbindung zwischen Browser und entferntem Knoten hergestellt.
- Der Download-Link hat eine begrenzte Gültigkeit, so dass der Nutzer ihn nicht sinnvoll weitergeben kann. Dies ist auch notwendig, damit der Eigentümer einmal erteilte Rechte auch wieder entziehen kann.

Technisch wird der Download über ein Servlet und *GET*-Requests realisiert. In der URL sind neben dem Servlet-Endpunkt folgende Parameter kodiert, wobei eine modifizierte Base64-Kodierung für die Parameter verwendet wird:

Parameter	Bedeutung
node_id	Zielknoten. Der lokale Knoten des Nutzers entscheidet hiermit, ob er die Anfrage selbst bearbeitet oder weiterleitet.
collection_id	Collection in der das angefragte Dokument gespeichert ist
Zeitstempel	Über den Zeitstempel wird gesteuert, wie lange ein Link gültig bleibt
Dateiname	Name der angefragten Datei (ein MD5-Hash)
Nonce	Eine Zufallszahl
Signatur	Signatur mit dem privaten Schlüssel des Zielknotens zur Gültigkeitsprüfung

Bei der Anforderung eines lokalen Dokuments liefert wird zunächst die Gültigkeit der Anforderung geprüft. Anschließend wird das betreffende Dokument über Port 443 an den Nutzer ausgeliefert. Bei Anforderung eines Dokuments von einem entfernten Knoten, wird die Anforderung zum entfernten Knoten (Port 8443) weitergeleitet. Der lokale Knoten übernimmt dann die Funktion einer Relais-Station und leitet die Antwort des entfernten Knotens an den Nutzer weiter.

Synchronisierung von Entitäten

Bei der Synchronisierung von Entitäten ist zwischen zwei Fällen zu unterscheiden:

- Die synchronisierten Entitäten werden in der Datenbank persistiert. Da die Integrität der Datenbank, z.B. bei Ausscheiden eines Knotens, schwierig sicher zu stellen ist, soll die Zahl der in der Datenbank persistierten Remote-Entitäten minimiert werden. Daher werden momentan nur folgende Remote-Entitäten persistiert:
  - Informationen über den eigenen und entfernte Knoten sowie ihre Zuordnung zu ein oder mehreren Clouds: *Node* und *CloudNode*
  - Nutzer, Gruppen und Gruppenmitgliedschaften
- Die synchronisierten Entitäten der entfernten Knoten stehen nur temporär innerhalb einer Session zur Verfügung. Nur lokale Entitäten werden in der Datenbank persistiert. Zukünftig hinzukommende Entitäten werden voraussichtlich ebenfalls in diese Klasse fallen.
  - Kollektionen
  - Topics und Postings im Nutzerforum

Die Abwicklung der Synchronisierung erfolgt über REST-Endpoints. Die Verantwortlichkeit liegt üblicherweise in den Klassen ... *WebClient* bzw. ... *WebService*. Die meisten dieser REST-Endpoints sind über einen Authentifizierungsmechanismus auf der Basis "Zeitstempel, Nonce, verschlüsselter Hash" geschützt. Aufgrund eines Henne-Ei-Problems kann dieses Verfahren für *Nodes* / *CloudNodes* nicht angewendet werden. Außerdem findet die Synchronisierung in der Regel asynchron statt, um das Nutzererlebnis nicht durch blockierendes Warten zu trüben. Die Steuerung der asynchronen Prozesse erfolt üblicherweise durch ... *Orchestrator*-Klassen.

Nodes and CloudNodes

Über die Entitäten bzw. Klassen *Node* und *CloudNode* werden Teilnehmerinformationen ausgetauscht. Die Klasse *Node* speichert vor allem den Namen der Institution und die Internetadresse, unter der der Knoten erreichbar ist. Die Klasse *CloudNode* repräsentiert eine Cloud-Mitgliedschaft und speichert den Rang des Knotens (gewöhnlicher Knoten oder Master-Knoten), den öffentlichen Schlüssel für Authentifizierungszwecke sowie Informationen zur aktuellen Knotenerreichbarkeit. Die Synchronisierung dieser Informationen erfolgt Timer-gesteuert, indem jeder Knoten sich bei den Master-Knoten aller Clouds meldet. Bei der Meldung werden die Daten des *Node*-Objekts, der Cloud-Name und der öffentliche Schlüssel zum Master übertragen. Als Antwort erhält der Knoten eine Liste mit allen beim Master bekannten Knoten für die jeweilige Cloud. Es ist sichergestellt, dass ein Knoten keine Informationen über Knoten aus anderen Clouds erhält. Ein Master-Knoten stellt selbst keine Anfragen, es sei denn, er ist als gewöhnlicher Knoten Mitglied in einer weiteren Cloud.

Nutzer, Gruppen und Mitgliedschaften

Die Übertragung und Persistierung von Nutzern, Gruppen und Mitgliedschaften ist notwendig, damit einem Nutzer auf entfernten Knoten Zugriffsrechte eingeräumt werden können. Die Übertragung erfolgt dabei nicht zeitgesteuert sondern als Annoncierung bei der Anmeldung eines Nutzers. So ist sichergestellt, dass alle Knoten über die aktuellen Informationen zu den Gruppenmitgliedschaften verfügen. Änderungen werden so schnellstmöglich wirksam. Bei der Synchronisierung ist zu beachten, dass sensitive Informationen (v.a. Passwort-Hashes) vor der Übertragung entfernt werden, zumal die Passwort-Hashes auf den entfernten Knoten nicht benötigt werden. Da bei der Arbeit mit Nutzer-Objekten personenbezogene Daten verarbeitet werden, müssen die Regeln der DSGVO (u.a. das Recht auf Anonymisierung) beachtet werden.

Nichtpersistente Entitäten

Aktuell werden Kollektionen (*de.ipb_halle.lbac.entity.Collection*), Topics (Diskussionsfäden, Threads) und Postings zwischen den Knoten ausgetauscht. Eine Persistierung findet jeweils nur auf dem Knoten statt, der die Kollektion bzw. den Topic beheimatet. Der Grund hierfür ist, dass ansonsten jeder Knoten auch die Zugriffsrechte synchronisieren müsste. Dies würde z.B. bei Änderungen und beim Löschen jede Menge Probleme aufwerfen, da auch Nutzer

von Drittknoten (ggf. andere Cloud!) beteiligt sein können. Stattdessen findet die Zulässigkeitsprüfung auf dem Quellknoten statt und die Entitäten werden ohne Zugriffsrechte übertragen.

Authentifizierung, Nutzerverwaltung

Die Prozesse Authentifizierung und Nutzerverwaltung umfassen alle Aspekte, die während des Anlegen eines Nutzers, der Zuordnung zu Gruppen aber auch während seiner An- bzw. Abmeldung berücksichtigt werden müssen. Die überwiegende Zahl der Nutzer- und Gruppenkonten auf einem Knoten dürfte im Regelfall von externen Knoten gefolgt von lokalen Verzeichnisdiensten (LDAP) stammen. Nur ein kleiner Teil der Konten muss vom lokalen Administrator verwaltet werden. In diesem Zusammenhang sei auf den entsprechenden Abschnitt "Verwaltung von Nutzern und Gruppen" im Kapitel "Bausteinsicht" verwiesen. Die Verwaltung von Konten umfaßt das Anlegen von Nutzern, die Zuweisung bzw. das Löschen von Gruppenmitgliedschaften und die Anonymisierung von Nutzern. Ein Löschen von Nutzern ist nicht vorgesehen, da dies schwerwiegende Probleme mit der referentiellen Integrität der Datenbank aufwerfen würde. Stattdessen werden nicht mehr benötigte oder nicht mehr erwünschte Konten anonymisiert und gesperrt. Im Frontend sind an der Verwaltung der Nutzer, Gruppen und Gruppenmitgliedschaften vor allem die Beans *UserMgrBean* und *GroupMgrBean* beteiligt, die auf die entsprechenden Services *MemberService* (für Nutzer und Gruppen) und *MembershipService* (für Mitgliedschaften) zurückgreifen.

Wie bereits beschrieben ist eine Authentifizierung von Nutzern nur an ihrem Heimatknoten möglich. Sie erfolgt über das Bean *UserBean*. Im Rahmen des Anmeldeprozess wird zunächst geprüft, ob das angegebene Login als lokaler Nutzer in der Datenbank bekannt ist. Falls ja, müssen zwei Fälle unterschieden werden: a) es handelt sich um einen lokalen Nutzer und b) es handelt sich um einen LDAP-Nutzer. Bei lokalen Nutzern wird aus dem eingegebenen Passwort (und dem "Salz" des gespeicherten Passworthashs) ein gesalzener Hash berechnet (*CredentialHandler*) und mit dem in der Datenbank gespeicherten Hash verglichen. Bei Übereinstimmung hat sich der Nutzer authentifiziert. Der Prozess bei LDAP-Nutzern ist wesentlich komplexer: Nach erfolgreicher Authentifizierung des Nutzers gegen das LDAP-Verzeichnis werden die Nutzerattribute (Login, UniqueId, Name, Telefon, Email) und sämtliche Mitgliedschaften (auch verschachtelte) im LDAP-Verzeichnis nachgeschlagen. Anschließender werden die ermittelten Werte mit den in der Datenbank gespeicherten Werten verglichen und die Datenbank entsprechend aktualisiert.

Falls der Nutzer der Datenbank nicht bekannt ist, kann es sich um einen LDAP-Nutzer handeln, der sich zum ersten Mal am Knoten anmeldet. Das Procedere ist im Prinzip das selbe wie bei der Anmeldung eines bereits bekannten LDAP-Nutzers; sämtliche Datenbank-Einträge (Nutzer und Mitgliedschaften) müssen jedoch neu angelegt werden.

Nach erfolgreicher Anmeldung am System muss der Nutzer an verschiedenen Stellen innerhalb der Webanwendung bekannt gemacht werden. Die Klasse *UserBean* feuert zu diesem Zweck ein *LoginEvent* und interessierte Klassen implementieren eine entsprechende Methoden mit @*Observes*-Annotation. Außerdem müssen der Nutzer und seine Mitgliedschaften MultiCloud-weit bekannt gemacht werden (s..a. Abschnitt "Synchronisierung von Entitäten"), was die Klasse *MembershipOrchestrator* übernimmt und asynchron erledigt. Vor Annoncierung des Nutzers werden sensitive Informationen (Passwort-Hash) *obfusziert*.

Die Abmeldung eines Nutzers ist gleichbedeutend mit der Anmeldung des *BUILTIN*-Nutzers *Public Account*, wobei die Passwort-Überprüfung und die MultiCloud-weite Annoncierung entfallen.

Ebenfalls zur Nutzerverwaltung gehört die Änderung von Nutzerattributen (Name, Telefonnummer, Email) und des Passworts.

2.18 Verteilungssicht

Üblicherweise beschreibt die Verteilungssicht die technische Infrastruktur und wie sich die einzelnen (Software-)Bausteine auf die einzelnen Infrastrukturkomponenten verteilen. In diesem Projekt geht diese Information bereits weitgehend (?) aus den Beschreibungen in den Kapiteln Bausteinsicht bzw. Laufzeitsicht hervor.

2.19 Querschnittliche Konzepte

In diesem Kapitel werden übergreifende Konzepte und Regelungen beschrieben.

2.19.1 Entwicklungskonzepte

Programmiersprache

Der Hauptteil der Entwicklung erfolgt mit Java, wobei zunächst keine Festlegung auf eine bestimmte Implementierung (Oracle JDK, OpenJDK) getroffen wird. Die Lizenzbedingungen einiger JDKs zwingen jedoch quasi zur Benutzung von OpenJDK. Hinzu kommen – vor allem für Konfiguration, Installation und Betrieb – einige Shell-Skripte und SQL für das Datenbank-Setup.

Coding-Conventions

n.n.

Entwicklungsumgebung (IDE)

Das Projekt ist nicht an eine bestimmte IDE gebunden, jeder Entwickler entscheidet selbständig mit welchem Werkzeug er am besten zurecht kommt. Nach einer Phase mit IntelliJ IDEA findet die Entwicklung derzeit (Frühjahr 2020) mit NetBeans statt. Darüber hinaus findet Entwicklung auch ohne IDE mit vi unter Linux statt.

Build-System

Als Build-System wurde Maven ausgewählt, weil damit bereits Erfahrungen vorliegen und es in seiner Funktionalität weit über das ebenfalls bekannte Build-System Ant hinausgeht. Andere Build-Systeme wie Gradle, Grape, Buildr usw. wurden nicht betrachtet; eine Umstellung, sollte sie notwendig werden, dürfte aufgrund der geringen Projektgröße unkritisch sein.

Code-Repository

Der Quellcode wird mit Git verwaltet. Das zentrale Repository liegt auf GitHub https://github.com/ipb-halle/CRIMSy.

> **Info:** In der Anfangsphase des Projekts wurde der Quellcode mit Subversion (und das Gesamtprojekt mit Trac) verwaltet und später nach Git / Bitbucket / Confluence / Jira migriert. Um die Sichtbarkeit des Projekts zu verbessern, den Administrationsaufwand im IPB zu minimieren und die Aktivitäten des IPB an einer Stelle zu bündeln, wurde entschieden, das Projekt in einem öffentlichen Repository (GitHub) weiterzuentwickeln. Dies zieht entsprechende Migrationsarbeiten für die Dokumentation und die Projektverwaltung nach sich. Zur Vertuschung unserer "Verbrechen" haben wir jedoch nicht die gesamte Projekthistorie nach GitHub übertragen ;-).

Qualitätssicherung

Um Qualität einer komplexen Software sicher zu stellen, sind umfangreiche Tests unerlässlich. Dieses Projekt pflegt zu diesem Zweck eine ganze Reihe von Testfällen und fügt dem Quelltext laufend neue Testfälle hinzu. Da es sich um eine verteilte Webanwendung handelt, sind einfach Unit-Tests nicht ausreichend. Bestimmte Szenarien erfordern das Zusammenspiel mehrerer Komponenten - das Framework Arquillian https://arquillian.org ist in diesem Zusammenhang sehr nützlich. Die Testabdeckung des Projekts beträgt über 40 Prozent (Stand März 2020). Vollautomatische Browser-Tests (Selenium-Framework o.ä.) sind momentan in Vorbereitung (siehe Abschnitt Integration Tests).

Dokumentation

Die Dokumentation erfolgt als Markup innerhalb des Projekts, damit das bisherige interne Projekt-Wiki (Confluence) abgelöst werden kann. Alle wesentlichen Informationen (inkl. Graphiken) wurden bzw. werden aus Confluence übertragen. Dabei wird die Trennung zwischen Handbüchern und Wiki aufrecht erhalten werden.

Projektverwaltung

Die Projektverwaltung wird mittelfristig von Jira auf die bei GitHub verfügbare Projektverwaltung umgestellt (siehe auch den Info-Kasten im Abschnitt Code-Repository).

Continuous Integration

CI im eigentlichen Sinne, d.h. inclusive Deployment, findet nicht statt. Das Projekt nutzt jedoch GitHub Actions, um das fehlerfreie Durchlaufen der Unit-Tests sicherzustellen. Das bislang genutzte Produkt Bamboo wurde abgelöst.

2.19.2 Integration Tests

Für Integrationstests wird ein Rechner mit der üblichen Softwareausstattung (siehe Handbuch Konfiguration & Installation) und zusätzlich folgenden Komponenten benötigt:

- OpenJDK 8
- git
- ssh
- nodejs
- npm
- selenium-side-runner (installieren mit `npm install -g selenium-side-runner`)
- eine Liste der Zielhosts (im weiteren Verlauf `HOSTLIST`)

Der durchführende Nutzer muss sich mit `ssh` zu allen Rechnern betreffenden Rechnern verbinden können und mittels `sudo` Superuser-Privilegien erlangen können. Dies muss jeweils ohne Passwortabfrage funktionieren. Die Liste der Zielhosts enthält Schlüssel-Wert-Paare, jeweils ein Schlüssel und Wert durch Leerzeichen getrennt auf einer Zeile. Die Schlüssel spezifizieren dabei den jeweiligen Testknoten (`node1` usw.) und die Werte den (vollqualifizierten) Hostnamen des Zielhosts. Die Zeilenzahl bestimmt, wieviele Knoten für die Tests instantiiert werden. Nach dem Auschecken der Quellen von github kann der Integrationstest mit

```
./util/bin/testSetup.sh HOSTLIST
```

angestoßen werden.

2.20 Risiken und Datenschutz

2.20.1 Risikoanalyse

Dieses Kapitel nimmt eine Risikoanalyse vor und versucht, die mit dem Betrieb der Leibniz Bioactives Cloud verbundenen Risiken und die Bedrohungen, denen die Leibniz Bioactives Cloud ausgesetzt ist, möglichst vollständig darzustellen und die Maßnahmen zu ihrer Minimierung zu skizzieren.

Es ist ausdrücklich nicht Ziel des Projekts, eine hochsichere oder hochverfügbare Lösung zu schaffen. Die Betreiber und Nutzer der Leibniz Bioactives Cloud sollen jedoch darauf vertrauen können, dass alle Maßnahmen zur Risikominimierung ergriffen wurden, die man vernünftigerweise erwarten kann.

Die Risiken bzw. Bedrohungen können dabei aus verschiedenen Blickwinkeln betrachtet werden. So kann man zunächst zwischen Risiken für den Betrieb der Leibniz Bioactives Cloud an sich und darüberhinausgehenden Risiken unterscheiden, wobei das Schadenpotential der zweiten Klasse leider erheblich höher ist. Zweitens ist eine Unterscheidung nach der Quelle der Bedrohung, nähmlich durch anonyme Angreifer aus dem Internet oder durch Insider aus dem Intranet möglich. Drittens sind bei der Betrachtung die mögliche Schadenshöhe und die geschätzte Eintrittswahrscheinlichkeit hilfreiche Kennziffern. Insgesamt ist der Betrieb der Leibniz Bioactives Cloud durch folgende Bedrohungen gekennzeichnet:

- Feindliche Übernahme der Infrastruktur
- Unautorisierter lesender Zugriff auf Daten
- Unautorisierte Datenmanipulation
- Denial of Service
- technisches Versagen

Mit diesen Bedrohungen sind die Risiken

- Sach- und Vermögensschäden / Haftung
- Reputationsschaden
- Betriebsunterbrechung
- Datenverlust

verbunden, die durch technische und organisatorische Maßnahmen zur Bedrohungsabwehr minimiert werden sollen. Bei allen Betrachtungen wird angenommen, dass aktuelle kryptographische Verfahren und speziell auch die Implementierungen sicher sind. Diese Einschätzung der Algorithmen beruht dabei auf der Beurteilung durch das Bundesamt für Sicherheit in der Informationstechnik (BSI). Sollten während des Betriebs der Cloud Schwachstellen bekannt werden, so muss mit entsprechenden Updates darauf reagiert werden.

Da strenggenommen personenbezogene Daten verarbeitet werden ist auch eine Betrachtung zum Datenschutz notwendig (Kapitel im Anschluss).

Feindliche Übernahme

Die feindliche Übernahme der Infrastruktur hat insgesamt das höchste Schadenpotential, da der Angreifer damit die volle Kontrolle über das System erlangt und es für seine Zwecke mißbrauchen kann. Denkbar sind unter anderem:

- Spam-Versand
- Teilnehmer in einem Bot-Net
- Mining von Crypto-Währungen
- Abfluß aller Daten
- Nutzung als Ausgangsbasis für weitere Angriffe

Die feindliche Übernahme muss daher unter allen Umständen vermieden werden. Im Falle eines Falles sollte ein Einbruch bzw. ein Einbruchsversuch frühzeitig durch entsprechende Sensoren (intrusion detection) entdeckt werden. Eine zentrale Rolle kommt hier den Firewalls der DMZ zu, die die Möglichkeiten des Angreifers wirksam beschränken sollen. Ebenso wichtig sind eine sichere Grundkonfiguration des Knotens, auf dem nur die tatsächlich benötigte Software installiert sein soll, und das regelmäßige Einspielen von Sicherheitsupdates.

Neben direkten Angriffen über das Netzwerk (von Außen über das Internet bzw. von Innen über das Intranet) besteht ein weiterer Angriffsvektor über bösartige Dokumente, die durch autorisierte Nutzer hochgeladen, z.B. durch Pufferüberläufe während der Indexierung, das System kompromittieren. Neben intrusion detection besteht die Gegenwehr in diesem Szenario hauptsächlich im regelmäßigen Update der eingesetzten Softwarekomponenten.

Unautorisierter Lesezugriff

Der Lesezugriff auf Dokumente darf nur über definierte Schnittstellen (DocumentServlet) möglich sein. Die Schnittstelle muss dabei prüfen, ob der Zugriff autorisiert ist. Die Autorisierung kann beispielsweise mittels eines Access-Tokens erfolgen, dass eine zeitlich begrenzte Gültigkeit hat. Auf diese Weise soll sichergestellt werden, dass Links auf Dokumente (z.B. aus dem Browser-Cache) nicht von unautorisierten Personen mißbraucht werden können. Die Infrastruktur für die Autorisierung ist im System selbst verankert und funktioniert über die Tripel (Asset, Zugriffsprivileg, Person/Gruppe). Die Authentifizierung als Person bzw. Gruppenmitglied kann gegen eine interne Nutzerdatenbank erfolgen. Zusätzlich besteht die Möglichkeit der Anbindung an externe Systeme (LDAP / Active Directory).

Die Datenübertragung von Knoten zu Knoten oder zum Nutzer muss verschlüsselt erfolgen, so dass es nicht zum Datenabfluß während des Transits kommen kann.

Ebenso muss das System gegen Lesezugriffe auf Zertifikatsschlüssel oder Passwörter geschützt sein. Der Zugriff auf Passwörter wird dadurch verhindert, dass Nutzerpasswörter nur gesalzen und mehrfach gehasht gespeichert werden. Verbindungen zu externen Authentifizierungssystemen (LDAP) werden verschlüsselt aufgebaut.

Die Speicherung der Daten in einem verschlüsselten Dateisystem wird nicht für notwendig erachtet; die Leibniz Bioactives Cloud ist nicht für die Speicherung von Daten vorgesehen, die solche Sicherheitsmaßnahmen erfordern.

Unautorisierte Datenmanipulation

Ebenso wie Lesezugriffe dürfen Schreiboperationen (einschließlich Löschen) nur über definierte Schnittstellen (z.B. FileUpload) und nur mit Autorisierung möglich sein. Das im Abschnitt "Unautorisierter Lesezugriff" geschriebene gilt entsprechend analog.

Denial of Service

Für die Bedrohung Denial of Service sind verschiedene Szenarien zu betrachten:

1. **Denial of Service gegen die Leibniz Bioactives Cloud**, indem die Ressourcen (Speicher, CPU, Bandbreite) einzelner oder mehrerer Knoten durch gezielte Abfragen erschöpft werden Das Risiko dieses Szenarios soll dadurch minimiert werden, dass Eingabeparameter (z. B. in URLs oder Post-Requests) einer sorgfältigen Zulässigkeitsprüfung (Minimale und maximale Länge, Zeichenvorrat, Vollständigkeit , reguläre Ausdrücke, usw.) unterzogen werden. Ausserdem müssen sinnvolle Begrenzungen für Prozessorzeit und Upload-Dateigrößen definiert werden.
2. **Denial of Service gegen andere interne (LDAP-Server, DNS-Server, usw.) oder externe (Software-Repositories, sonstige Server) Dienste:** Hier sind vor allem Maßnahmen zum Schutz des LDAP-Servers gegen die Blockierung von Nutzer-Accounts zu nennen. Durch Intruder Lockout verhindert die Infrastruktur, dass Nutzeraccounts über den Cloud-Knoten gezielt angegriffen werden. Die Risiko für sonstige Server und Dienste (DNS, Software-Repositories, sonstige Server) wird als gering eingestuft, so dass bislang keine Maßnahmen geplant sind. Im Zweifel können auch hier die Firewalls eine Schutzfunktion übernehmen.
3. **Denial of Service gegen Nutzer**, z.B. durch Browser-Absturz oder Computerviren Im normalen Betrieb der Cloud sind die Nutzer vor allem durch bösartige Dokumente gefährdet. Hierunter werden Dokumente (PDF, DOCX, DOC, . . .) verstanden, in deren aktiven Inhalten sich Schadcode verbirgt. Solche Dokumente können (feindliche Übernahme des Systems ausgeschlossen) die Nutzer nur erreichen, wenn sie zuvor von einem autorisierten Nutzer hochgeladen wurden. Idealerweise wird dieses Risiko dadurch minimiert, dass sowohl der

hochladende als auch der herunterladende Benutzer Virenscanner auf ihren Computern installiert haben. Unter Umständen kann auch die Firewall zum Intranet helfen, das Risiko dieser Bedrohung zu senken, wenn sie in der Lage ist, den Datenverkehr (als man-in-the-middle) transparent zu entschlüsseln und auf Viren zu scannen.

Technisches Versagen

Gegen technisches Versagen sollte die Installation durch adäquate Maßnahmen wie ECC-RAM oder fortschrittliche Storage-Systeme (RAID, ZFS, ...) geschützt werden. Sollte es dennoch zu einer Betriebsunterbrechung bei einem einzelnen Knoten kommen, so bleibt die Leibniz Bioactives Cloud als ganzes durch ihre dezentrale Organisiation weiterhin nutzbar. Gegen Datenverluste des Knotens ist ein externes Backup vorzusehen. Die Software der Leibniz Bioactives Cloud wird hierfür regelmäßig Datenbank-Dumps erzeugen, die von individuell zu konfigurierenden Backup-Agenten gesichert werden können. Sofern ein funktionsfähiges Backup existiert, ist der zu erwartende Schaden bei Ausfall eines Knotens vergleichsweise gering und erfordert somit keine weiteren Maßnahmen.

2.20.2 Datenschutz

Die Betrachtung der Datenverarbeitung durch die Leibniz Bioactives Cloud kann unter verschiedenen Aspekten erfolgen:

- aus der Perspektive der Anwender
- aus der Perspektive der gespeicherten Daten
- aus der Perspektive der Betreiber

Anwenderperspektive

Für die Anwender der Leibniz Bioactives Cloud wurde eine Datenschutzerklärung verfasst, die über den jeweiligen Knoten abgerufen werden kann. Darin ist erklärt, welche Daten der Anwender zu welchem Zweck gespeichert und verarbeitet werden und nach welchen Fristen diese Daten gelöscht werden. Im Wesentlichen handelt es sich bei anonymen Zugriffen um die üblichen Verkehrsdaten (IP-Adresse usw.); bei authentifizierter Nutzung kommen noch der Name, Nutzername ("Login"), evtl. eindeutige externe Kennungen, Gruppenmitgliedschaften und ggf. dienstliche Telefonnummern bzw. dienstliche Emailadressen hinzu. Bei authentifizierter Nutzung eines Knotens werden diese Daten an die übrigen Knoten der Cloud annonciert, damit dem Nutzer Rechte an Objekten eingeräumt werden können. Der Nutzer kann vom Betreiber seines Knotens die Anonymisierung seines Nutzerkontos verlangen. Die Anonymisierung erfolgt dadurch, dass jegliche personenbezogene Daten (Name, Login, Telefonnummer, Emailadresse, ggf. eindeutige externe Kennungen) aus einem Nutzerkonto entfernt werden.

Datenperspektive

Auf den Knoten werden wissenschaftliche Daten und Dokumente (u.a. wissenschaftliche Veröffentlichungen) gespeichert. Diese Dokumente können personenbezogene Daten enthalten. Üblicherweise umfasst dies die Name der Autoren, ihre Affiliation und Kontaktdaten. In einigen wissenschaftlichen Publikationen finden sich gelegentlich auch Bilder und Lebensläufe. Die Leibniz Bioactives Cloud ist jedoch keinesfalls zur Speicherung darüberhinausgehender personenbezogener Informationen (Patientendaten, nicht anonymisierte Umfragedaten) geeignet.

Betreiberperspektive

Dem Betreiber der Leibniz Bioactives Cloud ist für die Einhaltung der datenschutzrechtlichen Vorschriften auf seinem Knoten verantwortlich. Er muss insbesondere sicherstellen, dass keine unzulässigen Dokumente in die Leibniz Bioactives Cloud hochgeladen werden und dass eventuellen Anonymisierungsbegehren von Nutzern Rechnung getragen

wird. Darüber hinaus steht er in der Verantwortung der ordnungsgemäßen Datenlöschung bei Ausscheiden seines Knotens.

Die im laufenden Betrieb anfallenden Log-Dateien, die personenbezogene Daten der Nutzer enthalten können, werden automatisch nach 7 Tagen gelöscht.

2.21 Technische Schulden

Dieses Kapitel listet technische Schulden auf, die mit bestimmten Design-Entscheidungen in Zusammenhang stehen. Den IT-Verantwortlichen und Systemadministratoren der Institute soll damit die Einschätzung des Betriebsrisikos ermöglicht werden. Gleichzeitig sollen regelmäßige Reviews dieser Liste den Abbau technischer Schulden ermöglichen. Desweiteren werden einige offene Punkte gelistet, deren Umsetzung mangels Ressourcen bislang noch nicht möglich war.

- **Verschlüsselung der Konfigurationsdatei** Die Konfigurationsdatei wird vor dem Versand nicht signiert sondern nur verschlüsselt, weil zum Zeitpunkt der Konfiguration noch kein Zertifikat für den Absender generiert wurde. Im Prinzip könnte daher jedermann eine Konfigurationsdatei erzeugen und versenden. Das Risiko wird als gering beurteilt, da zusätzliche Kommunikationskanäle (Email, Telefon) bestehen, über die eine Abstimmung des Versands erfolgt. Außerdem sind die Cloud-Administratoren gehalten, die Konfigurationsdateien vor dem Einlesen gründlich zu prüfen.
- **Härtung** Die Härtung des Knotens durch Mechanismen wie SELinux wurde aufgrund mangelnder Ressourcen noch nicht unternommen.
- **Init-System** Das System unterstützt momentan hauptsächlich Systemd. Alternativ können SystemV-Init-Skripte ausgewählt werden, die jedoch praktisch ungetestet sind.

2.21.1 Offene Punkte

Folgende Punkte konnten bislang nicht umgesetzt werden, die zukünftige Umsetzung ist jedoch vorgesehen:

- Verwendung des TomEE-internen Mechanismus für Zugriffsbeschränkung

2.22 Glossar

- **`$LBAC_DATASTORE`** Bei dieser Variable handelt es sich um die zentrale Konfigurationsvariable eines Knotens. Der Wert dieser Variable wird an mehreren Stellen (u.a. `$HOME/.lbac`, `/root/.lbac`, `$LBAC_DATASTORE/etc/config.sh`) gespeichert. Inkonsistente Änderungen an diesen Einstellungen sind ein Rezept für Disaster.
- **Collection** Eine Collection ist eine Sammlung von Dokumenten, die auf einem Knoten lokalisiert (gespeichert) ist und dort verwaltet wird. Dies schließt die Indexierung der Dokumente und die Verwaltung der Zugriffsrechte ein. Eine gewisse Verwechslungsgefahr besteht mit dem Java Interface `java.util.Collection`, weshalb auf entsprechende Klarheit zu achten ist.
- **Container** In Verbindung mit der Leibniz Bioactives Cloud kann der Begriff Container zwei verschiedene Bedeutungen haben.
  - Zum einen beschreibt der Begriff ein virtuelles Betriebssystem für die Ausführung von Diensten. Im Gegensatz zu herkömmlicher Virtualisierung wird nicht die komplette Hardware virtualisiert, wodurch der Ressourcenverbrauch erheblich geringer ist. Dies ermöglicht, für jeden Dienst einen eigenen Container anzulegen und die Dienste so zu isolieren. Durch die Isolation ist Interaktion nur noch über definierte Schnittstellen möglich, wodurch die Sicherheit verbessert wird.

  - In der zweiten Bedeutung beschreibt Container eine Software zur Verwaltung von sogenannten JavaEE-Beans. In der Leibniz Bioactives Cloud übernimmt die Software Apache TomEE diese Funktion und sorgt u.a. für die Authentifizierung und Autorisierung von Nutzern und organisiert die Speicherung der Daten in einer Datenbank.

- **DMZ** Eine Demilitarisierte Zone ist ein durch Firewalls nach innen und außen abgesichertes Netzwerk, an das Server angeschlossen werden, um Dienste (z.B. Webanwendungen) anzubieten.
- **Docker** Docker ist eine vergleichsweise neue Technologie, mit der einzelne Anwendungen in Containern verpackt ausgeführt werden können. Auf einem Host können viele verschiedene Container parallel ausgeführt werden, die gegeneinander isoliert sind und nur über definierte Pfade miteinander interagieren können. Man erreicht dadurch eine saubere Abgrenzung der Systeme untereinander, wodurch die Sicherheit erhöht wird.
- **Dokument** Ein Dokument ist eine Datei in beliebigem Format. Bei geeigneten Formaten (insbes. PDF, DOC, DOCX usw.) findet eine Indexierung statt.
- **JPA / JPA2** Java Persistence API - eine Sammlung von Schnittstellen und Methoden um in Java auf eine Datenbank zuzugreifen.
- **Knoten** Ein Knoten (gelegentlich auch englisch: Node) bezeichnet einen Server, in dem die Softwarekomponenten für die Cloud installiert sind. Jeder Knoten wickelt die Kommunikation mit seinen Benutzern ab und tritt bei Bedarf mit allen anderen Knoten in der Cloud direkt in Verbindung. Es besteht jedoch keine permanente Verbindung und eine Weiterleitung (indirekter Zugriff, Relay) findet nicht statt. Prinzipiell reicht bereits ein einzelner Knoten für den Betrieb aus; die Idee ist jedoch, dass jedes beteiligte Institut einen eigenen Knoten beherbergt und dass Nutzer direkt nur mit dem Knoten in ihrem Institut kommunizieren.
- **Kollektion** siehe Collection
- **Node** siehe Knoten
- **Repository** Im Kontext dieses Projekts ist ein Repository eine Sammlung von Software oder Daten, aus dem Teile heruntergeladen werden können. Entsprechend kann dieses Projekt auch als (verteiltes) Repository aufgefasst werden, das jedoch nur den teilnehmenden Institutionen zugänglich ist. Beispiele für öffentliche Repositories sind

2.23 Förderer

2.23.1 Leibniz-Institut für Pflanzenbiochemie

Das Leibniz-Institut für Pflanzenbiochemie in Halle (Saale) trägt von Beginn an die Lohnkosten für einen Entwickler und finanziert den Großteil der für die Entwicklung benötigten Infrastruktur.

2.23.2 Europäischer Fonds für regionale Entwicklung (EFRE)

SACHSEN-ANHALT

EUROPÄISCHE UNION
EFRE
Europäischer Fonds für regionale Entwicklung

Seit November 2019 unterstützt der Europäische Fonds für regionale Entwicklung (EFRE) die Entwicklung des "Cloud Resource and Information Management System" im Rahmen des Projekts "ProCognito - Pflanzliche Wirkstoffe zum Erhalt der kognitiven Leistungsfähigkeit im Alter".

2.23.3 Leibniz-Forschungsverbund "Wirkstoffe und Biotechnologie"

Leibniz
Wirkstoffe und Biotechnologie

Das Projekt wurde 2016 als "Leibniz Bioactives Cloud" vom Leibniz-Forschungsverbund "Wirkstoffe und Biotechnologie" initiiert und bis 2020 aus Mitteln des Leibniz-Senatsausschuss Strategie (SAS) unterstützt.

2.24 Lizenzierung

Das Leibniz-Institut für Pflanzenbiochemie lizenziert die Software "Cloud Resource & Information Management System - CRIMSy" (vormals *Leibniz Bioactives Cloud*) unter der Apache License Version 2.0. Der Lizenztext ist nachfolgend wiedergegeben und kann im Original unter https://www.apache.org/licenses/LICENSE-2.0 abgerufen werden (Irrtümer vorbehalten).

```
                              Apache License
                        Version 2.0, January 2004
                     http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

   "License" shall mean the terms and conditions for use, reproduction,
   and distribution as defined by Sections 1 through 9 of this document.

   "Licensor" shall mean the copyright owner or entity authorized by
   the copyright owner that is granting the License.

   "Legal Entity" shall mean the union of the acting entity and all
   other entities that control, are controlled by, or are under common
   control with that entity. For the purposes of this definition,
   "control" means (i) the power, direct or indirect, to cause the
```

(continues on next page)

(continued from previous page)

```
   direction or management of such entity, whether by contract or
   otherwise, or (ii) ownership of fifty percent (50%) or more of the
   outstanding shares, or (iii) beneficial ownership of such entity.

   "You" (or "Your") shall mean an individual or Legal Entity
   exercising permissions granted by this License.

   "Source" form shall mean the preferred form for making modifications,
   including but not limited to software source code, documentation
   source, and configuration files.

   "Object" form shall mean any form resulting from mechanical
   transformation or translation of a Source form, including but
   not limited to compiled object code, generated documentation,
   and conversions to other media types.

   "Work" shall mean the work of authorship, whether in Source or
   Object form, made available under the License, as indicated by a
   copyright notice that is included in or attached to the work
   (an example is provided in the Appendix below).

   "Derivative Works" shall mean any work, whether in Source or Object
   form, that is based on (or derived from) the Work and for which the
   editorial revisions, annotations, elaborations, or other modifications
   represent, as a whole, an original work of authorship. For the purposes
   of this License, Derivative Works shall not include works that remain
   separable from, or merely link (or bind by name) to the interfaces of,
   the Work and Derivative Works thereof.

   "Contribution" shall mean any work of authorship, including
   the original version of the Work and any modifications or additions
   to that Work or Derivative Works thereof, that is intentionally
   submitted to Licensor for inclusion in the Work by the copyright owner
   or by an individual or Legal Entity authorized to submit on behalf of
   the copyright owner. For the purposes of this definition, "submitted"
   means any form of electronic, verbal, or written communication sent
   to the Licensor or its representatives, including but not limited to
   communication on electronic mailing lists, source code control systems,
   and issue tracking systems that are managed by, or on behalf of, the
   Licensor for the purpose of discussing and improving the Work, but
   excluding communication that is conspicuously marked or otherwise
   designated in writing by the copyright owner as "Not a Contribution."

   "Contributor" shall mean Licensor and any individual or Legal Entity
   on behalf of whom a Contribution has been received by Licensor and
   subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
   this License, each Contributor hereby grants to You a perpetual,
   worldwide, non-exclusive, no-charge, royalty-free, irrevocable
   copyright license to reproduce, prepare Derivative Works of,
   publicly display, publicly perform, sublicense, and distribute the
   Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of
   this License, each Contributor hereby grants to You a perpetual,
   worldwide, non-exclusive, no-charge, royalty-free, irrevocable
```

(continues on next page)

(continued from previous page)

```
   (except as stated in this section) patent license to make, have made,
   use, offer to sell, sell, import, and otherwise transfer the Work,
   where such license applies only to those patent claims licensable
   by such Contributor that are necessarily infringed by their
   Contribution(s) alone or by combination of their Contribution(s)
   with the Work to which such Contribution(s) was submitted. If You
   institute patent litigation against any entity (including a
   cross-claim or counterclaim in a lawsuit) alleging that the Work
   or a Contribution incorporated within the Work constitutes direct
   or contributory patent infringement, then any patent licenses
   granted to You under this License for that Work shall terminate
   as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the
   Work or Derivative Works thereof in any medium, with or without
   modifications, and in Source or Object form, provided that You
   meet the following conditions:

   (a) You must give any other recipients of the Work or
       Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices
       stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works
       that You distribute, all copyright, patent, trademark, and
       attribution notices from the Source form of the Work,
       excluding those notices that do not pertain to any part of
       the Derivative Works; and

   (d) If the Work includes a "NOTICE" text file as part of its
       distribution, then any Derivative Works that You distribute must
       include a readable copy of the attribution notices contained
       within such NOTICE file, excluding those notices that do not
       pertain to any part of the Derivative Works, in at least one
       of the following places: within a NOTICE text file distributed
       as part of the Derivative Works; within the Source form or
       documentation, if provided along with the Derivative Works; or,
       within a display generated by the Derivative Works, if and
       wherever such third-party notices normally appear. The contents
       of the NOTICE file are for informational purposes only and
       do not modify the License. You may add Your own attribution
       notices within Derivative Works that You distribute, alongside
       or as an addendum to the NOTICE text from the Work, provided
       that such additional attribution notices cannot be construed
       as modifying the License.

   You may add Your own copyright statement to Your modifications and
   may provide additional or different license terms and conditions
   for use, reproduction, or distribution of Your modifications, or
   for any such Derivative Works as a whole, provided Your use,
   reproduction, and distribution of the Work otherwise complies with
   the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
   any Contribution intentionally submitted for inclusion in the Work
   by You to the Licensor shall be under the terms and conditions of
```

(continues on next page)

(continued from previous page)

```
   this License, without any additional terms or conditions.
   Notwithstanding the above, nothing herein shall supersede or modify
   the terms of any separate license agreement you may have executed
   with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
   names, trademarks, service marks, or product names of the Licensor,
   except as required for reasonable and customary use in describing the
   origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
   agreed to in writing, Licensor provides the Work (and each
   Contributor provides its Contributions) on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
   implied, including, without limitation, any warranties or conditions
   of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
   PARTICULAR PURPOSE. You are solely responsible for determining the
   appropriateness of using or redistributing the Work and assume any
   risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
   whether in tort (including negligence), contract, or otherwise,
   unless required by applicable law (such as deliberate and grossly
   negligent acts) or agreed to in writing, shall any Contributor be
   liable to You for damages, including any direct, indirect, special,
   incidental, or consequential damages of any character arising as a
   result of this License or out of the use or inability to use the
   Work (including but not limited to damages for loss of goodwill,
   work stoppage, computer failure or malfunction, or any and all
   other commercial damages or losses), even if such Contributor
   has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
   the Work or Derivative Works thereof, You may choose to offer,
   and charge a fee for, acceptance of support, warranty, indemnity,
   or other liability obligations and/or rights consistent with this
   License. However, in accepting such obligations, You may act only
   on Your own behalf and on Your sole responsibility, not on behalf
   of any other Contributor, and only if You agree to indemnify,
   defend, and hold each Contributor harmless for any liability
   incurred by, or claims asserted against, such Contributor by reason
   of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

   To apply the Apache License to your work, attach the following
   boilerplate notice, with the fields enclosed by brackets "[]"
   replaced with your own identifying information. (Don't include
   the brackets!)  The text should be enclosed in the appropriate
   comment syntax for the file format. We also recommend that a
   file or class name and description of purpose be included on the
   same "printed page" as the copyright notice for easier
   identification within third-party archives.

Copyright [yyyy] [name of copyright owner]
```

(continues on next page)

(continued from previous page)

```
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

CHAPTER

THREE

ADMINISTRATIONSHANDBUCH

3.1 Administrator-Handbuch

Die Administrator- und Entwicklerdokumentation des Projekts setzt sich aus der JavaDoc-Dokumentation, der Systembeschreibung und diesem Handbuch zusammen. Die JavaDoc-Dokumentation spezifiziert dabei Details, während die Systembeschreibung einen größeren Zusammenhang herstellt. Dieses Handbuch beschreibt, wie man aus den Projektquellen eine neue Cloud aufsetzt, d.h. die Quellen übersetzt, die Cloud konfiguriert und die Softwareverteilung organisiert. Das Handbuch fokussiert dabei auf die praktischen Aspekte dieser Tätigkeiten.

3.1.1 Setup

Zur Einrichtung und zum Betrieb einer Cloud-Instanz mit ein oder mehreren Knoten sind folgende Komponenten notwendig:

- Ein Linux-Rechner mit Java, Maven, Git, OpenSSL usw. für den Build-Prozess; dieser Rechner sollte gut abgesichert sein (Backup, sichere Konfiguration), da die Zertifizierungsstelle der Cloud-Instanz auf diesem Rechner betrieben wird.
- Ein Web-Server mit SSH-Zugang für die Softwareverteilung; der SSH-Zugang kann auf den Buildrechner beschränkt sein.
- Ein oder mehrere Cloud-Knoten; einer dieser Knoten muss die Rolle des Master-Knotens übernehmen

Notfalls können alle Komponenten auf einem Rechner zusammengefasst werden - dies sollte bei Produktivsystemen aber vermieden werden. Das Handbuch setzt grundlegende Fertigkeiten in der Administration von Linux-Rechnern voraus. Soweit nicht explizit anders gefordert, sollten alle Schritte mit einem unprivilegierten Nutzeraccount durchgeführt werden.

3.2 Phase 1

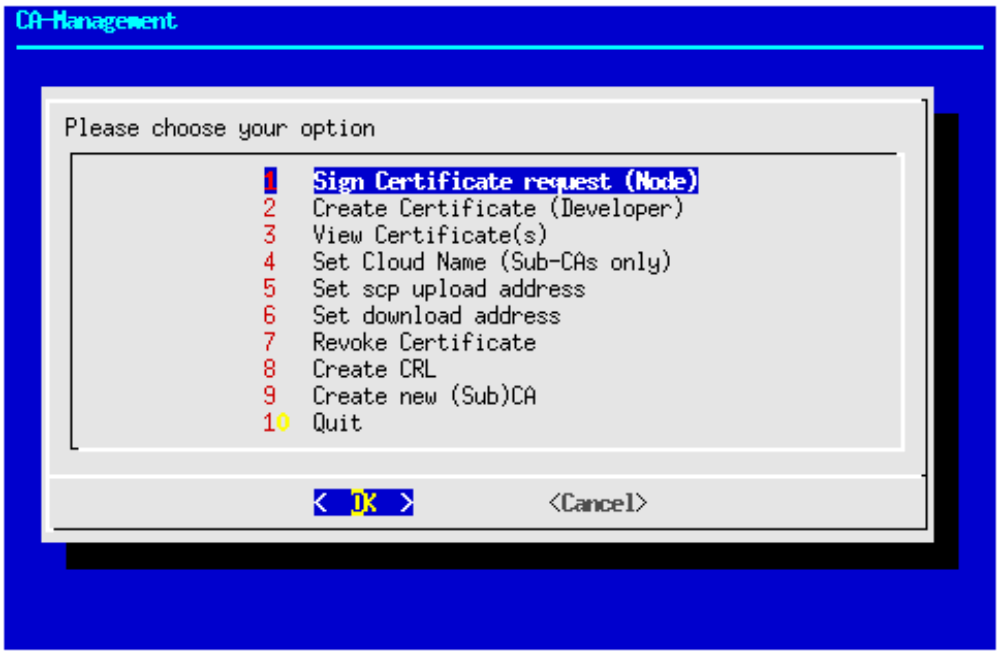
Die Einrichtung einer Cloud beginnt mit dem Clonen des Repositories bzw. Auschecken oder Herunterladen der Quellen. Seit März 2020 sind die Quellen über GitHub verfügbar:

```
git clone https://github.com/ipb-halle/CRIMSy.git
```

Durch das Clonen wird ein Verzeichnis CRIMSy mit den Quellen angelegt. Alle weiteren Aktionen finden - soweit nichts anderes angegeben ist - in diesem Verzeichnis statt.

Als nächstes muss die Distribution konfiguriert werden. Dies beinhaltet das Anlegen einer Stammzertifizierungsstelle (Root CA) und einer Zertifizierungsstelle für die geplante Cloud (eine Sub-CA). Die Einrichtung erfolgt jeweils mit

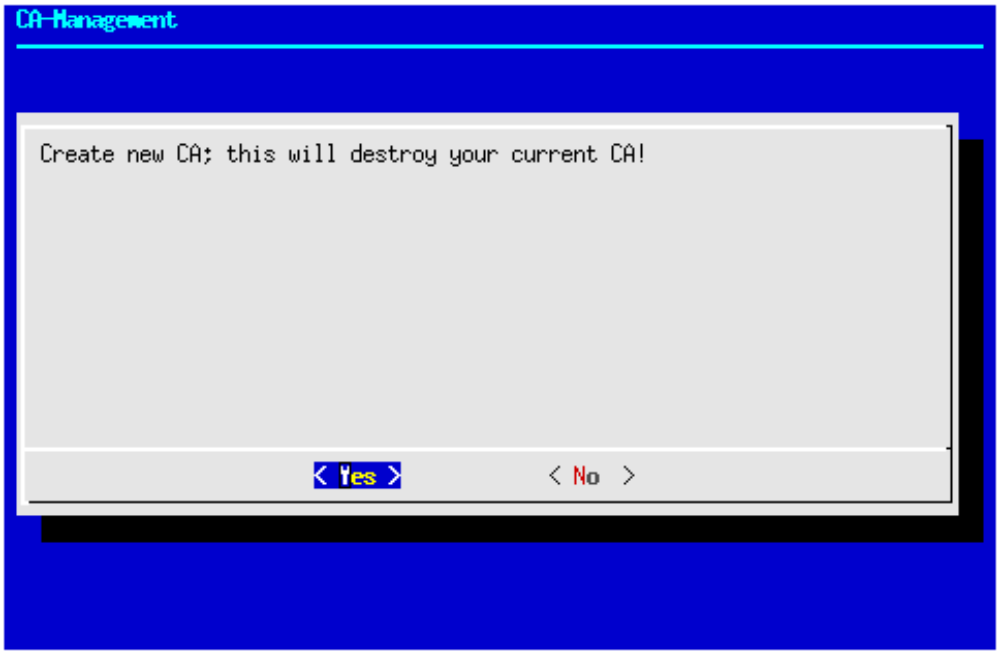
dem Script *util/bin/camgr.sh*. Alle Konfigurationsdaten der Cloud einschließlich der CAs werden im Source-Tree unterhalb des Verzeichnis config/ gespeichert. Durch eine entsprechende Einstellung in .gitignore ist sichergestellt, dass die Konfiguration nicht in das Sourcecode-Repository eingecheckt wird. Viele Funktionen des *camgr.sh*-Scripts sind menügesteuert und verwenden (weitestgehend) eine Curses-Oberfläche. Einige Parameter (z.B. die Cloud) müssen jedoch über Kommandozeilenoptionen übergeben werden. Andere Funktionen (Ausstellen eines Zertifikats, Erzeugung der Zertifikatssperrliste / Certificate Revocation List - CRL) sind auch über die Kommandozeile zugänglich, da das Skript *camgr.sh* auch in automatisiert ablaufenden Prozessen benutzt wird. Mit der Kommandozeilenoption "–help" kann man sich einen Hilfetext anzeigen lassen. Das Hauptmenü des Scripts sieht wie folgt aus:



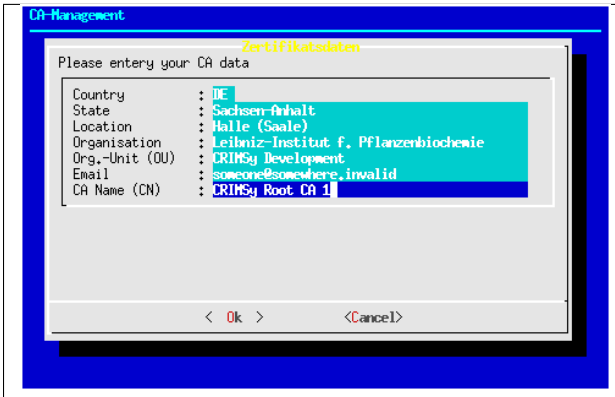
Note: Das camgr.sh-Script beendet sich momentan nach jedem Schritt und muss demzufolge für einen darauffolgenden Schritt neu gestartet werden.

In einem ersten Schritt muss eine neue CA (= Zertifizierungsstelle) mit einem neuen CA-Zertifikat (Zertifizierungsstellenzertifikat) angelegt werden. Das aktuelle Betriebskonzept sieht eine dreistufige Zertifikatshierarchie vor: An oberster Stelle steht eine Stammzertifizierungsstelle (Root-CA), die Zertifikate für weitere Zertifizierungsstellen (Sub-CAs bzw. Zwischen-CAs) ausstellt. Es ist nicht vorgesehen, dass die Root-CA Zertifikate für Knoten ausstellt. Die Sub-CAs (eine Sub-CA für jede Cloud) stellen die Zertifikate für Knoten ihrer jeweiligen Cloud aus. Auf der untersten Ebene erhält jeder Knoten ein Zertifikat der Zertifizierungsstelle (Sub-CA) seiner Cloud.

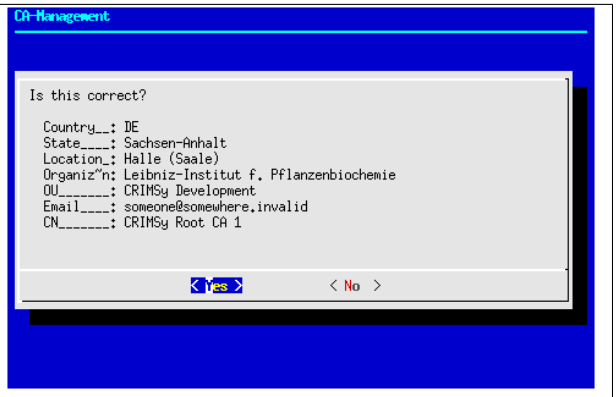
Note: Ursprünglich (d.h. in der *Leibniz Bioactives Cloud*) wurden die Zertifikate für die Knoten direkt von der Root-CA ausgestellt. Da CRIMSy einen dezentralen Ansatz verfolgt, können die Sub-CAs von unterschiedlichen Root-CA beglaubigt werden.



Sofern noch keine Root-CA existiert, die die Sub-CA für eine Cloud beglaubigt, sollte zunächst eine neue Root-CA angelegt werden. Rufen Sie dazu *camgr.sh* ohne Kommandozeilenargumente auf und wählen Sie Menüpunkt 9. Es ist möglich, ohne eigene Root-CA zu arbeiten, und die Sub-CA der Cloud von einer fremden Root-CA zertifizieren zu lassen. Das Script *camgr.sh* unterstützt jedoch hauptsächlich ein spezifisches Szenario, so dass abweichende Szenarien manuelle Eingriffe nötig machen können. **Achtung: Beim Anlegen einer neuen CA wird eine eventuell bestehende CA zerstört!**

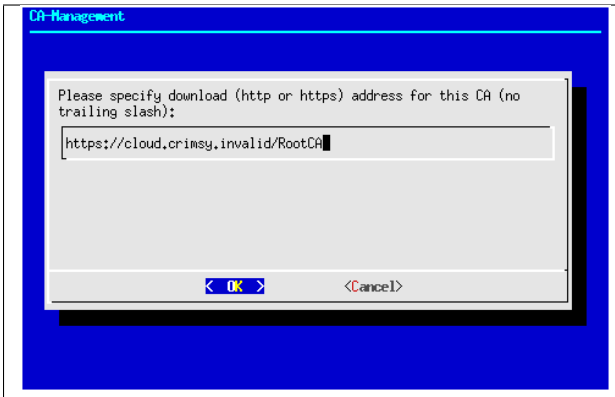


CA-Warnung



CA-Daten

Wie in der Abbildung zu sehen, werden im nächsten Schritt die Daten für die Root-CA abgefragt. Daran schließt sich eine Überprüfung der Daten an.



CA: URL

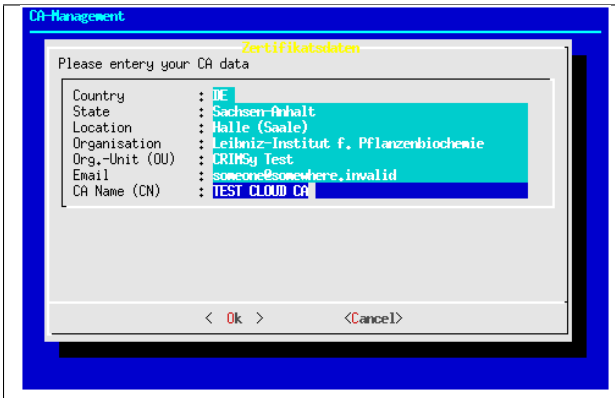
CA: SCP

Für die Root-CA müssen außerdem eine Download-URL und eine *Secure Copy*-Adresse angegeben werden (beide ohne *trailing slash*). Über die Download-URL sind das Zertifikat der CA, die Sperrliste (CRL) sowie weitere öffentliche Informationen abrufbar. Aus der URL wird die URL der Zertifikatssperrliste durch Anhängen von *crl.pem* gebildet und in das Zertifikat übertragen. Die URL im Zertifikat kann später nicht mehr geändert werden! Die *Secure Copy*-Adresse dient dazu, das Zertifikat, die Sperrliste und weitere Informationen (einen Java-Truststore und eine Adressliste) hochzuladen. Idealerweise sollte *scp* mit Public Key Authentifizierung konfiguriert sein, damit die Kopiervorgänge ohne Passworteingabe funktionieren.

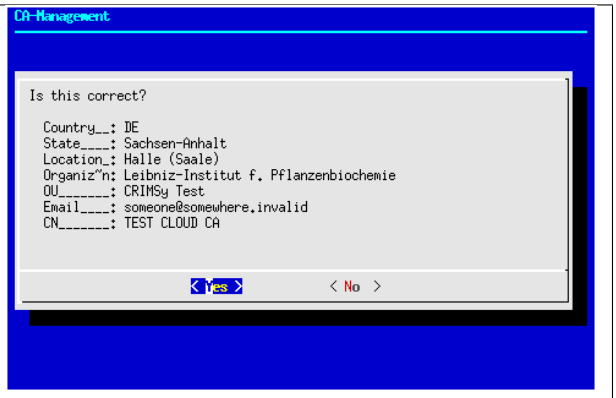
Für die Einrichtung eine Sub-CA für die Cloud *TEST* muss das Skript mit der Option *–cloud TEST* aufgerufen werden:

```
./util/bin/camgr.sh --cloud TEST
```

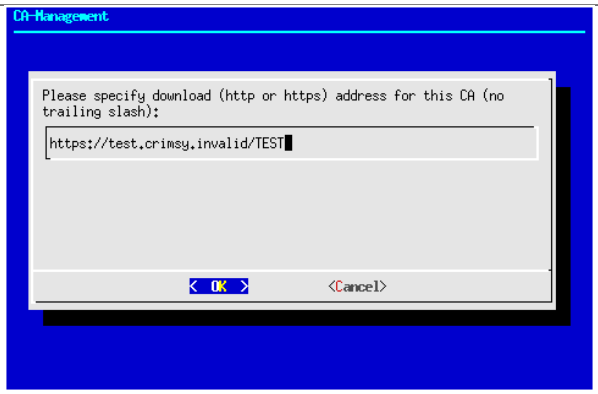
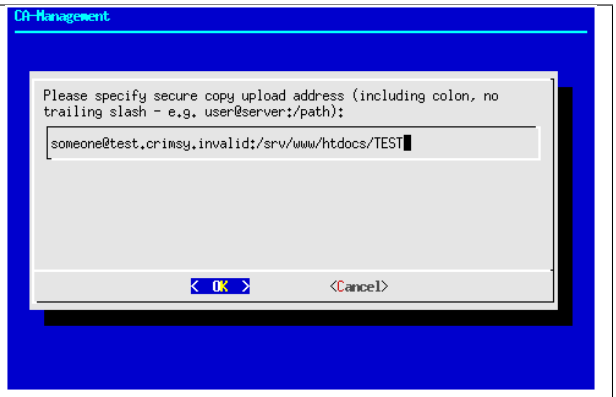
Die nächsten Schritte (Warnhinweis, Erfassung und Prüfung der Daten, Download-URL und *Secure Copy*-Adresse) laufen dann zunächst analog zur Einrichtung der Root-CA ab:



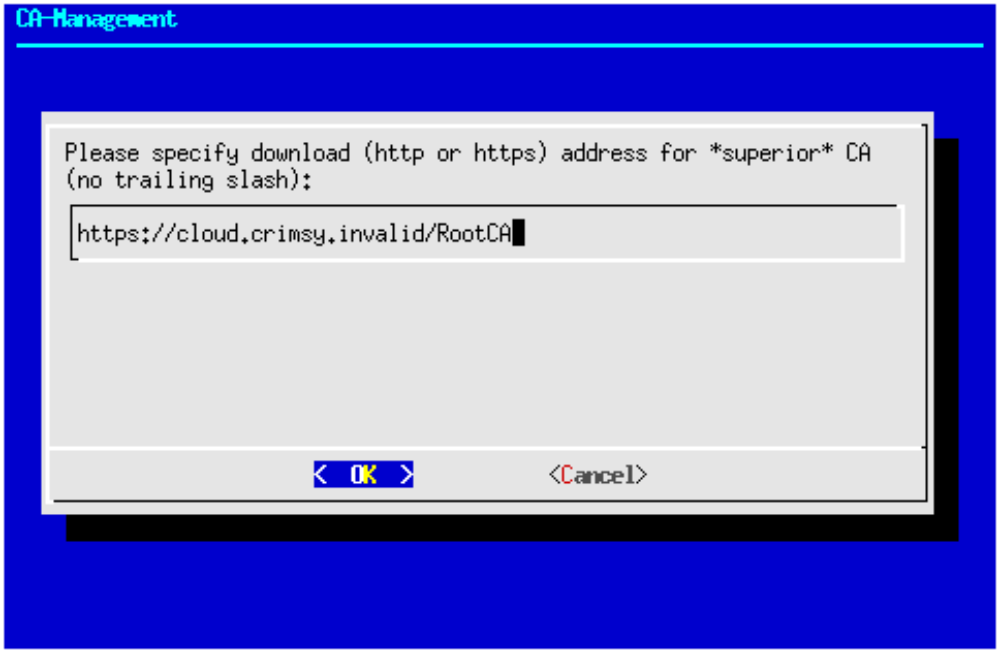
Sub-CA: Daten



"Sub-CA: Prüfung

	
Sub-CA: URL	“Sub-CA: SCP

Die Unterschiede beginnen, wenn die Download-URL der übergeordneten CA (***superior***, also der Root-CA) abgefragt wird. Falls Sie die übergeordnete CA selbst betreiben, geben Sie bitte die gleiche URL wie bei der Erstellung der Root-CA an:



Sub-CA: URL der Root-CA

Falls Sie die Root-CA nicht selbst betreiben, müssen Sie sicherstellen, dass alle benötigten Informationen unter der Download-URL verfügbar sind:

- die Zertifikatskette der übergeordneten CA in der Datei *chain.txt*; auf diese Weise sind auch mehrstufige CA-Hierarchien möglich
- die Zertifikatssperrliste in der Datei *crl.pem*
- ein Java-Keystore mit allen Zertifikaten der Zertifikatskette in der Datei *truststore* sowie das zugehörige Keystore-Passwort in der Datei *truststore.passwd*
- eine Datei *addresses.txt* mit dem (Kurz-)Namen der CAs, *subject hashes*, Fingerprints, sowie den URLs von CA-Zertifikat und CRL aller Zertifikate der Zertifikatskette; 5 Spalten jeweils durch Tabulator getrennt.

Obwohl im Dialog nur die Protokolle *http:* bzw. *https:* angegeben sind, können Sie hier ausnahmsweise auch *file:* benutzen. Das Skript *camgr.sh* wird dann die Dateien der übergeordneten CA in das Verzeichnis der Sub-CA kopieren und einen Zertifikatsrequest für die Sub-CA erstellen. Im nächsten Schritt muss anhand dieses Zertifikatsrequests ein Zertifikat für die Sub-CA ausgestellt werden. Falls Sie die übergeordnete CA nicht selbst betreiben, müssen Sie den Zertifikatsrequest (die Datei *config/…/CA/cacert.req*) an die übergeordnete CA übermitteln. Ansonsten reicht der Aufruf der beiden Kommandos (hier gezeigt am Beispiel einer Cloud *TEST*):

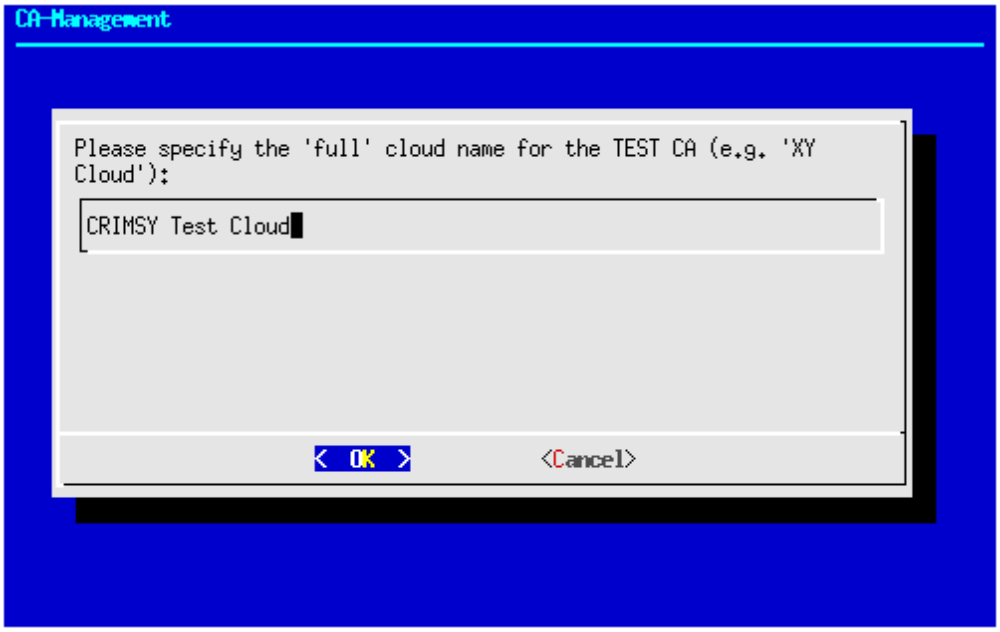
```
./util/bin/camgr.sh --mode sign \
  --extension v3_subCA \
  --input ./config/TEST/CA/cacert.req \
  --output ./config/TEST/CA/cacert.pem

./util/bin/camgr.sh --mode importSubCA \
  --cloud TEST
```

Das erste Kommando veranlasst die Ausstellung des Zertifikats mit der Erweiterung *v3_subCA* durch die Root-CA, so das mit diesem Zertifikat andere Zertifikate beglaubigt werden können. Die Pfade können entweder absolut oder relativ zum aktuellen Verzeichnis angegeben werden.

```
            fc:af:77:6d:0b:ef:37:96:cd:22:d6:80:f2:37:25:b6:ce:0f:
            99:0d:36:c7:96:23:07:a3:d2:5d:9e:fc:47:d7:d3:3a:d0:4f:
            ac:31:53:7e:17:62:62:0b:57:60:9e:96:40:d6:3a:17:fa:f6:
            92:94:40:29:77:a2:9b:da:da:83:71:ec:1a:0f:27:f1:87:ef:
            c6:81:f4:76:2a:0f:f6:44:91:a8:57:c7:39:d4:3d:66:e9:70:
            d1:2d:0c:42:be:50:2b:d9:2d:2f:3c:37:96:49:15:49:45:fd:
            cc:15:9b:b6:8f:0e:67:fd
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:'DE'
stateOrProvinceName   :ASN.1 12:'Sachsen-Anhalt'
localityName          :ASN.1 12:'Halle (Saale)'
organizationName      :ASN.1 12:'Leibniz-Institut f. Pflanzenbiochemie'
organizationalUnitName:ASN.1 12:'CRIMSy TEST'
commonName            :ASN.1 12:'TEST CLOUD CA'
Everything appears to be ok, creating and signing the certificate
Successfully added extensions from config
The subject name appears to be ok, checking data base for clashes
Certificate is to be certified until Mar 25 15:12:26 2025 GMT (1825 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
```

Root-CA: Zertifikatsbestätigung

Das zweite Kommando "importiert" anschließend das Zertifikat und stellt schon praktisch die Betriebsbereitschaft der Sub-CA her. Zu guter Letzt sollte aber noch der Langname der Cloud gesetzt werden:



Sub-CA: langer Cloud Name

Dabei ist zu beachten, dass das Skript *camgr.sh* immer mit der Option *–cloud* und dem Kurznamen der Cloud (z.B. *TEST*)aufgerufen werden muss:

```
./util/bin/camgr.sh --cloud TEST
```

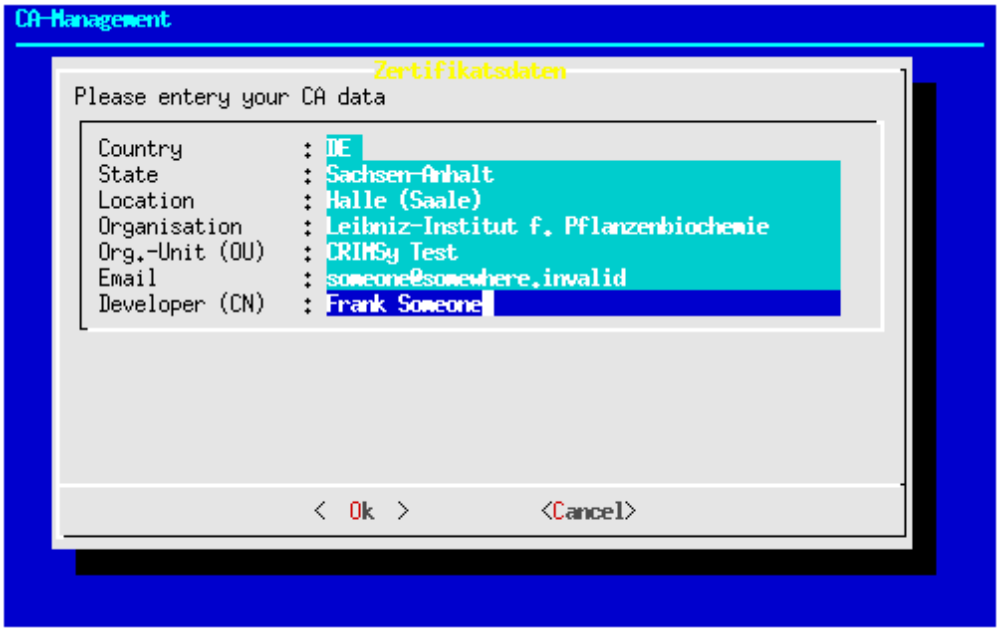
da ansonsten die Einstellungen der Root-CA verändert werden.

Für die Funktion der Cloud sind aktuelle Zertifikatssperrlisten notwendig. Am besten werden diese regelmäßig (z.B. 1x täglich) durch einen CRON-Job aktualisiert. Es ist wichtig, sämtliche Zertifikatssperrlisten der Zertifikatskette aktuell zu halten. Ein CRON-Eintrag könnte wie folgt aussehen:

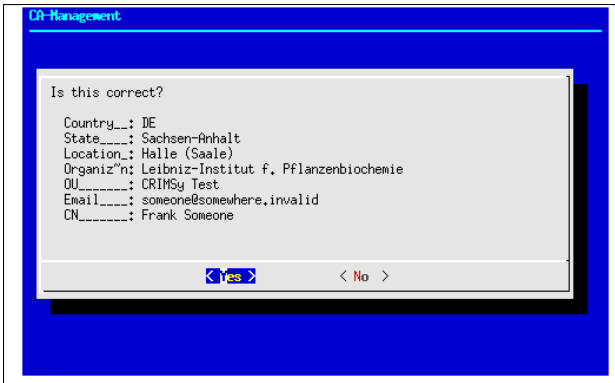
```
#
# CRL erzeugen
#
#Min Hour Day Month Week Cmd
# Root-CA
20 10 * * * /home/someone/git/CRIMSy/util/bin/camgr.sh --mode genCRL 2>/dev/null >/
↪dev/null
#
# Sub-CA
21 10 * * * /home/someone/git/CRIMSy/util/bin/camgr.sh --mode genCRL --cloud TEST 2>/
↪dev/null >/dev/null
```

Der CRON-Job sollte zum selben Nutzer gehören, der auch die Quellen ausgecheckt hat.

Für die Absicherung der Softwarepakete durch digitale Signaturen wird ein Entwicklerzertifikat benötigt. Dies kann ebenfalls mit dem Skipt *camgr.sh* ausgestellt werden. Es ist darauf zu achten, dass das Entwicklerzertifikat von der CA der jeweiligen Cloud ausgestellt wurde. Entwickler-Zertifikate der Root-CA sind momentan ohne Nutzen. Nachfolgend ist die Ausstellung eines Entwicklerzertifikats für die Cloud *TEST* illustriert (d.h. Aufruf des Skripts mit *./util/bin/camgr.sh –cloud TEST*). Im Skript ist dazu Menüpunkt 2 “Create Certificate (Developer)” aufzurufen:



Sub-CA: Entwickler-Zertifikat



Entwicklerzert. Prüfung

Entwicklerzert. Bestätigung

Wie üblich schließen sich daran die Überprüfung der Eingaben und die Bestätigung der Zertifikatsausstellung an.

Die erste Setup-Phase schließt mit dem Hochladen des Konfigurationsscripts ab. Dies erfolgt ganz einfach durch Aufruf des Skripts *upload.sh* mit dem Kurznamen der Cloud als Argument:

```
./util/bin/upload.sh TEST
```

Anschließend muss unter der Download-Adresse der Distribution in etwa folgende Seite aufrufbar sein:

CRIMSy Download Page

localhost/TEST/

CRIMSY Test Cloud: TEST

Auf dieser Seite sind die für die Installation relevanten Ressourcen (Handbücher, das signierte Konfigurations-Skript und die zugehörigen Zertifikate) zum Herunterladen zusammengefasst. Ausführlichere Dokumentation finden Sie entweder hier oder auf unserer Projektseite.

Installation

- Konfigurationshandbuch
- configure.sh.sig das PEM-kodierte und signierte Installationsskript
- chain.txt LBAC Zertifikatskette
- devcert.pem Code-Signing-Zertifikat

Verwenden Sie am besten den folgenden Code-Schnipsel, der auch die Prüfung der Signatur übernimmt:

```
wget -O configure.sh.sig http://localhost/TEST/configure.sh.sig
wget -O chain.txt http://localhost/TEST/chain.txt
wget -O devcert.pem http://localhost/TEST/devcert.pem
sha256sum chain.txt
openssl verify -CAfile chain.txt devcert.pem
openssl smime -verify -in configure.sh.sig -certfile devcert.pem -CAfile chain.txt -out configure.sh
```

Idealerweise haben Sie die SHA-256 Prüfsumme der Zertifikatskette der *CRIMSY Test Cloud CA* auf unabhängigem Weg (Email, Telefon, Fax) von uns bekommen. Vergleichen Sie bitte die Ausgaben der letzten drei Kommandos mit folgenden Werten (in Rot dargestellt). **Bitte informieren Sie uns, wenn es irgendeine Unstimmigkeit gibt.**

```
[...]
~> sha256sum chain.txt
208b6b29e0e4eba670c4b84133617f0f19c294ba65e68c1dcfac57c76b6eac40  chain.txt
~> openssl verify -CAfile chain.txt devcert.pem
devcert.pem: OK
~> openssl smime -verify -in configure.sh.sig -certfile devcert.pem -CAfile chain.txt -out configure.sh
Verification successful
```

Falls Sie keine Abweichungen feststellen, können Sie das Skript aufrufen:

```
chmod +x configure.sh
./configure.sh
```

Distibutions-Webseite

In der zweiten Phase müssen die einzelnen Knoten der Cloud und vor allem der Master-Knoten konfiguriert werden. Ausgangspunkt ist die eben erwähnte Distributions-Seite. Die genaue Vorgehensweise ist im Manual Konfiguration und Installation und speziell im Abschnitt Konfigurationsskript beschrieben.

3.3 Phase 2

Die zweite Phase beginnt damit, dass der Distributor den Quellcode übersetzt:

```
mvn package
```

Die *pom.xml*-Datei im Hauptordner des Projekts sorgt dafür, dass einige zusätzliche Java-Bibliotheken im Verzeichnis Verzeichnis *config/extralib/* bereitgestellt werden. Diese Bibliotheken werden benötigt, damit Hibernate zusammen mit TomEE funktioniert (Näheres unter [https://stackoverflow.com/questions/10852035/how-to-use-tomee-with-hibernate{]}(https://stackoverflow.com/questions/10852035/how-to-use-tomee-with-hibernate)). Im Einzelnen handelt es sich um die Bibliotheken (ggf. in neueren Versionen):

```
antlr-2.7.7.jar
classmate-1.3.0.jar
dom4j-1.6.1.jar
hibernate-commons-annotations-5.0.1.Final.jar
hibernate-core-5.2.6.Final.jar
hibernate-jpa-2.1-api-1.0.0.Final.jar
hibernate-validator-5.2.4.Final.jar
javassist-3.20.0-GA.jar
jboss-logging-3.3.0.Final.jar
postgresql-9.3-1102.jdbc4.jar
```

Zwischenzeitlich wurden (bzw. nachfolgend werden) die verschlüsselten Konfigurationsdateien der Teilnehmerknoten empfangen und im Verzeichnis config/nodes/ abspeichert. Diese Dateien sollten zuvor einer Sichtprüfung unterzogen werden. Das Format der Konfigurationsdateien sollte folgendem Muster entsprechen:

```
#
# LBAC_INSTITUTION=IPB Demo
# CERTIFICATE_ID=06:30:D8:B9:86:54:B5:69:CF:79:3F:C9:A9:BC:D4:8F:D8:94:39:78
# Wed Oct 10 14:30:29 UTC 2018
#
# ----- SMIME ENCRYPTED CONFIG BEGIN -----
-----BEGIN PKCS7-----
MIINnQYJKoZIhvcNAQcDoIINjjCCDYoCAQAxggLBMIICvQIBADCBpDCBnjELMAkG
[...]
YnVrNTz4AlCAvtlG+QDZrbaw/iwAUkaLVI+Hk/mMpB9e
-----END PKCS7-----
# ----- SMIME ENCRYPTED CONFIG END -----
```

Der entscheidende Schritt in Phase 2 ist die Festlegung des Master-Knotens und des Cloudnamens, wobei letzerer identisch mit dem Namen der Sub-CA ist. Ein Cloudname wie "TEST", "CLOUD" oder "PRODUKTION" ist vollkommen ausreichend; es dürfen jedoch innerhalb einer Multi-Cloud keine Kollisionen des Namens vorkommen. In den oben angeführten Beispielen wurde der Cloudname "CRIMSY Test Cloud" bzw. kurz "*TEST*" verwendet.

Der Masterknoten wird durch Aufruf des Scripts util/bin/package.sh festgelegt. Ohne weiteres Argument gibt das Script eine kurze Syntaxinformation aus. Ansonsten gibt es für das Script drei Einsatzszenarien:

1. Festlegung des Masterknotens: Aufruf mit *./util/bin/package.sh CLOUDNAME MASTER*
2. Hinzufügen eines Knotens zur Cloud: Aufruf mit *./util/bin/package.sh CLOUDNAME*
3. Neuzusammenstellung der Installationspakete aller aus dem Verzeichnis *config/CLOUDNAME/* bekannten Knoten, z.B. bei einer neuen Software-Version: *./util/bin/package.sh CLOUDNAME AUTO*

Während Szenario 3 später automatisch durchläuft, öffnen sich in den Szenarien 1 und 2 die nachfolgend dargestellten Dialoge zur Auswahl eines Knotens:

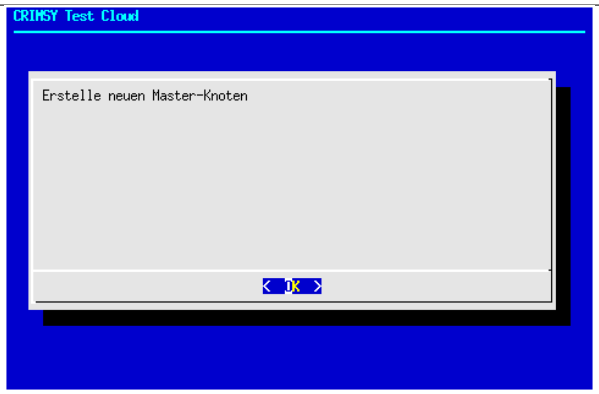
Zunächst kann aus der Liste der eingesendeten Konfigurationsdateien (*config/nodes/*) der zukünftige Master-Knoten bestimmt werden

Anschließend wird die Konfigurationsdatei geöffnet. **Hier muss unbedingt gründlich geprüft werden, ob von den**

angezeigten Daten eine Gefahr ausgeht, da die Konfiguration anschließend im den Kontext der aktuellen Shell interpretiert wird! Gefährlich sind Aufrufe von Kommandos (pars pro toto das berüchtigte rm -f /) die sich mittels Backticks auch in Variablenzuweisungen verstecken können (konkretes Beispiel: *LBAC_DATASTORE="/home/``rm -rf /``cloud/"*). Selbstverständlich sollte die angezeigte Institution auch mit der zuvor getätigten Auswahl korrespondieren.

> **Warning:** Gründlich prüfen, eine bösartige Konfigurationsdatei kann beliebig großen Schaden anrichten!

Das Ergebnis der Prüfung wird anschließend abgefragt.

	
Master-Knoten	Info

Bei der Erstellung eines Master-Knotens erscheint der links oben angeführte Informations-Dialog, der bei regulären Knoten entfällt. Darauf folgt ein Dialog, der darüber informiert, dass die Ausstellung des Knoten-Zertifikats unmittelbar bevorsteht.

Die Ausstellung des Zertifikats verläuft analog zur Ausstellung des Entwickler- oder Zertifizierungsstellenzertifikats (Sub-CA). Der Zertifikatsrequest ist in der übermittelten Konfigurationsdatei enthalten. Falls für einen Knoten bereits zu einem früheren Zeitpunkt ein Zertifikat ausgestellt wurde, wird dieses wiederverwendet, sofern es noch nicht widerrufen wurde. Ein widerrufenes Zertifikat erfordert zwingend die Einsendung einer neuen Konfigurationsdatei mit einem neu ausgestellten Zertifikatsrequest. Die Identifizierung des möglicherweise existierenden Zertifikats erfolgt über die md5-Summe des Zertifikatsrequests.

Anschließend werden vom Script ohne weiteren Nutzereingriff

- alle Artefakte im Verzeichnis *target/dist/* gesammelt
- mittels *tar*, *gzip* und *uuencode* in ein komprimiertes, Base64-kodiertes Archiv gepackt und
- an die Datei *setup.sh* angehängt. Der entstehende Datenstrom wird
- mittels *openssl smime* mit dem öffentlichen Schlüssel des Empfängerknotens verschlüsselt
- und dem Entwicklerzertifikat signiert. Das fertige Installationspaket wird anschließend
- mittels *scp* auf den Distributionsserver hochgeladen.

Für den Multi-Cloud-Betrieb wird außerdem ein kompaktes tar-Archiv ohne Software erstellt, das nur die Zertifikatskette, Truststores und Informationen über den Master-Knoten enthält. Dadurch wird es möglich, dass ein Knoten Mitglied in mehreren Clouds ist. Gleichzeitig wird durch unterschiedliche SubCAs sichergestellt, das nur Mitgliedsknoten miteinander kommunizieren können.

Die Aufgaben des Distributors sind damit zunächst erledigt. Im weiteren Verlauf werden die Administratoren der einzelnen Knoten die Installation gemäß Handbuch (Konfiguration und Installation) durchführen. Zweckmäßigerweise sollte der Masterknoten als erster Knoten eingerichtet werden.

3.3.1 Management

Die Managementaufgaben des Distributors umfassen neben allgemeinen Wartungsaufgaben (Überwachung der Dienste, Patch-Management, ...) vor allem die Verwaltung der Zertifikate und der Konfigurationsdateien. Der Widerruf eines Zertifikats kann dabei sowohl bei Kompromittierung als auch beim Ausscheiden einer Cloud-Instanz notwendig werden. In diesem Fall dürfen für den betroffenen Knoten keine Softwarepakete mehr erstellt werden. Insbesondere muss auch sichergestellt werden, dass der Zertifikatsrequest nicht wiederverwendet wird (der private Schlüssel könnte kompromittiert sein).

Trifft das Script package.sh auf eine Konfiguration mit widerrufenem Zertifikat, wird eine Warnmeldung ausgegeben und das Skript beendet.

Die fragliche Konfigurationsdatei mit dem Zertifikatsrequest muss manuell gelöscht oder verschoben werden.

CHAPTER

FOUR

FULLTEXT SEARCH

- search