Hyland™

Hyland™

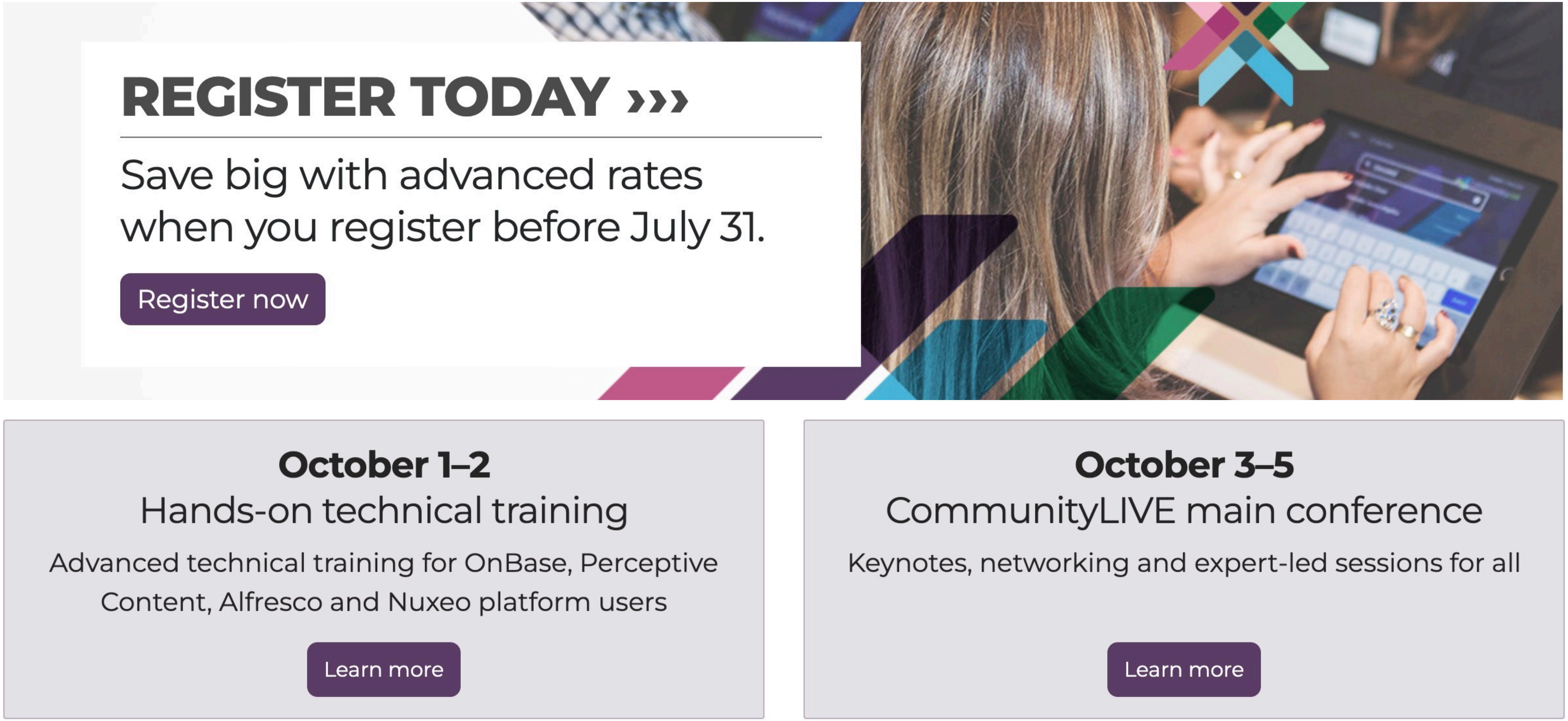
Alfresco
Tech Talk Live #149

July 19, 2023

Agenda

- Community news
- Alfresco mTLS authentication made simple(r) with Ansible

Hyland Community Live



REGISTER TODAY ›››

Save big with advanced rates when you register before July 31.

Register now

October 1–2

Hands-on technical training

Advanced technical training for OnBase, Perceptive Content, Alfresco and Nuxeo platform users

Learn more

October 3–5

CommunityLIVE main conference

Keynotes, networking and expert-led sessions for all

Learn more

Upcoming Hyland Summit in your area

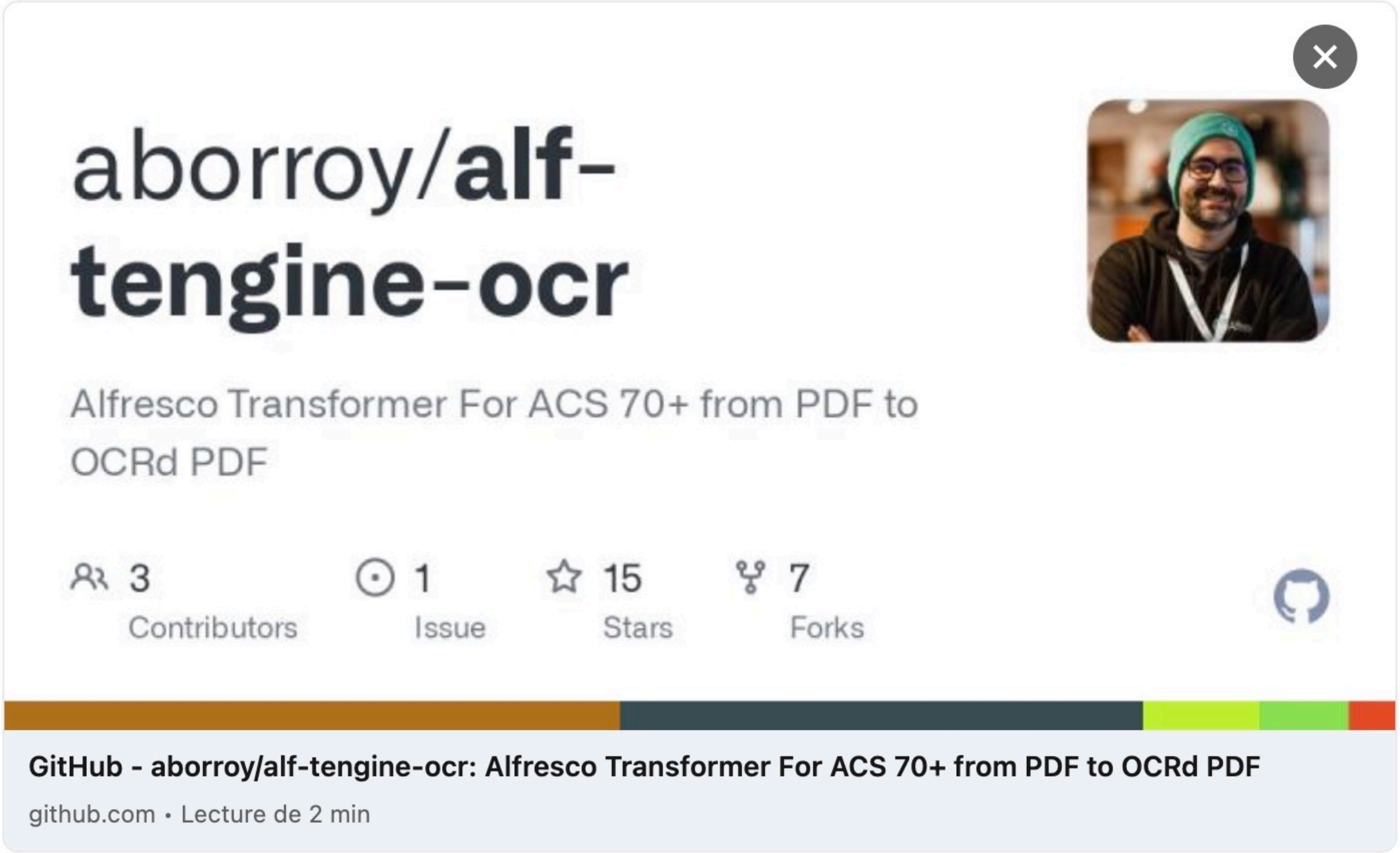
Hyland Summit Colombia	Bogotá, Colombia	Thursday, 17 August
Hyland Summit Brazil	Brasília, Brazil	Thursday, 24 August
Hyland Summit London	London, UK	Tuesday, 10 October
Hyland Summit Paris	Paris, France	Thursday, 19 October
Hyland Summit Düsseldorf	Düsseldorf, Germany	Tuesday, 24 October
Hyland Summit Sydney	Sydney, Australia	Thursday, 2 November
Hyland Summit Madrid	Madrid, Spain	Tuesday, 7 November

Big thanks

To

Douglas C R Paes

For testing ats-transformer-ocr with different languages



aborroy/**alf-tengine-ocr**

Alfresco Transformer For ACS 70+ from PDF to OCRd PDF

3 Contributors · 1 Issue · 15 Stars · 7 Forks

GitHub - aborroy/alf-tengine-ocr: Alfresco Transformer For ACS 70+ from PDF to OCRd PDF

github.com • Lecture de 2 min

Ressources

Alfresco 7.4

- Secure Communications with Alfresco 7.4 - Alfresco Hub
- Offline/parallel re-indexing with ElasticSearch - Alfresco Hub
- How to migrate from Alfresco Search Services to Alfresco SearchEnterprise (From Apache Solr to Elasticsearch or Amazon Opensearch – Slideshare)

Contribute to Alfresco Community

- Adapt Order of the Bee support tools to Alfresco 7.4 in the use of log4j2 (LinkedIn)

Resources to come

- Adapting your logging configuration to log4jv2
- Migrating to Search Enterprise
- Share to ADF migration guide (thanks Loftux for the feedback!)
- Using Spring Security with ACS 7.4
- ActiveMQ deep dive

TTL Speakers wanted!

- Take the opportunity to showcase your work with the community
- About Alfresco, Nuxeo, and associated technologies
- Best practices, integration, scaling, cloud, …

Today’s talk

Alfresco mTLS authentication made simple(r) with Ansible

Alexandre Chapellon & Giovanni Toraldo
DevOps Engineering, Hyland

Hyland™

Hyland™

ATS mTLS authentication made simple(r) with Ansible

Alexandre Chapellon – Giovanni Toraldo

DevOps Engineers @ Hyland

July 20, 2023

About us

Alexandre Chapellon

Guîtres, France 🇫🇷

Free software 🐃 enthusiast and Alfresco old timer 👴

I like to injure 🩹 myself doing 🛹 and tries to forget about it watching ⭐ in a 🔭

Twitter: @alxgomz

GitHub: alxgomz

Giovanni Toraldo

Lucca, Italy 🇮🇹

Open source 🐧 remote software developer 👨‍💻 writer ✍️ speaker 🎤 Often teleported into the 14th century 🛡️⚔️🎯

Twitter: @gionn

GitHub: gionn

Agenda

- What is mTLS
- Integration with Alfresco ATS
- Alternative solutions & other deployments

What is mTLS

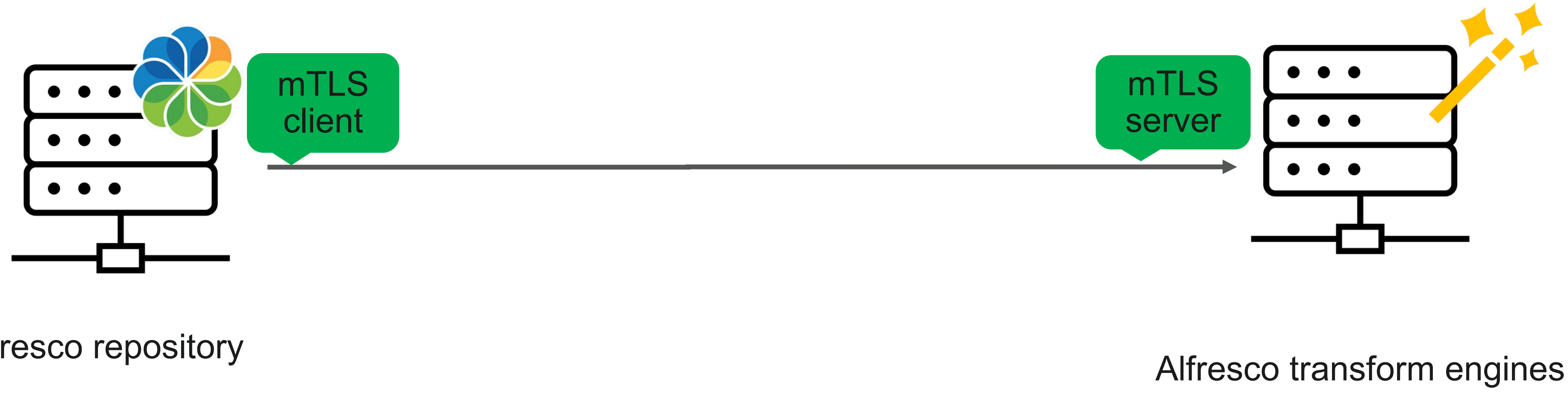
- TLS is a well established security protocol to encrypt data connections over an insecure channel (https)
  - Clients doesn’t assume the server identity and requires a valid certificate

- Mutual TLS ensure that both parties involved in a secure network connection are both who they claim to be
  - Server doesn’t assume clients can access and ask them to provide a valid certificate (not expired, revoked or untrusted)

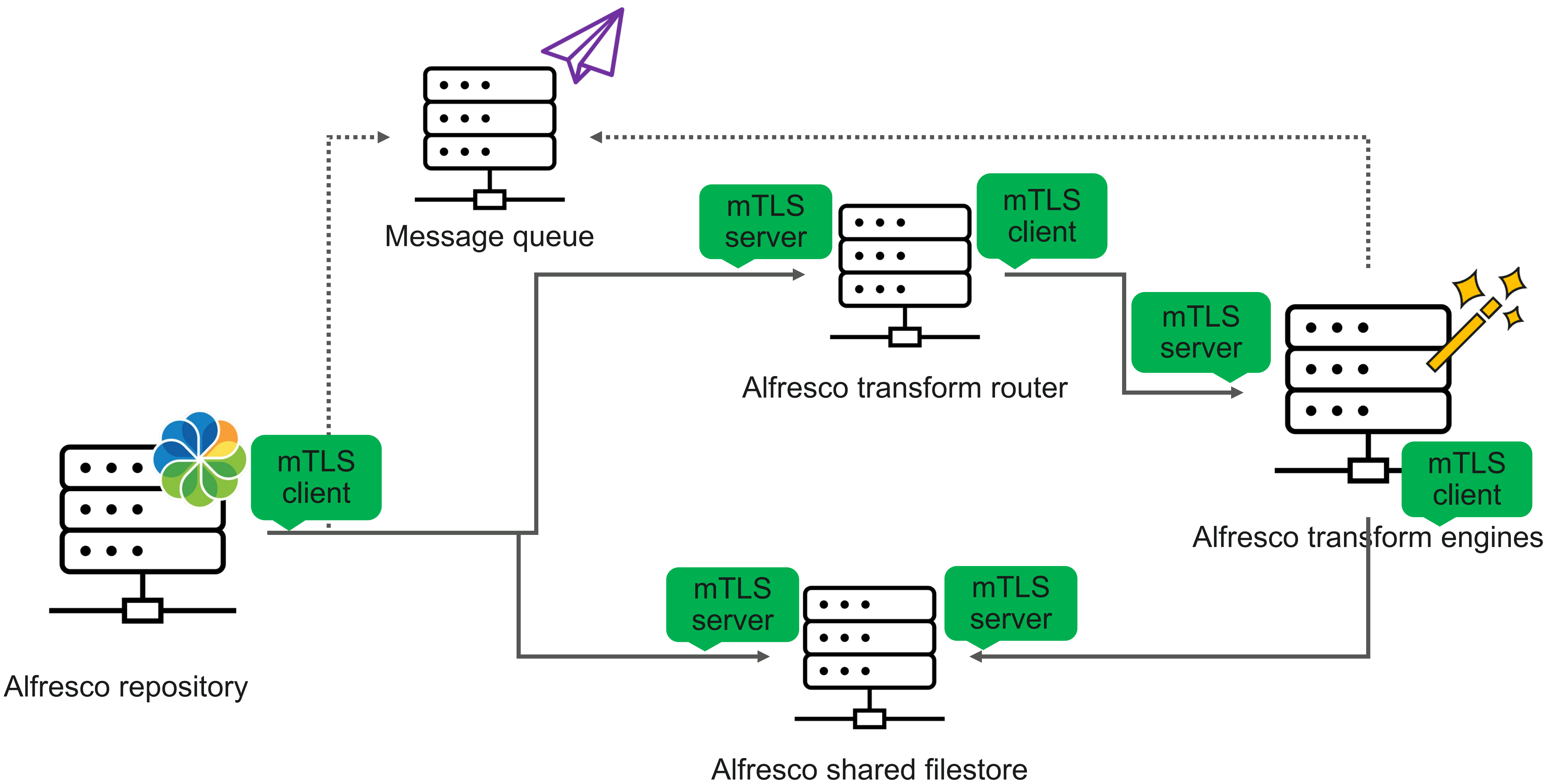
What are the advantages of mTLS

- Traffic encryption (even on internal networks)
  - High level of security compliance in regulated industries
- Traffic integrity
  - Data cannot be forged or tampered with
- Hosts authentication for service access
  - Do not assume that a client can access a service just because it’s in the same network (zero trust networks)

Alfresco Community Architecture



Alfresco Enterprise Architecture



What is Alfresco Ansible deployment offering?

- Automated certificate generation
- Certificates deployments
- Certificates updates

- Use cases:
  - Corporate PKI
    - From a signin CA cert
    - From plain certificates
  - Fully automated

Playbook implementation

- playbooks/pki.yml:
  - Responsible for generating the keys & certificates with appropriate configuration and bundling them into a PKCS12 keystore for each host in the inventory.
- java role (keystore.yml entrypoint):
  - Responsible for copying the p12 to their respective targets
  - Import certificates & keys into the java keystore
- repository, sfs, trouter & transformers roles:
  - Responsible for deploying the appropriate configuration so mTLS is properly configured
- playbooks/acs.yml:
  - Chooses when to enabled/disable mTLS

pki.yml playbook configuration

- Playbook variables
  - pki_dir: directory where to generate/find the hosts certificates on the control node
  - ca_cn: Name of the auto-generated Certification Authority
  - secret_ca_passphrase: CA passphrase
  - ca_key_size: size of the cryptographic key used to generate our own CA
  - ca_key_type: type of cryptographic key used to generate our own CA
  - ca_days_valid_for: lifetime of generated CA
  - p12_passphrase: p12 container passphrase
  - cert_key_size: size of the cryptographic key used to generate certificates
  - cert_key_type: type of the cryptographic key used to generate certificates
  - cert_days_valid_for: lifetime of generated certificates
- Hostname’s checks are disabled (mostly to cope with non generated certs)

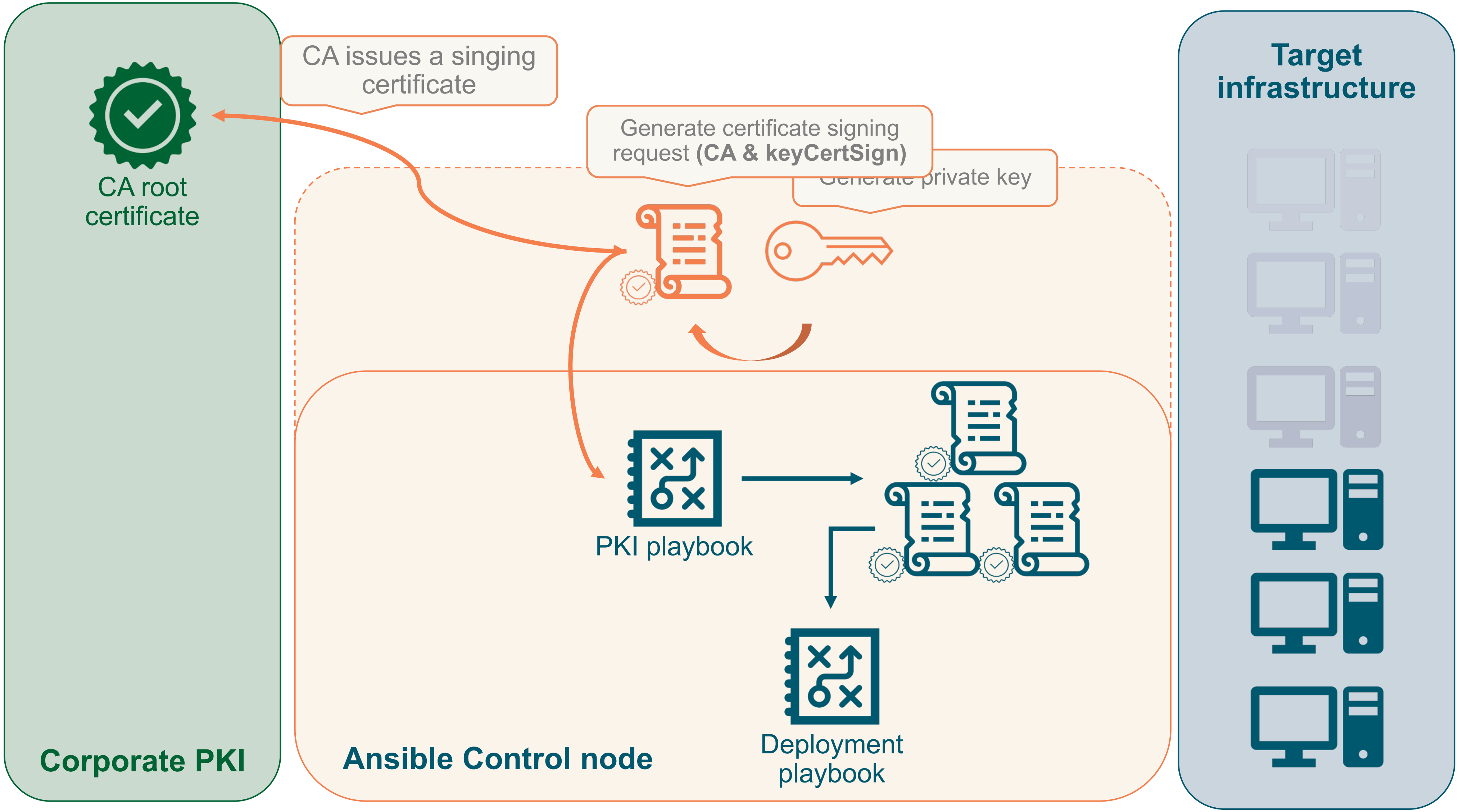
Java role configuration

- Main role’s arguments (check roles/java/meta/argument_specs.yml for a full list
  - java_truststore: path to the truststore to import CA chain to
  - java_truststore_pass: passphrase to unlock the truststore
  - java_keystore.path: path to the keystore to import certificates to
  - java_keystore.pass: passphrase to unlock the keystore
  - java_keystore.type: type of keystore
  - java_keystore.cert_containers[*]: p12 container passphrase
    - pass
    - path

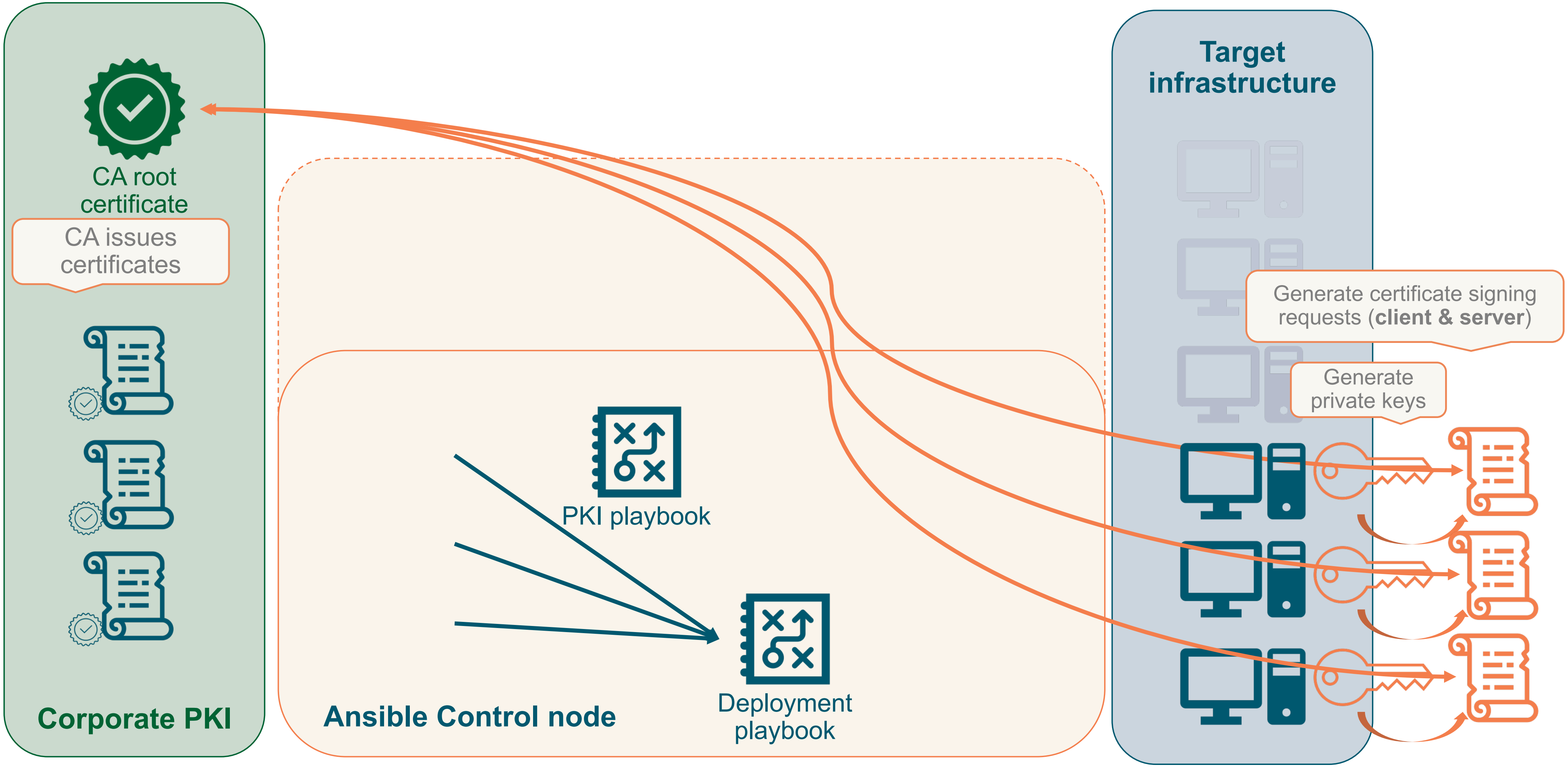
Components' role configuration

- Each component has its own _keystore argument:
  - A "proxy argument" to the java role's java_keystore argument
  - e.g for the repository role:
    - repository_keystore => java_keystore as java is a dependency of repository

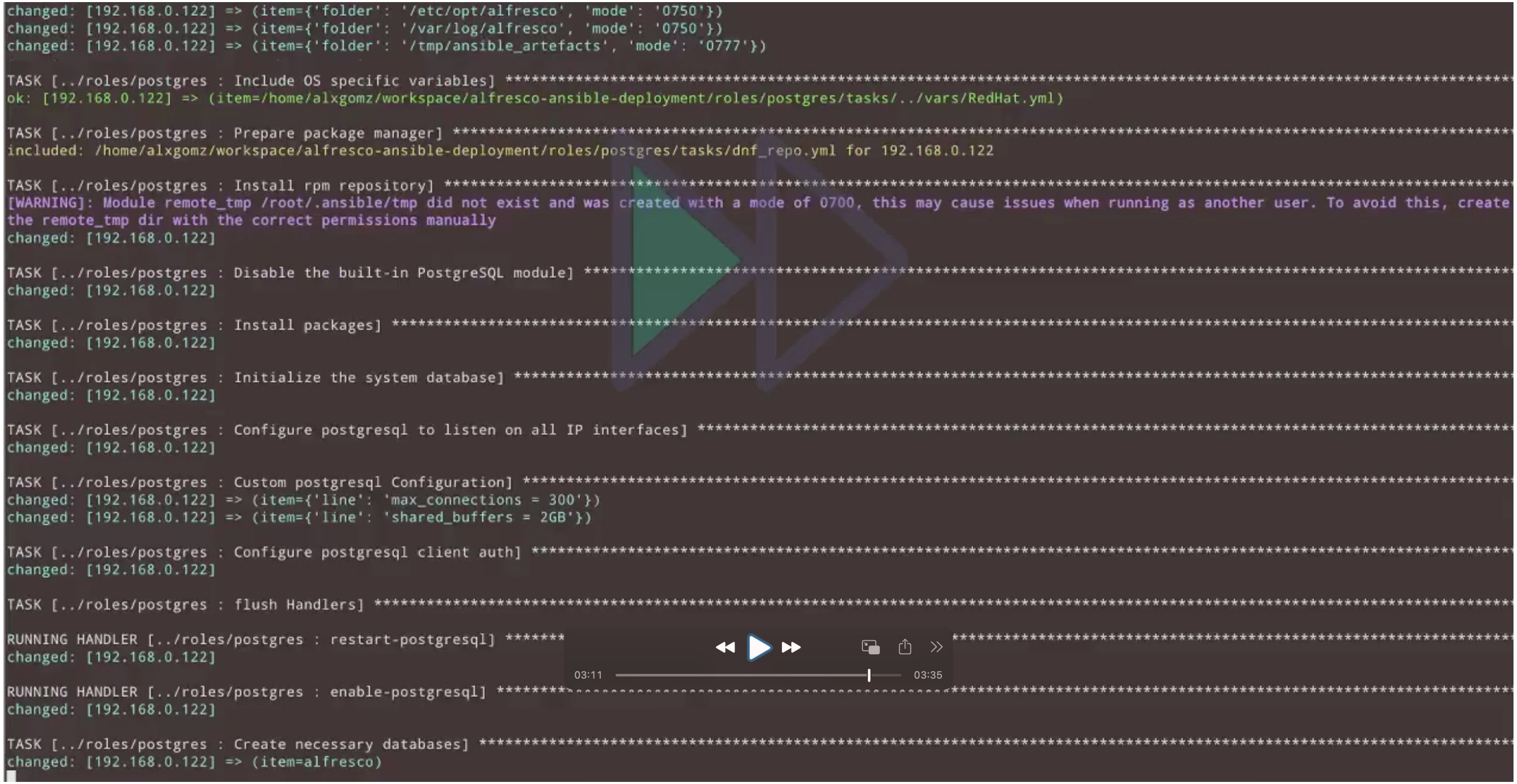
Corporate PKI (issuing a signing certificate)



Corporate PKI (issuing certificates)



Live demo video



```
changed: [192.168.0.122] => (item={'folder': '/etc/opt/alfresco', 'mode': '0750'})
changed: [192.168.0.122] => (item={'folder': '/var/log/alfresco', 'mode': '0750'})
changed: [192.168.0.122] => (item={'folder': '/tmp/ansible_artefacts', 'mode': '0777'})

TASK [../roles/postgres : Include OS specific variables] *******************************************************************************************************************
ok: [192.168.0.122] => (item=/home/alxgomz/workspace/alfresco-ansible-deployment/roles/postgres/tasks/../vars/RedHat.yml)

TASK [../roles/postgres : Prepare package manager] *************************************************************************************************************************
included: /home/alxgomz/workspace/alfresco-ansible-deployment/roles/postgres/tasks/dnf_repo.yml for 192.168.0.122

TASK [../roles/postgres : Install rpm repository] **************************************************************************************************************************
[WARNING]: Module remote_tmp /root/.ansible/tmp did not exist and was created with a mode of 0700, this may cause issues when running as another user. To avoid this, create
the remote_tmp dir with the correct permissions manually
changed: [192.168.0.122]

TASK [../roles/postgres : Disable the built-in PostgreSQL module] **********************************************************************************************************
changed: [192.168.0.122]

TASK [../roles/postgres : Install packages] ********************************************************************************************************************************
changed: [192.168.0.122]

TASK [../roles/postgres : Initialize the system database] ******************************************************************************************************************
changed: [192.168.0.122]

TASK [../roles/postgres : Configure postgresql to listen on all IP interfaces] *********************************************************************************************
changed: [192.168.0.122]

TASK [../roles/postgres : Custom postgresql Configuration] *****************************************************************************************************************
changed: [192.168.0.122] => (item={'line': 'max_connections = 300'})
changed: [192.168.0.122] => (item={'line': 'shared_buffers = 2GB'})

TASK [../roles/postgres : Configure postgresql client auth] ****************************************************************************************************************
changed: [192.168.0.122]

TASK [../roles/postgres : flush Handlers] **********************************************************************************************************************************

RUNNING HANDLER [../roles/postgres : restart-postgresql] *******************************************************************************************************************
changed: [192.168.0.122]

RUNNING HANDLER [../roles/postgres : enable-postgresql] ********************************************************************************************************************
changed: [192.168.0.122]

TASK [../roles/postgres : Create necessary databases] **********************************************************************************************************************
changed: [192.168.0.122] => (item=alfresco)
```

Checking truststore on deployed hosts

```
$ keytool -list -cacerts | grep alfresco

alfresco ansible imported ca (bf9b1a5b48e7e527ef15518174f19e620710493d), 16 juil. 2023, trustedCertEntry,

$ keytool -list -cacerts -rfc -alias 'alfresco ansible imported ca (bf9b1a5b48e7e527ef15518174f19e620710493d)' | openssl x509 -noout
-text

Certificate:

    Data:

        Version: 3 (0x2)

        Serial Number:

            37:37:09:8e:6c:58:8b:c6:98:7a:32:d5:aa:72:5d:a5:a5:17:5d:86

        Signature Algorithm: sha256WithRSAEncryption

        Issuer: CN = Hyland - Alfresco signing CA

        Validity

            Not Before: Jul 16 18:41:57 2023 GMT

            Not After : Jul 13 18:41:57 2033 GMT

        Subject: CN = Hyland - Alfresco signing CA
```

```
X509v3 extensions:
    X509v3 Key Usage: critical
        Certificate Sign
    X509v3 Basic Constraints: critical
        CA:TRUE
    X509v3 Subject Key
Identifier:

        57:91:E2:93:58:C0:65:F0:1D:93:0D:16:52:6C:BD:28:53:79:7D:8
B
```

Checking keystore on deployed hosts

```
$ keytool -list -keystore /etc/opt/alfresco/pki/192.168.0.121.keystore -alias 192.168.0.121 -rfc | openssl x509 -noout -text
 Entrez le mot de passe du fichier de clés :  bx)2HsdRYs5wX0YchdsNn5wGxC^KhN.WK
 Certificate:
     Data:
         Version: 3 (0x2)
         Serial Number:
             45:c3:64:a3:bc:65:0e:6e:33:c7:4c:36:69:28:4f:ef:36:21:df:69
         Signature Algorithm: sha256WithRSAEncryption
         Issuer: CN = Hyland - Alfresco signing CA
         Validity
             Not Before: Jul 16 18:42:19 2023 GMT
             Not After : Jul 13 18:42:19 2033 GMT
         Subject: CN = alioth.home
```

```
X509v3 extensions:
    X509v3 Subject Alternative Name:
        DNS:192.168.0.121, DNS:alioth
    X509v3 Key Usage:
        Digital Signature, Non Repudiation, Key Encipherment
    X509v3 Extended Key Usage:
        TLS Web Server Authentication, TLS Web Client
Authentication
    X509v3 Basic Constraints:
        CA:FALSE
```

Alternatives & useful tools

- EasyRSA
  - Offers a full fledge PKI management tooling through a set of shell scripts: https://github.com/OpenVPN/easy-rsa
- ssl-generator
  - Alfresco community provides a set of script to ease the configuration of everything which involves certificate generation
  - Supports more use case than the one currently supported in the Alfresco ansible playbook
  - Sources: https://github.com/Alfresco/alfresco-ssl-generator

mTLS on Kubernetes deployments

- Kubernetes is often used for microservices/service-oriented architectures
- Service mesh are becoming a standard plug-and-play solutions
- mTLS is a standard feature in a service mesh
- Managing cert and keystores on Kubernetes can be tricky
- It’s highly recommended to adopt a service mesh to provide mTLS support for any application running on Kubernetes

Stay connected

- https://hub.alfresco.com
- Alfresco Discord
- https://github.com/Alfresco/alfresco-ansible-deployment

Hyland™

Thanks for listening!