CAS STANDALONE versione 5

Pacchetti che servono:

1. Tomcat8
2. JDK8
3. CAS Overlay Template

1.) **Prima di procedere con tutte le configurazioni si deve generare una chiave per il server CAS.**

Eseguire il seguente comando per generare la chiave:

`keytool -keystore /opt/jdk1.8.0_101/jre/lib/security/cacerts -genkey -alias cas -keyalg RSA`

Importante inserire nel nome e cognome il nome del server(hostname) oppure localhost

2.) **Attivare SSL su tomcat. Modifica server.xml:**

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
             maxThreads="150" scheme="https" secure="true"
             clientAuth="false" sslProtocol="TLS"
             keystoreFile="/opt/jdk1.8.0_101/jre/lib/security/thekeystore"
             keystorePass="changeit"
             trustoreFile="/opt/jdk1.8.0_101/jre/lib/security/cacerts"  />
```

3.) **Installazione CAS Overlay Template**

- Scarica CAS dal: https://github.com/apereo/cas-overlay-template
- Dopo aver scompattato il pacchetto eseguire il commando nella cartella del cas:

`run build.sh package`

- Nella directory cas-overlay-master/target si trova il war file per il tomcat.
- In /etc/cas/config/users.properties modifica in `casuser=notused,ROLE_ADMIN,enabled` nel caso si voglia usera l'utente casuser
- Modifica il /etc/cas/config/cas.properties o /cas/WEB-INF/classes/application.properties:

```
 # CAS Server Context Configuration

cas.server.name=https://localhost:8443
cas.server.prefix=https://localhost:8443/cas

cas.host.name=localhost
server.context-path=/cas
server.port=8443

server.ssl.key-store=file:/opt/jdk1.8.0_101/jre/lib/security/thekeystore
server.ssl.key-store-password=changeit
server.ssl.key-password=changeit


management.contextPath=/status
management.security.enabled=true
management.security.roles=ACTUATOR,ADMIN,ROLE_ADMIN
management.security.sessions=if_required

cas.adminPagesSecurity.ip=127\.0\.0\.1
logging.config=file:/etc/cas/config/log4j2.xml

cas.serviceRegistry.watcherEnabled=true
cas.serviceRegistry.repeatInterval=120000
cas.serviceRegistry.startDelay=15000
cas.serviceRegistry.initFromJson=true
cas.serviceRegistry.config.location=file:/etc/cas/services

#cas.authn.accept.users=casuser::Mellon
cas.authn.accept.users=
#logging.level.org.apereo=DEBUG
```

```
cas.authn.file.separator=::
cas.authn.file.filename=file:///home/utente/users.txt
cas.serviceRegistry.initFromJson=true
```

- Per autentificare tramite una lista file TXT, si deve mettere la dependency nel file pom.xml.
- Scaricare nel cas/WEB-INF/lib il jar cas-server-support-generic-5.2.0-RC4.jar dal maven repository. https://mvnrepository.com/
- Ricorda di copiare il codice della dependecy come suggerito nel sito maven

```
<dependency>
    <groupId>org.apereo.cas</groupId>
    <artifactId>cas-server-support-generic</artifactId>
    <version>5.2.0-RC4</version>
    <scope>test</scope>
</dependency>
```

- Aggiungere nel webapps/cas/META-INF/maven/org.apereo.cas/cas-overlay/pom.xml

```
<dependency>
        <groupId>org.apereo.cas</groupId>
        <artifactId>cas-server-support-json-service-registry</artifactId>
        <version>${cas.version}</version>
    </dependency>
```

- Crea file json in /cas/WEB-INF/classes/services con il nome allservices-101.json con il contenuto

```
{
  "@class" : "org.apereo.cas.services.RegexRegisteredService",
  "serviceId" : "^(http|https)://.*",
  "name" : "allservices",
  "id" : 101,
  "accessStrategy" : {
    "@class" : "org.apereo.cas.services.DefaultRegisteredServiceAccessStrategy",
    "enabled" : true,
    "ssoEnabled" : true
  }
}
```

Attenzione! *Il nome del file deve essere name-id.json come specificato nello script.*

CAS su DB MONGO versione 6.0.5

Si parte dal progeto cas overlay template https://github.com/apereo/cas-overlay-template

Bisogna aggiungere le dipendenze di gradle nel file build.gradle

build.grandle

```
...
dependencies {
    compile "org.apereo.cas:cas-server-webapp${project.appServer}:${casServerVersion}"
    compile "org.apereo.cas:cas-server-support-mongo:${casServerVersion}"
    compile "org.apereo.cas:cas-server-support-mongo-ticket-registry:${casServerVersion}"
    compile "org.apereo.cas:cas-server-support-events-mongo:${casServerVersion}"
    compile "org.apereo.cas:cas-server-support-json-service-registry:${casServerVersion}"
    // compile "org.apereo.cas:cas-server-support-mongo-service-
registry:${casServerVersion}"
    // Other CAS dependencies/modules may be listed here...

}
...
```

Compilare con

```
./gradlew clean build
```

Copiare le configurazioni base con

```
./gradlew copyCasConfiguration
```

Editare i file in /etc/cas/config

cas.properties

```
cas.server.name=https://localhost:8443
cas.server.prefix=${cas.server.name}/cas

logging.config: file:/etc/cas/config/log4j2.xml


# Lasciare la seguente property vuota per avere l autenticazione su mongo
cas.authn.accept.users=


 cas.serviceRegistry.watcherEnabled=true
 cas.serviceRegistry.schedule.repeatInterval=120000
 cas.serviceRegistry.schedule.startDelay=15000
# Auto-initialize the registry from default JSON service definitions
 cas.serviceRegistry.initFromJson=true
# cas.serviceRegistry.managementType=DEFAULT|DOMAIN
#cas.serviceRegistry.json.location=classpath:/services
cas.serviceRegistry.json.location=file:/etc/cas/services

#############################################################
#### MONGO PASSWORD ENCODING - encoding delle password ####
#############################################################
cas.authn.mongo.passwordEncoder.type=BCRYPT
# cas.authn.mongo.passwordEncoder.characterEncoding=
# cas.authn.mongo.passwordEncoder.encodingAlgorithm=
# cas.authn.mongo.passwordEncoder.secret=
cas.authn.mongo.passwordEncoder.strength=10
#################################################################
#### MONGO USER STRUCTURE - mapping struttura dati utente ####
#################################################################
cas.authn.mongo.attributes=first_name,last_name
cas.authn.mongo.usernameAttribute=username
cas.authn.mongo.passwordAttribute=password
# cas.authn.mongo.principalIdAttribute=
# cas.authn.mongo.name=
#####################################################################
#### MONGO AUTHN - connessione a mongo per autenticazione cas ####
#####################################################################
# cas.authn.mongo.host=localhost
cas.authn.mongo.clientUri=mongodb://localhost:27017/cas?safe=true&w=1
# cas.authn.mongo.idleTimeout=30000
# cas.authn.mongo.port=27017
# cas.authn.mongo.dropCollection=false
# cas.authn.mongo.socketKeepAlive=false
# cas.authn.mongo.password=
# Depending on the feature at hand, CAS may decide to dynamically create its own collections
and ignore this setting.
cas.authn.mongo.collection=users
cas.authn.mongo.databaseName=cas
# cas.authn.mongo.timeout=5000
# cas.authn.mongo.userId=
# cas.authn.mongo.writeConcern=NORMAL
# cas.authn.mongo.authenticationDatabaseName=
# cas.authn.mongo.replicaSet=
# cas.authn.mongo.sslEnabled=false
```

```
# cas.authn.mongo.conns.lifetime=60000
# cas.authn.mongo.conns.perHost=10
################################################
### MONGO TICKETIING - salvataggio ticketing  ####
################################################
# cas.ticket.registry.mongo.host=localhost
cas.ticket.registry.mongo.clientUri=mongodb://localhost:27017/cas?safe=true&w=1
# cas.ticket.registry.mongo.idleTimeout=30000
# cas.ticket.registry.mongo.port=27017
# cas.ticket.registry.mongo.dropCollection=false
# cas.ticket.registry.mongo.socketKeepAlive=false
# cas.ticket.registry.mongo.password=
# Depending on the feature at hand, CAS may decide to dynamically create its own collections
and ignore this setting.
# se si decommenta questa SI SPACCA
# perchè tickets estende direttamente BaseMongoDbProperties e non
SingleCollectionMongoDbProperties
# cas.ticket.registry.mongo.collection=cas-tickets
cas.ticket.registry.mongo.databaseName=cas
# cas.ticket.registry.mongo.timeout=5000
# cas.ticket.registry.mongo.userId=
# cas.ticket.registry.mongo.writeConcern=NORMAL
# cas.ticket.registry.mongo.authenticationDatabaseName=
# cas.ticket.registry.mongo.replicaSet=
# cas.ticket.registry.mongo.sslEnabled=false
# cas.ticket.registry.mongo.conns.lifetime=60000
# cas.ticket.registry.mongo.conns.perHost=10
###############################################
#### MONGO AUDIT - auditing degli accessi ####
###############################################
# cas.events.mongo.host=localhost
cas.events.mongo.clientUri=mongodb://localhost:27017/cas?safe=true&w=1
# cas.events.mongo.idleTimeout=30000
# cas.events.mongo.port=27017
# cas.events.mongo.dropCollection=false
# cas.events.mongo.socketKeepAlive=false
# cas.events.mongo.password=
# Depending on the feature at hand, CAS may decide to dynamically create its own collections
and ignore this setting.
cas.events.mongo.collection=audits
cas.events.mongo.databaseName=cas
# cas.events.mongo.timeout=5000
# cas.events.mongo.userId=
# cas.events.mongo.writeConcern=NORMAL
# cas.events.mongo.authenticationDatabaseName=
# cas.events.mongo.replicaSet=
# cas.events.mongo.sslEnabled=false
# cas.events.mongo.conns.lifetime=60000
# cas.events.mongo.conns.perHost=10
#############################################################################
#### MONGO SERVICE REGISTRY - persistenza delle configurazioni dei service ####
#############################################################################
# cas.serviceRegistry.mongo.host=localhost
#cas.serviceRegistry.mongo.clientUri=mongodb://localhost:27017/cas?safe=true&w=1
# cas.serviceRegistry.mongo.idleTimeout=30000
# cas.serviceRegistry.mongo.port=27017
# cas.serviceRegistry.mongo.dropCollection=false
# cas.serviceRegistry.mongo.socketKeepAlive=false
# cas.serviceRegistry.mongo.password=
# Depending on the feature at hand, CAS may decide to dynamically create its own collections
and ignore this setting.
#cas.serviceRegistry.mongo.collection=services
#cas.serviceRegistry.mongo.databaseName=cas
# cas.serviceRegistry.mongo.timeout=500
# cas.serviceRegistry.mongo.userId=
```

```
# cas.serviceRegistry.mongo.writeConcern=NORMAL
# cas.serviceRegistry.mongo.authenticationDatabaseName=
# cas.serviceRegistry.mongo.replicaSet=
# cas.serviceRegistry.mongo.sslEnabled=false
# cas.serviceRegistry.mongo.conns.lifetime=60000
# cas.serviceRegistry.mongo.conns.perHost=10

cas.logout.followServiceRedirects=true
```

In log4j2.xml modificare solo la baseDir

```
...
<Property name="baseDir">/opt/tomcat8/logs</Property>
...
```

Creare una cartella services in /etc/cas/ creare i file per i servizi come da esempio:

CollaudoEc-10000001.json

```
{
  "@class" : "org.apereo.cas.services.RegexRegisteredService",
  "serviceId" : "^(http|https)://collaudo-ibc-ec.xdams.org.*",
  "name" : "CollaudoEc",
  "theme" : "ec",
  "id" : 10000001,
  "description" : "Autenticazione per EC/xDams di Collaudo",
  "evaluationOrder" : 1
}
```

Il nome del file deve fare match con il name e l'id all'interno del json

Volendo si può usare un template generico per fare sempre match

allservices-101.json

```
{
  "@class" : "org.apereo.cas.services.RegexRegisteredService",
  "serviceId" : "^(http|https)://.*",
  "name" : "allservices",
  "id" : 101,
  "accessStrategy" : {
    "@class" : "org.apereo.cas.services.DefaultRegisteredServiceAccessStrategy",
    "enabled" : true,
    "ssoEnabled" : true
  }
}
```

Per personalizzare i temi bisogna creare un file di properties in .../webapps/cas/WEB-INF/classes/ con il nome [theme].properties

ec.properties

```
cas.standard.css.file=/themes/ec/css/cas.css
cas.javascript.file=/themes/ec/js/cas.js
```

All'iterno della cartella /opt/tomcat8/webapps/cas/WEB-INF/classes/static/themes/ copiare la cartella apereo con il nome del tema nb: deve fare match con il file properties e alla fine il risultato sarà:

```
total 8
```

```
drwxr-x--- 5 cas cas 4096 Oct 29 15:45 apereo
drwxr-x--- 5 cas cas 4096 Oct 29 15:45 ec
```