

Real life challenges and configurations when implementing HCL Sametime V12.0.1 FP1

Herwig W. “Wickerl” Schauer
Technical Lead - HCL Lab-Services Austria

Erik Schwalb
Technical Advisor – HCL Software Germany



Disclaimer

HCL's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice and at HCL's sole discretion.

Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract.

The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Performance is based on measurements and projections using standard HCL benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.

HCLSoftware

Herwig W. "Wickerl" Schauer

works as a Technical Lead at HCL Lab-Services and is based in Austria.

Looking back at 25 years in the services business, being with Lotus Professional Services as well as almost 20 years at IBM Software Group Services.
He works at HCL Lab-Services since 2020 and currently primarily supports several Austrian key customer accounts through the Advanced Technical Services (ATS) program.

<mailto:wickerl@pnp-hcl.com>
@TheWickerl



HCLSoftware

Erik Schwalb

Working as a Technical Advisor at HCL Software in Germany, Erik is responsible for consulting and sales of the HCL Digital Solutions portfolio with a focus on Domino, Sametime and Connections.

He has 30+ years of experience in various technical sales roles at Lotus, IBM and HCL.

Erik started working with Sametime in version 1 and later co-authored the Lotus Sametime 2.0 Deployment Guide. More recently he created a Do-it-yourself monitoring solution for HCL Sametime Meetings on Docker and keeps his hands-on experience current by maintaining the HCL Sametime Sandbox and several other Sametime environments.

Contact: erik.schwalb@hcl.com



HCLSoftware

Creating Truststores and Keystores for Sametime

Creating Truststores and Keystores for Sametime

It's all about PKCS#12, but ...

Sametime 12.0.1 requires PKCS#12 file format for Truststores and Keystores.

Due to security enhancements, PKCS#12 files created with newer versions of Java will not work with older versions of Java !



Either use an older version of OpenJDK keytool (eg OpenJDK 8) or add the `-J-Dkeystore.pkcs12.legacy` parameter when using newer versions of keytool !

Example for converting an existing PKCS#12 file to the desired file format:

```
keytool -importkeystore -srckeystore <existingkeystore.p12> -srcstoretype PKCS12 \  
-srcstorepass <existingkeystorepassword> -destkeystore <sametimekeystore.p12> \  
-deststoretype PKCS12 -deststorepass <sametimekeystorepassword> \  
-J-Dkeystore.pkcs12.legacy
```

https://help.hcltechsw.com/sametime/1201/admin/t_create_truststore.html

HCLSoftware

Securing connections between Sametime servers and LDAP

Securing connections between Sametime servers and LDAP

Important things to know

1. Requires a properly created PKCS#12 truststore containing the TLS certificates of your LDAPs server.
We already learned about that earlier ...
2. Requires specific TLS Ciphers to work correctly, ie for TLS1.2:
RSA_WITH_AES_256_GCM_SHA384 (0x009D)
RSA_WITH_AES_128_CBC_SHA (0x002F)



Sametime 12.0 TLS required ciphers to connect to Domino 12.0.2 LDAP

https://support.hcltechsw.com/csm?id=kb_article&sysparm_article=KB0099644

https://help.hcltechsw.com/sametime/1201/admin/securing_connections_sametime_community_and_ldap.html

Securing connections between Sametime servers and LDAP

Docker configuration

You can expand your configuration for your LDAP truststore with a `tlsldap.env` file (as documented) or simply add the required parameters to your `custom.env` file:

```
STI__Config__STLDAP_TLS_TRUST_STORE_TYPE=p12
STI__Config__STLDAP_TLS_TRUST_STORE_FILE=/local/notesdata/ldaptruststore.p12
STI__Config__STLDAP_TLS_TRUST_STORE_PASSWORD=<sametimetruststorepassword>
```

Important:

- `STI__Config__STLDAP_TLS_TRUST_STORE_FILE`
describes the path to your PKCS#12 file as seen by the community container, it does not refer to the path of the file on the host system.
- File permissions
Ensure that your PKCS#12 file permissions are not restricted to root only, otherwise the community container won't be able to read the file.

https://help.hcltechsw.com/sametime/1201/admin/securing_ldap_docker.html

Securing connections between Sametime servers and LDAP

Docker configuration (cont.)

Change your docker-compose.yml to map your PKCS#12 file from the host system to the community container:

```
[...]
community:
  image: hclcr.io/st/chat-server:${BUILD_LEVEL}
  restart: ${RESTART_POLICY}
  env_file: custom.env
  environment:
    - JWT_SECRET_ENV=${JWT_APP_SECRET}
[...]
networks:
  - sametime.test
volumes:
  - <sametimekeystore.p12>:/local/notesdata/ldaptruststore.p12
[...]
```

Important:

- <sametimekeystore.p12> describes the path to your PKCS#12 file on the host system. Either use an absolute path or a relative path (relative to the directory containing your docker-compose.yml file).
- The right part of your volumes configuration (after the colon) must match the path as specified earlier in the `STI__Config__STLDAP_TLS_TRUST_STORE_FILE` parameter.

https://help.hcltechsw.com/sametime/1201/admin/securing_ldap_docker.html

Securing connections between Sametime servers and LDAP

Kubernetes configuration

In a Kubernetes configuration the LDAP truststore is stored in a Kubernetes Secret.

```
kubect1 -n <sametimenamespace> create secret generic <ldapsecretname> \
--from-literal=KeyStorePassword=<sametimetruststorepassword> \
--from-file=ldaptruststore.p12=<sametimetruststore.p12>
```

Important:

- `<ldapsecretname>` is the name of the secret you refer to in your values.yaml file.
The default value in the Sametime documentation is "ldap-secret".
- Have a closer look at the `--from-file` option:
The value after the first equal sign specifies the filename in the Kubernetes Secret (which has to be "ldaptruststore.p12").
The value after the second equal sign specifies the path to the PKCS#12 file you want to import into the Secret.
So the file imported can actually have a name different than "ldaptruststore.p12".

https://help.hcltechsw.com/sametime/1201/admin/securing_ldap_kubernetes.html

Securing connections between Sametime servers and LDAP

Kubernetes configuration (cont.)

Change your helm/values.yaml file to refer to your Kubernetes Secret and enable LDAPs.

```
global:
  [...]
  ldapHost: <ldapsfqhn>
  # ldapPort: 389
  ldapPort: <ldapsport>
  # ldapTls: false
  ldapTls: true
  ldapConfigSecret: <ldapsecretname>
  [...]
```

Remarks:

<ldapsport> specify your LDAP server's secure port. The default port for LDAPs is 636.

https://help.hcltechsw.com/sametime/1201/admin/securing_ldap_kubernetes.html

HCLSoftware

Setting up SSO using LTPA

Setting up SSO using LTPA

Additional things to consider ...

In real life scenarios you will have to deal with existing LTPA Keys, generated in other systems, eg

```
#IBM WebSphere Application Server key file
#Tue Jan 24 15:26:53 CET 2017
com.ibm.websphere.CreationDate=Tue Jan 24 15\:26\:53 CET 2017
com.ibm.websphere.ltpa.version=1.0
com.ibm.websphere.ltpa.3DESKey=T+mMs2ekH83ev0csh8FpwhFduUkonkB33/cGjIn+mR4\=
com.ibm.websphere.CreationHost=connections
com.ibm.websphere.ltpa.PrivateKey=28mfdluonFfUPuZzX9RRScpfBQRJJ5Xan9m1SDLdpXwQAWqvDVIZCog9MF5ithR9NaAVYWVTaaToxVV6mWA0
XD+vfWmVumUtPfcJzPhOGS+gHetcqdPOQLmCkuaq0tvfZ8uaxhLttZ7pLCuKST7B9ogjks7bfOSOs3HDUl2tmlJRWLlnUiieE6Vd7dCOhqD/garvyzAD6Z
R9lxqEzi+kEQ68i9DprHxG+t4PqYT6ygQaEyLTYwTzBa96zrirdoDuL5sizQ9FqSc07lX4AnaLnRdR96l1jpuVW+l2uNoABIuxE7seFG2Mb8ratszaQd8d
OLcMFom3yeaSVKmCXXy31EbWs4V2AsvdXVoU8DL4xts\=
com.ibm.websphere.ltpa.Realm=defaultWIMFileBasedRealm
com.ibm.websphere.ltpa.PublicKey=AKulAHlIHCGpiwOKAMVaKqBGO8GikB7U3n7zMAB6O2E459Tm6vgMzChrnSmY4IY6U20VpkbXzirF4stoi2PMr
uC0DlktaeTXsMt7kfe6yCCMALi8xiB35hQoOJ4ktQi4YtIpBvBWtDCjbxGYZa0wmAK+0ovSUZUaKRIm0OqkxPtzAQAB
```

Important:

An existing LTPA Key, created in another system like Connections, might contain a different LTPA Realm than the default Realm of WebSphere Liberty ("defaultRealm").

To integrate Sametime with existing LTPA configurations you also have to properly configure the Realm !

https://help.hcltechsw.com/sametime/1201/admin/enabling_sso_ltpa.html

Setting up SSO using LTPA Docker

Change your .env file:

```
ENABLE_LTPA=true
LTPA_KEYS_FILE_PATH=<ltpakeyfilepath>
LTPA_KEYS=/ltpa-config/ltpa.keys
LTPA_KEYS_PASSWORD=<ltpakeyfilepassword>
LTPA_REALM=<ltparealm>
```

Important:

- <ltpakeyfilepath> specifies the path to your LTPA Key file.
The actual file name does not matter.
- Do not change the value of the parameter LTPA_KEYS.
This is how the container refers to the LTPA Key file and must not be changed.
- If your LTPA Realm differs from the default name "defaultRealm", specify your existing LTPA Realm name in the LTPA_REALM variable as <ltparealm>.
- You have to make additional changes to your docker-compose.yml for configuring your LTPA Realm.

Change the value STI_ST_BB_NAMES_ST_AUTH_TOKEN in your custom.env file to

```
STI_ST_BB_NAMES_ST_AUTH_TOKEN=Fork:Jwt,Ltpa
```

https://help.hcltechsw.com/sametime/1201/admin/ltpa_configure_docker.html

Setting up SSO using LTPA

Docker (cont.)

If you had to specify a different LTPA Realm name, you have to change your docker-compose.yml as follows:

```
[...]
  auth:
    image: hclcr.io/st/meetings-auth.node:${BUILD_LEVEL}
    restart: ${RESTART_POLICY}
    env_file: custom.env
    volumes:
      - ${LTPA_KEYS_FILE_PATH}:/ltpa-config/ltpa.keys:Z
    environment:
[...]
```

- LTPA_KEYS
- LTPA_KEYS_PASSWORD
- **LTPA_REALM**

```
[...]
```

https://help.hcltechsw.com/sametime/1201/admin/ltpa_configure_docker.html

Setting up SSO using LTPA

Kubernetes

In a Kubernetes configuration the LTPA Key File is stored in a Kubernetes Secret.

```
kubectl -n <sametimenamespace> create secret generic <ltpasecretname> \
--from-file=ltpa.keys=<ltpakeyfilepath>
```

Important:

- `<ltpasecretname>` is the name of the secret you refer to in your values.yaml file.
The default value in the Sametime documentation is "ltpa-keys".
- Have a closer look at the `--from-file` option:
The value after the first equal sign specifies the filename in the Kubernetes Secret (which has to be "ltpa.keys").
The value after the second equal sign specifies the path to the LTPA Keys file you want to import into the Secret.
So the file being imported can have a name different than "ltpa.keys".

https://help.hcltechsw.com/sametime/1201/admin/ltpa_configure_kubernetes.html

Setting up SSO using LTPA

Kubernetes (cont.)

Change your helm/values.yaml file to enable LTPA, if required also specify your existing LTPA Realm:

```
global:
  [...]
  enableLtpa : true
  ltpaRealm: <ltparealm>
  [...]
```

Add the base64 encoded LTPA Key Password to helm/templates/sametime-secrets.yaml:

```
data:
  [...]
  LtpaKeysPassword: <ltpakeyfilepasswordbase64>
  [...]
```

Remarks:

Calculate <ltpakeyfilepasswordbase64> **by executing** `echo -n <ltpakeyfilepassword> | base64` **where** <ltpakeyfilepassword> **is your plain text LTPA Key Password.**

https://help.hcltechsw.com/sametime/1201/admin/ltpa_configure_kubernetes.html

HCLSoftware

Integrating with other applications

Integrating with other applications

Things to know ...

Beginning with HCL Sametime 12.0 the legacy web-client is not enabled by default, but it can be enabled when needed for integration with other products, eg HCL Connections, HCL Verse, ...

Things to do:

- Enable (legacy) web-client integration
- Enable content security headers
See <https://content-security-policy.com/frame-ancestors/> for details.



HCL Connections



HCL Verse

Remarks:

SSO is required for integrating with other products, ie LTPA or SAML.

https://help.hcltechsw.com/sametime/1201/admin/verse_integration.html

Integrating with other applications

Docker

Enable the (legacy) web-client in docker-compose.yml:

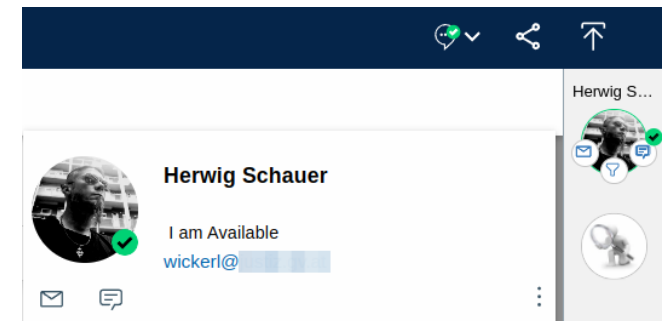
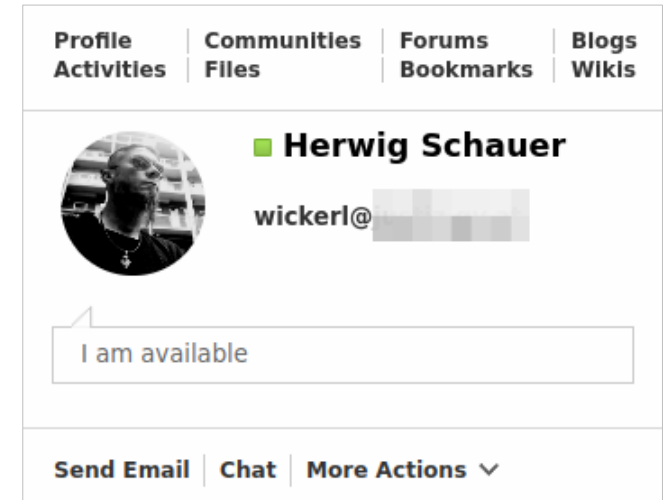
```
[...]
  proxy:
    image: hclcr.io/st/chat-proxy:${BUILD_LEVEL}
    restart: ${RESTART_POLICY}
    env_file: custom.env
    volumes:
      - proxy-workspace:/workspace/proxy-storage
    environment:
      - JAVA_TOOL_OPTIONS=-XX:MaxDirectMemorySize=64M -XX:MaxMetaspaceSize=192M
      - SAMETIME_EXTERNAL_WARINTEGRATION=true
[...]
```

Enable content security policy by adding the following parameter to custom.env, eg:

```
[...]
CONTENT_SECURITY_POLICY=frame-ancestors https://*.<yourdomain.tld>
[...]
```

https://help.hcltechsw.com/sametime/1201/admin/verse_integration_docker.html

https://help.hcltechsw.com/sametime/1201/admin/verse_integration_contentsecurity_docker.html

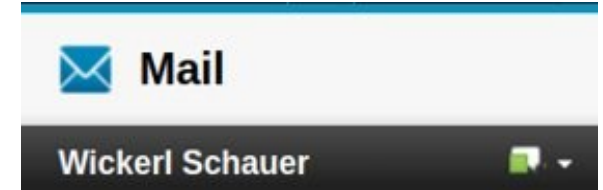


Integrating with other applications

Kubernetes

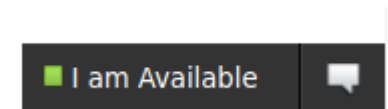
Enable the (legacy) web-client in values.yaml:

```
global:
  [...]
  enableLegacyChatClient: true
  [...]
```



Enable content security policy by adding the following parameter to values.yaml, eg:

```
global:
  [...]
  enableLegacyChatClient: true
  contentSecurityPolicy: frame-ancestors https://*.<yourdomain.tld>
  [...]
```



https://help.hcltechsw.com/sametime/1201/admin/verse_integration_kubernetes.html
https://help.hcltechsw.com/sametime/1201/admin/verse_integration_contentsecurity_kubernetes.html

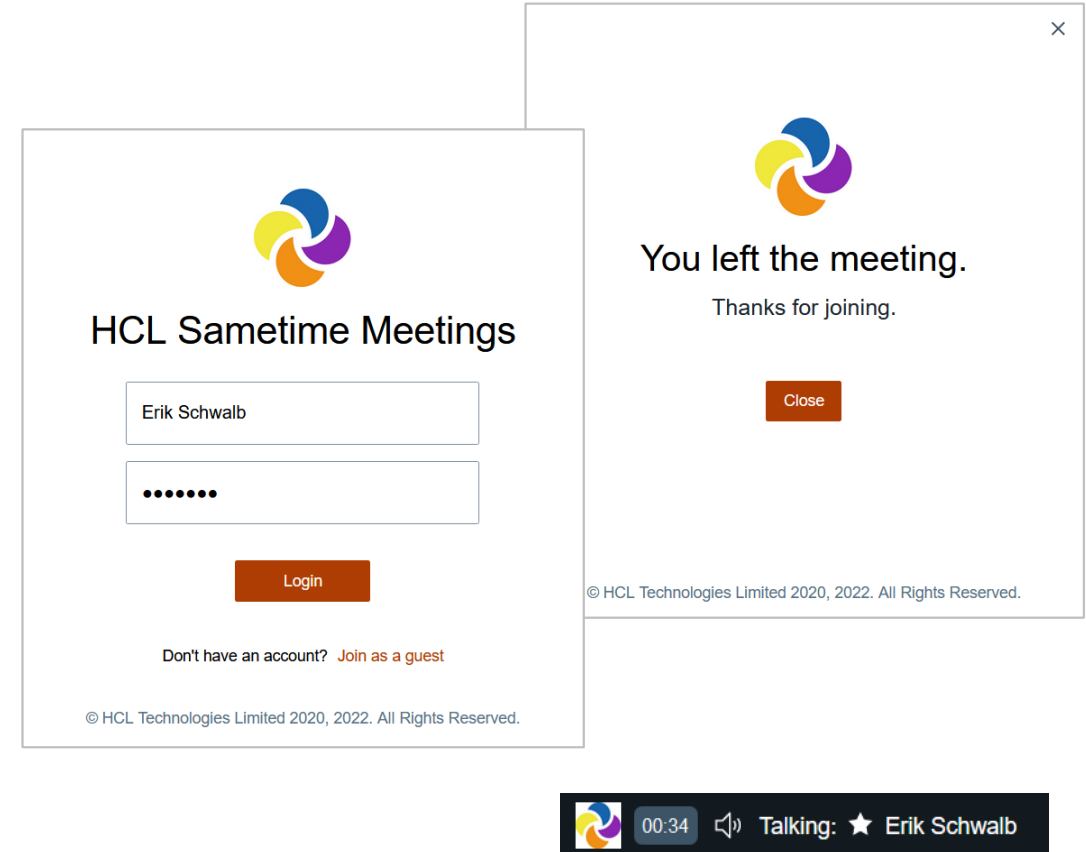
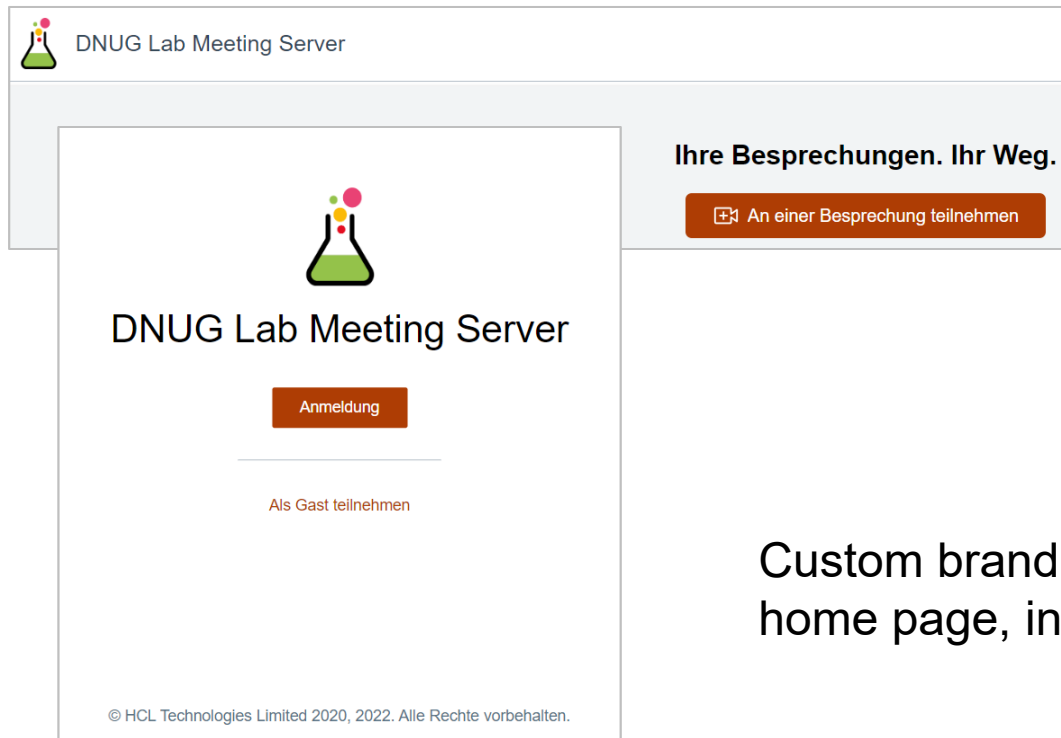
HCLSoftware

Branding

Branding

You can customize different elements of the Meeting Web UI

- Define a custom product / meeting service name
- Replace the Sametime product logo with your company logo
- Use a custom meeting background image



Custom branding can be displayed on the login page, on the meeting home page, in a meeting, on the logout page and in meeting reports.

Adding corporate branding to Sametime Meetings on Docker

Settings in `custom.env`

```
REACT_APP_PRODUCT_NAME=DNUG Lab Meeting Server
REACT_APP_PRODUCT_LOGO=/images/branding/dnug-lab-logo-large.png
REACT_APP_MEETING_BANNER_IMAGE=/images/branding/dnug-lab-logo-large.png
REACT_APP_PRODUCT_LOGO_URL=https://sametime.lab.dnug.eu/images/branding/dnug-lab-logo-large.png
REACT_APP_MEETING_BACKGROUND_IMAGE=/images/branding/dnug-background.jpg
```

`/images/branding` in `custom.env` corresponds to `./sametime-config/web/branding`

```
[root@sametime sametime]# ls -l ./sametime-config/web/branding
total 1848
-rw-r--r--. 1 root root 1825010 Dec  6 11:02 dnug-background.jpg
-rw-r--r--. 1 root root    7398 Dec  6 11:02 dnug-lab-large.png
-rw-r--r--. 1 root root   12221 Dec  6 11:02 dnug-lab-logo-large.png
-rw-r--r--. 1 root root    5703 Dec  6 11:02 dnug-lab-medium.png
-rw-r--r--. 1 root root    2977 Dec  6 11:02 dnug-lab-small.png
-rw-r--r--. 1 root root   30321 Dec  6 11:02 dnug-lab-xlarge.png
```

Alternatively, you can use a URL that points to an accessible image to specify your custom meeting background image:

```
REACT_APP_MEETING_BACKGROUND_IMAGE=https://mycompany.com/assets/theme.png
```

Adding corporate branding to Sametime Meetings on Kubernetes

Specify your custom branding settings in the `global:` section of `values.yaml`

Define your custom product / meeting service name

`productName: <YourCustomProductName>`

Use a custom logo

`productLogo: /images/branding/<your_logo_file>`

Use a custom meeting banner image

`meetingBannerImage: /images/branding/<your_banner_image_file>`

Use a custom meeting background image

`meetingBackgroundImage: /images/branding/<your_background_image_file>`

Alternatively, you can use an accessible URL that points to your custom logo and custom images. Then you do not have to copy these files to the persistent volume:

`productLogo: "http://mycompany.com/assets/<your_logo_file>"`

Custom branding settings begin

`productLogo: "https://sametime.lab.dnug.eu/images/branding/dnug-lab-logo-large.png"`

Custom branding settings end

```
[root@st10srv1 sametime]# ls -l /local/lab/storage/
total 0
drwxr-xr-x. 2 root root 6 May 1 19:49 backgrounds
drwxr-xr-x. 2 root root 6 May 1 19:49 downloads
drwxr-xr-x. 2 root root 6 May 1 19:49 files
drwxr-xr-x. 2 root root 6 May 1 19:49 quarantine
drwxr-xr-x. 2 centos centos 33 May 1 19:48 recordings
drwxr-xr-x. 2 root root 6 May 1 19:49 reports
drwxr-xr-x. 2 root root 6 May 1 19:49 server-backgrounds
drwxr-xr-x. 2 root root 6 May 1 19:49 server-branding
[root@st10srv1 sametime]#
```

HCLSoftware

Migrating contact lists (vpuserinfo.nsf)

Migrating contact lists (vpuserinfo.nsf)

What do you need to know ?

Starting with HCL Sametime 12.0 contact list data is now also stored in MongoDB as Community and Proxy is now containerized.

Existing contact list data stored in vpuserinfo.nsf must be migrated to MongoDB.



Important:

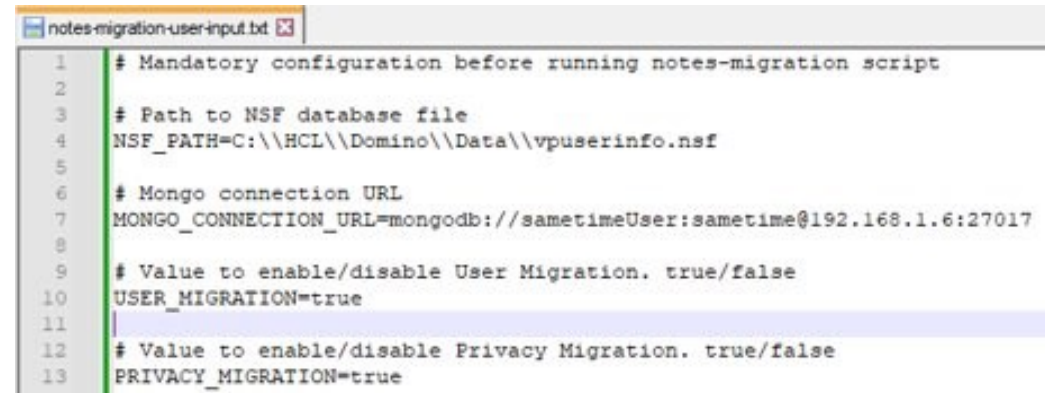
- Required tooling is part of HCL Sametime since 12.0 (notes-migration.zip) and available for Windows and Linux.
- Supports HCL Sametime 9.x, 10.x and 11.x.
Prior versions are not supported due to outdated Java versions.
- Designed to be unzipped and run directly on the HCL Sametime Server.
- Be aware of limited character encoding capabilities for Mongo user and Mongo password.
Currently only %, @, :, /, ?, #, [and] are supported characters in the migration scripts provided.

https://help.hcltechsw.com/sametime/1201/admin/migrating_upgrading.html#ariaid-title4

Migrating contact lists (vpuserinfo.nsf)

Running the migration tool - Windows

- The migration tool is designed to be **unzipped into the Domino Program Directory**.
- The notes-migration.bat script can only be **executed out of the Domino Program Directory**.
- Change **notes-migration-user-input.txt** to match your environment.
- If you specify a full path for **NSF_PATH**, either use **** or **/** as delimiter for the script to execute correctly.
- **Run** the notes-migration.bat script in an **elevated ("Administrator") command window**.
- Don't mind the "additional output" after successful execution... ;-)



```
1 # Mandatory configuration before running notes-migration script
2
3 # Path to NSF database file
4 NSF_PATH=C:\\HCL\\Domino\\Data\\vpuserinfo.nsf
5
6 # Mongo connection URL
7 MONGO_CONNECTION_URL=mongodb://sametimeUser:sametime@192.168.1.6:27017
8
9 # Value to enable/disable User Migration. true/false
10 USER_MIGRATION=true
11
12 # Value to enable/disable Privacy Migration. true/false
13 PRIVACY_MIGRATION=true
```

https://help.hcltechsw.com/sametime/1201/admin/migrating_upgrading.html#ariaid-title4

Migrating contact lists (vpuserinfo.nsf)

Migration tool output - Windows

```
Administrator: Command Prompt
Trace: Tue Mar 21 10:44:12 CET 2023 NotesUtils getCustomDataAttribute() fieldName: 0 fieldType: UbqOpaque headerSize: 4
Trace: Tue Mar 21 10:44:12 CET 2023 Attribute 0 value:

Trace: Tue Mar 21 10:44:12 CET 2023 Size: 106
Trace: Tue Mar 21 10:44:12 CET 2023 Header: 6a 0 0 0 0 68 56 65 72 73 69 6f 6e 3d 33 2e 31 2e 33 d
Trace: Tue Mar 21 10:44:12 CET 2023 NotesUtils getCustomDataAttribute() fieldName: 8194 fieldType: UbqOpaque headerSize: 4
Trace: Tue Mar 21 10:44:12 CET 2023 Attribute 8194 value:

Trace: Tue Mar 21 10:44:12 CET 2023 Size: 97
Trace: Tue Mar 21 10:44:12 CET 2023 Header: 61 0 0 0 3c 3f 78 6d 6c 20 76 65 72 73 69 6f 6e 3d 22 31
Trace: Tue Mar 21 10:44:12 CET 2023 NotesUtils getCustomDataAttribute() fieldName: 8195 fieldType: UbqOpaque headerSize: 4
Trace: Tue Mar 21 10:44:12 CET 2023 Attribute 8195 value:

Trace: Tue Mar 21 10:44:12 CET 2023 Size: 391
Trace: Tue Mar 21 10:44:12 CET 2023 Header: 87 1 0 0 3c 3f 78 6d 6c 20 76 65 72 73 69 6f 6e 3d 22 31
Userinfo records migration progress 100% <[33m???<[0m 37591/37591 (0:12:22 / 0:00
User migration completed.
37591 userinfo records are migrated.
Privacy records migration progress 100% <[33m????<[0m 69/69 (0:00:00 / 0:00:00)
Privacy migration completed.
69 privacy records are migrated.
The syntax of the command is incorrect.
The syntax of the command is incorrect.
The syntax of the command is incorrect.
The syntax of the command is incorrect.
The syntax of the command is incorrect.
```

https://help.hcltechsw.com/sametime/1201/admin/migrating_upgrading.html#ariaid-title4

Migrating contact lists (vpuserinfo.nsf)

Running the migration tool - Linux

- The migration tool is designed to be **unzipped into the Domino Binaries Directory**.
- **Change setenv.sh** to match your environment.
- **Source setenv.sh** by executing: `source ./setenv.sh`
- **Run the notes-migration.sh** script by executing: `./notes-migration.sh`

```
PROGRAMDIR=/opt/hcl/domino/notes/latest/linux
LOTUSDIR=$PROGRAMDIR/../../../../bin
DATADIR=/local/notesdata
PATH=/usr/bin:/bin:$PROGRAMDIR:/usr/ucb
export PATH
export PROGRAMDIR

Notes_ExecDirectory=$PROGRAMDIR
export Notes_ExecDirectory

export CLASSPATH=/opt/hcl/domino/notes/latest/linux/jvm/lib/ext/Notes.jar
export LD_LIBRARY_PATH=$PROGRAMDIR:$LD_LIBRARY_PATH:$PROGRAMDIR/STOpenSSL:$PROGRAMDIR/sticc
```

https://help.hcltechsw.com/sametime/1201/admin/migrating_upgrading.html#ariaid-title4

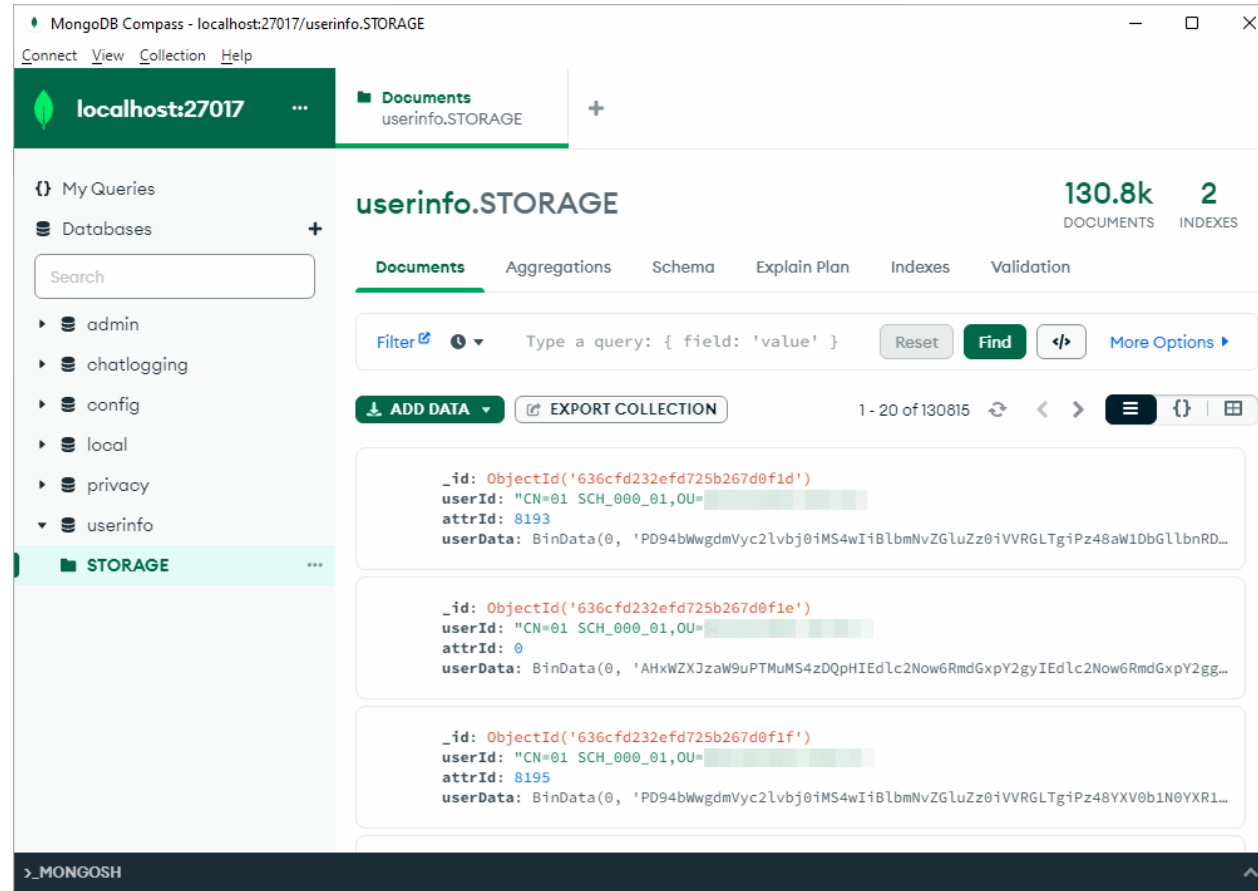
Migration tool output - Linux

```
Trace: Tue Mar 21 12:31:55 UTC 2023 Size: 97  
Trace: Tue Mar 21 12:31:55 UTC 2023 Header: 61 0 0 0 3c 3f 78 6d 6c 20 76 65 72 73 69 6f 6e 3d 22 31  
Trace: Tue Mar 21 12:31:55 UTC 2023 NotesUtils getCustomDataAttribute() fieldName: 8195 fieldType: UbqOpaque headerSize: 4  
Trace: Tue Mar 21 12:31:55 UTC 2023 Attribute 8195 value:  
  
Trace: Tue Mar 21 12:31:55 UTC 2023 Size: 391  
Userinfo records migration progress 100% | ████████████████████████████████████████████████████████████ | 37591/37591 (0:11:28 / 0:00:00)  
User migration completed.  
37591 userinfo records are migrated.  
Privacy records migration progress 100% | ████████████████████████████████████████████████████████████ | 69/69 (0:00:00 / 0:00:00)  
Privacy migration completed.  
69 privacy records are migrated.
```

https://help.hcltechsw.com/sametime/1201/admin/migrating_upgrading.html#ariaid-title4

Migrating contact lists (vpuserinfo.nsf)

Migrated contact lists in MongoDB



https://help.hcltechsw.com/sametime/1201/admin/migrating_upgrading.html#ariaid-title4

HCLSoftware

Upgrading MongoDB

Upgrading MongoDB

Considerations

Sametime 11.5/11.6 supports MongoDB 3.5 or 4.2

https://support.hcltechsw.com/csm?id=kb_article&sysparm_article=KB0082513

https://support.hcltechsw.com/csm?id=kb_article&sysparm_article=KB0089405

Sametime 12.0 supports MongoDB 4.4 or later (on best effort)

https://support.hcltechsw.com/csm?id=kb_article&sysparm_article=KB0097599



Sametime 12.0.1 supports MongoDB 4.2, 4.4, 5, 6 or later (on best effort)

https://support.hcltechsw.com/csm?id=kb_article&sysparm_article=KB0100619

Upgrading HCL Sametime to 12.0.x or higher requires the migration of contact lists from vuserinfo.nsf to MongoDB. Given that requirement, you might also have/want to upgrade your existing MongoDB to a newer release.

<https://www.mongodb.com/docs>

Upgrading MongoDB

In-place approach

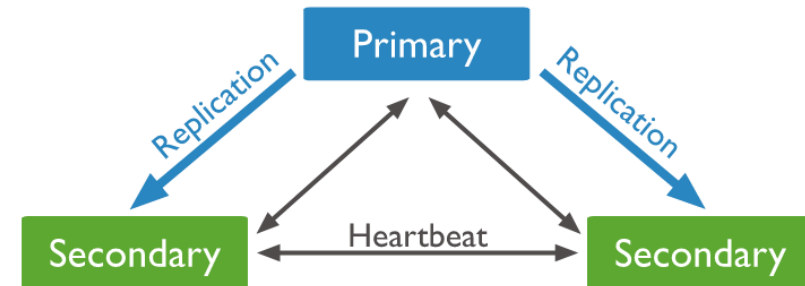
Important:

Always follow the official MongoDB documentation for detailed upgrade instructions.



In-place upgrading a MongoDB ReplicaSet - in a nutshell:

- Upgrade the MongoDB binaries to next major release on all secondaries.
- Step down the replica set primary to force an election of a new primary.
- Upgrade the MongoDB binaries on the stepped-down primary.
- Increment the feature compatibility version.



<https://www.mongodb.com/docs/manual/release-notes/5.0-upgrade-replica-set>

Upgrading MongoDB

Side-by side approach

Implement a new MongoDB ReplicaSet and migrate your data with **mongodump** and **mongorestore**.

Remark: It is recommended to restore to a matching major version.

Back up your databases with mongodump, eg:

```
mongodump --host=<oldmongohost>:<port> --username=<mongouser> --password=<mongopassword> --forceTableScan  
--gzip --db=meeting --out /dump
```

```
mongodump --host=<oldmongohost>:<port> --username=<mongouser> --password=<mongopassword> --forceTableScan  
--gzip --db=mobileOffline --out /dump
```

```
mongodump --host=<oldmongohost>:<port> --username=<mongouser> --password=<mongopassword> --forceTableScan  
--gzip --db=chatlogging --out /dump
```

```
mongodump --host=<oldmongohost>:<port> --username=<mongouser> --password=<mongopassword> --forceTableScan  
--gzip --db=userinfo --out /dump
```

<https://www.mongodb.com/docs/database-tools/mongodump/>

<https://www.mongodb.com/docs/database-tools/mongorestore/>

Upgrading MongoDB

Side-by side approach (cont.)

Restore your databases with mongorestore, eg:

```
mongorestore --host=<newmongohost>:<port> --username=<mongouser> --password=<mongopassword> --drop  
--noIndexRestore --gzip --verbose --nsInclude=meeting.* /restore
```

```
mongorestore --host=<newmongohost>:<port> --username=<mongouser> --password=<mongopassword> --drop  
--noIndexRestore --gzip --verbose --nsInclude=mobileOffline.* /restore
```

```
mongorestore --host=<newmongohost>:<port> --username=<mongouser> --password=<mongopassword> --drop  
--noIndexRestore --gzip --verbose --nsInclude=chatlogging.* /restore
```

```
mongorestore --host=<newmongohost>:<port> --username=<mongouser> --password=<mongopassword> --drop  
--noIndexRestore --gzip --verbose --nsInclude=userinfo.* /restore
```

<https://www.mongodb.com/docs/database-tools/mongodump/>
<https://www.mongodb.com/docs/database-tools/mongorestore/>

HCLSoftware

Sametime Monitoring Dashboard

Sametime Statistics Monitoring



HCL Sametime

Provides statistics about chat and meeting services via several HTTP endpoints



You can

- collect
- store
- analyze
- visualize

the metrics from HCL Sametime using any tools of your choice



Provides a `docker-compose-monitoring.yml` file that creates a monitoring stack



Run a monitoring stack based on Prometheus and Grafana on the same Linux host as your Sametime server

Provides json files for Docker and Kubernetes to build a Grafana dashboard



Create a Sametime monitoring dashboard on a Grafana instance of your choice (e.g. on prem or in the cloud)

You need to make a few decisions

Where to run the different monitoring components?

- On the Sametime server / K8s cluster or on a separate server?
- Do you also want to collect system and container metrics (CPU & RAM usage, disk space left, etc.)?

How often do you want to collect metrics data?

- More often means more load on the Sametime server
- Keep in mind: You're NOT building a real time monitoring system!
- Hint: JVB stats are refreshed every 5 seconds

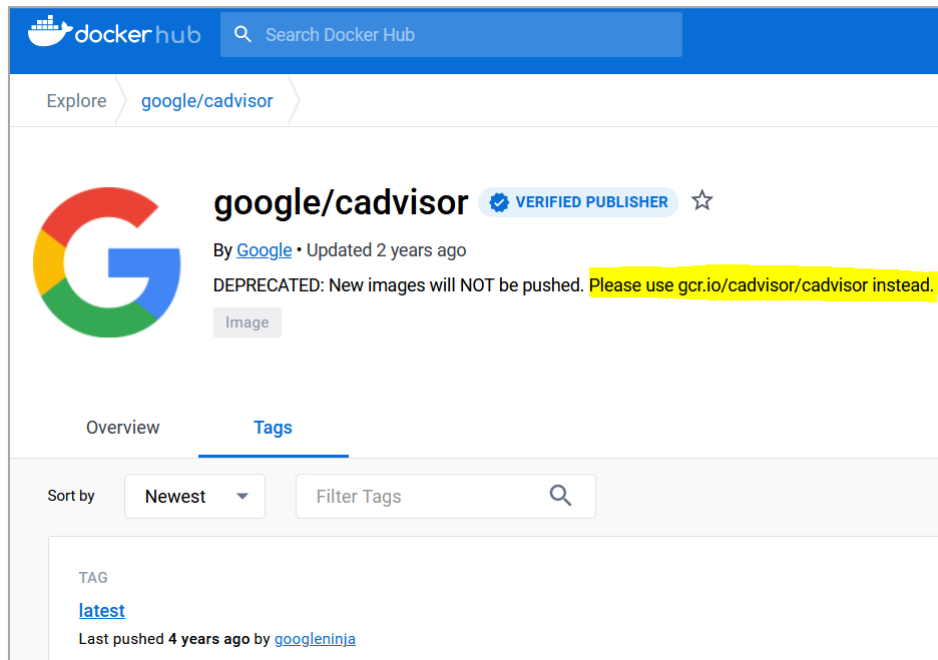
Where do you want to store the metrics data and how long do you want to keep it?

- A few hours...days...weeks...months?
- You may want to configure persistent volumes for Prometheus and Grafana

HCL Sametime Monitoring Dashboard on Docker (1)

The Sametime services defined in `docker-compose.yml` and the monitoring services defined in `docker-compose-monitoring.yml` run on the same Docker network.

- No need to expose any ports of the monitoring services to localhost except for Grafana
- No need to open firewall for any monitoring services ports except for Grafana port 3000
- 👉 Pull the cadvisor image from gcr.io, not from Docker hub.



```
cadvisor:
  image: gcr.io/cadvisor/cadvisor:latest
  ports:
    # - 8088:8080 no need to expose this port
  volumes:
    - /:/rootfs:ro
    - /var/run:/var/run:rw
    - /sys:/sys:ro
    - /var/lib/docker:/var/lib/docker:ro
  networks:
    - sametime.test
```

HCL Sametime Monitoring Dashboard on Docker (2)

Modify `docker-compose-monitoring.yml` to enable https for Grafana

You can use the same TLS certificate as Sametime

Open `https://SametimeServerFQDN:3000` and change default Grafana admin credentials

```
grafana:
  image: grafana/grafana
  user: root
  depends_on:
    - prometheus
  environment:
    - GF_SECURITY_ADMIN_USER=admin
    - GF_SECURITY_ADMIN_PASSWORD=admin
    - GF_INSTALL_PLUGINS=
    - GF_SERVER_PROTOCOL=https
    - GF_SERVER_CERT_FILE=/local/certs/cert.pem
    - GF_SERVER_CERT_KEY=/local/certs/key.pem
  ports:
    - 3000:3000
  volumes:
    - ./grafana/provisioning:/etc/grafana/provisioning/
    - grafana_data:/var/lib/grafana
    - ./sametime-config/web/acme-certs/sametime.dnug.eu/fullchain.pem:/local/certs/cert.pem
    - ./sametime-config/web/acme-certs/sametime.dnug.eu/key.pem:/local/certs/key.pem
  env_file:
    - ./env
  networks:
    - sametime.test
  restart: always
```

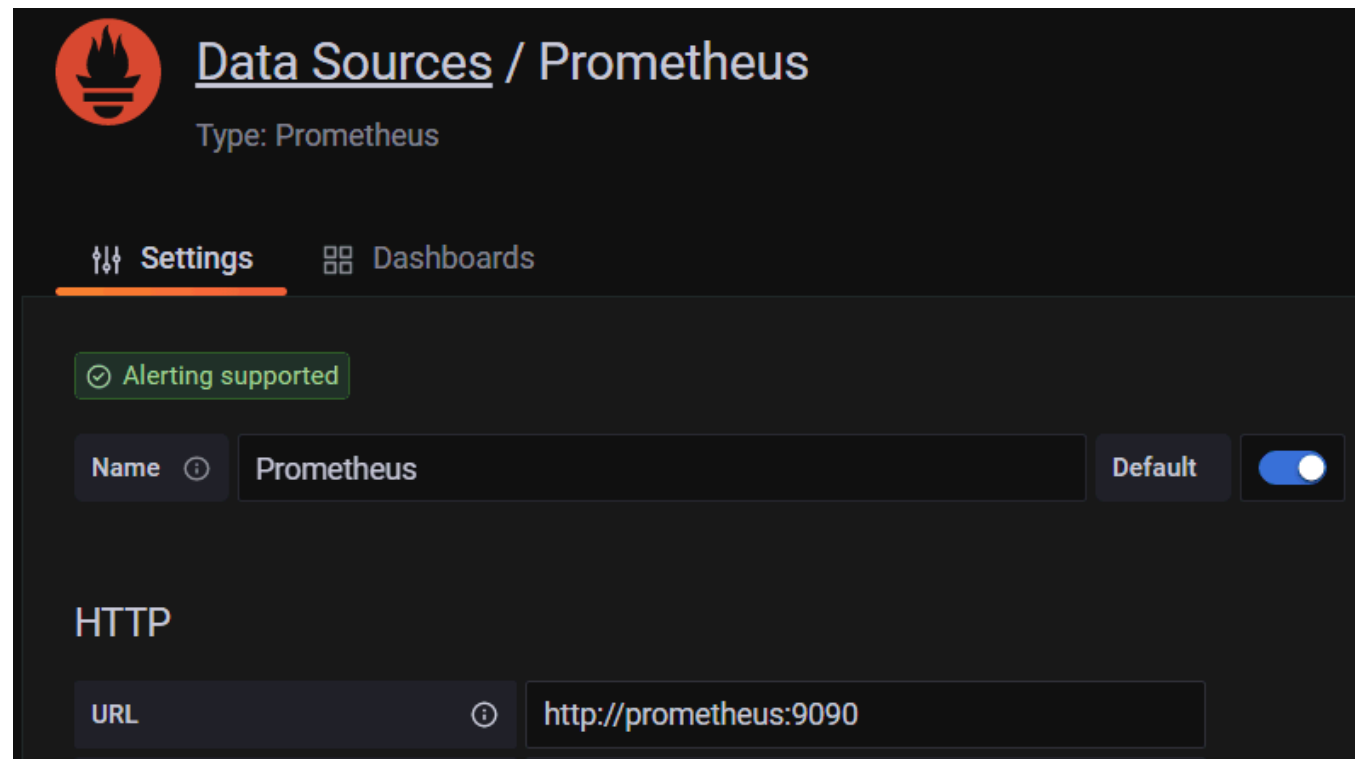
Other options

- Place a proxy server (e.g. built-in nginx) in front of Grafana Web UI
- Configure LDAP authentication in Grafana

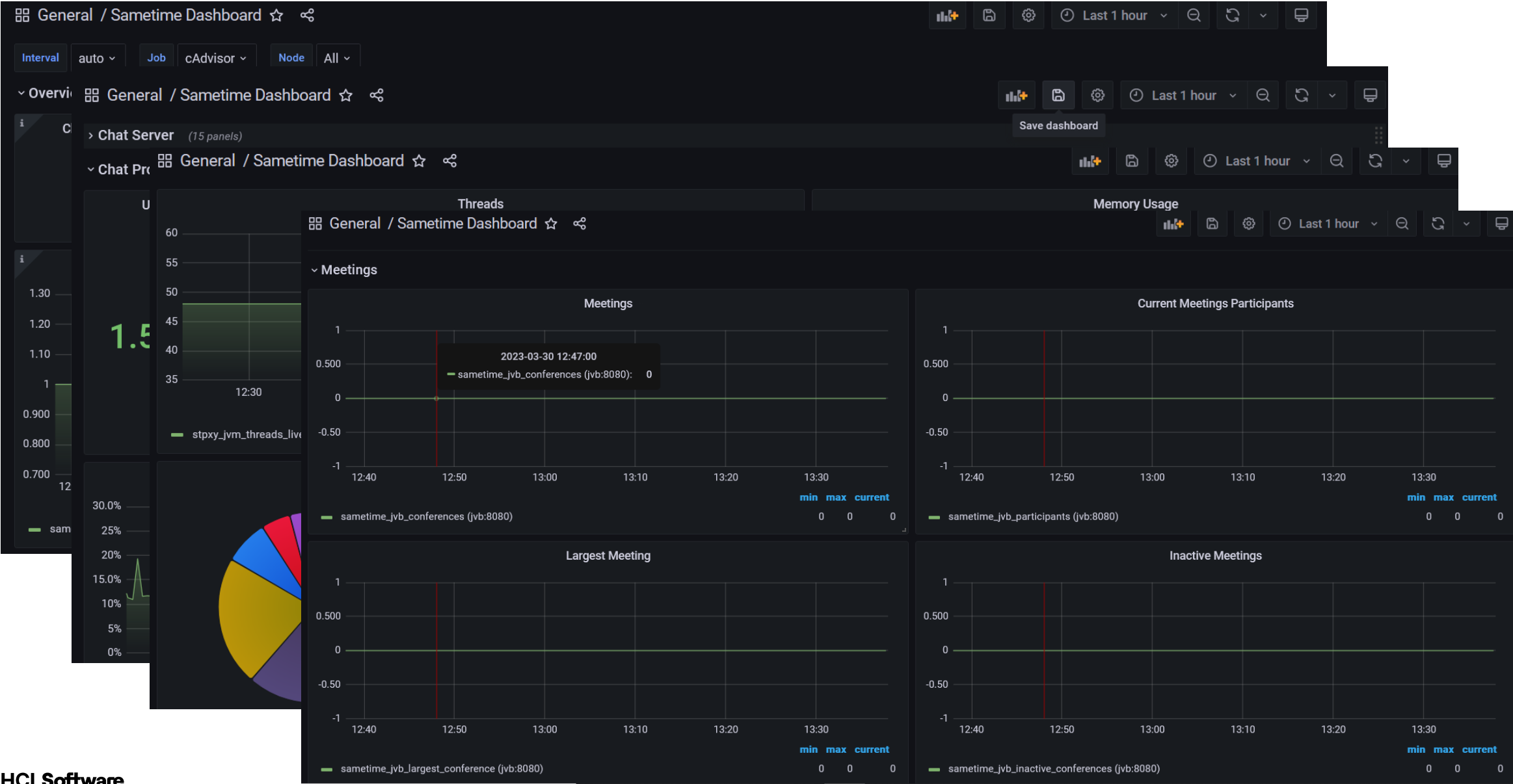
HCL Sametime Monitoring Dashboard on Docker (3)

The Sametime services defined in `docker-compose.yml` and the monitoring services defined in `docker-compose-monitoring.yml` run on the same Docker network.

➤ In Grafana you can refer to Prometheus service name on port 9090 instead of `http://host.docker.internal:9091`



HCL Sametime Monitoring on Docker – Grafana Dashboard



HCL Sametime Monitoring on Kubernetes

The Sametime product documentation tells you to install kube-prometheus-stack that already includes Grafana.

➤ No need for an additional Grafana installation

Familiarize yourself with this stack **before you install** it !

- What's included, how does it work, how to open Prometheus Web UI, etc.
- See values.yaml to find the initial admin password for Grafana (hint: “prom-operator”) and learn about customization options

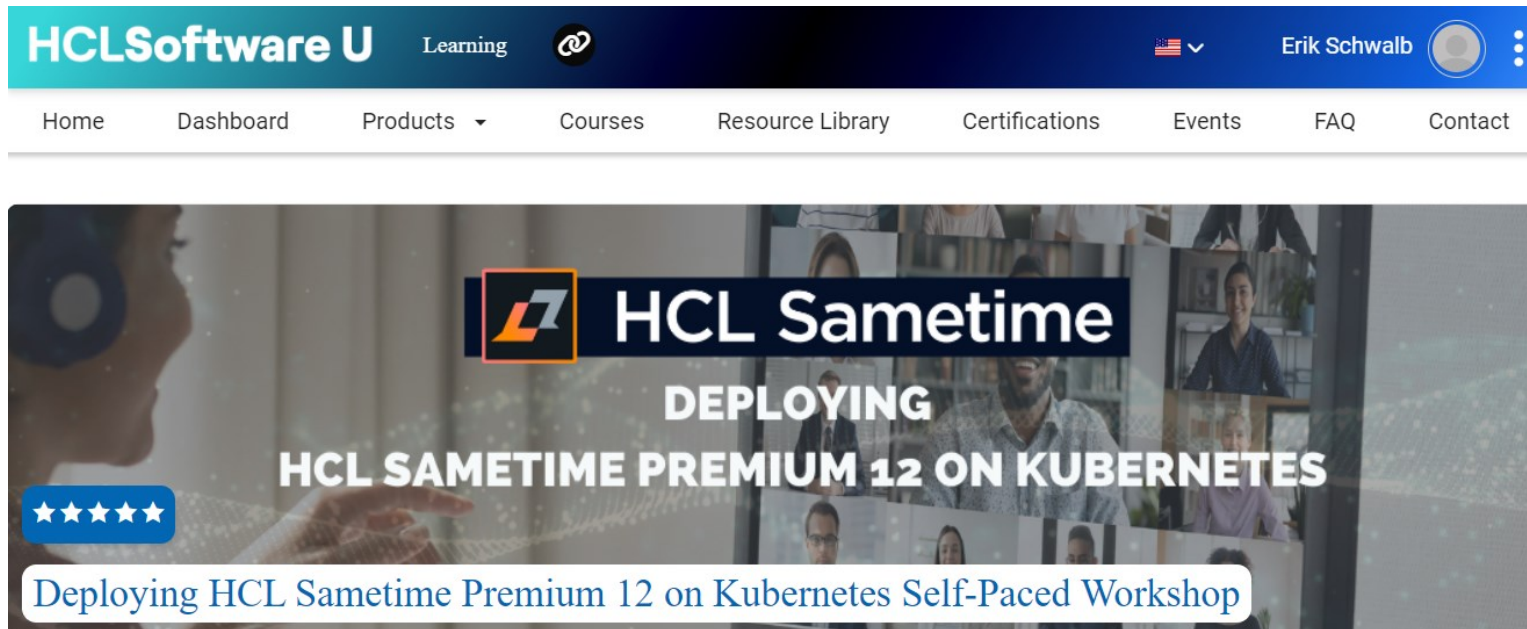
Accessing Grafana Web UI via kubectl port-forward is okay for the initial setup. For regular use of the dashboard(s) you may want to consider other options to implement secure access to Grafana.

- Expose Grafana as a LoadBalancer service
- Create an Ingress for Grafana (also see values.yaml)

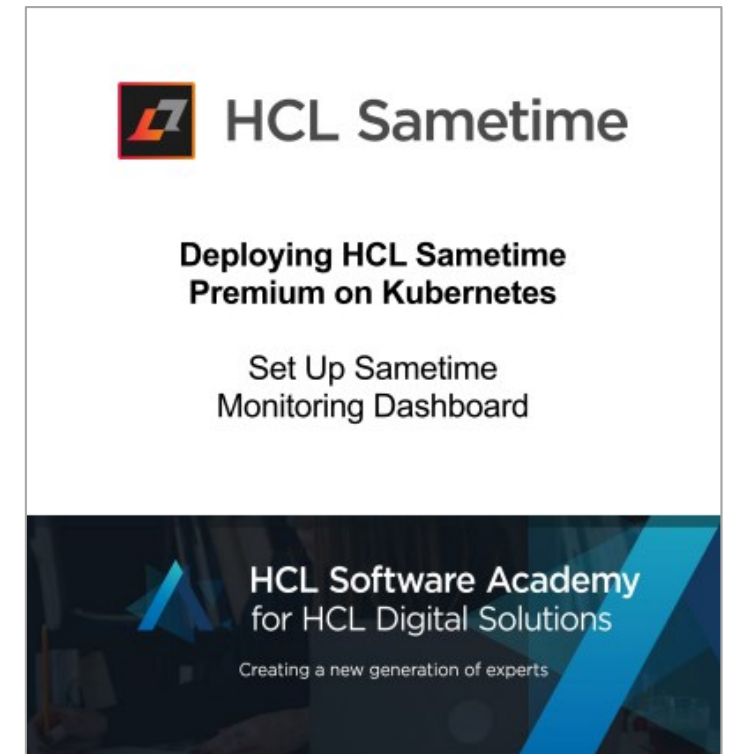
HCL Sametime Monitoring on Kubernetes

Register for the free of charge [Deploying HCL Sametime Premium 12 on Kubernetes Self-Paced Workshop](#) and get **step-by-step instructions** in the [Set Up Sametime Monitoring Dashboard](#) chapter.

➤ <https://hclsoftwareu.hcltechsw.com/hclsoftwareu-courses/course/sametime-on-kubernetes-self-paced>



The screenshot shows the HCLSoftware U Learning portal. The header includes the HCLSoftware U logo, 'Learning' tab, a user profile for Erik Schwalb, and a navigation menu with links: Home, Dashboard, Products, Courses, Resource Library, Certifications, Events, FAQ, and Contact. The main banner features the HCL Sametime logo and the text 'DEPLOYING HCL SAMETIME PREMIUM 12 ON KUBERNETES'. Below the banner, a blue button with five stars is visible, and a white text box contains the course title: 'Deploying HCL Sametime Premium 12 on Kubernetes Self-Paced Workshop'.



The course cover features the HCL Sametime logo at the top. Below it, the title 'Deploying HCL Sametime Premium on Kubernetes' is displayed. Underneath the title, the subtitle 'Set Up Sametime Monitoring Dashboard' is shown. At the bottom, the HCL Software Academy logo is present, along with the text 'HCL Software Academy for HCL Digital Solutions' and the tagline 'Creating a new generation of experts'.

HCLSoftware



Integration with Let's Encrypt

Integrate Sametime on Docker with Let's Encrypt

Sametime 12.0.1 on Docker can request, retrieve and apply a TLS certificate from Let's Encrypt. It will also automatically renew the TLS certificate before it expires.

The built-in integration is based on the ACME protocol (**A**utomatic **C**ertificate **M**anagement **E**nvironment) using an HTTP-01 challenge to verify, that you are the owner of the requesting website.

1. ACME server (= Let's Encrypt) sends a challenge to ACME client (= NGINX service in Sametime)
2. ACME server will ask via **in-bound HTTP request on port 80** for the “secret” at a well-known URL

- **The FQDN of your Sametime server must be registered in public DNS**
- **Your Sametime server must be accessible on the public Internet via http port 80**



The built-in integration is currently **broken in 12.0.1 FP1**...but there is a **workaround**



Integrate Sametime 12.0.1 on Docker with Let's Encrypt

Settings in `.env`

```
# Exposed HTTP port
HTTP_PORT=80

# Exposed HTTPS port
HTTPS_PORT=443

# Redirect HTTP traffic to HTTPS
# Necessary for Let's Encrypt, relies on standard HTTPS port (443)
ENABLE_HTTP_REDIRECT=1

# Let's Encrypt configuration
# Enable Let's Encrypt certificate generation
ENABLE_LETSENCRYPT=1

# Domain for which to generate the certificate
LETSENCRYPT_DOMAIN=<your.sametimeserver.com>

# E-Mail for receiving important account notifications (mandatory)
LETSENCRYPT_EMAIL=<YourAdminEmailAddress>

# Use the staging server (for avoiding rate limits while testing)
# LETSENCRYPT_USE_STAGING=1
```

If you set up the integration for the first time, make sure to test it first with the Let's Encrypt staging service enabled!

Integrate Sametime 12.0.1 on Docker with Let's Encrypt



Settings in `docker-compose.yml`

nginx:

```
image: hclcr.io/st/meetings-web:${BUILD_LEVEL}
restart: ${RESTART_POLICY}
ports:
  - '${HTTP_PORT}:80'
  - '${HTTPS_PORT}:443'
...
```

environment:

```
...
- LETSENCRYPT_DOMAIN
- LETSENCRYPT_EMAIL
- LETSENCRYPT_USE_STAGING
...
```

Settings in `custom.env`

```
# Example: PUBLIC_URL=https://sametime.company.com
PUBLIC_URL=https://<your.sametimeserver.com>
```

A setting for `PUBLIC_URL` can be found both in `.env` and in `custom.env`.

Make sure you define a value for this setting only in `custom.env`.

If you want to use the Let's Encrypt staging service you must also add `LETSENCRYPT_USE_STAGING` to the environment variables of the nginx container.

Temporary fix for Sametime 12.0.1 FP1 on Docker with Let's Encrypt



Settings in `.env`

```
# Exposed HTTP port
HTTP_PORT=8000

# Exposed HTTPS port
HTTPS_PORT=443

# Redirect HTTP traffic to HTTPS
# Necessary for Let's Encrypt, relies on standard HTTPS port (443)
ENABLE_HTTP_REDIRECT=1

# Let's Encrypt configuration
# Enable Let's Encrypt certificate generation
ENABLE_LETSENCRYPT=1

# Domain for which to generate the certificate
LETSENCRYPT_DOMAIN=<your.sametimeserver.com>

# E-Mail for receiving important account notifications (mandatory)
LETSENCRYPT_EMAIL=<YourAdminEmailAddress>

# Use the staging server (for avoiding rate limits while testing)
# LETSENCRYPT_USE_STAGING=1
```

Settings in `docker-compose.yml`

nginx:

```
image: hclcr.io/st/meetings-web:${BUILD_LEVEL}
user: "0:0" # temporary fix
restart: ${RESTART_POLICY}
ports:
  - '${HTTP_PORT}:8080'
  - '${HTTPS_PORT}:4443'
  - 80:80 # temporary fix
```

...

environment:

...

```
- LETSENCRYPT_DOMAIN
- LETSENCRYPT_EMAIL
- LETSENCRYPT_USE_STAGING
```

...

Integrate Sametime 12.0.1 on Docker with Let's Encrypt

The nginx container will use the ACME protocol to register an account with Let's Encrypt. If successful it will then request and retrieve a TLS certificate.

```
[Fri Sep 2 18:06:09 CEST 2022] Using CA: https://acme-v02.api.letsencrypt.org/directory
[Fri Sep 2 18:06:09 CEST 2022] Run pre hook:'if [[ -d /var/run/s6/services/nginx ]]; then s6-svc -d /var/run/s6/services
[Fri Sep 2 18:06:09 CEST 2022] Standalone mode.
[Fri Sep 2 18:06:09 CEST 2022] Create account key ok.
[Fri Sep 2 18:06:09 CEST 2022] Registering account: https://acme-v02.api.letsencrypt.org/directory
[Fri Sep 2 18:06:11 CEST 2022] Registered
[Fri Sep 2 18:06:11 CEST 2022] ACCOUNT_THUMBPRINT='IGj4UFtbU5Z4FQ2HEgo_jRTM02RzdLET36DrDURCu_U'
[Fri Sep 2 18:06:11 CEST 2022] Creating domain key
[Fri Sep 2 18:06:11 CEST 2022] The domain key is here: /config/acme.sh/sametime.dnug.eu/sametime.dnug.eu.key
[Fri Sep 2 18:06:11 CEST 2022] Single domain='sametime.dnug.eu'
[Fri Sep 2 18:06:12 CEST 2022] Getting domain auth token for each domain
[Fri Sep 2 18:06:14 CEST 2022] Getting webroot for domain='sametime.dnug.eu'
[Fri Sep 2 18:06:14 CEST 2022] Verifying: sametime.dnug.eu
[Fri Sep 2 18:06:14 CEST 2022] Standalone mode server
[Fri Sep 2 18:06:19 CEST 2022] Success
[Fri Sep 2 18:06:19 CEST 2022] Verify finished, start to sign.
[Fri Sep 2 18:06:19 CEST 2022] Lets finalize the order.
[Fri Sep 2 18:06:19 CEST 2022] Le_OrderFinalize='https://acme-v02.api.letsencrypt.org/acme/finalize/714169367/1218760562
[Fri Sep 2 18:06:21 CEST 2022] Downloading cert.
[Fri Sep 2 18:06:21 CEST 2022] Le_LinkCert='https://acme-v02.api.letsencrypt.org/acme/cert/049e2ec117270078520a43948fb99
[Fri Sep 2 18:06:21 CEST 2022] Cert success.
-----BEGIN CERTIFICATE-----
MIIFJjCCBA6gAwIBAgISBJ4uwRcnAHhSCk0Uj7mV4CzYMA0GCSqGSIb3DQEBCwUA
```

Integrate Sametime 12.0.1 on Docker with Let's Encrypt

```
| Bk2faD7ys1QxKQfS6R4h5v93pZXYizWfroI=  
| -----END CERTIFICATE-----  
| [Fri Sep  2 18:06:22 CEST 2022] Your cert is in /config/acme.sh/sametime.dnug.eu/sametime.dnug.eu.cer  
| [Fri Sep  2 18:06:22 CEST 2022] Your cert key is in /config/acme.sh/sametime.dnug.eu/sametime.dnug.eu.key  
| [Fri Sep  2 18:06:22 CEST 2022] The intermediate CA cert is in /config/acme.sh/sametime.dnug.eu/ca.cer  
| [Fri Sep  2 18:06:22 CEST 2022] And the full chain certs is there: /config/acme.sh/sametime.dnug.eu/fullchain.cer  
| [Fri Sep  2 18:06:22 CEST 2022] Run post hook:'if [[ -d /var/run/s6/services/nginx ]]; then s6-svc -u /var/run/s6/services  
| [Fri Sep  2 18:06:22 CEST 2022] Installing key to:/config/acme-certs/sametime.dnug.eu/key.pem  
| [Fri Sep  2 18:06:22 CEST 2022] Installing full chain to:/config/acme-certs/sametime.dnug.eu/fullchain.pem
```

The TLS certificate will be placed in a subdirectory below the `./sametime-config` directory, that is named after the FQDN of your server.

```
# List the TLS certificate
```

```
ls -l sametime-config/web/acme-certs/SametimeServerFQDN
```

```
-rw-r--r--. 1 root root 5597 Sep  2 18:06 fullchain.pem  
-rw-----. 1 root root 1679 Sep  2 18:06 key.pem
```

Integrate Sametime on Kubernetes with Let's Encrypt



Sametime 12.0.1 on Kubernetes does not include built-in integration with Let's Encrypt. However, you can use cert-manager to get a TLS certificate from Let's Encrypt and use it with your Sametime deployment.

cert-manager adds custom objects such as Certificates, CertificateRequests and Issuers as resource types in Kubernetes clusters, and simplifies the process of obtaining, renewing and using those certificates.

- Install and configure cert-manager including CRDs
- Create Let's Encrypt `ClusterIssuer` for staging and production
- Edit the file `ingress.yaml` that is included with the Sametime helm charts and insert a new line with the string `cert-manager.io/cluster-issuer: "letsencrypt-staging"` in the `annotations` section

When you then deploy Sametime, an ingress will be created as part of the deployment and cert-manager will automatically provision a TLS certificate from Let's Encrypt.



➤ <https://cert-manager.io>

Integrate Sametime on Kubernetes with Let's Encrypt



GNU nano 2.3.1

File: /opt/hcl/sametime/helm/charts/web/templates/ingress.yaml

```
{{ if .Values.global.sofySolutionContext }}
{{ else }}
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: {{ include "web.fullname" . }}
  labels:
    {{- include "web.labels" . | nindent 4 }}
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
    networking.gke.io/v1beta1.FrontendConfig: {{ include "web.fullname" . }}-frontend-config
    nginx.ingress.kubernetes.io/proxy-body-size: "0"
    nginx.ingress.kubernetes.io/ssl-redirect: {{ not (default false .Values.global.tlsTermination) }}
    nginx.ingress.kubernetes.io/force-ssl-redirect: {{ not (default false .Values.global.tlsTermination) }}
    cert-manager.io/cluster-issuer: "letsencrypt-prod"
spec:
  {{ if .Values.global.ingressClassName }}
  ingressClassName: {{ .Values.global.ingressClassName }}
  {{ else }}
  ingressClassName: nginx
  {{ end }}
```

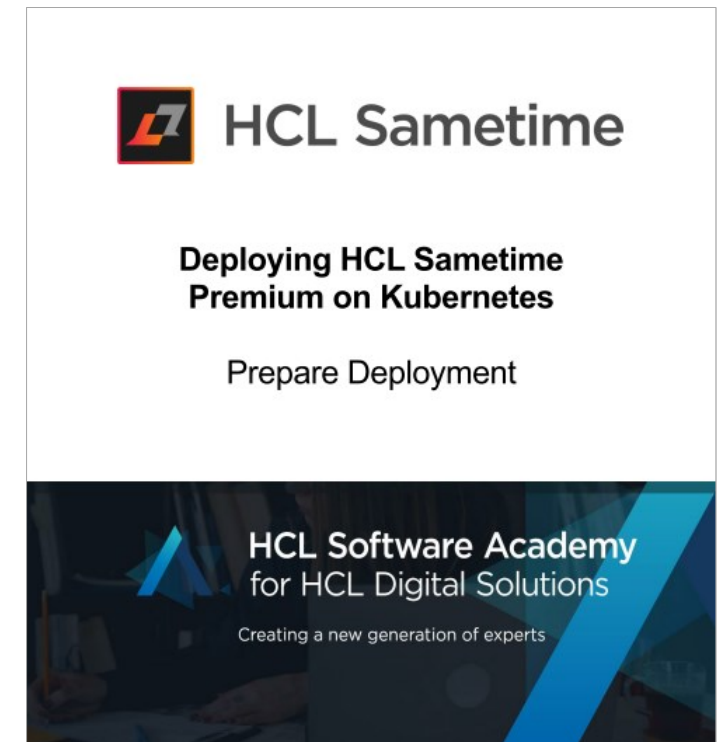
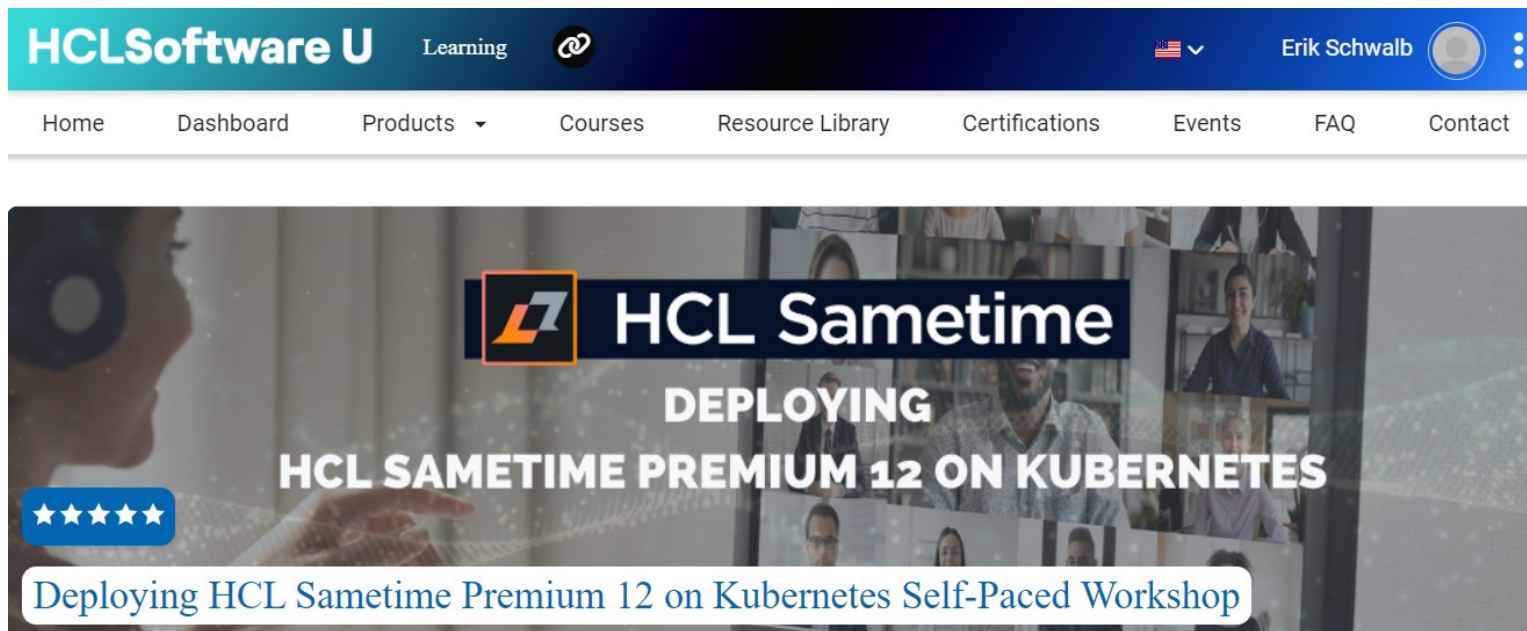


Integrate Sametime on Kubernetes with Let's Encrypt



Register for the free of charge [Deploying HCL Sametime Premium 12 on Kubernetes Self-Paced Workshop](#) and get **step-by-step instructions** in the [Prepare Deployment](#) chapter.

➤ <https://hclsoftwareu.hcltechsw.com/hclsoftwareu-courses/course/sametime-on-kubernetes-self-paced>



HCLSoftware

Implementing and configuring an internal STUN server

Implementing and configuring an internal STUN server

Things you need to know



What is STUN ?

- Session Traversal Utilities for NAT (STUN) is a standardized set of methods, including a network protocol, for NAT traversal of Network address translation (NAT) gateways in applications of real-time voice, video, messaging, and other interactive communications.
- STUN is a tool used by other protocols, such as Interactive Connectivity Establishment (ICE), the Session Initiation Protocol (SIP), and WebRTC. It provides a tool for hosts to discover the presence of a network address translator, and to discover the mapped, usually public, Internet Protocol (IP) address and port number that the NAT has allocated for the application's User Datagram Protocol (UDP) flows to remote hosts.
- The protocol requires assistance from a third-party network server STUN server located on the opposing public side of the NAT, usually the public Internet.

Why do we need STUN ?

Simply put, we use STUN as a tool to help clients determine their public IP address so that they can connect to each other and the Sametime Meeting Server to send and receive audio and video data.

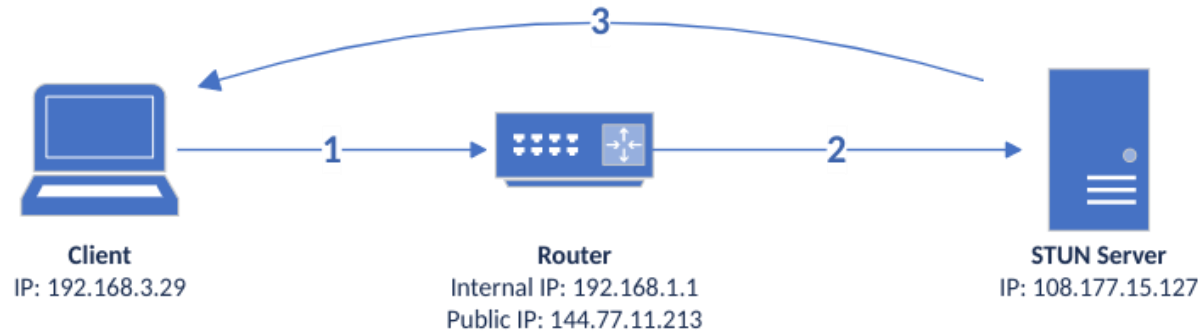
<https://en.wikipedia.org/wiki/STUN>

https://help.hcltechsw.com/sametime/1201/admin/session_traversal_utilities.html

Implementing and configuring an internal STUN server

How does it work ?

1. Client 192.168.3.29 sends a STUN request through Router 192.168.1.1 to a STUN server outside the network, listening on 108.177.15.127, using source port 5090.
2. Router 192.168.1.1 forwards the request to STUN server 108.177.15.127 and changes port 5090 to port 15090.
3. STUN Server 108.177.15.127 sends a response back to Client 192.168.3.29 through the Router with public IP 144.77.11.213 specifying that the request was received from IP 144.77.11.213 and port 15090.



<https://en.wikipedia.org/wiki/STUN>

Implementing and configuring an internal STUN server

Considerations

Common reasons for implementing an internal STUN server

- Internal only deployment in a NAT network
- DMZ deployment, but internal clients are not allowed to send UDP traffic to public IP
- etc. ...

Important:

- HCL Sametime ships with **Google STUN servers** in the default configuration, ie
stun.l.google.com:19302,stun1.l.google.com:19302,stun2.l.google.com:19302
- The internal STUN server **must not** be installed on the same host running the Sametime JVB !



Implementing and configuring an internal STUN server

Sample setup and configuration (RedHat)

```
yum install epel-release coturn coturn-utils -y
mv /etc/coturn/turnserver.conf /etc/coturn/turnserver.conf.orig
cat <<EOF > /etc/coturn/turnserver.conf
listening-port=3478
server-name=<stun.yourdomain.tld>
realm=<yourdomain.tld>
total-quota=100
stale-nonce=600
proc-user=coturn
proc-group=coturn
EOF

firewall-cmd --zone=public --add-port=3478/udp --permanent
firewall-cmd --reload

systemctl enable coturn
systemctl start coturn
```

Implementing and configuring an internal STUN server

Docker

Modify your .env file:

```
[...]  
# STUN servers used to discover the server's public IP.  
JVB_STUN_SERVERS=stun.l.google.com:19302,stun1.l.google.com:19302,stun2.l.google.com:  
19302,<stun.yourdomain.tld>:<stunport>  
[...]
```

eg

```
[...]  
# STUN servers used to discover the server's public IP.  
JVB_STUN_SERVERS=stun.l.google.com:19302,stun1.l.google.com:19302,stun2.l.google.com:  
19302,stun.yourdomain.tld:3478  
[...]
```

https://help.hcltechsw.com/sametime/1201/admin/configuring_stun.html
https://help.hcltechsw.com/sametime/1201/admin/session_traversal_utilities.html

Implementing and configuring an internal STUN server

Kubernetes

Modify your helm/values.yaml file:

```
global:
[...]
  jvbStunServers: stun1.l.google.com:19302,stun1.l.google.com:19302,stun2.l.google.com
:19302,<stun.yourdomain.tld>:<stunport>
[...]
```

eg

```
global:
[...]
  jvbStunServers: stun1.l.google.com:19302,stun1.l.google.com:19302,stun2.l.google.com
:19302,stun.yourdomain.tld:3478
[...]
```

https://help.hcltechsw.com/sametime/1201/admin/configuring_stun.html
https://help.hcltechsw.com/sametime/1201/admin/session_traversal_utilities.html

HCLSoftware

Sametime Limited Use

Set Up Sametime Limited Use

HCL customers with an entitlement for Sametime Limited Use can download HCL Sametime 12 from FlexNet. The file [Sametime_12.0.1_FP1.zip](#) contains all server components for a “[Chat-only](#)” deployment of Sametime. If you install that file on Docker or Kubernetes, the deployment will [automatically configure the Sametime clients](#), so that they only provide such capabilities, that are [compliant with a Sametime Limited Use entitlement](#).

If your organization has both **Sametime Limited Use** and **Sametime Premium users**, you can **deploy Sametime Premium and configure multiple policies** to make sure, that Sametime Limited Use users will be restricted to their entitlement while Sametime Premium users can use all capabilities of Sametime.

- Modify the **default** user **policy** so that it allows only **Sametime Limited Use capabilities**
- Create a **custom policy** that allows **Sametime Premium capabilities** and assign it to your Sametime Premium users

Modifying a user policy to allow Sametime Limited Use capabilities

The file `policies.user.xml` is located in the `/local/notesdata` directory of the Sametime Community container. Change the following settings in the **default policy** in `policies.user.xml` to allow only Sametime Limited Use capabilities.

Policy ID	set this to	Description
im.2019	current-value=1	requires this community to be the default, primary community
im.2011	current-value=0	disables the ability to add multiple communities to the installed Sametime client
im.2001	current-value=0	disables the ability to add external users via the Sametime Gateway
im.enableOrganizationTreeView	current-value=0	disables the organization tree view
im.3000	current-value=0	disables the use of some integrated features of the client (e.g. Audio / Video)
im.2009	current-value=0	disables Screen Captures and Image Transfer
im.2005	current-value=0	disables client-to-client File Transfer
im.1	current-value=0	disables File Transfer through server
im.thirdPartyMeetingEnabled	current-value=0	disables Instant Meeting invite ("Meet me here")
im.meetingsEnabled	current-value=0	disables Sametime 11.6 or Sametime 12 Meetings

Creating a custom policy for Sametime Premium users

Create a group in your LDAP directory, that contains your Sametime Premium users (e.g. "SametimePremiumUsers").

Multi-purpose group : SametimePremiumUsers

Basics | Comments | Administration

Basics

Group name:	SametimePremiumUsers
Group type:	Multi-purpose
Category:	Application
Description:	This group contains all Sametime Premium users

Example showing HCL Domino as the LDAP directory

Create a **custom policy** in policies.user.xml with settings for your Sametime Premium users and **assign it a weight**, that is **higher than the weight of the default policy**.

```
<policies>
  <product id="im">
    <policy weight="10"
            id="im.premium.policy">
      <p:policy-template xmlns:p="ht
                        xmlns:xsi='

```


Assigning a custom policy to Sametime Premium users

Assign the custom policy to the group, that you created for your Sametime Premium users.

`assignment type="1"` means you are assigning the policy to a group.

```
<policyAssignment>  
  <assignment type="1"  
              id="SametimePremiumUsers"/>  
</policyAssignment>
```

How to customize the policies.user.xml file in Sametime on Docker (1)

Change to your Sametime directory

```
cd /opt/hcl/sametime
```

Best practice: Create a separate directory for your customizations

```
mkdir custom-config
```

Copy the file policies.user.xml from the Community container to your custom-config directory while Sametime is running

```
docker cp sametime_community_1:/local/notesdata/policies.user.xml custom-config/policies.user.xml
```

Stop the Sametime server

```
docker compose down
```

How to customize the policies.user.xml file in Sametime on Docker (2)

Modify the policies.user.xml file in your custom-config directory as needed

Edit the file `docker-compose.yml` to mount your customized policies file to the community container

Note: In the example below, we changed the file name of the customized policy to `customized.policies.user.xml`

```
version: "2"
services:
  community:
    image: hclcr.io/st/chat-server:${BUILD_LEVEL}
    restart: ${RESTART_POLICY}
    env_file: custom.env
    environment:
      - JWT_SECRET_ENV=${JWT_APP_SECRET}
      - DOMINO_SERVER_HOST_ENV=domino
      - DOMINO_SERVER_NAME_ENV=CN\=domino\0\=test
      - DOMINO_SERVER_DOMAIN_ENV=test
      - ST_BRANDING_INFO_ENV=standard
    volumes:
      - ./custom-config/customized.policies.user.xml:/local/notesdata/policies.user.xml
    networks:
      - sametime.test
```

Start the Sametime server

```
docker compose up -d
```

How to customize the policies.user.xml file in Sametime on Kubernetes (1)

Create a subdirectory for the customized policy and switch to this directory.

```
mkdir my-custom-policy  
cd my-custom-policy
```

Copy the existing policy files from the Community container to the current directory.

You need to **copy both files**, even if you plan to update only one of the files.

```
kubectl exec -it <podID> --container community -- cat /local/notesdata/policies.user.xml >  
./policies.user.xml  
kubectl exec -it <podID> --container community -- cat /local/notesdata/policies.server.xml >  
./policies.server.xml
```

Modify the policy files as needed, then create a ConfigMap from these files.

```
kubectl create configmap custom-community-policy --from-file=.
```

Modify the values.yaml file and add the following parameter to use the ConfigMap:

```
overrideCommunityPolicy: custom-community-policy
```

How to customize the policies.user.xml file in Sametime on Kubernetes (2)

Change to the `helm` directory, run 'helm upgrade' and scale the Community pod.

```
helm upgrade <deploymentName> . # run this command from the helm directory
```

```
kubectl scale deploy community --replicas=0
```

```
kubectl scale deploy community --replicas=1
```

If you need to make further changes to your policies, update the policy files again, then recreate the ConfigMap and scale the Community pod to apply your changes.

```
cd my-custom-policy
```

Update the policy files, then run the commands below

```
kubectl delete cm custom-community-policy
```

```
kubectl create configmap custom-community-config --from-file=.
```

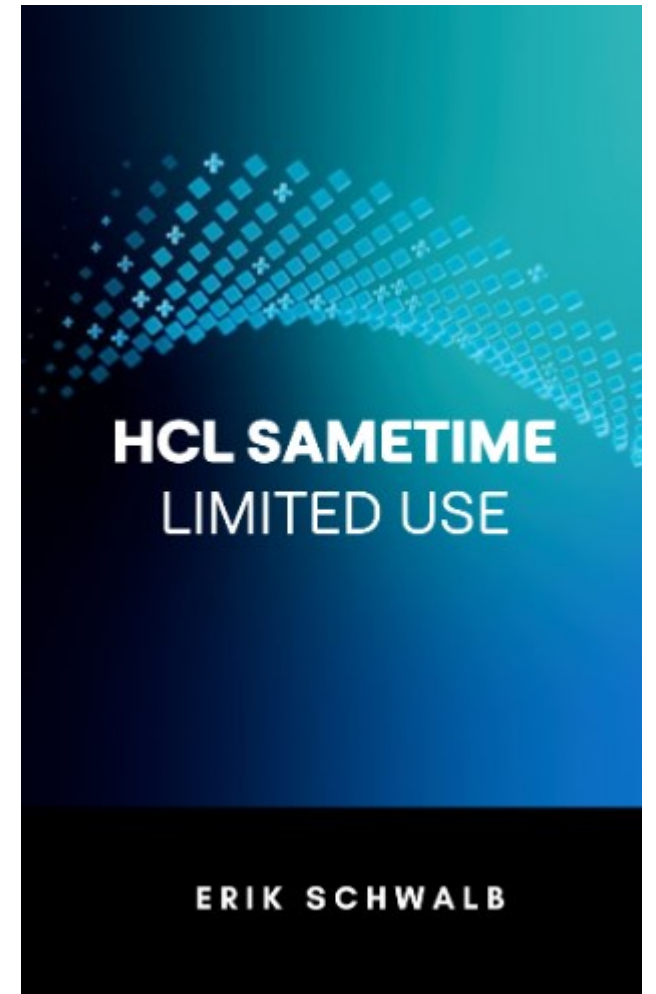
```
kubectl scale deploy community --replicas=0
```

```
kubectl scale deploy community --replicas=1
```

Sametime Limited Use

Download the [Sametime Limited Use Whitepaper](https://support.hcltechsw.com/csm?id=kb_article&sysparm_article=KB0108328)

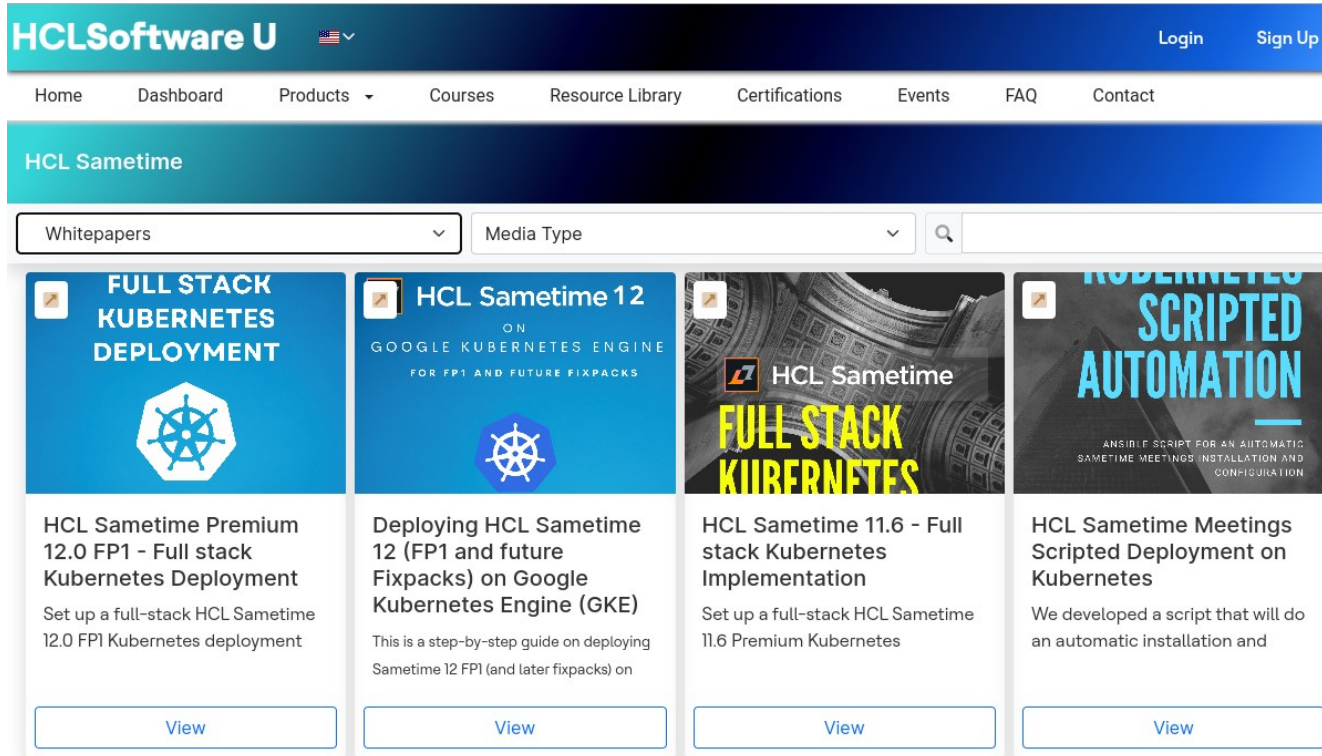
https://support.hcltechsw.com/csm?id=kb_article&sysparm_article=KB0108328



HCL Sametime Premium 12.0 FP1 - Full stack Kubernetes Deployment

HCLSoftware U - Whitepapers

You want to get in touch with implementing Sametime on Kubernetes ?
Get your “free copy” now ! ... ;-)



<https://hclsoftwareu.hcltechsw.com>

https://support.hcltechsw.com/csm?id=kb_article&sysparm_article=KB0100042

HCLSoftware

Elevate Your Business Success with Domino v14

More Open, Mobile and Future-facing than Ever

December 7 @ 10am ET

[Register now](#)



HCL Domino

Join us for the official launch of HCL Domino 14 **and HCL Sametime 12.0.2**

<https://register.gotowebinar.com/register/3240097374153477467>

HCLSoftware

MongoDB on K8s

SAML

Q&A

12.0.2

TURN

HCLSoftware

hcltechsw.com