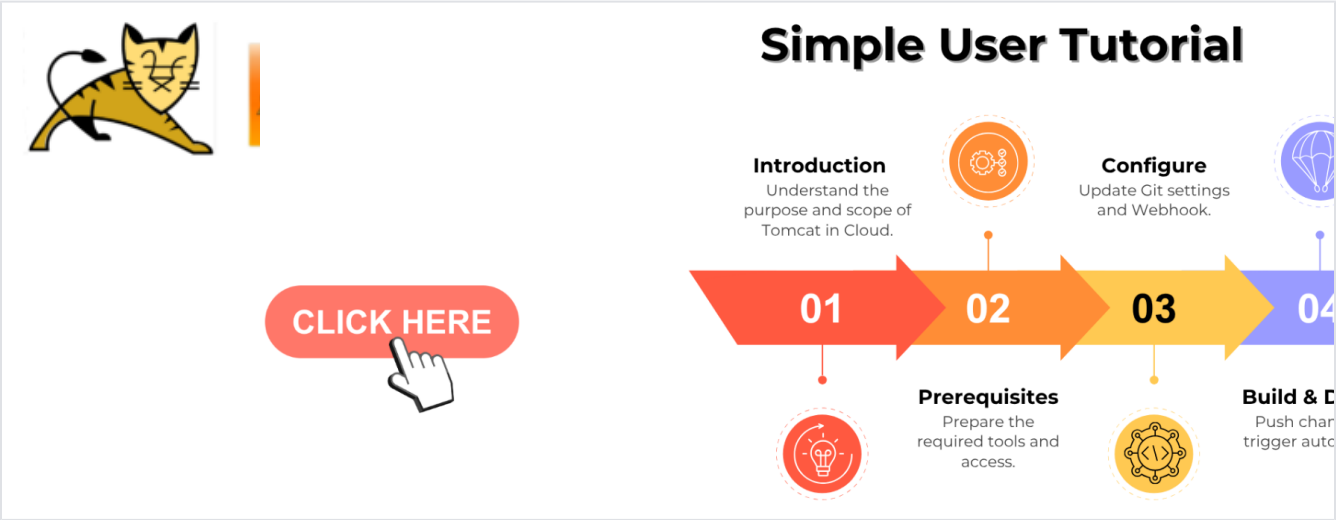
Tomcat in Cloud: User Guide



ⓘ Check the Change Log to see what's new and track past changes.

ⓘ Request Subscription to the Service to gain full access to Tomcat in the Cloud.

ⓘ Explore Tomcat in Cloud further by watching the available Service Introduction Videos.

Click through the infographic to explore the **Journey Map**.

Service Description & Architecture

Service Description

Benefits of Adopting Tomcat Cloud Offering

Architecture Overview

Environment Setups

SNC Ready **(New)**

Subscribe

How to Request Subscription to the Service?

What requests can you make?

Accessing & Connecting to Tomcat

Secure Your Environment

Security Components

Manage Secrets

Security Troubleshooting

Image Build

Base Image

Build Customer Image

Nexus Repository Structure

Versions & Patches

Gitlab Pipelines for Artifact Creation

Storage

Persistent and Ephemeral Volumes

Database Storage

S3 Storage

Image Deployment

Deployment Chain

Create Custom Deployment

Static Content Deployment

Pod Restart

Enable auto-scaling

Maintenance Mode

Workload Scheduler

Defining Probes

Infrastructure as Code (IaC) Blueprints

Configuration of the Git Repo

Flow-as-Code (FaC)

Reverse Proxy Mapping As Code (RPMaC)

Monitoring as Code (MONaCo)

Srv4Dev

Serv4Dev User Guide

Tomcat Development Container

Monitoring & Logging

Monitoring

Application Log Files

Monitoring as Code (MONaCo)

Auditing & Retention

Git Feedback Notifications

Tomcat Console / Logs PVCs

FAQ & Supporting Links

FAQ (**NEW UPDATE**)

Support & Useful Links

Glossary

Tomcat in Cloud: Change Log

Below you will find a series of updates across our our service aimed at improving functionality, performance, and addressing known issues.

These updates reflect our ongoing commitment to providing you with a high-quality service and ensuring your experience is as productive as possible.

Date	Category	Change Description	Link
2026-01-29	Fix Fix	Fixed link to Monaco Changed PVC storages	Tomcat in Cloud: Monitoring as Code (MONaCo) Tomcat in Cloud: Create Custom Deployment
2026-01-21	New	Introduction of Service Now Un-Subscription form	Tomcat in Clohttps://webgate.ec.europa.eu/fpfis/wikis/spaces/DIGITCUG/pages/1993476485/Tomcat+in+Cloud+Monitoring+as+Code+MONaCoud: What requests can you make?
LAST DEPLOYMENT 2025-11-12	Improvement	New Versions Released (5.1.x) • EULogin update to version 9.15.7	Tomcat in Cloud: Version & Patches
2026-01-12	New	New page Added component links section	Tomcat in Cloud: Gitlab Pipelines for Artifact Creation Tomcat in Cloud: Support & Useful Links
2025-12-05	Improvement	Changed PVCv2 limit to 1024 GB	Tomcat in Cloud: Persistent and Ephemeral Volumes
2025-12-02	Improvement	Removed PVCv1 from documentation	Tomcat in Cloud: Persistent and Ephemeral Volumes
2025-12-01	Improvement	Added max memory values for heap, meta and direct	Tomcat in Cloud: Create Custom Deployment
2025-11-24	New	Added k8sApplicationKind in Deployment documentation Added a Probe configuration section in Tutorial Wrong name for JMS mailsession fixed	Tomcat in Cloud: Create Custom Deployment Tomcat in Cloud: Simple User Tutorial Tomcat in Cloud: Simple User Tutorial
2025-11-21	Improvement	Ephemeral volume maximum size changed to 5 gb	Tomcat in Cloud: Create Custom Deployment Tomcat in Cloud: Persistent and Ephemeral Volumes
2025-11-20	New	Page renamed Added Ephemeral volumes documentation	Tomcat in Cloud: Persistent and Ephemeral Volumes Tomcat in Cloud: Create Custom Deployment

2025-11-12	Improvement	New Versions Released (5.0.x) • Tomcat upgrade to versions 9.0.112, 10.1.49, 11.0.14 • Java upgrade to versions 25.0.1, 21.0.9, 17.0.17, 11.0.29, 8u472 • MSSQL JDBC driver upgrade to version 13.2.1 • MySQL JDBC driver upgrade to version 9.5.0 • EULogin update to version 9.14.20	Tomcat in Cloud: Version & Patches
2025-11-11	Improvement	Added certificate to S3 page	Tomcat in Cloud: S3 Storage Documentation
2025-11-11	Improvement	Added notes about default naming of datasources and mailsessions	Tomcat in Cloud: Simple User Tutorial
2025-11-03	Improvement	Added smtp mail server and valid "from" field in mail sessions examples	Tomcat in Cloud: Create Custom Deployment
2025-10-28	New	Publication of the Service Now form for subscribing to the service	Tomcat in Cloud: Request Subscription to the Service
2025-10-24	Improvement	Added description for ReadWriteOnce	Tomcat in Cloud: Persistent Volumes
2025-10-21	Improvement	Added AWS values	Tomcat in Cloud: Persistent Volumes
2025-10-20	Improvement	Add the full list of PVCv2 StorageClass (0d, 7d, 35d, 1y)	Tomcat in Cloud: Persistent and Ephemeral Volumes
2025-10-09	New	New Tomcat releases and updated versions	Tomcat in Cloud: Version & Patches
2025-10-09	Improvement	Remove JFM documentation from public space	Tomcat in Cloud: Configure Application Log Files
2025-10-08	Improvement	Removed host from mailSession (only smtpHost is used)	Tomcat in Cloud: Create Custom Deployment
2025-10-07	New	Added new section for Packaging Accompanying Files	Tomcat in Cloud: Simple User Tutorial
2025-10-06	Improvement	Default values and descriptions added where missing	Tomcat in Cloud: Create Custom Deployment
2025-10-01	New	New conventions for PVCv2 creations	Tomcat in Cloud: Persistent and Ephemeral Volumes
2025-09-23	Improvement	Added Tomcat and TomEE version specifications	Tomcat in Cloud: Version & Patches
2025-09-23	Improvement	Updated URL and parameters	Tomcat in Cloud: Configure Application Log Files
2025-09-19	Improvement	Added prerequisites section in User Journey	Tomcat in Cloud: User Journey
2025-09-18	Improvement	Changed HPA threshold from 80% to 70% in Autoscaling	Enable auto-scaling

2025-09-12	Improvement	Added warnings for clientFilesKey and clientParamsKey	Tomcat in Cloud: Create Custom Deployment
2025-09-08	New	Added type to DataSource definition (default is 'javax.sql.DataSource')	Tomcat in Cloud: Create Custom Deployment Tomcat in Cloud: Simple User Tutorial
2025-09-05	New	Change datasource example with provider field factoryName renamed to factory	Tomcat in Cloud: Simple User Tutorial Tomcat in Cloud: Create Custom Deployment
2025-09-04	New	Added params logArgs, logEnv, logProps to tomcat.jvm section	Tomcat in Cloud: Create Custom Deployment
2025-09-03	New	New page for jfm application	Tomcat in Cloud: Configure Application Log Files
2025-09-02	New	Added Workload Scheduler documentation	Tomcat in Cloud: Workload Scheduler Tomcat in Cloud: Create Custom Deployment
2025-09-02	New	Added link to Dynatrace monitoring for Tomcat	Tomcat in Cloud: Monitoring
2025-08-07	New	debugJPDA removed from documentation (remote debugging not allowed)	Tomcat in Cloud: Create Custom Deployment
2025-08-06	Improvement	Added debugLogEnabled parameter	Tomcat in Cloud: Create Custom Deployment
2025-08-01	Improvement	Removed deprecated driverClassName property from config.yaml	Tomcat in Cloud: Create Custom Deployment
2025-07-31	Improvement	Added database providers in examples, mssql as a db provider	Tomcat in Cloud: Create Custom Deployment
2025-07-29		Added logRetentionPeriod parameter	Tomcat in Cloud: Create Custom Deployment Tomcat in Cloud: Configure Application Log Files
2025-07-26	Improvement	Internal URL part removed given that customer does not have to care about	Tomcat in Cloud: Create Custom Deployment
2025-07-26	New	Ingress name composition explained in Custom Deployment	Tomcat in Cloud: Create Custom Deployment

Tomcat in Cloud: Simple User Tutorial

Homepage

Page Topics

Related Links

Jumpstart your experience

Service Introduction Videos

Service Description & Architecture

Getting Started

Introduction

This user journey maps out the sequential steps a user takes to achieve specific goals in Tomcat Cloud. It enables users to have a quick-tour of the service offering by manipulating different features.

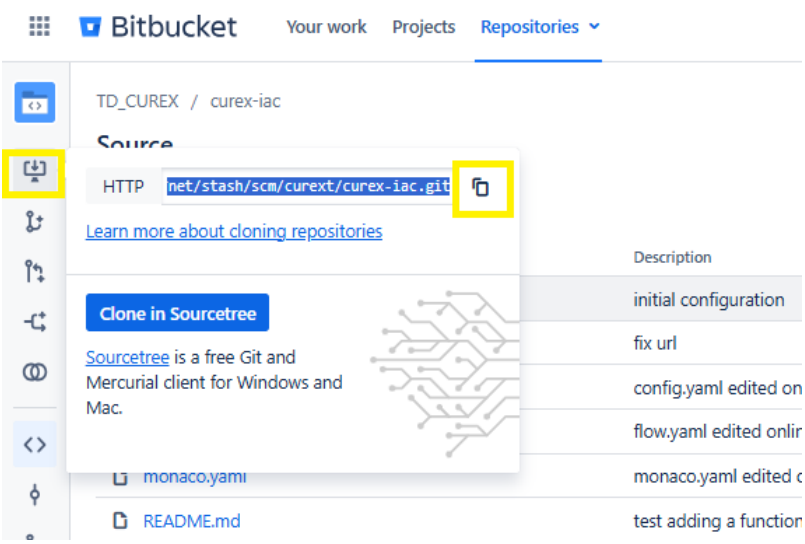
If more information is needed on a specific subject, links are provided accordingly.

Sit back, relax and enjoy your user journey !

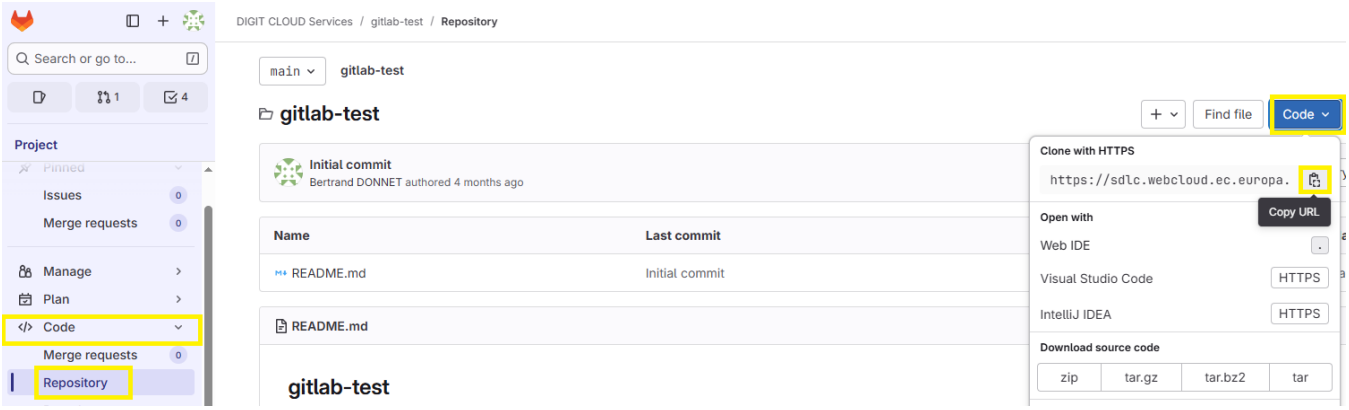
Prerequisites

Onboarding must have been done first ! For this, the URL of the git repository is needed

To find this required URL on **Bitbucket**:



To find the URL on **Gitlab** :



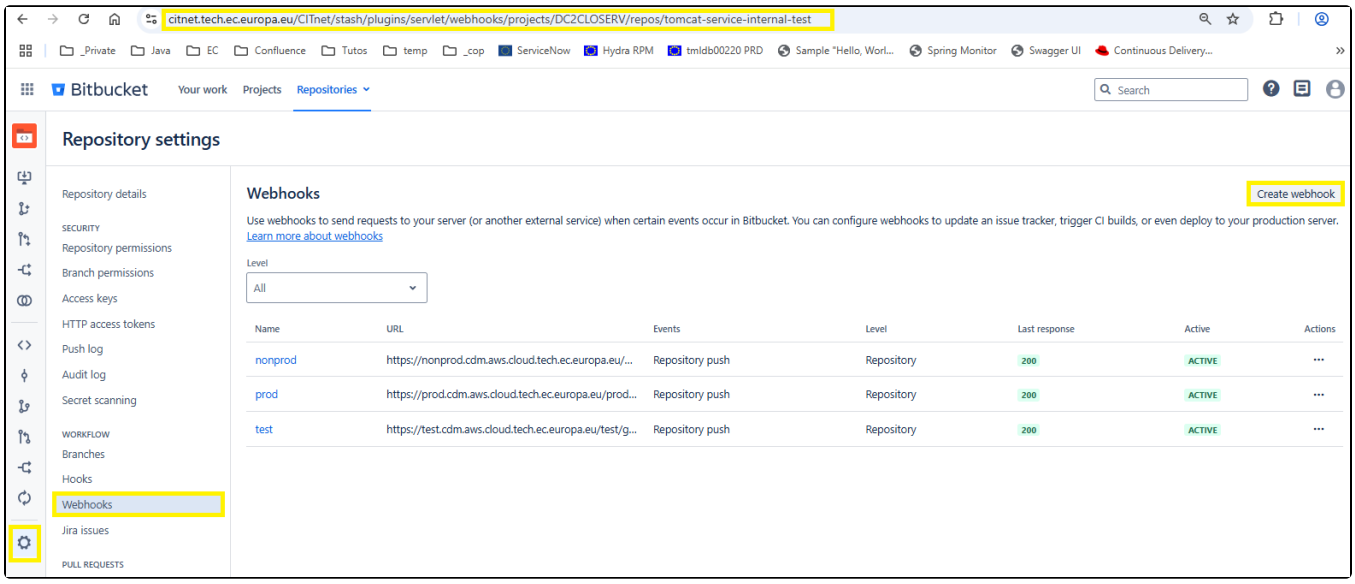
Configure the Git Webhook [only for Bitbucket users]

A webhook must first be configured and tested in the Git repository provided during onboarding. This webhook is essential for triggering pipeline executions and will be activated each time a configuration change is made.

- If the webhook is not correctly configured, the pipeline will not be triggered.
- If the Git repository name does not exactly match the URL provided during onboarding, the pipeline will not start, and no feedback will be returned to the customer.
- If the Git repository name is changed after onboarding (e.g., due to a migration from Bitbucket to GitLab), the onboarding process must be repeated; otherwise, the pipeline will not function.

To create and test the required webhook in Bitbucket, follow these steps:

1. Click on the **Settings** button on the lower-left corner.
2. Select the **Webhooks** entry.
3. Click on **Create webhook** button.

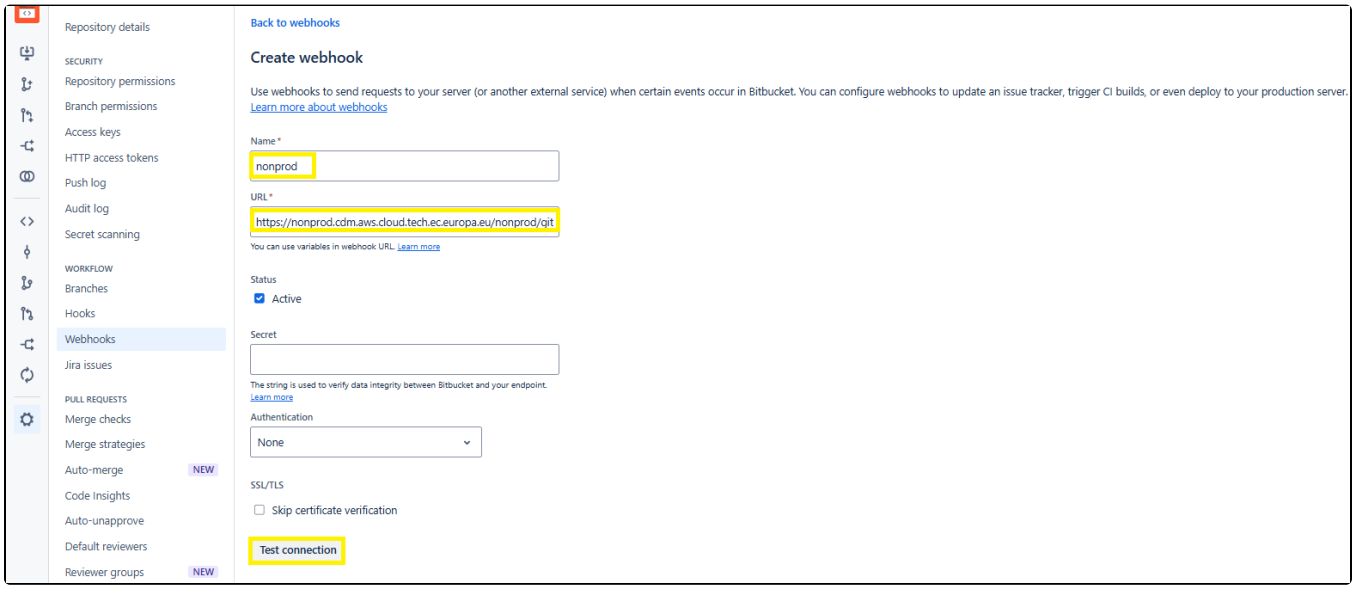


A second screen will be displayed:

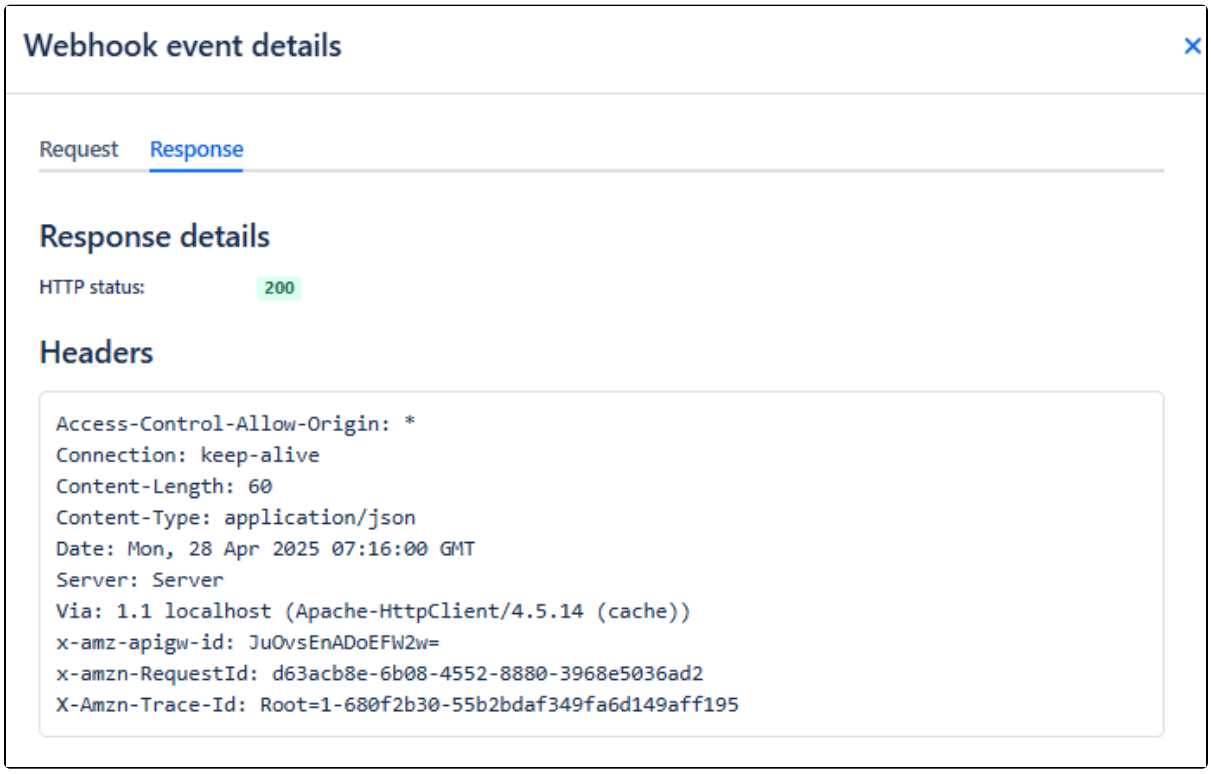
4. Enter a **name** for the webhook (free text)
5. Enter a valid **URL** pointing to the desired environment

   NONPROD environments : https://**nonprod**.cdm.aws.cloud.tech.ec.europa.eu/**nonprod**/git-webhook

   PROD environments : https://**prod**.cdm.aws.cloud.tech.ec.europa.eu/**prod**/git-webhook
6. Click on the **Save** button.



During the creation process, clicking the **Test Connection** button will display the following pop-up window.



If the HTTP response code is not **200**, this indicates an issue, and the webhook is not functioning correctly.
The configuration problem must be resolved **before proceeding**.

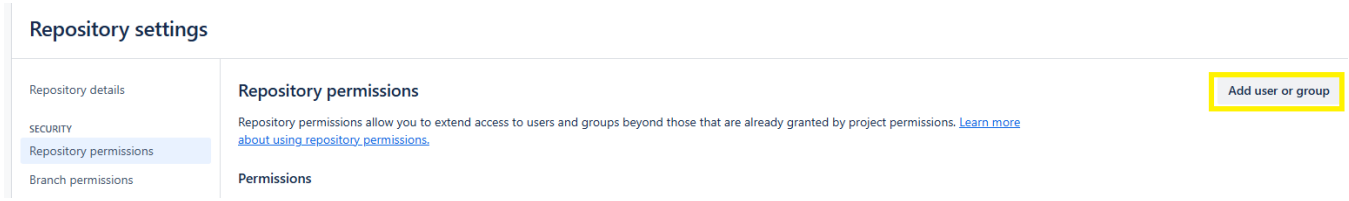
More information : Tomcat in Cloud: Configuration of the GitOps Repo

Configure the Git Access [only for Bitbucket users]

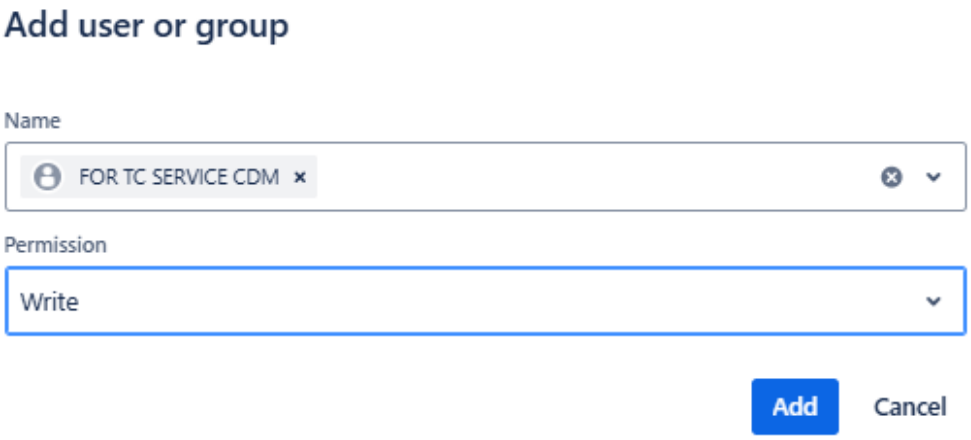
A **Tomcat functional user**, provided during onboarding, must be configured in the Git repository. This access is essential for enabling feedback to be triggered and delivered to the customer.

Perform the following steps in **Bitbucket** to add the functional user.

1. Click on the **Repository settings** button on the lower-left corner.
2. Select the **Repository permissions** entry.
3. Click on **Add user or group** button.



4. Add "FOR TC SERVICE CDM" with write access



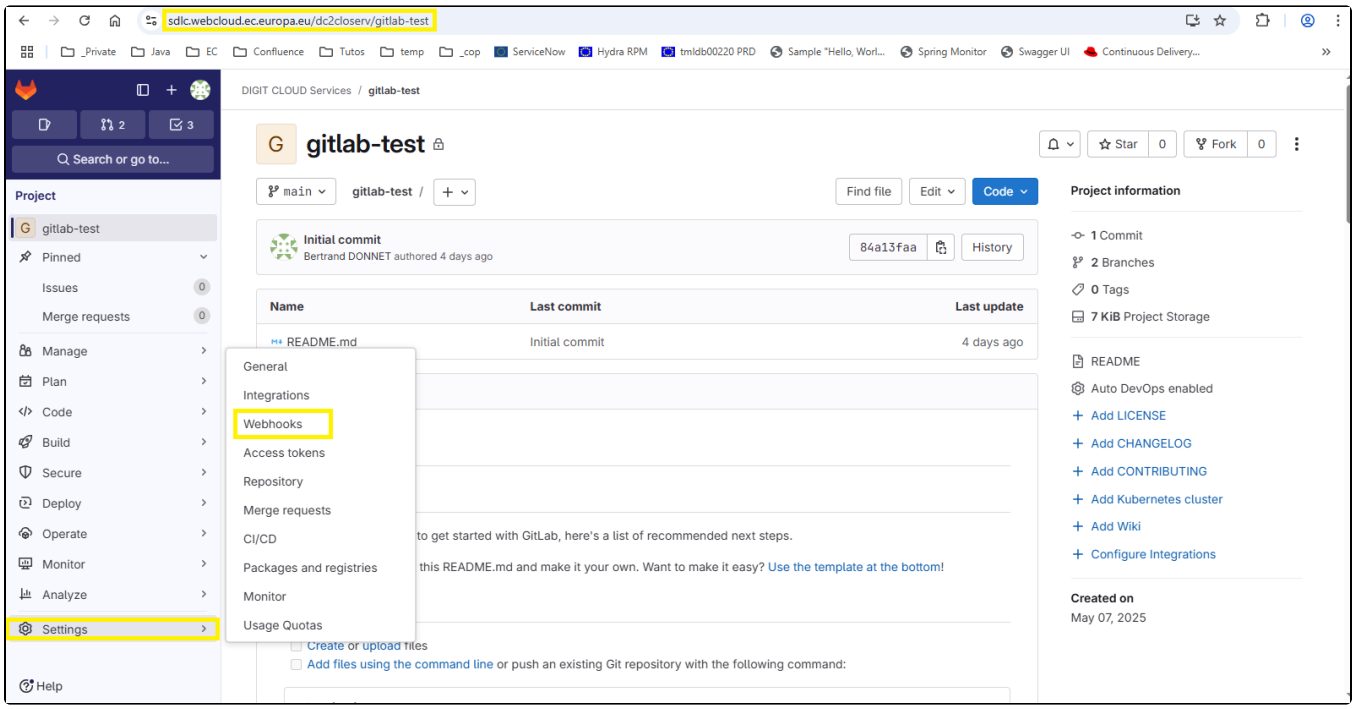
Configure the Git Webhook [only for GitLab users]

A webhook must first be configured and tested in the Git repository provided during the onboarding process. This webhook is essential for triggering pipeline executions and is activated whenever a configuration change is made.

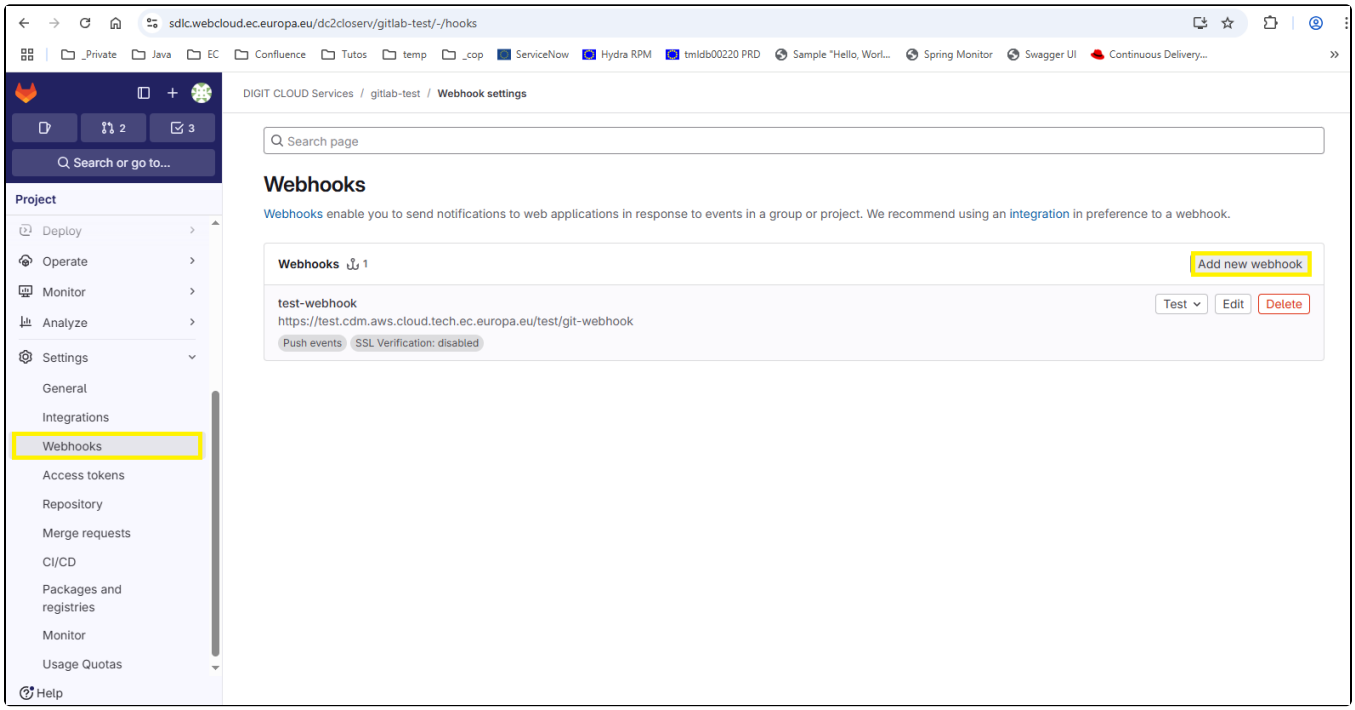
- If the webhook is not properly configured, the pipeline cannot be triggered.
- If the Git repository name does not exactly match the URL provided during onboarding, the pipeline will not start, and no feedback will be sent to the customer.
- If the Git repository name is changed after onboarding (e.g., due to a migration from Bitbucket to GitLab), the onboarding process must be repeated; otherwise, the pipeline will not function.

To create and test the required webhook in GitLab, follow these steps:

1. Click on the **Settings** icon in the lower-left corner.
2. Select **Webhooks** from the menu.
3. Click the **Create webhook** button.



4. Click on **Add new webhook**



A second screen will be displayed:

5. Enter a valid **URL** pointing to the desired environment

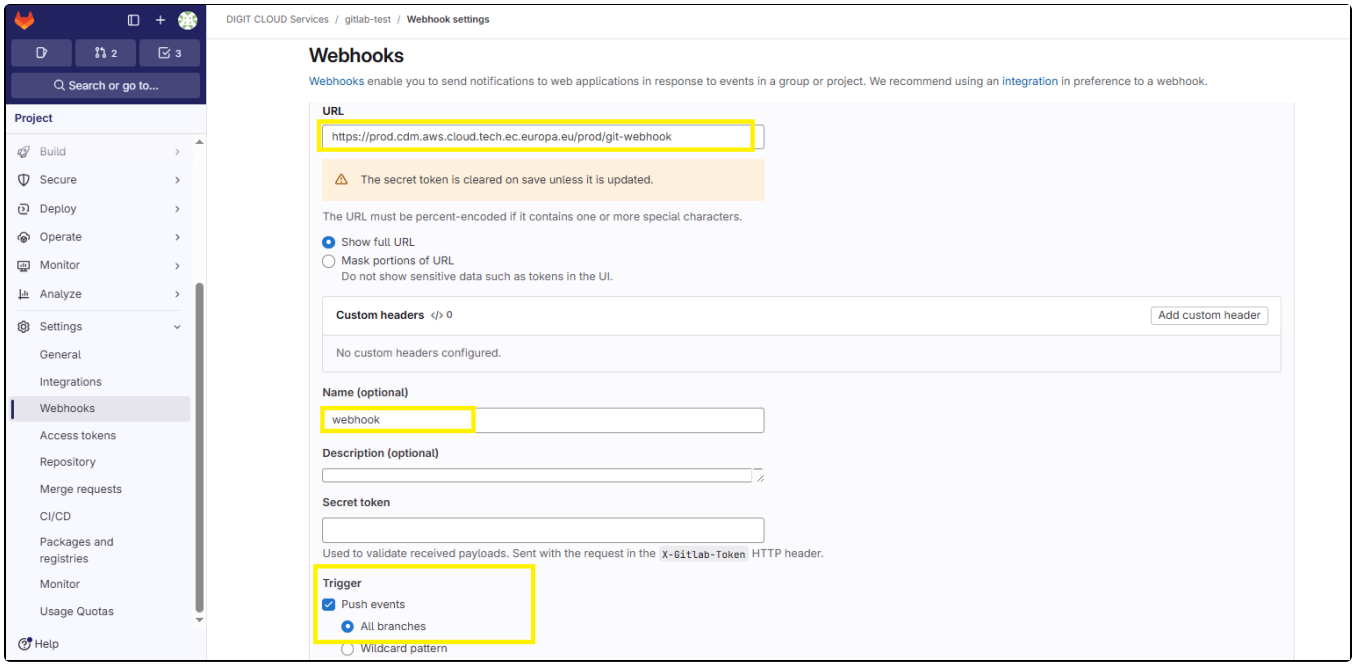
   NONPROD environment : https://**nonprod**.cdm.aws.cloud.tech.ec.europa.eu/**nonprod**/git-webhook

   PROD environment : https://**prod**.cdm.aws.cloud.tech.ec.europa.eu/**prod**/git-webhook

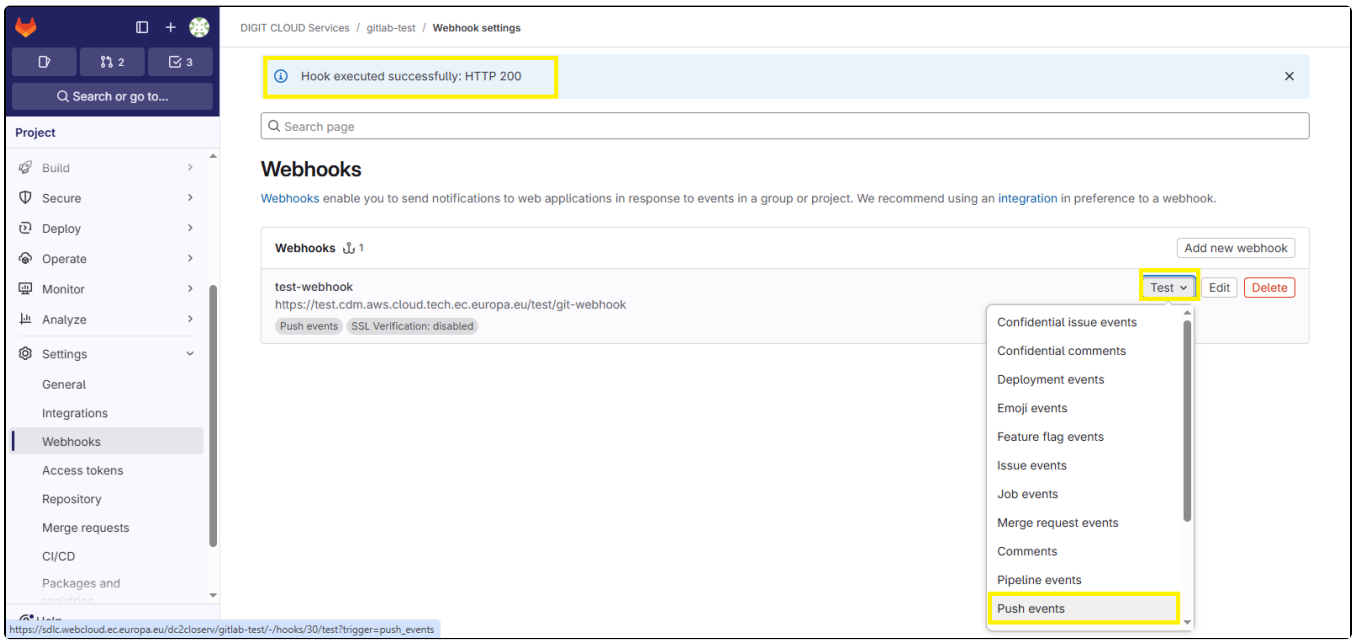
   ....

XXX environment : https://xxx.cdm.aws.cloud.tech.ec.europa.eu/xxx/git-webhook

6. Enter a **name** for the webhook (optional free text)
7. Check the **Push events** trigger with option "**All Branches**"
8. Disable **SSL Verification** option
9. Click on **Add Webhook** button



During the creation process, clicking the **Test Connection** button will display the following pop-up.



If the HTTP response code is anything other than **200**, it indicates an issue and the webhook is not functioning correctly. This configuration problem must be resolved **before proceeding**.

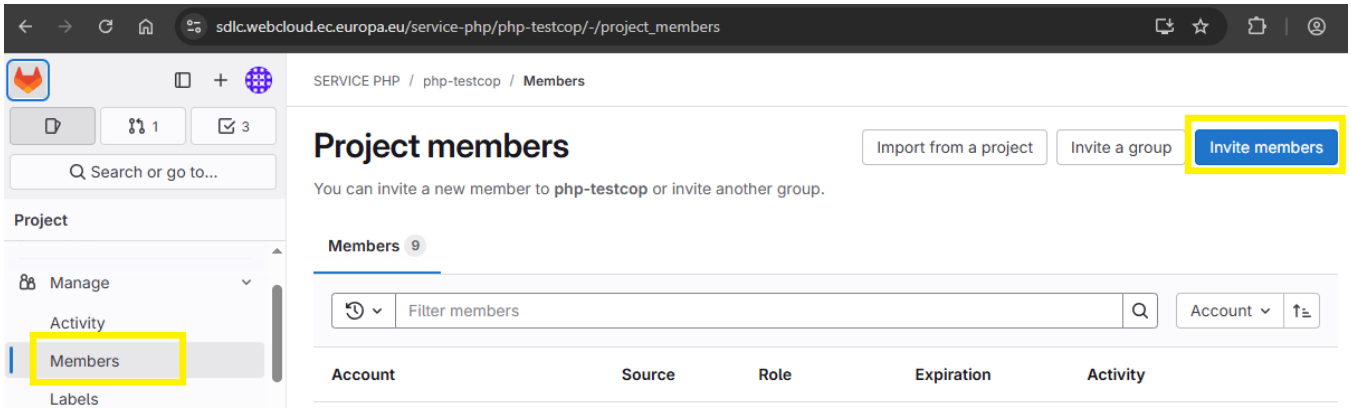
More information : Tomcat in Cloud: Configuration of the GitOps Repo

Configure the Git Access [only for GitLab users]

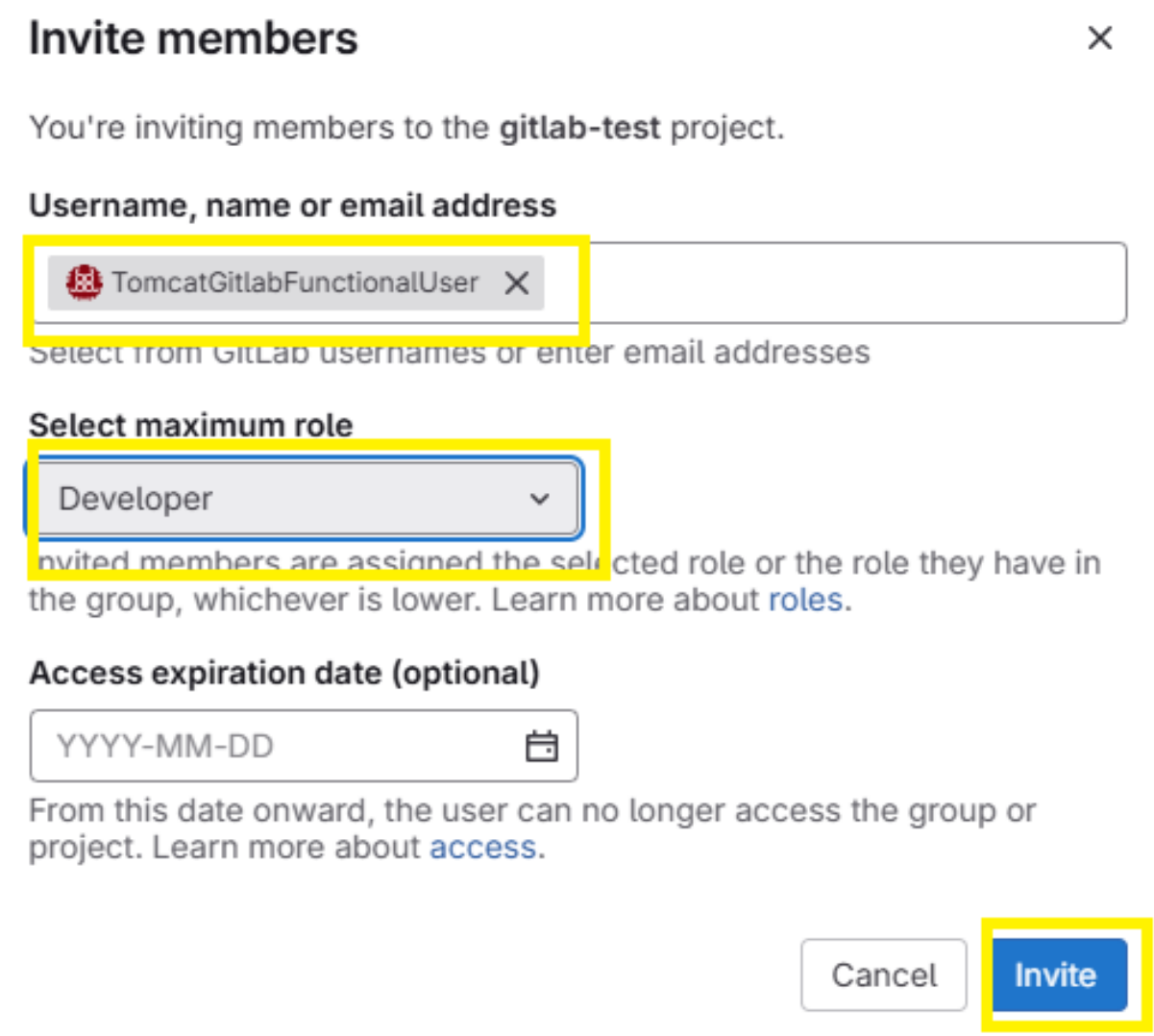
A Tomcat functional user provided during onboarding must be configured in the git repository. This access is crucial for triggering fe edback to the customer.

Perform the following steps in **Gitlab** to add the functional user.

1. Click on the **Manage → Members** button on the lower-left corner.
2. Click on **Invite members** button.



3. Add "TomcatGitlabFunctionalUser" with Developer role access (with no expiration date defined) and click on **Invite** button

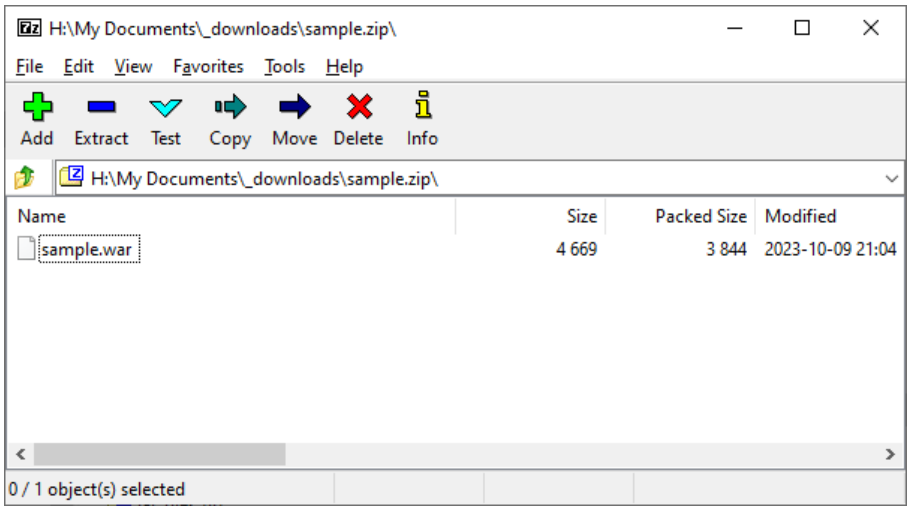


Package the Application Files

Before building and deploying the image, an artifact must be provided in Nexus.

This artifact consists of a ZIP containing one or more Java applications (static WARs, dynamic WARs, EARs). These applications will be copied in the Tomcat webapps directory of the generated image.

For example, the following ZIP file containing a single application simply named "sample" can be used.

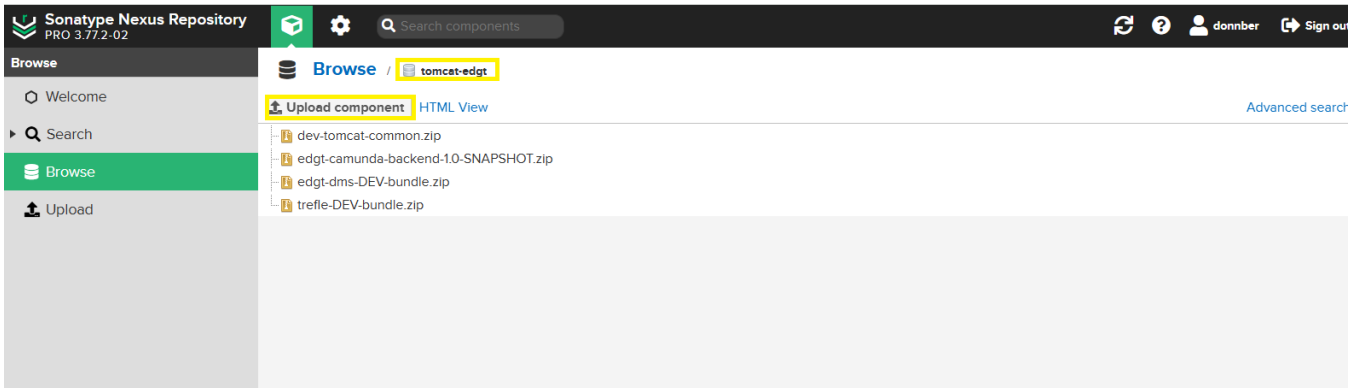


- A sample ZIP file can be downloaded here : sample.zip
- Please be aware that the **case-sensitive name of the WAR file** will be used as the WebContext
  - **sample.war** file will be deployed under the **/sample** web context
  - **x-y-z.war** file will be deployed under the **/x-y-z** web context
  - **ROOT.war** file will be deployed under the / web context
- If your ZIP contains more than one artifact, they will be deployed sequentially in alphabetical order
  - e.g. a ZIP containing **back-end.war** and **front-end.war** artifiacts
    - **back-end.war** will first be deployed under the **/back-end** web context
    - **front-end.war** will then be deployed under the **/front-end** web context

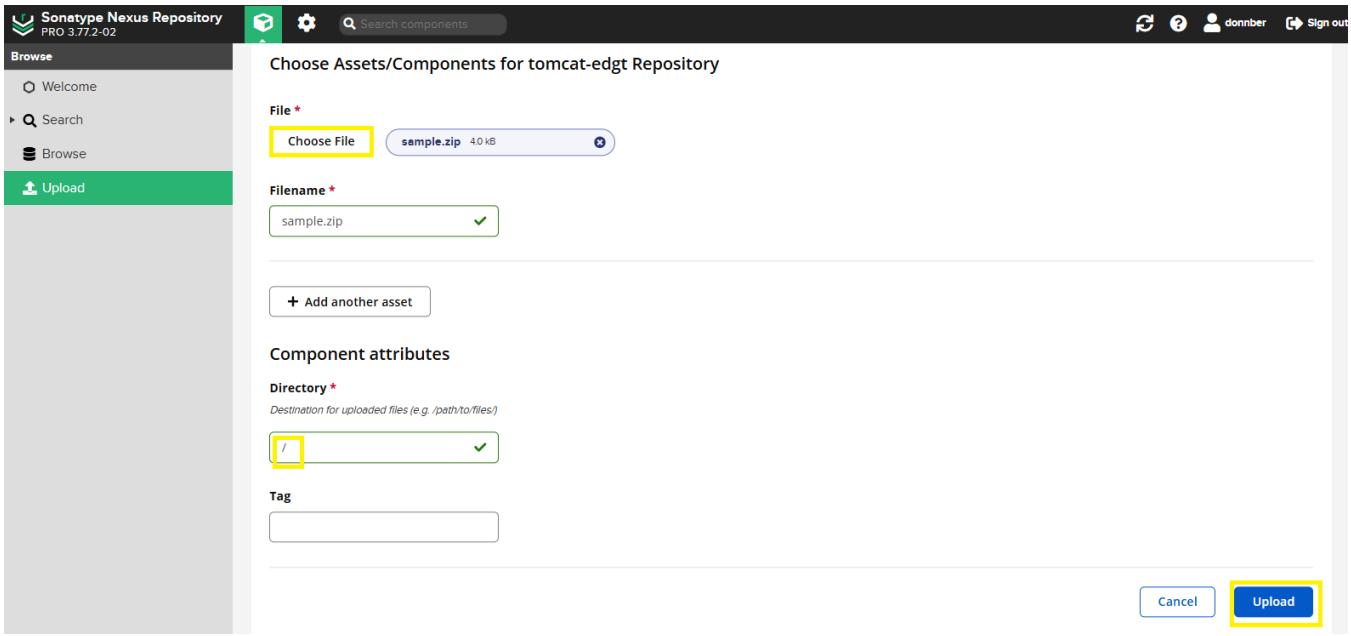
Once created, this ZIP file must be uploaded into the Nexus Repository.

A link should have been sent after onboarding. For instance if your IS is "edgt", the url will be the following https://tc-nexus.devops.tech.ec.europa.eu/#browse/browse:tomcat-**edgt**

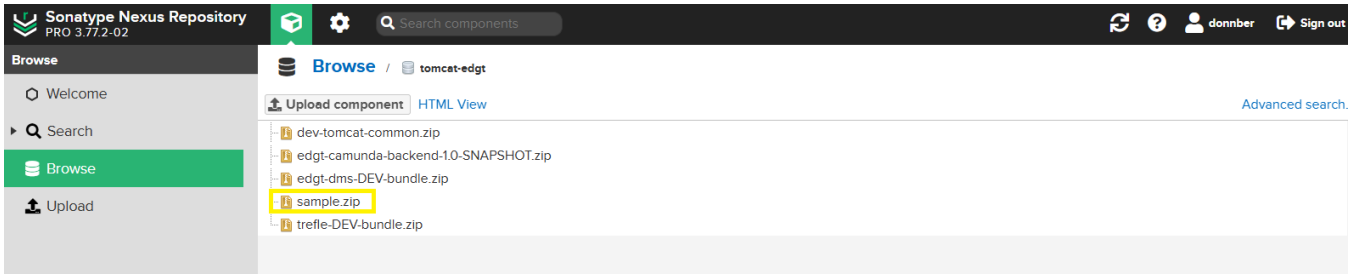
The current content of the repository will be listed as below.



1. Click on the **Upload component** button. A new form will be displayed.
   a. choose the ZIP file to upload.
   b. choose a destination path ("/" for root level).
   c. click on the **Upload** button.



The repository content will be updated with the following list.



The artifact is now ready for the Build and Deploy Application step.

It must be identified under the **artifactName** property of the **config.yaml** file.

Package the Application Accompanying Files

If needed, non-Java files can be packaged in a separate ZIP file containing all the accompanying files (eg properties, yaml, xml, ...)

This ZIP containing all the files at root level will be unzipped under the /lib-client directory which will be automatically added to the classpath

Here is an example of a zip file containing properties

It must be identified under the **applicationPropertiesZip** property of the **config.yaml** file.

Build and Deploy the Application

During onboarding, a git repository and a branch were selected for configuration.

To start the Build and Deploy step, a simple **commit and push** must be performed.

A sample **config.yaml** file can be created with the following minimal content.

> ⚠ All values not provided will default to predefined values.
>
> A full description of all the parameters can be found here : Tomcat in Cloud: Create Custom Deployment

```
applicationName: "tccop-int-app"
ingressName: "tccop-int"
ingressPath: "/"
replicas: "1"
customerImageName: "tccop-int-app"
customerImageTag: "test338"
jvmVersion: 8
tomcatVersion: "9.0"
artifactName: sample.zip
buildEnabled: true
```

After a successful commit on the git repository on the correct branch, the build and deploy will start and notifications will be sent by email to the FMB (Functional Mail Box) defined during onboarding.

> ⚠ If you don't receive any notification or feedback, please first verify that :
>
> - the push is done on the right **git repository** (the one that was defined during onboarding)
> - the push is done on the right **branch** of the repository (the one named like the environment name chosen during onboarding)
> - the pushed **filename** is exactly "config.yaml" ("Config.yaml" and "config.yml" won't work)

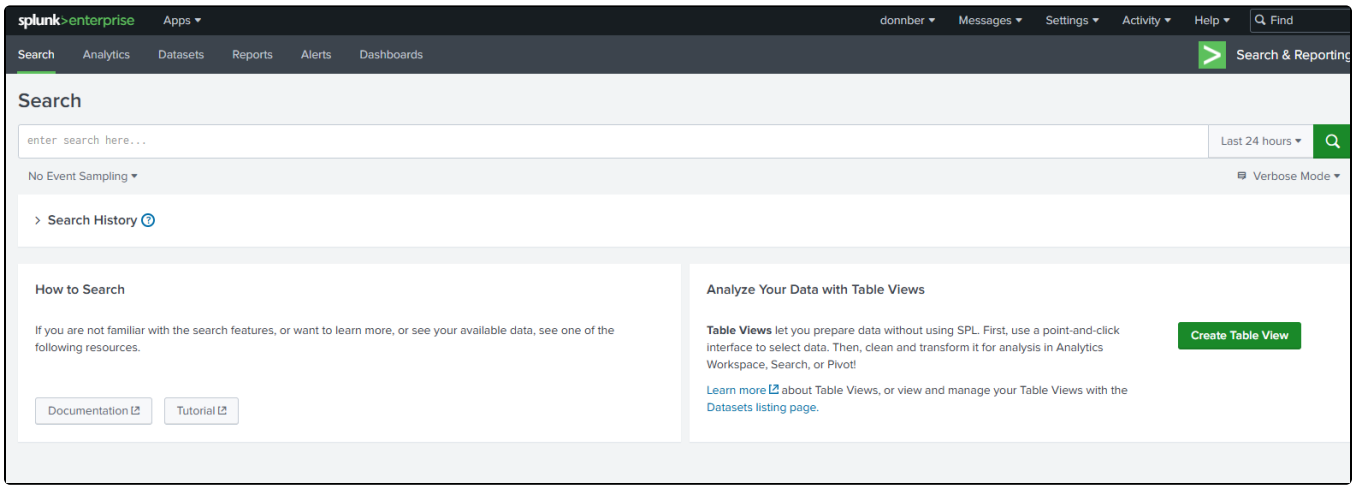
More information : Tomcat in Cloud: Create Custom Deployment

View the Application Logs with Splunk

Logs can be analysed using **Splunk**.

1. Go to the following URL : https://splunk.tech.ec.europa.eu/en-US/app/search/search.

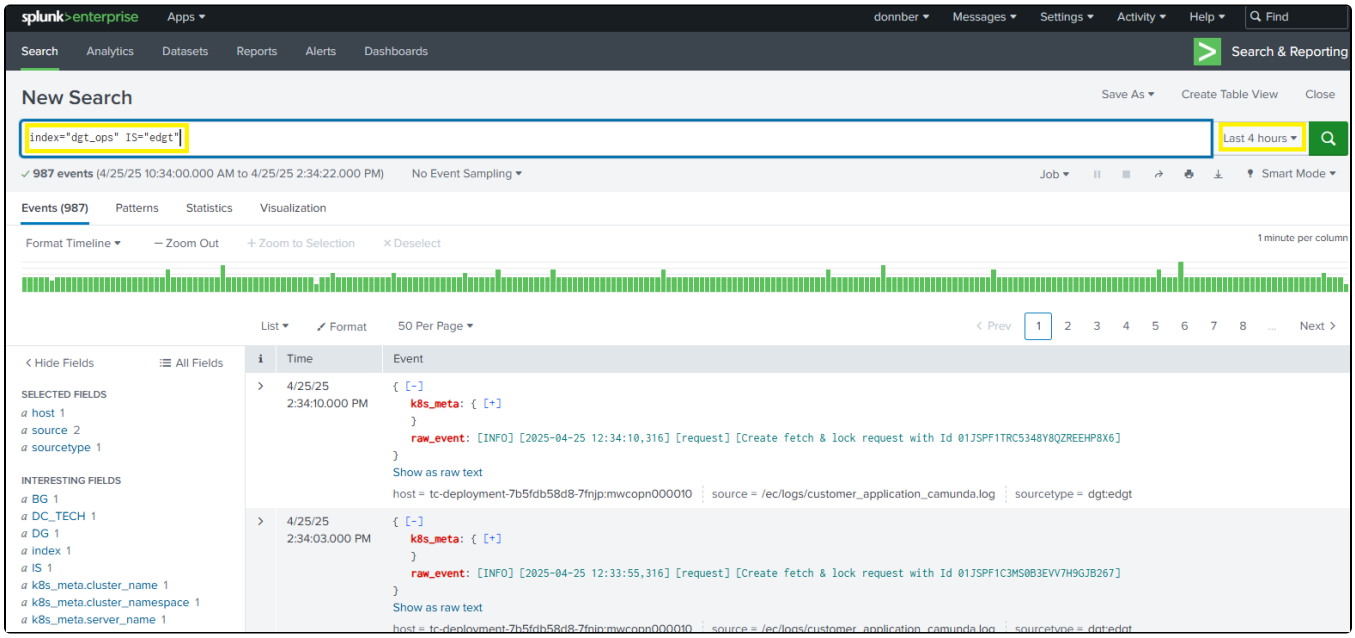
The following Splunk homepage will be displayed.



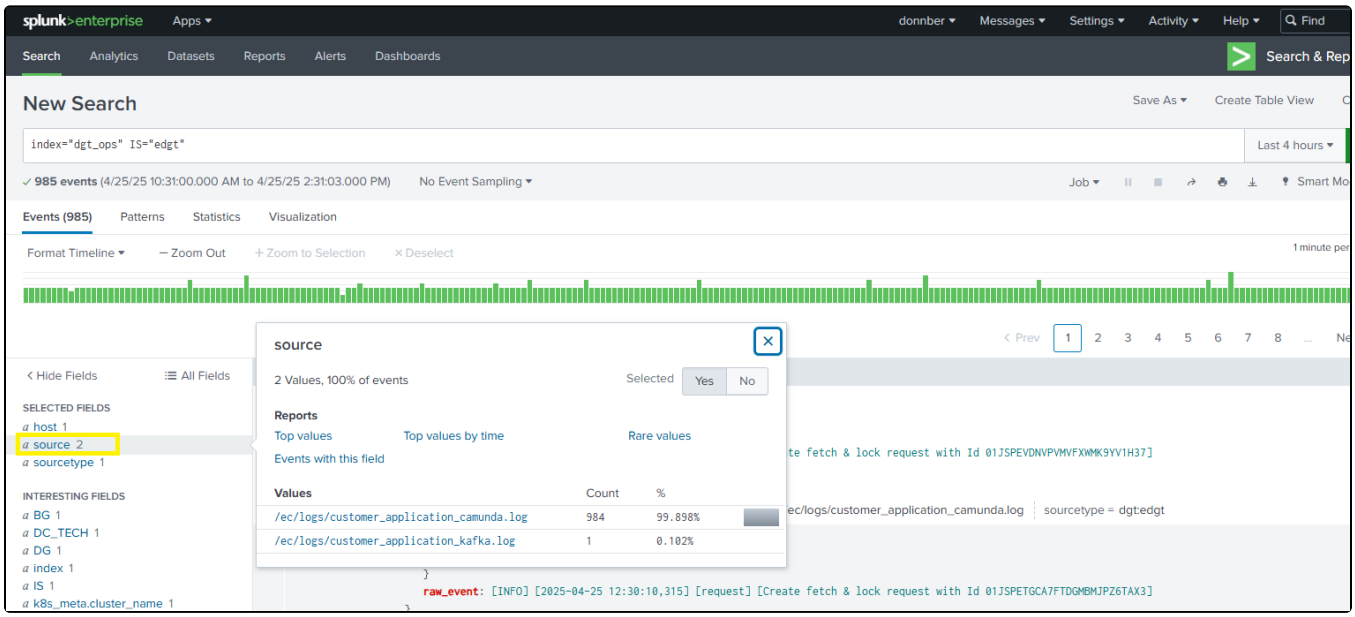
2. If your DG is 'dgt' and your IS is 'edgt', enter the following in the search bar:

   index="dgt_ops" IS="edgt"

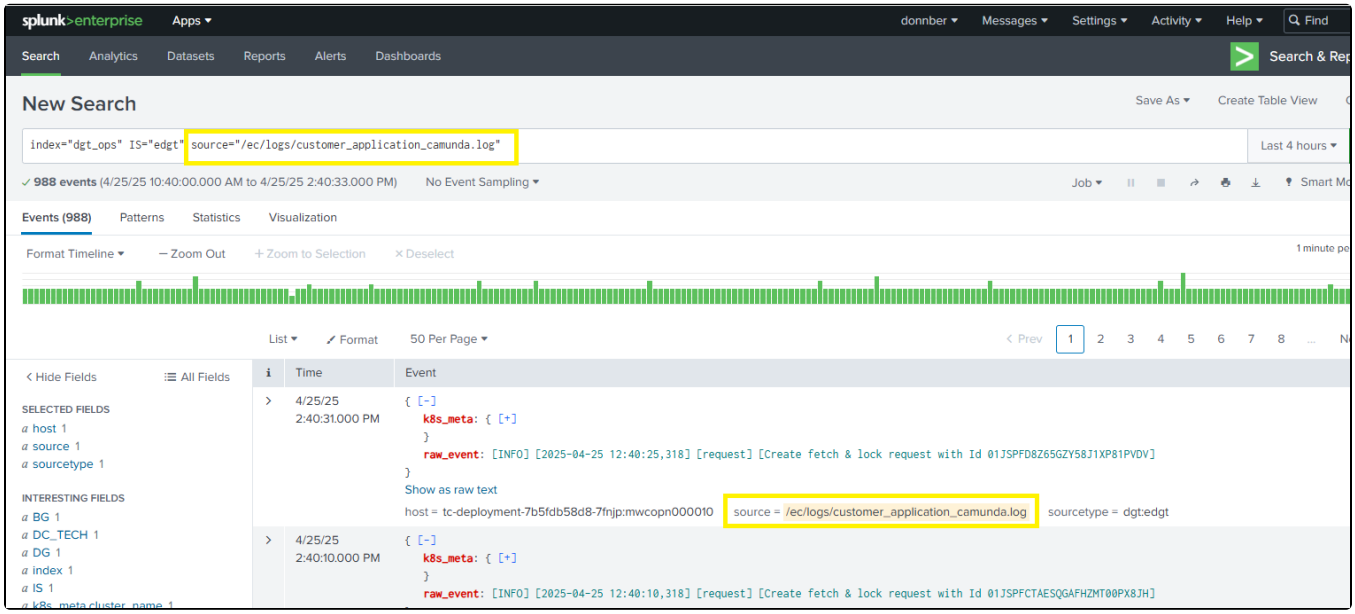
⚠ The time frame is set to 4 hours by default.



⚠ If you want to refine your search, click for instance on the **source** on the left and choose a specific source file.



The parameter will be automatically added to the query and only the entries with the specified source will be displayed.



More information : Tomcat in Cloud: Configure Application Log Files

---

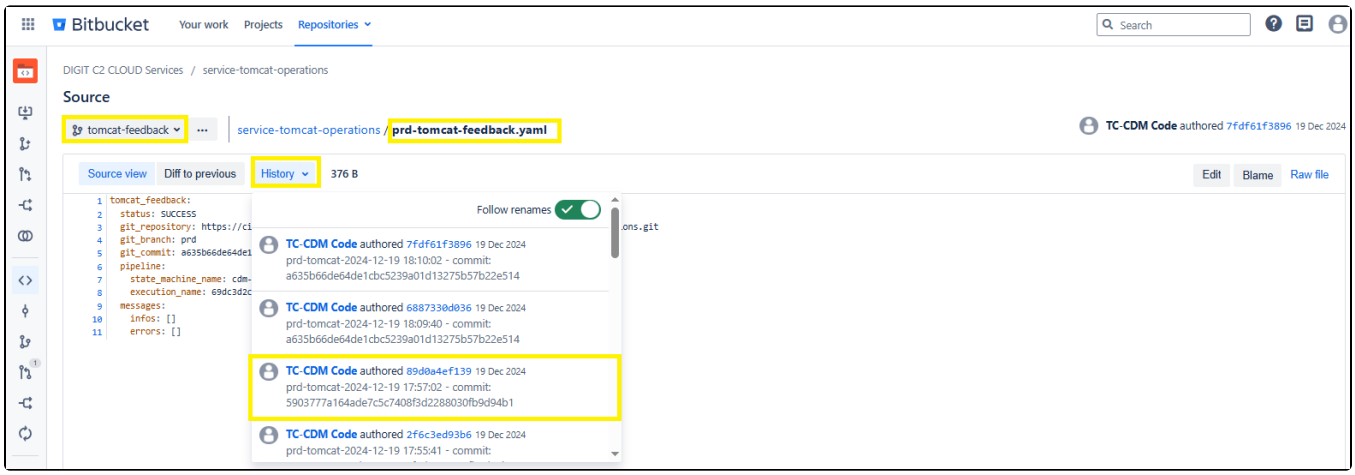
View the Application Feedback Branch

A branch named **`tomcat-feedback`** is created in the configuration repository to store all notification-related information.

Each environment has a dedicated file following the naming convention: **`[ENVIRONMENT_NAME]-tomcat-feedback.yaml`**.

For example, the **production (prd)** environment will use the file: **`prd-tomcat-feedback.yaml`**.

⚠ Once an environment file is selected, its history can be parsed to retrieve previously received messages.



More information : Tomcat in Cloud: Git Feedback Notifications

---

Access the Application through RPM

An RPM (Reverse Proxy Mapping) must be created to access the application.

RPM creation will take between 30 minutes and 2 hours.

```
rpms:
  - domain: intragate.development.ec.europa.eu
    targetResource: internal-test-api
    sourcecontextRoot: test
    targetcontextRoot: test
    rewrites: true
    ecas: true
    timeout: 300
    compression: false
    caching: true
    extendHttpMethod: false
    testPage: /sample
```

More information : Tomcat in Cloud: Reverse Proxy Mapping As Code (RPMaC)

Add OS Environment Variables

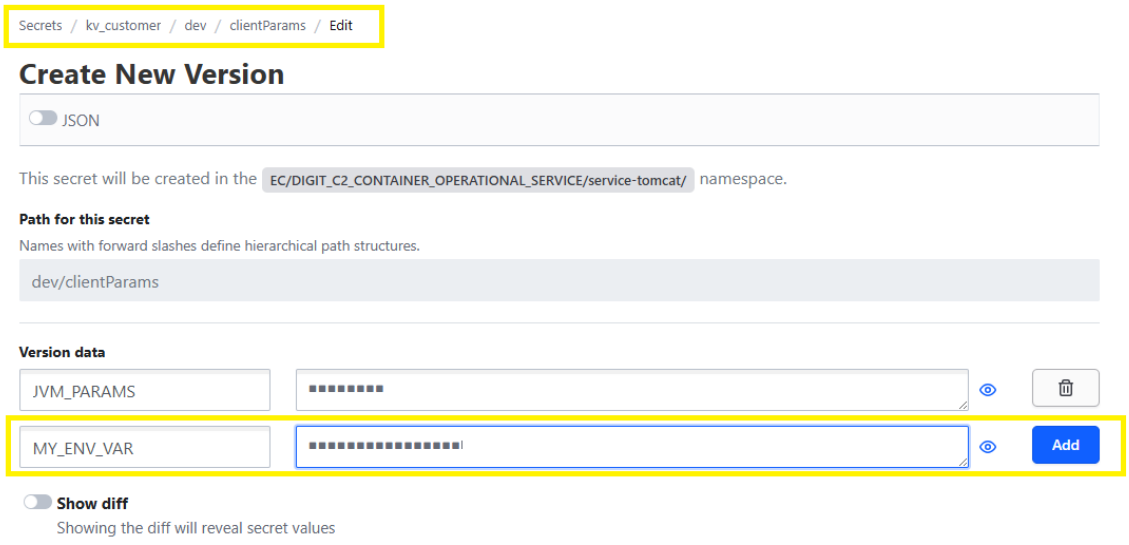
Environment variables can be defined in the Vault. They will be automatically

First, the *clientParamsKey* has to be added in the config.yaml file (line 14 in the following example)

```
applicationName: "tccop-int-app"
ingressName: "tccop-int"
ingressPath: "/"
replicas: "1"
customerImageName: "tccop-int-app"
customerImageTag: "test338"
jvmVersion: 8
tomcatVersion: "9.0"
artifactName: sample.zip
buildEnabled: true
clientParamsKey: clientParams
...
```

A secret name **MY_ENV_VAR** under **/<env_name_in_lowercase>/clientParams** has to be created :

NB: For instance, if your environment has been onboarded with name **My-Env**, path will be **/my-env/clientParams**



More information on parameters : Tomcat in Cloud: Create Custom Deployment

Add Specific JVM Parameters

There are 2 ways of adding JVM parameters to the application.

The simpler is to use the property **environmentVariables** like below (line 11-13 in the following example):

```
applicationName: "tccop-int-app"
ingressName: "tccop-int"
ingressPath: "/"
replicas: "1"
customerImageName: "tccop-int-app"
customerImageTag: "test338"
jvmVersion: 8
tomcatVersion: "9.0"
artifactName: sample.zip
buildEnabled: true
tomcat:
  jvm:
    environmentVariables: -Dparam1=a -Dparam2=b
```

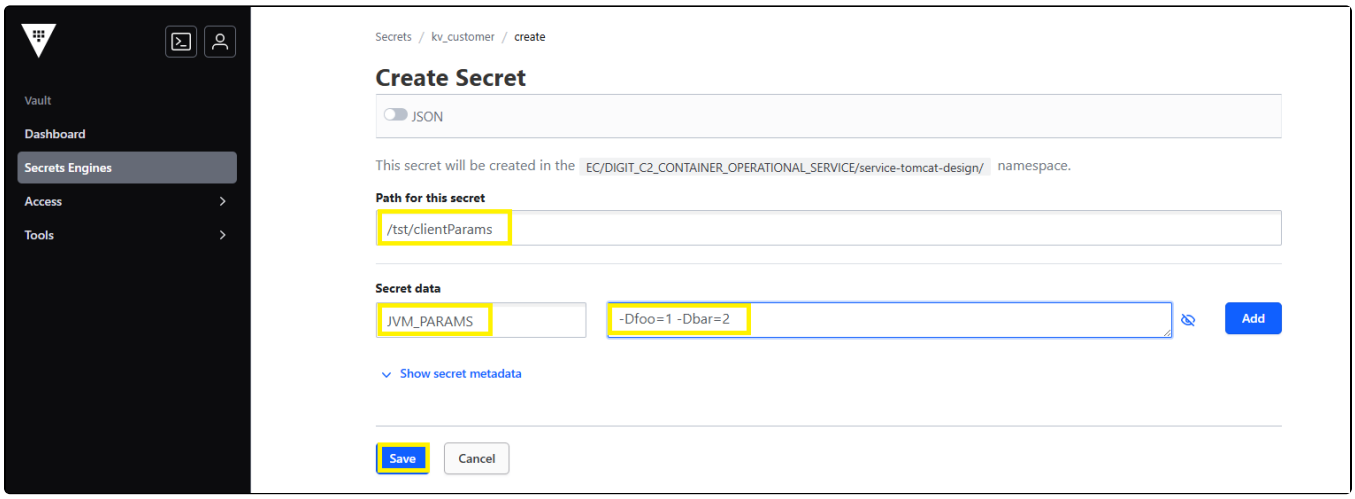
Another solution is to use the Vault to store the JVM Params, namely when some confidential data is provided (e.g. usernames, passwords, ...)

First, the *clientParamsKey* has to be added in the config.yaml file (line 14 in the following example)

```
applicationName: "tccop-int-app"
ingressName: "tccop-int"
ingressPath: "/"
replicas: "1"
customerImageName: "tccop-int-app"
customerImageTag: "test338"
jvmVersion: 8
tomcatVersion: "9.0"
artifactName: sample.zip
buildEnabled: true
clientParamsKey: clientParams
tomcat:
  jvm:
    environmentVariables: -Dparam1=a -Dparam2=b
```

A secret name **JVM_PARAMS** under **/<env_name_in_lowercase>/clientParams** has to be created :

NB: For instance, if your environment has been onboarded with name **My-Env**, path will be **/my-env/clientParams**



If both options are used, parameters will be concatenated on the command line.

More information on parameters : Tomcat in Cloud: Create Custom Deployment

---

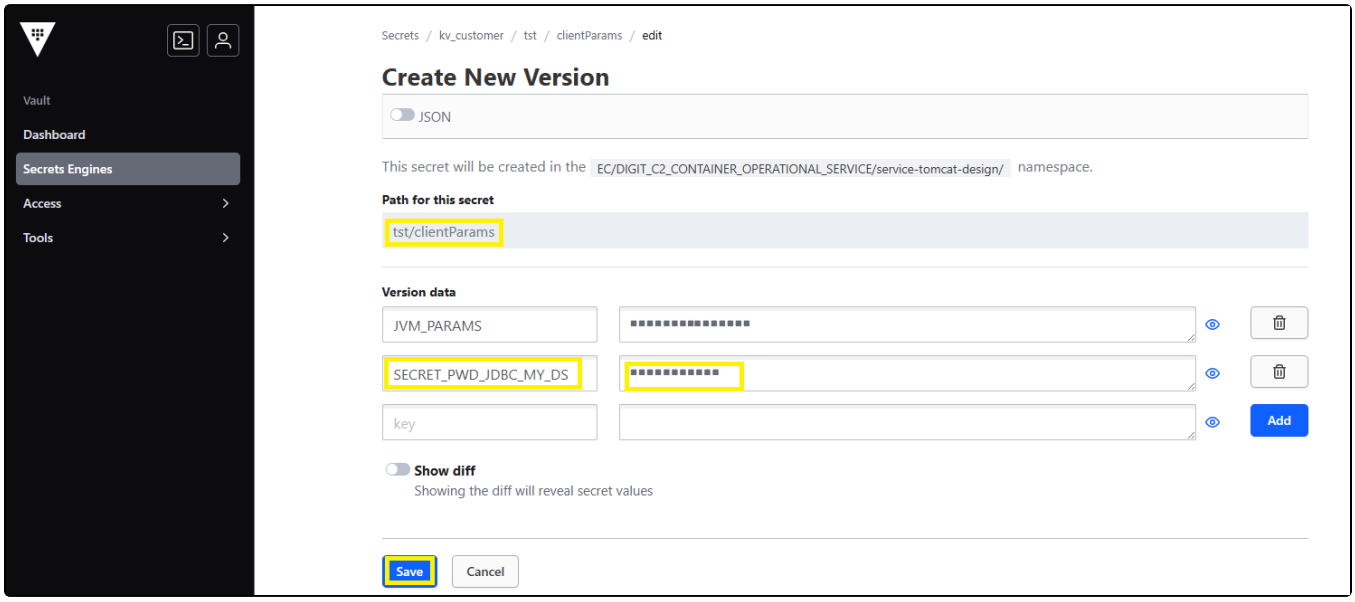
Add a JDBC Datasource to the Application

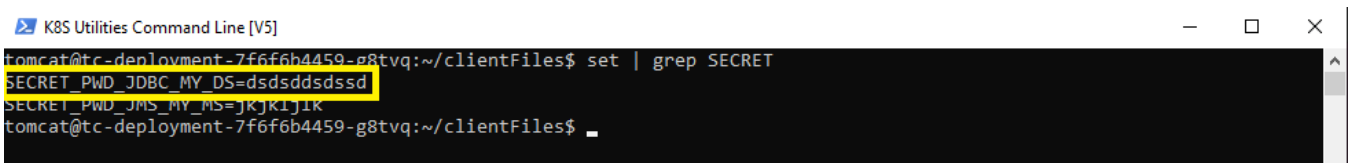
First, the datasource must be added in the YAML file. Here the datasource is named **jdbc/my-ds**

- In this tutorial, the datasource created will be simply named **my-ds**. In your specific case, please replace jdbc/**my-ds** and SECRET_PWD_JDBC_**MY_DS** with the desired name.
- For further explanations about naming, please go here : Tomcat in Cloud: Manage Secrets#DefineDatasourcesandMailsessionspasswordsintheVault

```
applicationName: "tccop-int-app"
ingressName: "tccop-int"
ingressPath: "/"
replicas: "1"
customerImageName: "tccop-int-app"
customerImageTag: "test338"
jvmVersion: 8
tomcatVersion: "9.0"
artifactName: sample.zip
buildEnabled: true
clientParamsKey: clientParams
tomcat:
  jvm:
    environmentVariables: -Dparam1=a -Dparam2=b
  datasources:
    - name: "jdbc/my-ds"
      provider: oracle
      type: javax.sql.DataSource
      url: jdbc:oracle:thin:@olrdev99.cc.cec.eu.int:1597/B4_COMM_01_D_TAF.cc.cec.eu.int
      username: "DSUSER"
      # For the password, a secret must be created in the Vault under kv_customer/<env>/clientParams
/SECRET_PWD_JDBC_MY_DS
      initialSize: 5
      minIdle: 5
      maxIdle: 10
      maxTotal: 15
```

The associated secret must created in the vault under the name kv_customer/<environment>/clientParams/**SECRET_PWD_JDBC_MY_DS**





More information on parameters : Tomcat in Cloud: Create Custom Deployment

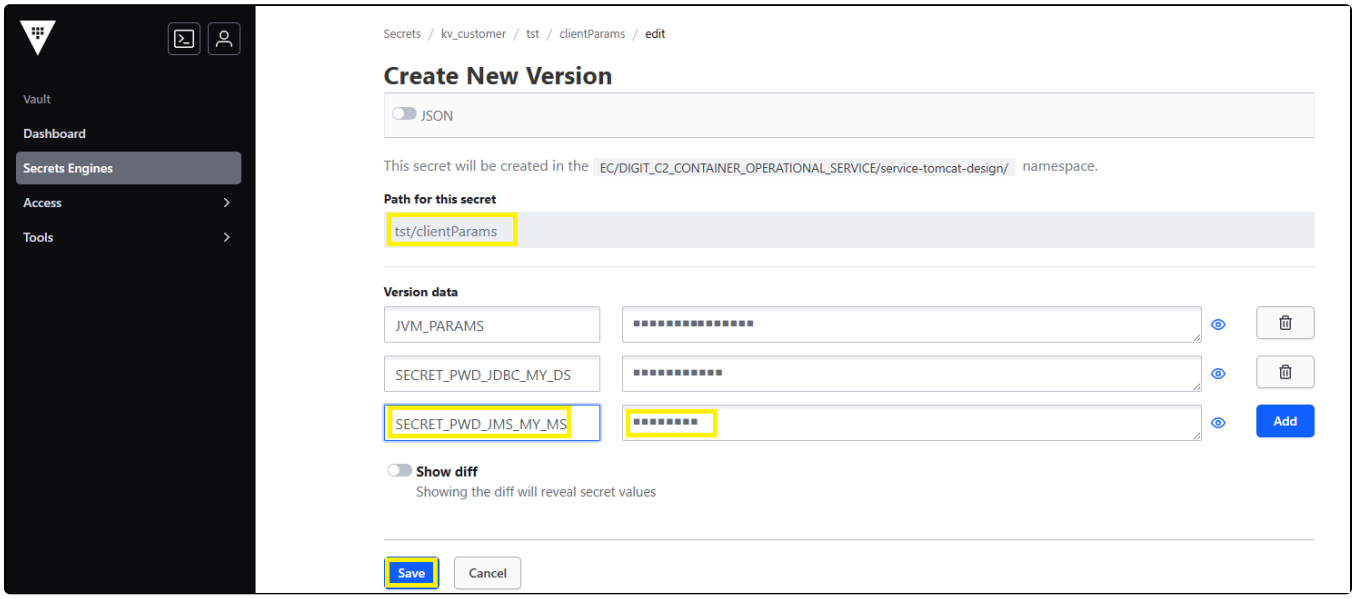
Add a JMS Mailsession to the Application

First, the mail session must be added in the YAML file.. Here the mailsession is named **jms/my-ms**

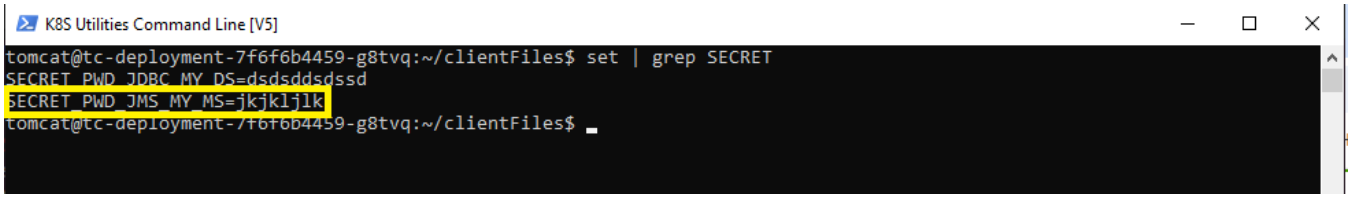
- In this tutorial, the mail session created will be simply named **my-ms**. In your specific case, please replace jms/**my-ms** and SECRET_PWD_JMS_**MY_MS** with the desired name.
- For further explanations about naming, please go here : Tomcat in Cloud: Manage Secrets#DefineDatasourcesandMailsessionspasswordsintheVault

```
applicationName: "tccop-int-app"
ingressName: "tccop-int"
ingressPath: "/"
replicas: "1"
customerImageName: "tccop-int-app"
customerImageTag: "test338"
jvmVersion: 8
tomcatVersion: "9.0"
artifactName: sample.zip
buildEnabled: true
clientParamsKey: clientParams
tomcat:
  jvm:
    environmentVariables: -Dparam1=a -Dparam2=b
  datasources:
    - name: "jdbc/my-ds"
      provider: oracle
      type: javax.sql.DataSource
      url:  jdbc:oracle:thin:@olrdev99.cc.cec.eu.int:1597/B4_COMM_01_D_TAF.cc.cec.eu.int
      username: "DSUSER"
      # For the password, a secret must be created in the Vault under kv_customer/<env>/clientParams
/SECRET_PWD_JDBC_MY_DS
      driverClassName: oracle.jdbc.OracleDriver
      initialSize: 5
      minIdle: 5
      maxIdle: 10
      maxTotal: 15
  mailsessions:
    - name: "jms/my-ms"
      host: host1
      from: from1
      type: javax.mail.Session
          # For the password, a secret must be created in the Vault under kv_customer/<env>/clientParams
/SECRET_PWD_JMS_MY_MS
```

The associated secret must be created in the vault under the name kv_customer/<environment>/clientParams/**SECRET_PWD_JMS_MY_MS**



A new environment variable is defined and can be seen from the pod.



More information on parameters : Tomcat in Cloud: Create Custom Deployment

Add Secret Files to the Application

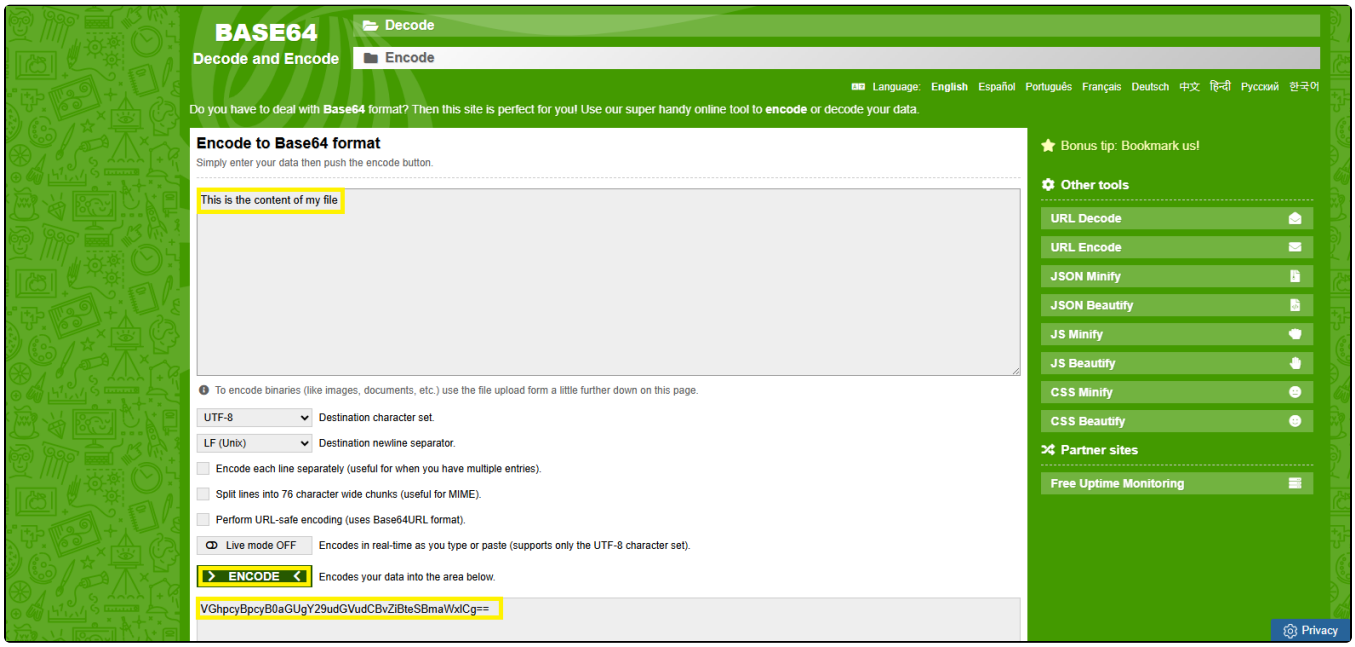
First, the **clientFilesKey** entry must be added to the **config.yaml** file (line

```
applicationName: "tccop-int-app"
ingressName: "tccop-int"
ingressPath: "/"
replicas: "1"
customerImageName: "tccop-int-app"
customerImageTag: "test338"
jvmVersion: 8
tomcatVersion: "9.0"
artifactName: sample.zip
buildEnabled: true
clientParamsKey: clientParams
clientFilesKey: clientFiles
tomcat:
  jvm:
    environmentVariables: -Dparam1=a -Dparam2=b
```

File content **must be encoded and padded** before being inserted into the vault.

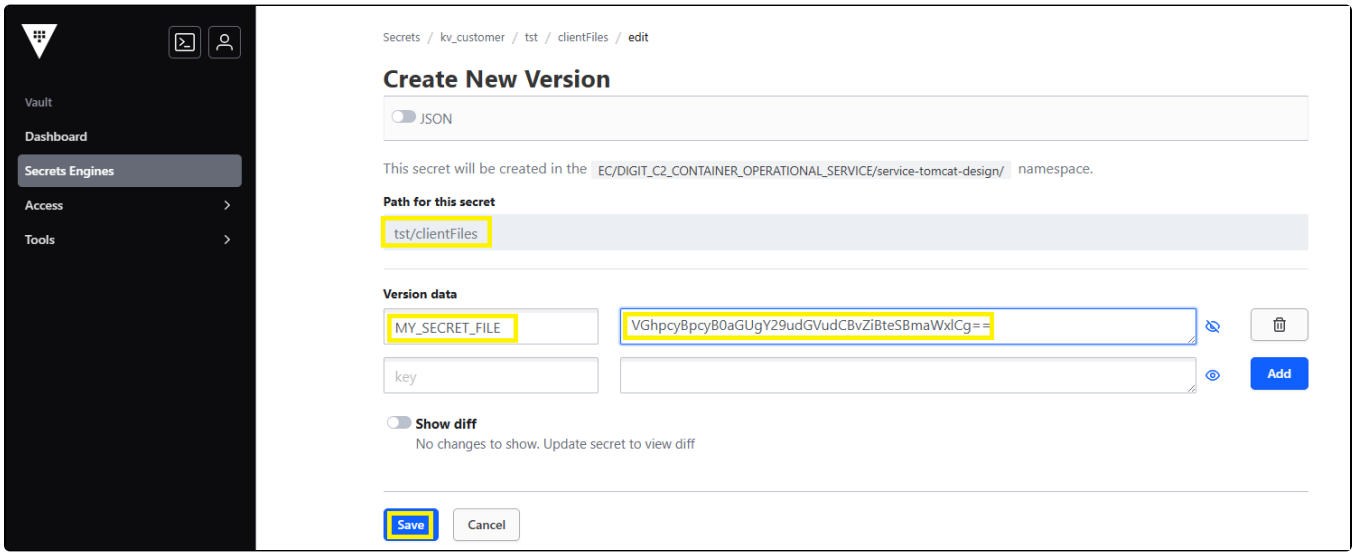
This can be done by using the following site : https://www.base64decode.org/ or with another tool if you prefer (linux command line, program ...)

NB: Even if files are encoded and padded for the Vault, they will be decoded and accessible with the original content (without encoding and padding)

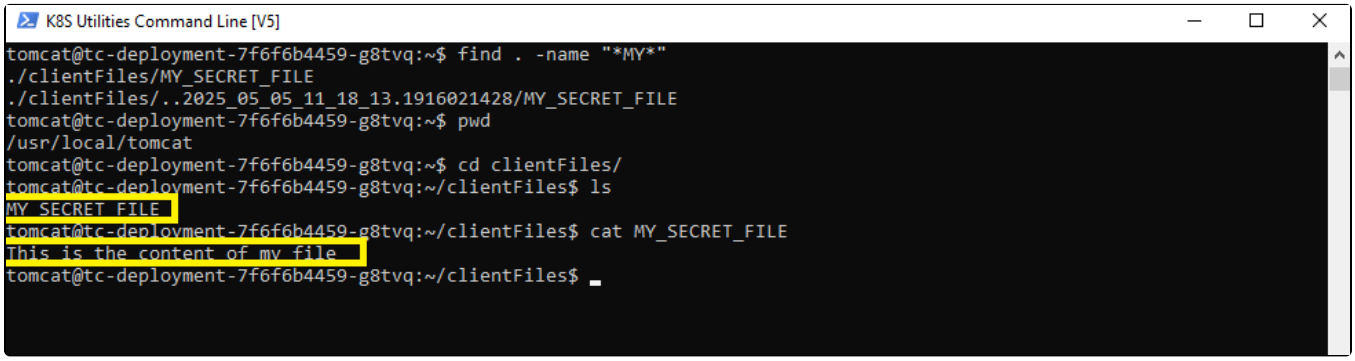


Add a key in the vault under the path **tst/clientFiles**.

For instance, add the key **MY_SECRET_FILE** with the encoded value of your file.



A new file is defined and can be seen directly from the pod.



Add a Liveness HTTP Probe for Automatic Restart

A liveness probe can be set up so that if the application does not respond after a certain amount of time, the application will be automatically restarted.

This can be set up like the following:

```
...
livenessProbe:
        probeEnabled: true
    probeType: HTTP_GET
        scheme: HTTPS
        path: /application/liveness-check
        timeoutSeconds: 10
    periodSeconds: 10
...
```

⚠ The path given here must return an adequate HTTP code, if the page returns 200 (OK) even if database is not working, the application won't be restarted.

More information on parameters : Tomcat in Cloud: Create Custom Deployment

Conclusion

Congratulations !

You have successfully finished the User Journey Tutorial and have gained some experience on the main features of the service.

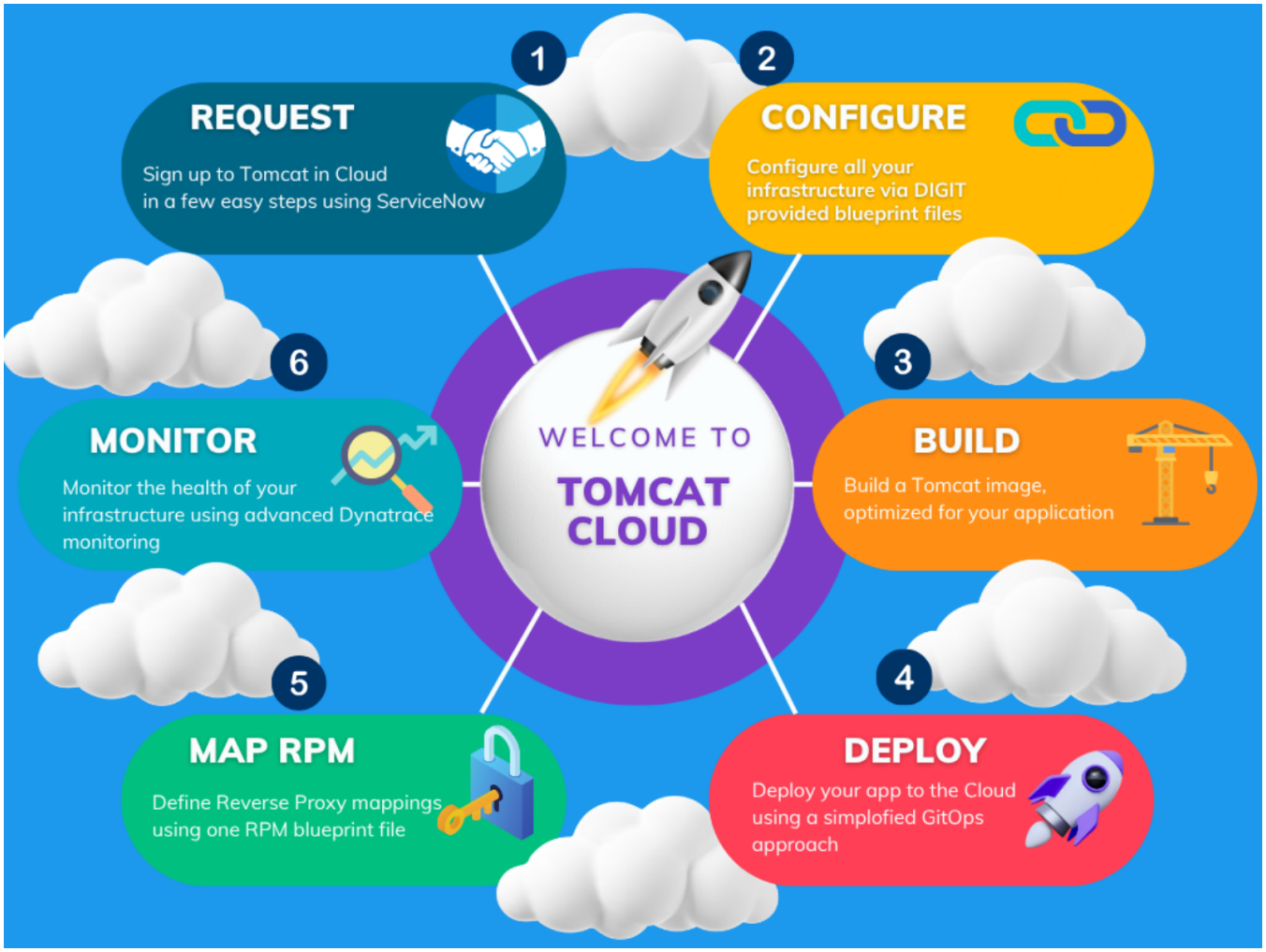
Feel free now to explore additional features or other subjects from the links provided under each section or directly from the homepage here : Tomcat in Cloud: User Guide

Tomcat in Cloud: Jumpstart your experience

- Tomcat in Cloud: Fast Track your journey
- Tomcat in Cloud Service Introduction Videos
- Tomcat in Cloud Sample Application

Tomcat in Cloud: Fast Track your journey

Click on any of the steps in the image below —**Subscribe**, **Configure**, **Build**, **Deploy**, **Map RPM**, and **Monitor**— to go directly to the relevant part of this User Guide.



Tomcat in Cloud Service Introduction Videos

Homepage

Page Topics

Related Links

Service Description

Benefits of Adopting Tomcat Cloud Offering

Architecture Overview

Environment Setups

Secure Hosting Service

Tomcat In Cloud Introduction Video

Please watch the following video for an overview of the Tomcat in Cloud service.

Your browser does not support the HTML5 video element

Tomcat In Cloud Deployment Video

Please watch the following video for an overview of the Tomcat deployment process.

Your browser does not support the HTML5 video element

Tomcat In Cloud RPM Video

Please watch the following video to learn how to define your Reverse Proxy Mapping in Tomcat in Cloud.

Your browser does not support the HTML5 video element

Tomcat in Cloud Sample Application

Homepage

Page Topics

Related Links

Service Description

Benefits of Adopting Tomcat Cloud Offering

Architecture Overview

Environment Setups

Secure Hosting Service

Sample Application

The following application can be used to test your installation

- Javax version for Tomcat 9.0.x : https://citnet.tech.ec.europa.eu/CITnet/stash/projects/DC2CLOSERV/repos/example-tomcat-war/browse?at=refs%2Fheads%2Fjavax
- Jakarta version for Tomcat 10.1.x and TomEE 10.1.x : https://citnet.tech.ec.europa.eu/CITnet/stash/projects/DC2CLOSERV/repos/example-tomcat-war/browse/src/main/resources?at=refs%2Fheads%2Fjakarta

Customize your ecas-config.properties file

Line 2 : edu.yale.its.tp.cas.client.filter.serverName=tccop-int.service-tomcat-design-tst-tc-00.mwcopn000010.digit.k8s.cec.eu.int

YAML file to use for deployment

Here is a minimal example of yaml file you can adapt for your deployment (namely parts in bold)

```
applicationName: "sample-dev-app"
ingressName: "sample-dev"
ingressPath: "/"
customerImageName: "sample-dev-app"
customerImageTag: "build001"
jvmVersion: 25
tomcatVersion: "10.1"
artifactName: "sample-dev.zip"
clientParamsKey: clientParams
tomcat:
  datasources:
    - name: jdbc/sample-db
      url: jdbc:oracle:thin:@myoraclesrv.cc.cec.eu.int:1597/MYORACLEDB_99_D_TAF.cc.cec.eu.int
      username: "oracleuser"
      driverClassName: oracle.jdbc.OracleDriver
      initialSize: 10
      minIdle: 10
      maxIdle: 10
      maxTotal: 10
```

Tomcat in Cloud: Service Description & Architecture

Tomcat in Cloud: Service Description

Homepage

Page Topics

Related Links
Service Introduction Videos
Benefits of Adopting Tomcat Cloud Offering
Architecture Overview
Environment Setups
SNC Ready

Service Description

Tomcat in Cloud is a managed service that enables the deployment of Tomcat applications to both on-premises and public cloud environments using a GitOps approach.

What is Apache Tomcat?

- World's most popular open-source Servlet container
- Simplifies Java web application hosting on open source technologies
- Supports Secure Socket Layer (SSL) to enable secure connection
- Multiple web apps can run on different ports
- Cross platform compatibility

What is Apache TomEE?

Term	Description
TomEE	TomEE stands for Tomcat + Java Enterprise Edition. It is an all-Apache Jakarta EE certified application server that extends Apache Tomcat.

TomEE Architecture	• Apache TomEE is built by starting with a vanilla Apache Tomcat zip file, adding necessary jars, and then zipping it up. • The result is Tomcat with EE features such as ActiveMQ, Apache CXF, OpenWebBeans, and OpenJPA.
Integration	TomEE integrates easily with Tomcat, reducing the effort and time required to add additional libraries.
Portability	TomEE, being a full and official JavaEE implementation, simplifies the process of porting applications from WebLogic.
More Information	Apache TomEE Website

Which Product Do You Need?

- If in doubt, use Apache Tomcat
- If you are migrating from WebLogic or WildFly, use Apache TomEE
- Both products are available under "Tomcat JEE Application" in JASSPR
- Click Here for Features and Versions Comparison

What are the Tomcat components?

- Java Servlet
- Java Server Pages
- Java Expression Language
- Java WebSocket Specification

Tomcat, when bundled with the optional **Tomcat Native component**, can leverage Apache Portable Runtime (APR) to enhance performance and scalability. This is achieved through closer integration with native server implementations that serve as the hosting platform for the service.

Applications developed using the aforementioned open-source implementation are fully compatible and eligible for deployment on Tomcat.

Tomcat in Cloud: Benefits of Adopting Tomcat Cloud Offering

Homepage

Related Links
Service Description
Architecture Overview
Service Introduction Videos
SNC Ready

Tomcat service leverages the **Cloud Deployment Model** (CDM) platform and offers the possibility to host Java web Information Systems on Tomcat containers based infrastructure running on Kubernetes.

Thanks to Continuous Integration and Continuous Deployment (CI/CD) pipelines, infrastructure can be provisioned fast and efficient as code, information systems deployed with just a few clicks and monitoring of infrastructure settings can quickly be configured, updated, managed and shared using one YAML file.

Tomcat (Legacy)	Tomcat in Cloud
Infrastructure updates	
Modify and request updates of infrastructure resources via JASSPR • DataSources • JVM memory	• On-demand provisioning of infrastructure resources, using Infrastructure as Code (IaC) making changes programmatically through YAML files
Deployment	
• Delivery of applications through a portal and API • Request app deployment to DIGIT via JASSPR	• Automatic and autonomous deployment of information systems (no JASSPR requests for this action)
Monitoring	
• Use a GUI to set up and configure Dynatrace infrastructure monitoring	• Dynatrace - Agent based as it's deployed by default • Giving full observability of one or multiple environments • Using infrastructure as code (IaC) - **monaco.yaml** file for further configuration (programmatically) of custom monitoring • Connect to Dynatrace – https://dynatrace.tech.ec.europa.eu/ • Use Splunk to run custom searches/visualize errors and events in order to instantly understand/ fix or predict problems • Connect to Splunk - https://splunk.tech.ec.europa.eu/

Approach	
• Time delay between code configuration and app deployment	• Continuous Integration and Deployment (CI/CD)
Manage secrets	
• Management of secrets within the application	• Default use of HashiCorp vault to manage the secrets

Tomcat in Cloud: Architecture Overview

Homepage

Page Topics

Related Links

Service Description

Benefits of Adopting Tomcat Cloud Offering

Service Introduction Videos

Environment Setups

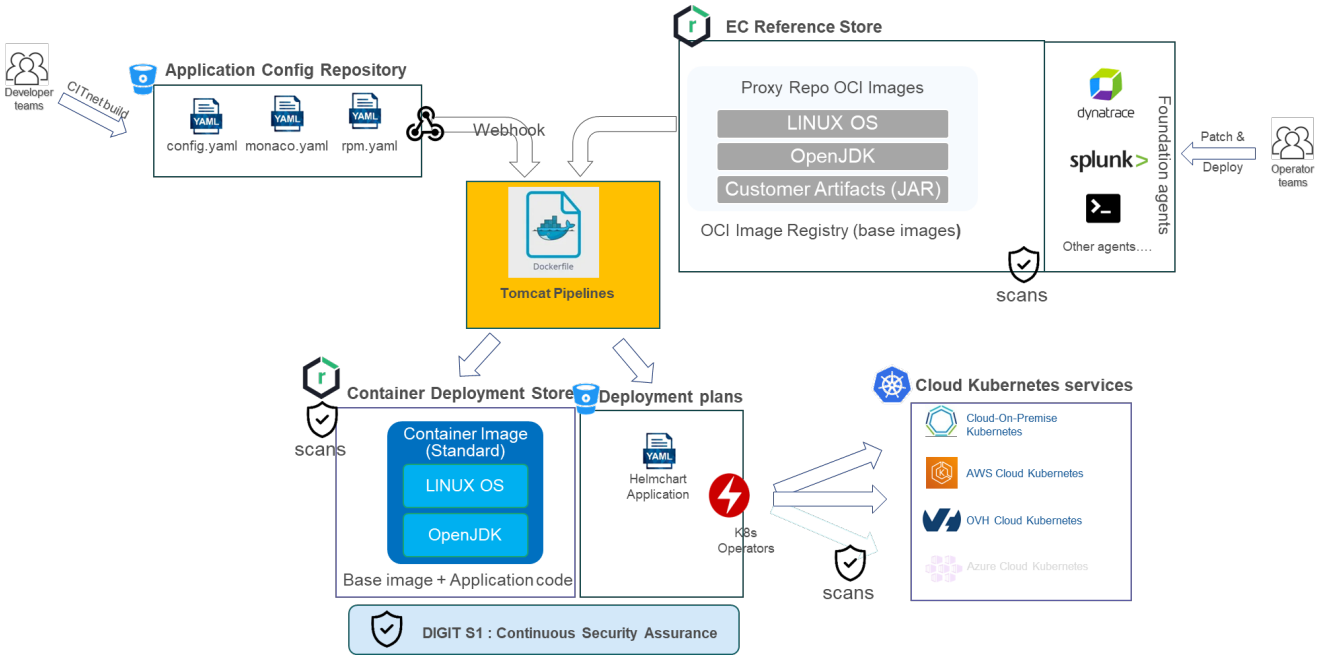
SNC Ready

What Is the Tomcat Architecture Comprised Of?

DIGIT provides **dedicated Tomcat namespaces** within Kubernetes clusters it manages, available for both **Production** and **Non-Production** environments.

Key architectural characteristics:

- Each namespace is reserved exclusively for **Tomcat** and must not be used to host other technologies.
- A namespace currently hosts a **single bundle of Tomcat applications**.
- **Environment hosts are not shared** between namespaces, ensuring isolation between Information Systems and improving both reliability and security.



What are the main dependencies?

Status	Recommended
Linux Operating System	Ubuntu LTS 'Jammy Jellyfish' (or newer)
Tomcat	9.0.x, 10.1.x
Java	OpenJDK LTS 8, 11, 17, 21

What does the Tomcat namespace comprise?

- End-users interact with an application via an RPM within the instance, which forms the front-end of the Information System (IS).
- Java JSP or Java Servlets provide dynamic functionality based on data returned by the database services, configured on the running instance (in the form of database connection pools)
- JMS and Web Services can be also used.
- **NB:** there is no kubectl access to namespace

What will you receive once the service subscription is completed ?

An email will be sent to service users with the following information :

- HashiCorp Vault Namespace
- HashiCorp Vault URL
- Tomcat CoP Nexus Repository

- Tomcat Functional Account Username
- WebHook to use
- Tomcat User Guide (link to this documentation)

Email example:

Dear Customer,

Welcome to the Tomcat service.

Please find below the information needed to use the service.

HashiCorp Vault Namespace: *[to be filled by the pipeline]*

HashiCorp Vault URL: *[to be filled by the pipeline]*

Tomcat CoP Nexus Repository: *[to be filled by the pipeline]*

Tomcat Functional Account Username: emfortccdm

WebHook: (depending on environment one of the following)

https://prod.cdm.aws.cloud.tech.ec.europa.eu/prod/git-webhook

https://nonprod.cdm.aws.cloud.tech.ec.europa.eu/nonprod/git-webhook

Tomcat User Guide: Tomcat in Cloud: User Guide

Tomcat on Cloud: Environment Setups

Homepage

Page Topics

Related Links

Service Description

Benefits of Adopting Tomcat Cloud Offering

Service Introduction Videos

Architecture Overview

SNC Ready

What Is the Recommended Minimum Environment Setup?

The effective minimum setup is one pod (Tomcat instance) per environment.

However, DIGIT recommends the following for optimal reliability and consistency:

- At least two pods in the Production environment to ensure high availability
- At least one non-Production environment (e.g. Test or Acceptance) configured identically to Production for validation and troubleshooting

DIGIT will only create namespaces explicitly requested by the customer as part of the provisioning process.

How are environment maintenance activities executed ?

Maintenance activities are executed in the following order:

- Non-Prod environments
- Prod environments

What size are the environments ?

K8S clusters have by default one worker node which has 2 virtual CPU's and 8 GB of memory.

The above specifications can be adapted (including memory an virtual processor cores) based on customer requirements.

→ Add worker nodes

How to request the configuration of an environment ?

1. Configure the webhook for your Git repository.

2. Use the correct branch (cf name of the Hosting Environment).
3. Execute a Git commit and push the change.
4. You will receive an email notification when the request is completed.

What is Horizontal Scaling ?

- Horizontal scaling is supported to accommodate higher volume of requests.
- Multiple K8S Pods for a Tomcat environment are used with a K8S ingress in order to load-balance the traffic.

What is Auto-scaling?

- Auto-scaling enables the system to respond to peak request volumes during short time frames.
- Additional K8S PODs are added to the instance where the traffic is controlled and distributed by the ingress service.
- This attribute is disabled by default and must be explicitly enabled by the customer. For additional information, refer to the Enable auto-scaling in the **config.yaml** file.

Tomcat on Cloud: SNC Ready

Homepage

Page Topics

Related Links
Service Description
Benefits of Adopting Tomcat Cloud Offering
Service Introduction Videos
Architecture Overview

SNC Ready

Sensitive Non-Classified (SNC) refers to information that is not formally classified as *EU Confidential*, *Secret*, or *Top Secret*, but still requires enhanced protection.

Full SNC compliance requires both the use of SNC Ready services and the implementation of application-level data protection.

Services designated as **SNC Ready** implement a number of security controls, defined by DIGIT.

Refer to the SNC User Guide for information.

Limitations

Service / Component	SNC Availability
Tomcat	Available only on Cloud-on-Premises deployments.
Database	Oracle is the only database service available for hosting SNC data.
Persistence Volumes	On-prem infra related to PVs is SNC ready.

How to find more information?

Please refer to the page in service catalog for additional information.

Tomcat in Cloud: Getting Started

Tomcat in Cloud: Request Subscription to the Service

Homepage

Page Topics

Related Links
What requests can you make? Access & Connection to the Environment

Request Subscription to the Service

You must use the following Service Now form to request subscription to the service: Subscribe to service

Please note it's mandatory to create a new Hosting Environment. You cannot re-use an existing HE.

Hosting Environment Recommendation

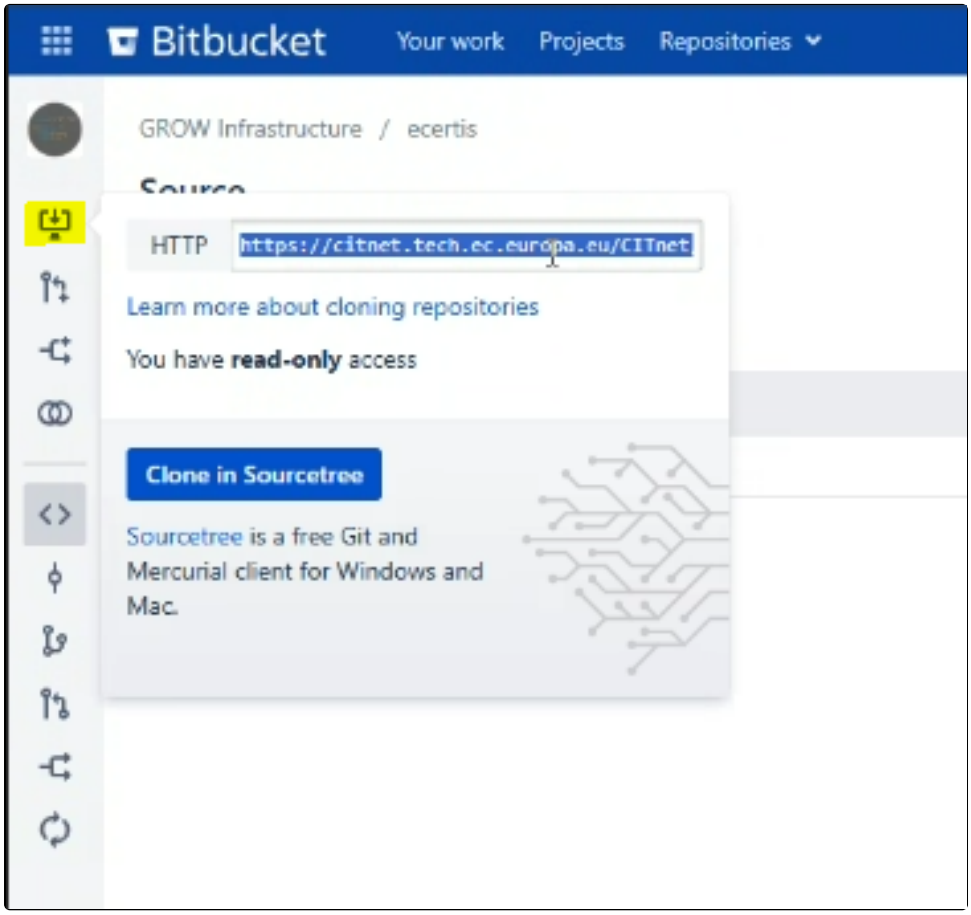
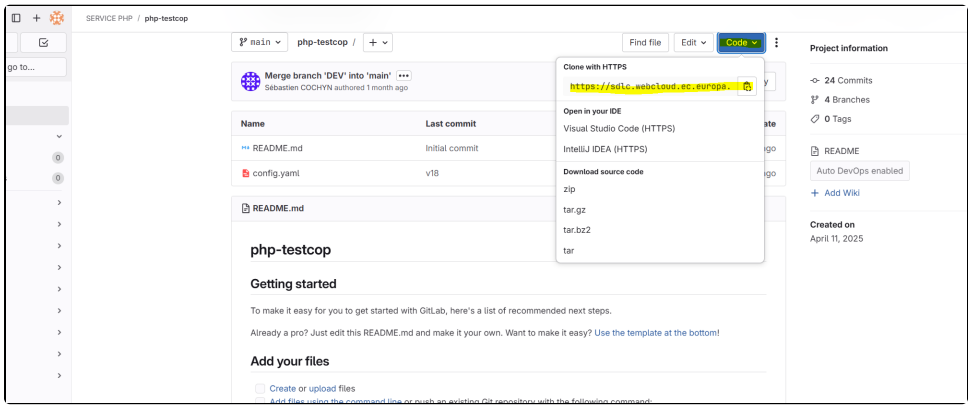
Our recommendation is a Hosting Environment name between 3 to 6 characters.

You must create a Branch in your IaC Git repo with the same name as your HE name.

Disclaimer: Service Levels are not defined yet. So the choice will not have any impact.

Required info:

Item	Description

Git Repository URL	This will be used by your IaC team to place the yaml files. The Git repo must be provisioned in advance by your own team. • **Bitbucket:** The URL will be of following format: https://citnet.tech.ec.europa.eu/CITnet/stash/scm/xxx/xxx**.git** It can be found by going to:  • **Gitlab:** The URL will be of following format: https://sdlc.webcloud.ec.europa.eu/xxx/xxx/xxx/xxx**.git** It can be found by going to: 

LDAP Group – Nexus Repository Access	• Used to access the Nexus repository where you will upload your artifacts. • This Nexus repository is created as part of the service subscription.
LDAP Group – HashiCorp Vault Access	• Grants access to the Vault namespace for storing secrets. • May be the same LDAP group used for Nexus access.
LDAP Group – Splunk Logs Access	Enables read access to your environment logs in Splunk.
LDAP Group – Dynatrace Monitoring Access	Grants access to Dynatrace for monitoring your environment.
Functional Mailbox (FMB)	• Used for all pipeline notifications. • Must not be a distribution list. • Final service details will also be sent here upon subscription completion.

For any general information on JASSPR, please refer to Access to the JASSPR User Interface.

For more information about requests, refer to the JASSPR Tutorial Videos,

Tomcat in Cloud: What requests can you make?

Homepage

Page Topics

Related Links

Request Subscription to the Service

Access & Connection to the Environment

How to request subscription to the Tomcat service in cloud?

Please refer to Tomcat in Cloud: Request Subscription to the Service for additional information.

How to request un-subscription from the Tomcat service in cloud?

You must use the following Service Now form to request un-subscription from the service: Service Now Un-subscription Form

If you have have Reverse Proxy Mappings related to the related Hosting Environment, please use rpm.yaml file to request deletion of these RPMs prior to submitting the un-subscription request.

How to request assistance ?

Through SERVICE NOW:

You can use the following form in Service Now to raise your enquiry to the Tomcat team: Service Now Support Enquiry Form

A ticket will be created to the Tomcat team who will reach back to you.

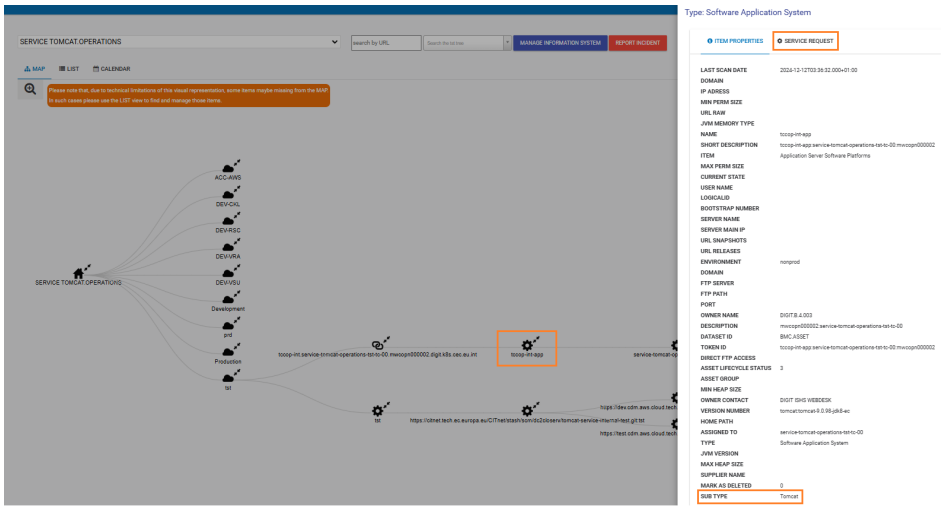
Through JASSPR:

If it is your first time subscribing to the Tomcat service and you cannot find the information you need in this user guide, please feel free to raise an RFI and request for an introduction to the Tomcat service in cloud.

If you have already deployed your environment but require assistance for additional configurations, you can use the Service Request form '**Other Enquiry**' (click on the Tomcat environment linked to your Hosting Environment in JASSPR) to raise your specific questions.

A ticket will be created to the Tomcat team who will reach back to you.

Example:



How to report an incident

Refer to Incident Management for more information.

Enter as much relevant information as necessary. As a minimum, send the following:

- Information System name and Hosting Environment name
- Ingress name and Port
- Error code and message
- **NB**: the more information is given, the quicker we will be able to advise accordingly

How to find more information

Watch one of the following JASSPR tutorial videos to learn more

Tomcat in Cloud: Access & Connection to the Environment

Homepage

Page Topics

Related Links

Request Subscription to the Service

What requests can you make?

What database is this service compatible with ?

- The service is compatible with the DIGIT B4 Oracle database service at COP.
- Accessing a legacy database outside COP is possible using OID to access the infrastructure.

Is the JMS layer supported ?

The following is supported:

- Access from outside of COP to COP resources **must** be done using an RPM, which is the entry point to access Tomcat resources.

Which access rights are required to access a Nexus repository ?

The customer can access the Nexus repository based on LDAP groups. (Depending on LDAP assigned rights)

How are access rights managed ?

Using **LDAP** groups. Once the user is in the required group, he will be able to access the Nexus and Docker repository.

What is the purpose of LDAP groups ?

- To manage Nexus repository access rights for users/groups and assign permissions.
- To access the HashiCorp Vault.
- To distribute access rights among their team(s).

How to find more info about LDAP Groups ?

For more information about LDAP Groups, refer to Manage LDAP Groups

Tomcat in Cloud: Secure Your Environment

Tomcat in Cloud: Security components

Homepage

Page Topics

Related Links

Manage Secrets

Security Troubleshooting

How Application Security Is Provided

Authentication and Authorization – EULogin

- Applications rely on **EULogin** for **Single Sign-On (SSO)**.
- **ECAS** has been superseded by EULogin, but some documentation may still refer to either term.
- **EULogin is the recommended authentication mechanism**; other methods like **LDAP** or **CAS** are **not currently supported**.
- Application developers can integrate **authentication and authorization** using EULogin.
- A dedicated **EULogin client for Tomcat** is available and pre-integrated in the provided image:
  - Includes `eulogin-tomcat-<TOMCAT_VERSION>-<EULOGIN_VERSION>.jar`
  - Includes `eulogin-tomcat-<TOMCAT_VERSION>-<EULOGIN_VERSION>-config.zip`
  - ⚠ **Do not override** these files with your own versions to avoid classpath conflicts.
- **You are responsible** for configuring EULogin correctly within your application.

Certificates

- Certificates are **automatically generated and managed** using **CERTMANA**, a DIGIT-managed service.

Secrets Management

- All sensitive data (e.g., credentials, tokens) must be stored as **secrets in HashiCorp Vault**.
- **Access to Vault and credentials** will be provided during the **onboarding process**.

Which files must be used for the ECAS/EULogin library ?

For version 9.11.3 of the EULogin library, the following files must be used :

EU-Supported Tomcat Version	Library JAR File	Configuration ZIP File

9.0.x	eulogin-tomcat-**8.0**-**9.13.0**.jar	eulogin-tomcat-**8.0**-**9.13.0**-config.zip
10.1.x	eulogin-tomcat-**10.0**-**9.13.0**.jar	eulogin-tomcat-**10.0**-**9.13.0**-config.zip
11.0.x	Not yet available	Not yet available

How to configure the ECAS/EULogin client for Tomcat ?

See explanation here from ECAS/EULogin documentation

https://citnet.tech.ec.europa.eu/CITnet/confluence/spaces/IAM/pages/24641879/EU+Login+Client+for+Apache+Tomcat

Additional documentation can be found here

https://citnet.tech.ec.europa.eu/CITnet/confluence/spaces/IAM/pages/24641912/ECAS+for+Developers

⚠ More details here if needed on how to package the application and provide your own **/META-INF/context.xml**

https://tomcat.apache.org/tomcat-9.0-doc/config/context.html#Defining_a_context

---

Where can I see a demo of the ECAS/EULogin app ?

View a demo of EULogin here.

Additional documentation can be found here : https://citnet.tech.ec.europa.eu/CITnet/confluence/spaces/IAM/pages/24641912/ECAS+for+Developers

---

How to provide additional application security ?

For higher levels of security, a Transport Layer Security layer (TLS) is used to encrypt the connection between the end-user and the RPM.

Tomcat in Cloud: Manage Secrets

Homepage

Page Topics

Related Links

Security Components

Security Troubleshooting

What is the purpose of secrets ?

Secrets can be used to encapsulate:

- usernames and passwords
  - Data sources
  - Mail sessions
  - other
- Sensitive or encoded file (in base64)

What is the HashiCorpVault ?

When the customer wants to use the Tomcat service, a dedicated namespace in HashiCorpVault will be created to store sensitive data

How to access the HashiCorpVault ?

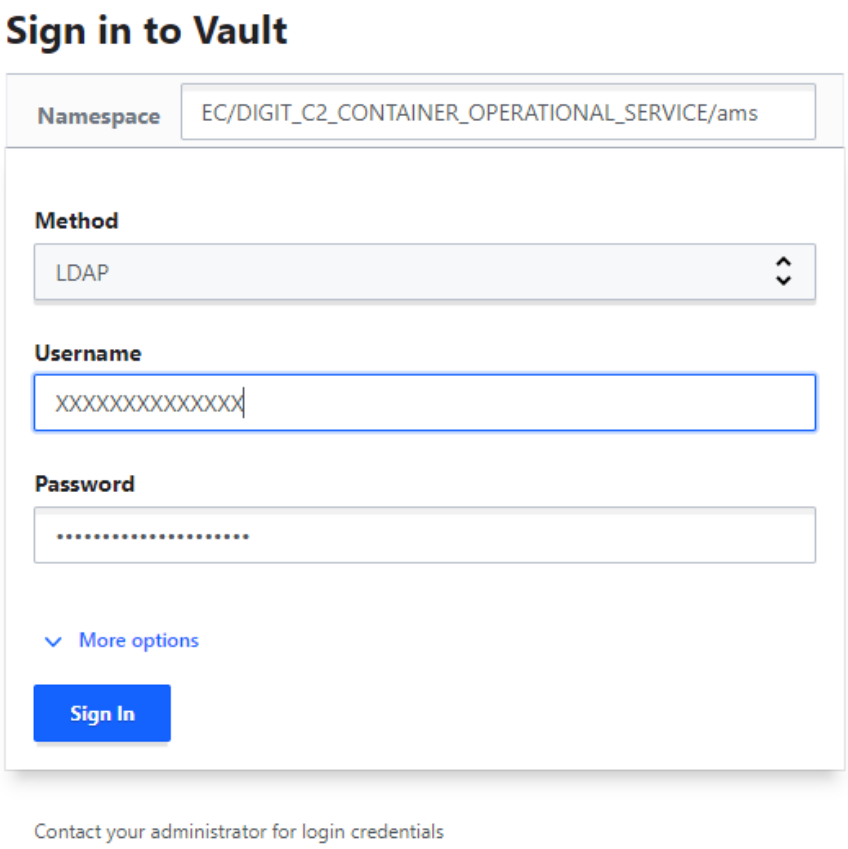
When the customer wants to consume the Tomcat service, DIGIT will create a dedicated namespace in HashiCorpVault that is accessed through PrivX.

To access your namespace:

1. Sign in to the Vault.
   a. Enter the namespace URL provided to you in summary of the service request.
   b. Select LDAP Method.
   c. Enter the relevant Username and password

Example

```
Namespace:EC/DIGIT_C2_CONTAINER_OPERATIONAL_SERVICE/ams
Method:LDAP
Username:<YOUR_LDAP_USERNAME>
Password:<YOUR_LDAP_PASSWORD>
```



> The customer secrets must be located in "**kv_customer**"
>
> - "**kv_customer**" is only accessible in **READ/WRITE** mode to customer
> - DIGIT ISHS Tomcat team doesn't have **WRITE** access
> - DIGIT ISHS Tomcat team doesn't have **READ** access so we cannot see sensitive information
> - DIGIT ISHS Tomcat can only **LIST** (useful to analyze configuration errors for example)

Unique Elements in NameSpace

Elements located in this namespace must be unique and all the secrets for each environments must be located within it.

The customer can name them according to their list of environments.

→ RFC1123 must be satisfied knowing we are adding string after the secret's name (due to naming convention)

Define Client Parameters in the Vault

Tomcat can be parameterized to use some custom client parameters coming from the Vault

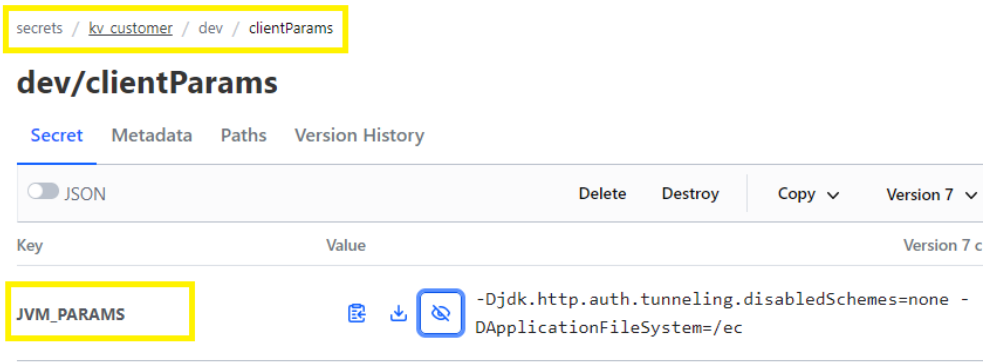
To define those parameters in the vault, the property **clientParamsKey** must be defined in the **config.yaml** file

- All parameters will have to be located under the path defined in **clientParamsKey**
- All secrets stored under this path will be automatically defined as environment variables for all pods
- Parameters can be different for each environment (dev/acc/prod/...). A specific path is used for each environment

For instance, if property is set to "myPath/myClientParams", the following paths will be used

Environment	Path
dev	kv_customer/dev/clientParams/
acc	kv_customer/acc/clientParams/
prod	kv_customer/prod/clientParams/

Here is an example screenshot from the HashiCorp Vault GUI



Once the pods started, the parameters will be available as environment variables.

Standard Directory Structure in HashiCorp Vault:

When adding sensitive values to HashiCorp Vault for a Java application running on Tomcat, it's essential to follow an organization pattern.

This guide describes the steps to configure Vault according to best practices.

1. **Environments:**
   - Sensitive data should be organized by environment. Environment names should be in **lowercase** and may include:
     - `dev` (development)
     - `tst` (testing)
     - `acc` (acceptance)
     - `ldt` (staging)
     - `prd` (production)
     - `my_env_name` (custom client value)
     - ...
   - The environment path must be followed by the type of secret you want to configure.
2. **Types of Secrets:**
   - Depends of the secrets you want to configure, you must to add in your path the following subdirectory:
     - **Subdirectory "clientParams":**
       - If the application has to add secrets to be consumed as environment variables, you must add this directory name in your Vault path (e.g. dev/clientParams)

        - The clientParams stores secrets the application will consume via variables.
        - Each value added here will be accessible through a variable whose name will be the secret key defined in Vault.
    - **Subdirectory "clientFiles":**
        - Allows the storage of files used by the application.
        - If the application needs to use file secrets you must add this directory name in your Vault path (e.g. dev/clientFiles)
        - The values of these files should be base64 encoded with padding.
        - The name (key) assigned to each file will be the name of the file generated for the application to access.
        - The path where these files will be stored is: `/usr/local/tomcat/clientFiles`.

Secrets must be configured using both best practices points mentioned above. Secrets created solely based on point 1 or 2 will not be considered.

Correct configuration example:

- **dev/clientParams/secret1**

Incorrect configuration example (will not be considered):

- **dev/secret1**

Define Datasources and Mailsessions passwords in the Vault

Passwords for datasources must be named after the name of the resource itself.

NB:

- **Environment directories are always lowercase (e.g. 'cop_dev')**
- **Secret names are always uppercase (e.g. 'SECRET_PWD_JDBC_ECERTIS')**

Description	Resource Name	Associated Secret Name in Vault
Generic Rule	**xxx/yyy**	kv_customer/**<env>**/**<clientParamsKey>**/SECRET_PWD_**XXX_YYY**
Example for Datasources in DEV environment with 'clientParams' key	**jdbc/datasource1**	kv_customer/**dev**/clientParams/SECRET_PWD_**JDBC_DATASOURCE1**
Example for Datasources in ACC environment with 'clientParams' key	**jdbc/myDS1**	kv_customer/**acc**/clientParams/SECRET_PWD_**JDBC_MYDS1**
Example for Mailsessions in DEV environment with 'clientParams' key	**mail/mailSession1**	kv_customer/**dev**/clientParams/SECRET_PWD_**MAIL_MAILSESSION1**
Example for Mailsessions in PROD environment with 'clientParams' key	**mail/my-ms**	kv_customer/**prod**/clientParams/SECRET_PWD_**MAIL_MY-MS**
Example for Datasources in COP-DEV environment with 'myParams' key	**jdbc/ecertis**	kv_customer/**cop-dev**/myParams/SECRET_PWD_**JDBC_ECERTIS**

Define Client Files in the Vault

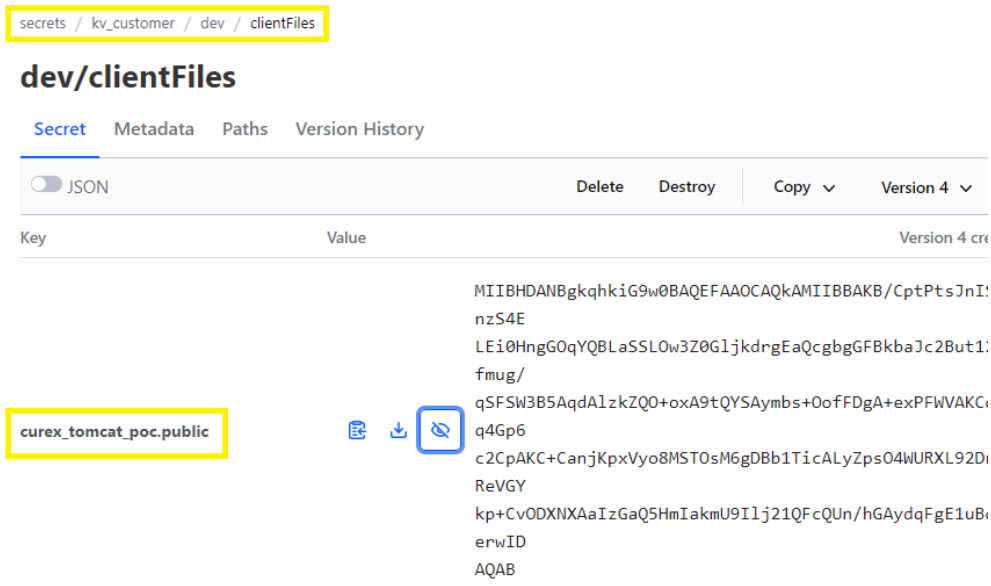
To use Client Files in the vault, the property **clientFilesKey** must be defined in the **config.yaml** file

- All parameters will have to be located under this path
- All files stored under this path will be automatically mounted as files for all pods
- Parameters can be different for each environment (dev/acc/prod/...). A specific path is used for each environment

For instance, if clientFilesKey property is set to "myClientFiles", the following paths will be used

Environment	Path
dev	kv_customer/**dev**/myClientFiles/
acc	kv_customer/**acc**/myClientFiles/
nonprod	kv_customer/**nonprod**/myClientFiles/
prod	kv_customer/**prod**/myClientFiles/
my_env_name	kv_customer/**my_env_name**/myClientFiles/

Here is an example screenshot from the HashiCorp Vault GUI



NB: Once the pods started, the files will be available in directory /usr/local/tomcat/<env>/clientFiles/

The value of the element **must be encoded with padding** - this is really important

The Tomcat secrets are synchronized with HashiCorpVault secrets.

- If you change a secret in HashiCorpVault for a secret defined in Tomcat model files (new password in case of DB migration for example), the **Tomcat instance must be at least restarted** (or a new image generated).
- This is required to "refresh" the value of secret from the Tomcat instance point of view.

Are secrets synchronized ?

- The HashiCorpVault secrets are synchronized with Tomcat secrets on Kubernetes cluster.
- If you change a secret in HashiCorpVault for a secret defined in a Tomcat model file (new password in case of DB migration for example), the **Tomcat instance must be at least restarted** (or a new image generated).

This is required to "refresh" the value of secret from the Tomcat instance point of view.

Tomcat in Cloud: Security Troubleshooting

Homepage

Page Topics

Related Links

Manage Secrets

Security Components

Q. I am getting the error "*Invalid ECAS Client Signature: corrupted JAR*" when trying to run a Tomcat app with an embedded server.

The exception is raised by the ECAS client itself because it cannot locate its class within the **s**`ignerVerifierClassLoader`**.**
I am unable to place the client in the Tomcat `lib` folder, as it is already embedded within the Tomcat distribution.

What am I missing?

A. The EU Login client is a signed JAR, therefore its content cannot be repackaged inside a fat JAR without removing the digital signatures.

- You cannot package the EU Login client together with your classes and dependencies as a **fat** JAR because the EU Login client is a signed JAR and this would remove the signatures.
- Instead, you need to preserve the JAR file formats of signed JARs in your Tomcat bundle.
- We have 2 PoCs with Tomcat at ECAS, Tomcat and Security

If you are using Maven, as suggested here, you can use the following code from *pom.xml.*

Maven snippet with requiresUnpack

```
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
            <configuration>
                <requiresUnpack>
                    <dependency>
                        <groupId>eu.europa.ec.digit.iam.eulogin.client</groupId>
                        <artifactId>eulogin-tomcat-8.0</artifactId>
                    </dependency>
                </requiresUnpack>
            </configuration>
        </plugin>
    </plugins>
</build>
```

See the following page for reference.

requiresUnpack

> *A list of the libraries that must be unpacked from fat jars in order to run.*
>
> *Specify each library as a <dependency> with a <groupId> and a <artifactId> and they will be unpacked at runtime.*

Tomcat in Cloud: Storage

- Tomcat in Cloud: S3 Storage Documentation
- Tomcat in Cloud: Database Storage
- Tomcat in Cloud: Persistent and Ephemeral Volumes

Tomcat in Cloud: S3 Storage Documentation

Homepage

Page Topics

Related topics

Persistence Volumes

Database Storage

If your application data needs to be directly accessible by third-party applications, you should store it either in Oracle databases or in S3 buckets.
Please note that **NAS filesystems are not supported on Tomcat containers**.

Please refer to COP S3 Storage User Guide for more information on this service.

The S3 service should be used **only** with instances provisioned on Cloud-on-Premise.

This COP S3 service is only offered through the COP API controller and CMP platform. For this reason the customer must request first the **creation of a Business Group (Project)** and then subsequently request to have the **S3 account service entitled to that Business Group**.

How to create a Business Group

This can be currently done using a non-standard DCTSC as shown in the following code:

```
Please create the following DCTSC Service Request ticket for the DATA CENTRE CONTROL TOWER team:

DG name:
IS name:
Business Group:
Description:
LDAP Group Name:
Environments: NONPROD/PROD
Service(s): S3 ACCOUNT

Incident type : REQUEST FOR SERVICE
SCIM : SERVICE / DCTSC / CMP / CREATE BUSINESS GROUP
```

The Project Name will be created as **Business Group** + "_" + **Environments**

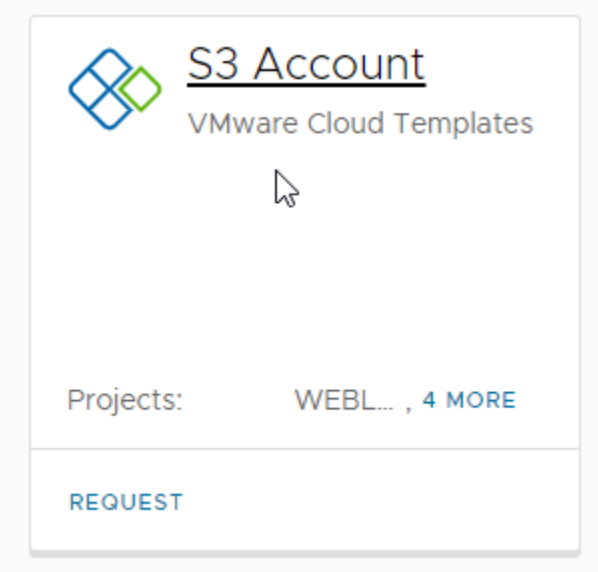
Create an S3 account

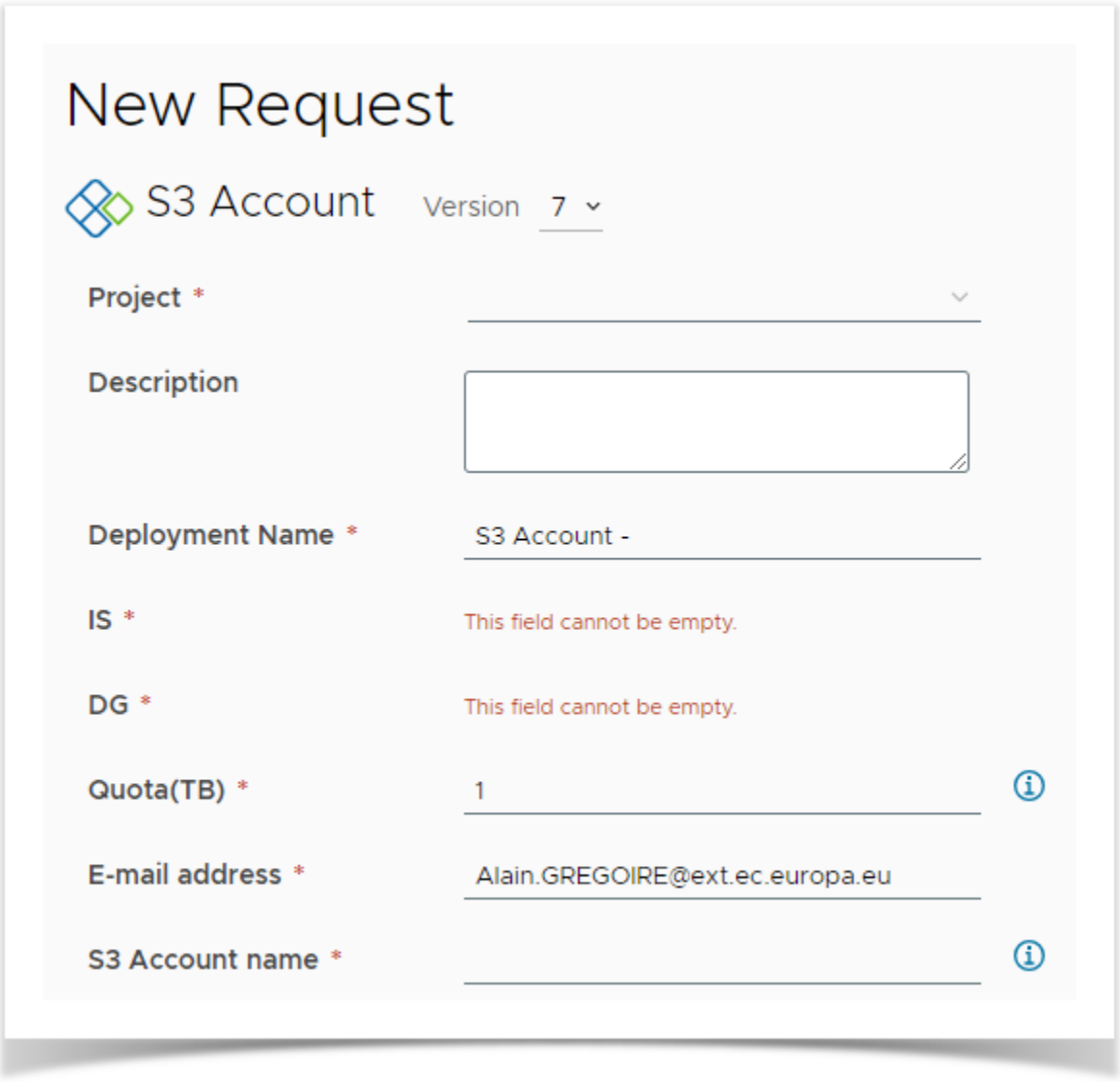
There are 2 possibilities to obtain an S3 account :

- Ask for it using CMP
- Send HTTP request

CMP

This access is possible from PrivX:

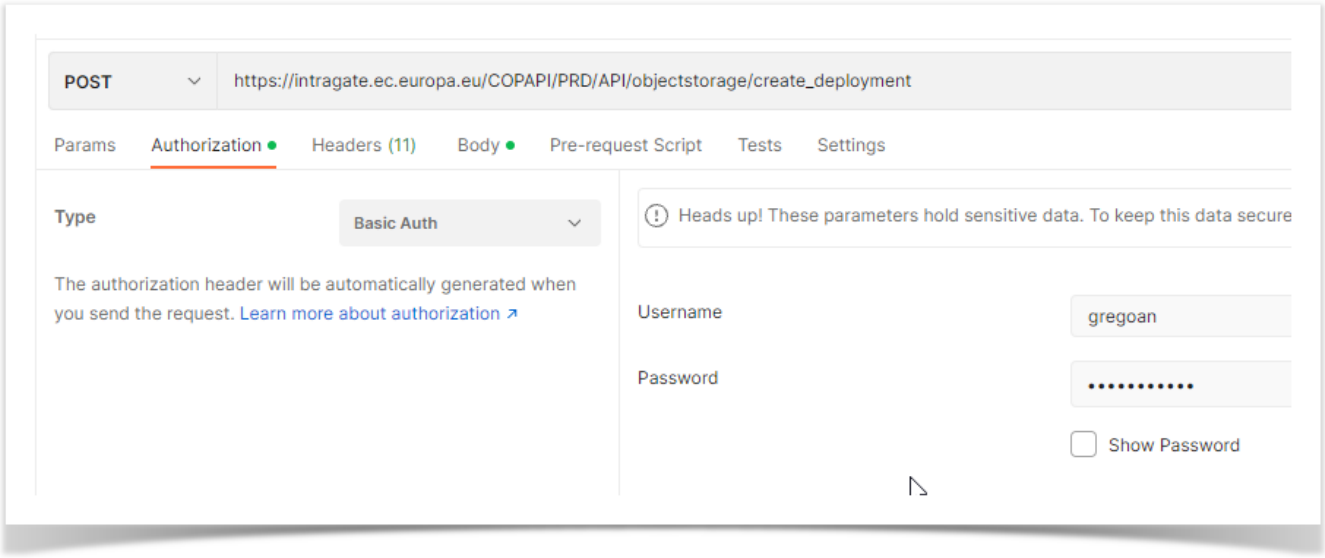




HTTP request

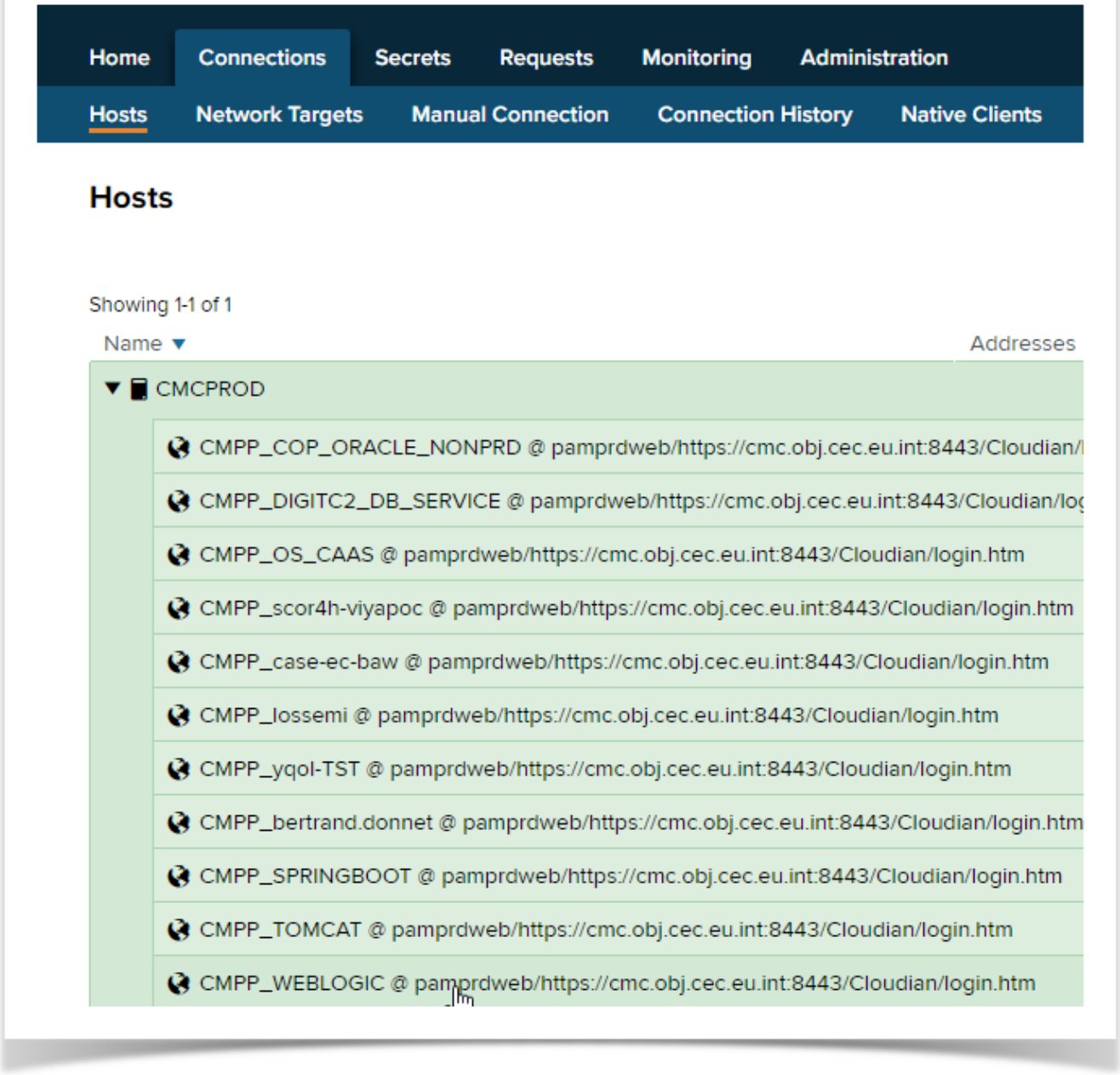
Another possibility is to send the following **POST** request to https://intragate.ec.europa.eu/COPAPI/PRD/API/objectstorage/create_deployment

```
{
 "ProjectName": "TO_BE_FILLED",
 "deploymentName": "TO_BE_FILLED",
 "quotaTB": "1",
 "s3account": "TO_BE_FILLED",
 "email_address": "VALID_EC_EMAIL_ADDRESS"
}
```



The credentials concerning S3 access will be sent to your email address.

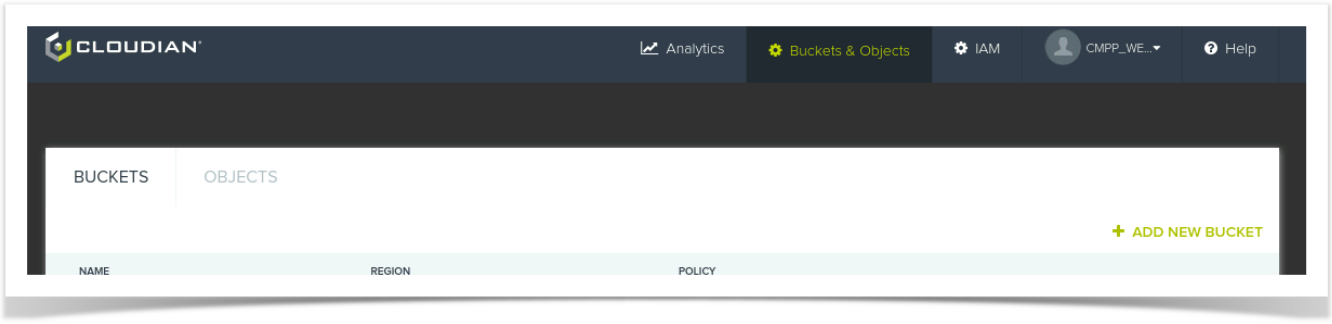
When the S3 account is created, you can access the S3 bucket from **PrivX**.

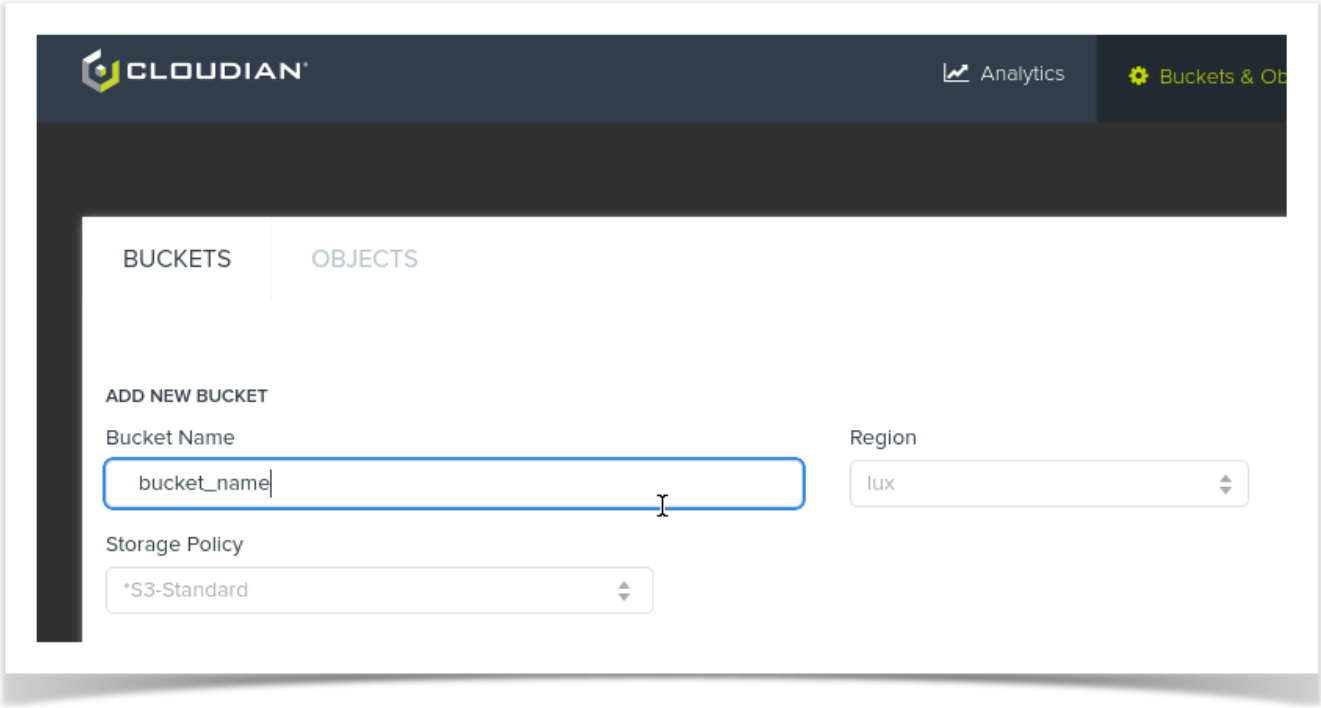

Home
Connections
Secrets
Requests
Monitoring
Administration
Hosts
Network Targets
Manual Connection
Connection History
Native Clients
Hosts
Showing 1-1 of 1
Name
Addresses
CMCPROD
CMPP_COP_ORACLE_NONPRD @ pamprdweb/https://cmc.obj.cec.eu.int:8443/Cloudian/
CMPP_DIGITC2_DB_SERVICE @ pamprdweb/https://cmc.obj.cec.eu.int:8443/Cloudian/log
CMPP_OS_CAAS @ pamprdweb/https://cmc.obj.cec.eu.int:8443/Cloudian/login.htm
CMPP_scor4h-viyapoc @ pamprdweb/https://cmc.obj.cec.eu.int:8443/Cloudian/login.htm
CMPP_case-ec-baw @ pamprdweb/https://cmc.obj.cec.eu.int:8443/Cloudian/login.htm
CMPP_lossemi @ pamprdweb/https://cmc.obj.cec.eu.int:8443/Cloudian/login.htm
CMPP_yqol-TST @ pamprdweb/https://cmc.obj.cec.eu.int:8443/Cloudian/login.htm
CMPP_bertrand.donnet @ pamprdweb/https://cmc.obj.cec.eu.int:8443/Cloudian/login.htm
CMPP_SPRINGBOOT @ pamprdweb/https://cmc.obj.cec.eu.int:8443/Cloudian/login.htm
CMPP_TOMCAT @ pamprdweb/https://cmc.obj.cec.eu.int:8443/Cloudian/login.htm
CMPP_WEBLOGIC @ pamprdweb/https://cmc.obj.cec.eu.int:8443/Cloudian/login.htm

- The GroupName is **CMPP_** + the **ProjectName** (part of RFC request)
- The UserID is **CMPP_** + **S3_ACCOUNT** (part of CMP/HTTP request)

How to create the S3 Bucket

In **Cloudian**, you can create the S3 bucket:





Example using the Tomcat service

The bucket can be accessed via Cloudian GUI with PAM PrivX , AWS CLI or the AWS Java SDK

Using Cloudian GUI with PAM Privx

Once on the login screen, enter the provided credentials.

The list of created buckets is displayed.

The content of each bucket can be seen by clicking on the bucket name.

Using the AWS CLI

Access has been tested and validated from tallinn to tc-bucket01:

```
unzip awscli-exe-linux-x86_64.zip
cd ./aws
./install --bin-dir ~/bin --install-dir ~/aws-cli
cd ~/aws-cli

echo '-----BEGIN CERTIFICATE-----

MIIFRTCCAy2gAwIBAgIBATANBgkqhkiG9w0BAQsFADAqMSgwJgYDVQQDDB9FdXJv
cGVhbiBDb21taXNzaW9uIFJvb3QgQ0EgLSAyMB4XDTE2MTAwNDExMjExN1oXDTQ2
MTAwNDExMjExN1owKjEoMCYGA1UEAwwfRXVyb3BlYW4gQ29tbWlzc2lvbiBSb290
IENBIC0gMjCCAiIwDQYJKoZIhvcNAQEBBQADggIPADCCAgoCggIBAL4JYI9CBISZ
uBOBknpxCRX306sYm4tQPm5H2l5f4fDESYbthbv8FEOFUPu/uh/L5FuCsPgjDkHp
6lQqfWV0QG8550pLWI82B5EgE/tN0F4Iwq5OVzOwK+qkHpcXLwxZATNYmfgTGAb2
mcvVZ8ZkhL4cm6fWqjGzpX9av4R1uqRMKxm/0xuUXx37034g1/fMvzZ3V4rLGOwE
GluagitBcZhpXXnAFZAu6QF07dokW7vgOOm392TlVgJrv94qN73gMfl/CGQd8Sb3
t75HhYQ9kGyXEkFzOyPvwBlvV6hCOvElU+2u/HPYYz5lrC0u3MHPos16XF5/XJ7M
/H9DDtA5mv3B9xO+/67fdaMyXaUJzoiE3decIUgLQC0Qh5hNs1kdkBnufmYRvpe9
sUHWCsk39cNwVX+vp8EKDjtkiQbFuYIqvFckBbm7AcJlUt4jj6SJHVhECM4SCVd+
oUtsaTStKUrrVjvuXgzN65qfzafesiaimXYWD60gRp7OoCiN9QwkCiRD1Iqs9irE
E2Drlz15suuTb2+esrKciiIqyENUeYQolLhPvfdsZhdrtlFsDxM+/IPl7xz0V66k
A97FtQuiVFOrZaj1YdjSpSfUlscpUfJHMebdd36zQ85oFyGERx7VHpjwE8bw4w4n
DfnCKsz4xe7gZZX3CaKBCng6F4KiVXZHAgMBAAGjdjB0MA8GA1UdEwEB/wQFMAMB
Af8wHQYDVR0OBBYEFC+klbkQluW624UvF9NUjFzbrNNXMB8GA1UdIwQYMBaAFC+k
lbkQluW624UvF9NUjFzbrNNXMBEGA1UdIAQKMAgwBgYEVR0gADAOBgNVHQ8BAf8E
BAMCAQYwDQYJKoZIhvcNAQELBQADggIBAFPypMzasOt82j0geV8hJOopri91Jc1d
/fpc6mlubXb8E/scI9qqWQVUMlqiJkCyZl1TVis0bCPFvSlDl/hhwS5vnC0rBmCT
XXEEsQmsEasw/IR4e7bNAF+l/pPmggh7u+Y00kjYt1XweA2Of/+xf4nAk3HiX02I
ToHmY/Y4nlLP3bt1oac01Zv7sHPogmQrFFAvuoC4k+e6vJP0XveSp/vBpfKrdCNj
nViZ3J8gUzrRowi10U812/A5NtZFvKOYXPTFi4vznYMmZsfgejUab5f/j+ycgrFl
svw8vhYwWsJhWM/oPVNGnfYusa/8aovhwOiCe6lnn3o2jlASIPy6ReSzqZpqImKm
UGdARWSFCJw4NX1m2dg4GnMjSlWFv5fEnyF0wZlqniarr2TsRek85N6vIaklzc0k
A5gNgWLTxXMbr8rNta1RtXcN+SH8QgQ8CKgjbq4PSD/WPoOxRcZemGTXBdgxhTjZ
JgwaQU4L810bScOcQ9cI1QB0/Iq+7fQOg9xIl3mvSoEhnP36Dr3uoi+yem1UhnjU
9DHE0uKYpHjlHXP6LHvjfQZyS3ba3S0/nYsVf24b4UEja3PehnHhdzyJx/cHRJpN
T5ibC5pZWL61QgOrDHuSBQnEQUMmYwNoqS+HQvu532NjlSfG6ffmDuEkGuBMM1jY
gTqhM3BGMuf+
-----END CERTIFICATE-----' > ca-bundle.pem

export AWS_ACCESS_KEY_ID=********
export AWS_SECRET_ACCESS_KEY=**********
export AWS_CA_BUNDLE="ca-bundle.pem"
export AWS_DEFAULT_REGION=lux
aws --endpoint-url https://s3-lux.obj.cec.eu.int s3 ls
2023-09-22 14:32:21 tc-bucket01
```

Using the AWS Java SDK

Once the required dependencies added in pom.xml :

```
...
<dependencies>

  <dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-java-sdk-s3</artifactId>
    <version>1.12.556</version>
  </dependency>

  <dependency>
    <groupId>javax.xml.bind</groupId>
    <artifactId>jaxb-api</artifactId>
    <version>2.3.1</version>
  </dependency>

</dependencies>
...
```

The following code can be used to test the connectivity to the bucket.

NB: In contrast with AWS client, Java code does NOT require to provide the CA Bundle.

Tomcat in Cloud: Database Storage

Homepage

The service is compatible with the following database storage:

- Oracle service : Oracle Database Cloud On-Premise (COP) User Guide
- PostgreSQL service : PostgreSQL: Cloud On-Premise (COP) User Guide
- MySQL service : MySQL: User Guide
- SQL Server service : SQL Server: User Guide

Flows must be opened for COP Databases following this link :

- COP Database: Opening Flows

Hybrid cloud architecture is **not allowed**.

The Tomcat environment and the database must be hosted with the same cloud provider, i.e. either both components are hosted on premise or both are hosted in public cloud (AWS).

Related topics

Persistence Volumes

S3 Storage

Tomcat in Cloud: Persistent and Ephemeral Volumes

Homepage

Page Topics

Related topics

Database Storage

S3 Storage

How to create Persistent Volumes in config.yaml file ?

The **persistentStorages** elements must be defined under the **tomcat** section of the YAML file.

```
...
tomcat:
  ...
  persistentStorages:

    - mountPath: /gs-fs1
      size: 1
      accessMode: ReadWriteMany
      storageClass: kube-repl-0d

    - mountPath: /gs-fs2
      size: 2
      accessMode: ReadWriteMany
      storageClass: kube-repl-0d
```

- You cannot request more than **3** elements.
- Maximum size is 1024 Gb
- All the volumes are accessible by each Tomcat server.

Only **Iron** storage can be requested for **PROD and NONPROD** environments for data that doesn't require backup !

All other storage will be rejected

For more information on storage, please read the Persistent Storage documentation here :

https://citnet.tech.ec.europa.eu/CITnet/confluence/spaces/COSMOS/pages/1315879774/PKSv2+Persistent+Storage

Details

Attribute	Description

<table>
<tr><td>mountPath</td><td>

This is a mandatory parameter
Represents the name of the persistent storage
Must match the following REGEX : ^[a-z]*[-a-z0-9]*[a-z0-9]$
The mount point cannot exceed 10 characters
The mount point is the name of persistent storage so /gs-fs1 in this example

</td></tr>
<tr><td>size</td><td>

This is a mandatory parameter
The maximum allowed size is 1024 (1024 GB or 1 TB)

</td></tr>
<tr><td>storageClass</td><td>

This is a mandatory parameter
The rules are :

<table>
<tr><th>Cloud Provider</th><th>Storage Class</th><th>Environments</th><th>Backup</th></tr>
<tr><td rowspan="4">CloudOnPrem</td><td>kube-repl-0d</td><td>Nonprod & Prod</td><td>No backup offered !</td></tr>
<tr><td>kube-repl-7d</td><td>Nonprod & Prod</td><td>Yes</td></tr>
<tr><td>kube-repl-35d</td><td>Nonprod & Prod</td><td>Yes</td></tr>
<tr><td>kube-repl-1y</td><td>Prod</td><td>Yes</td></tr>
<tr><td>AWS</td><td>0d</td><td>Nonprod & Prod</td><td>No backup offered !</td></tr>
</table>
</td></tr>
<tr><td>accessMode</td><td>

This is a mandatory parameter
The rules are :

<table>
<tr><th>Access Mode</th><th>Description</th></tr>
<tr><td>ReadWriteOnce</td><td>The volume can be mounted as read-write by a single node.

ReadWriteOnce access mode still can allow multiple pods to access (read from or write to) that volume when the pods are running on the same node.

For single pod access, please see ReadWriteOncePod.</td></tr>
<tr><td>ReadOnlyMany</td><td>The volume can be mounted read-only by many nodes</td></tr>
<tr><td>ReadWriteMany</td><td>The volume can be mounted as read-write by many nodes</td></tr>
<tr><td>ReadWriteOncePod</td><td>The volume can be mounted as read-write by a single pod</td></tr>
</table>
</td></tr>
</table>

- You cannot change the **storage_class** when volume is created
- You cannot decrease the **size** of storage when volume is created

How to delete Persistent Volumes ?

A Kyverno Policy is in place to prevent the deletion of Persistent Volume Claims (PVCs).

If you need to delete PVC, you will have to contact **DIGIT ISHS TOMCAT** by opening a ServiceNow ticket containing the following information:

- The DG
- The Information System
- The Hosting Environment
- The name(s) of the PVC(s) you wish to delete

Upon receiving your request, our CoE will contact you to remove the concerned PV(s) entries from your config.yaml file. Our CoE will then proceed with the decommissioning of PV(s).

- If you perform a new commit before our CoE has confirms back that PV(s) is deleted, you will receive a notification error by email.

Once the PV is deleted our CoE will ask you to commit your config.yaml file. The deployment would succeed without any errors.

How to create Ephemeral Volumes in config.yaml file ?

The **ephemeralStorages** elements must be defined under the **tomcat** section of the YAML file.

```
...
tomcat:
  ...
  ephemeralStorages:

    - mountPath: /tempDir
      size: 1

    - mountPath: /tempDirForMyApplication
      size: 5
```

- An Ephemeral storage is dedicated to a pod and is not backed up
- Maximum size is 5 gb
- Ephemeral storages cannot be accessed by other pods

Tomcat in Cloud: Image Build

Tomcat in Cloud: Base Image

Homepage

Page Topics

Related Links

Build Customer Image

Nexus Repository Structure

Version & Patches

What is a base image ?

- The base image is the Docker image that contains the following:
  - Ubuntu OS
  - Eclipse Temurin JDK
  - OpenSSL
  - OpenLDAP
  - JDBC drivers for Oracle
  - JDBC drivers for MySQL
  - JDBC drivers for PostgreSQL
  - JDBC drivers for Microsoft SQL Server
  - EULogin client
  - Reload4j (a binary compatible, drop-in replacement for log4j version 1.2.17)
- A GitOps approach is used to build this image. When a customer executes a push operation in the IaC Git repository, a pipeline is launched and uses docker to generate the target image.
- After a few minutes, the custom image is pushed to the Nexus Digit catalog repository

Why is a Base Image Needed?

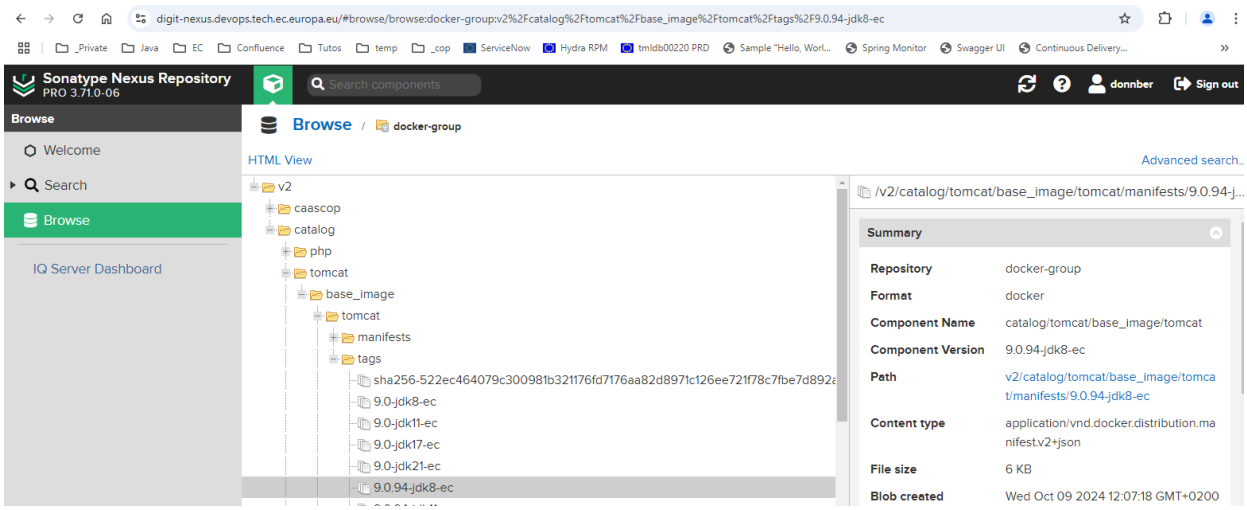
A **base image** provides a **common foundational layer** that includes essential runtime components and environment settings. On top of this base layer, additional layers containing the **client's application code and specific configurations** are added.

This approach ensures consistency, simplifies maintenance, and optimizes image reusability across different deployments.

Where are base images stored ?

Base images can be found in DIGIT Store reference catalog accessible here :

- https://digit-nexus.devops.tech.ec.europa.eu/#browse/browse:docker-group:v2%2Fcatalog%2Ftomcat%2Fbase_image%2Ftomcat
- https://digit-nexus.devops.tech.ec.europa.eu/#browse/browse:docker-group:v2%2Fcatalog%2Ftomcat%2Fbase_image%2Ftomee

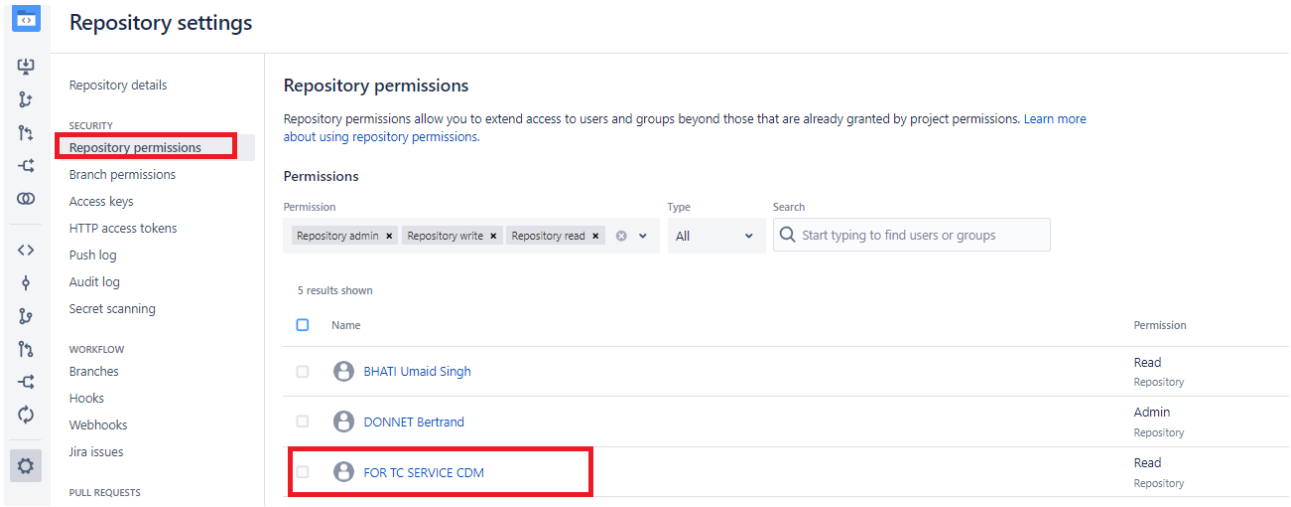


How is a Tomcat image structured ?

- The structure of a Tomcat image is composed of the following:
    - A Java virtual machine
    - A Tomcat application provided as an artifact by the customer
- Refer to the section **config.yaml** for more information.

How is Git used in relation to Tomcat ?

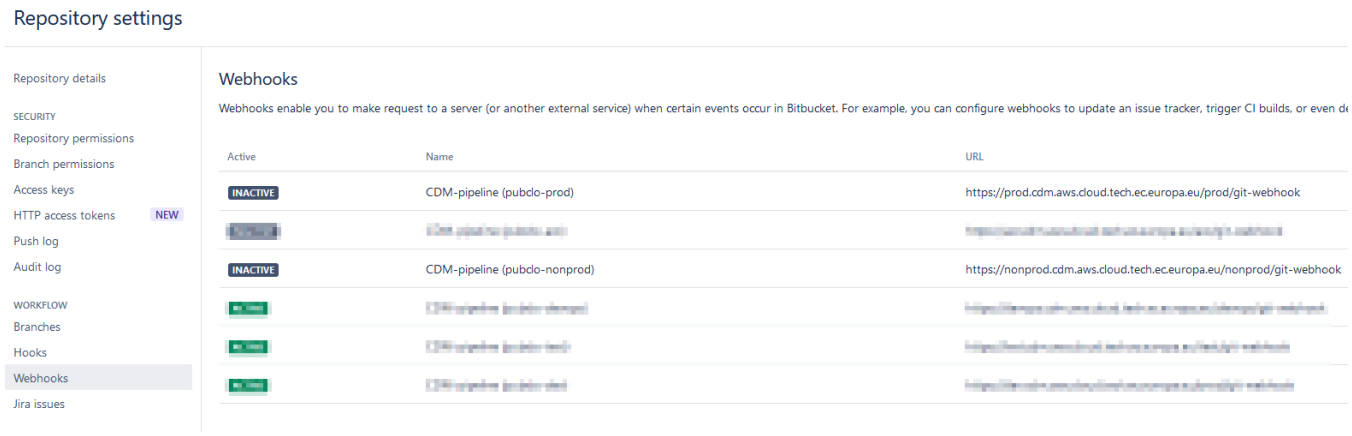
- The Git push operation triggers the pipeline, launching the creation of the Tomcat image
- The account **emfortccdm** must be granted by the customer to access the Git repository with **READ** access



Webhook Location

ENVIRONMENT (Tomcat Namespace)	WEBHOOK
NONPROD	https://**nonprod**.cdm.aws.cloud.tech.ec.europa.eu/**nonprod**/git-webhook
PROD	https://**prod**.cdm.aws.cloud.tech.ec.europa.eu/**prod**/git-webhook

[ENV_NAME]	https://**[ENV_NAME]**.cdm.aws.cloud.tech.ec.europa.eu/**[ENV_NAME]**/git-webhook



The compilation of the embedded application is the responsibility of the customer.

How can the customer application files directory be structured ?

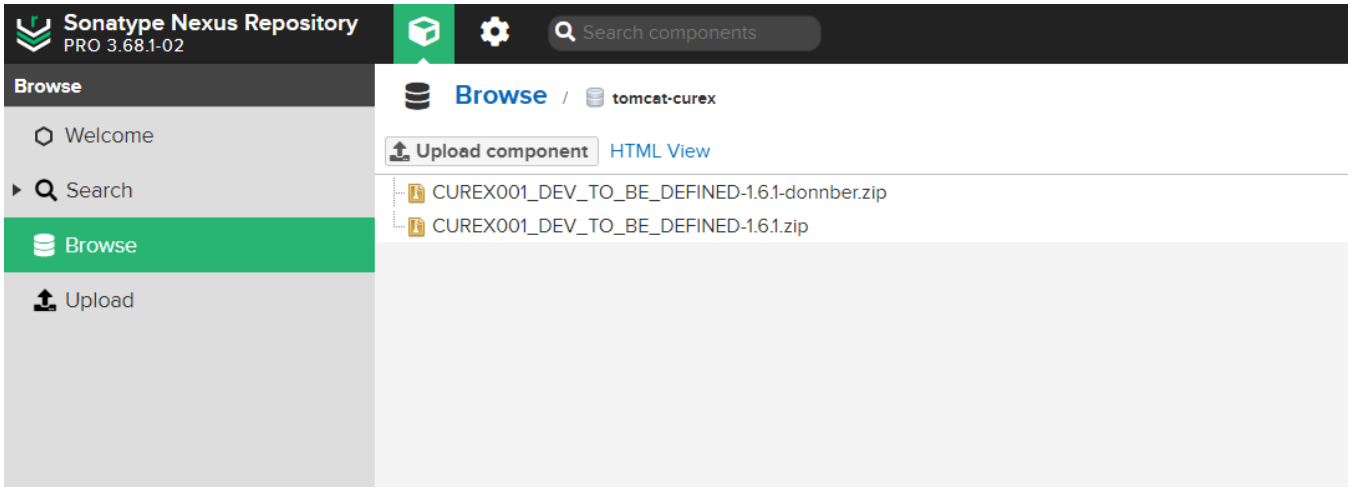
- The artifact must be provided as a ZIP file: tc-demo-0.0.5-SNAPSHOT.zip
- The zip file must contain the following:

Zip file contents

Location	Description	Remarks
<root of the ZIP file>	the Tomcat applications as WAR files	This is a classic WAR. All dependencies must be embedded in the WAR itself

View your zip file location

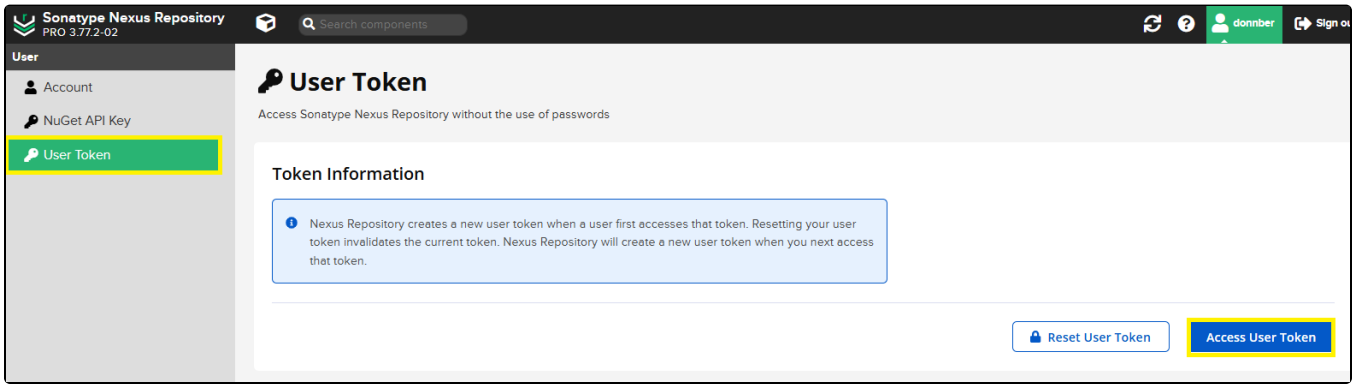
Once uploaded in Nexus, you can view it here:

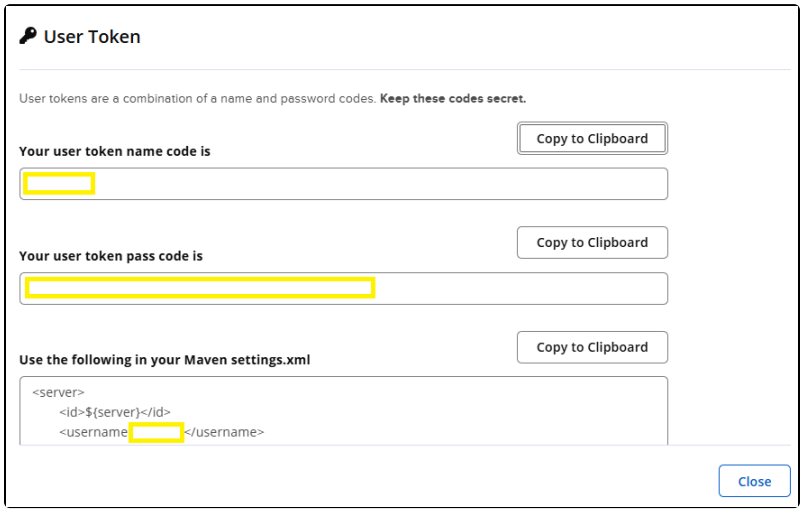


How to pull the image from docker ?

First, login must be performed with your user name and token.

> ⚠ Use your user token name & user token pass from https://digit-nexus.devops.tech.ec.europa.eu/#user/usertoken





Proceed with login :

`docker login` **`digit-docker`**`.devops.tech.ec.europa.eu`

Pull can be performed, for instance :

`docker pull` **`digit-docker`**`.devops.tech.ec.europa.eu/catalog/tomcat/base_image/`**`tomcat:10.1.40-jdk17-ec`**

Tomcat in Cloud: Build Customer Image

Homepage

Page Topics

Related Links

Base Image

Nexus Repository Structure

Version & Patches

Create a Docker image using **config.yaml** file

- Tomcat offers the capability to create a custom docker image with a YAML file.
- The **config.yaml** file defines this build with the buildEnabled property set to true
  - buildEnabled: true
- This file must be located in the root location of the customer Git repository, for which one webhook must be configured.
- **Case is important.**
  - If the filename case is not correct, the file will be ignored.
  - Inside the file, if the properties case is not correct, the property will be ignored and default value will be applied (if any)

What meta data is required ?

Attribute	Description	Example
buildEnabled	Indicates that the build must be forced with CDM (and not GitLab)	true
applicationName	Unique name of the application **(This must be in slug-format)**	ams-api-app

customerImageTag	• Represents the tag of the built image.
	  ○ This image should not be updated or overwritten
	  ○ Only the previous 10 images are saved.
	
	In Nexus, this tag will be used to reference the generated image (**applicationName:customerImageTag**)
		testimg85

tomcatVersion	The version of Tomcat that will be used to launch the artifact jar in the image	'9.0', '10.1', '11.0' are supported for the moment
tomeeVersion	The version of TomEE that will be used to launch the artifact jar in the image	'10.0' is supported for the moment
jvmVersion	The version of Java JDK that will be used to launch the artifact jar in the image	'8', '11', '17', '21' are supported for the moment

artifactName	• Name of the artifact to launch at startup of the image • It must be uploaded in Nexus first in ZIP format **Avoid spaces and exotic characters in the naming**	tc-demo-0.0.5-SNAPSHOT.zip

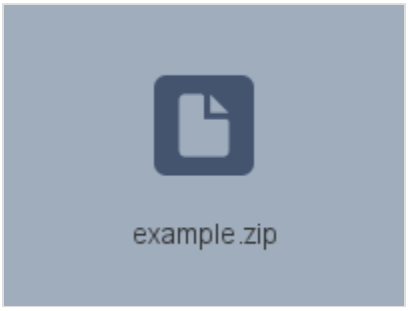
⚠ To convert a string to **slug-format**, please perform the following steps:

1. Convert to lower case
2. Trim any leading or trailing spaces
3. Remove any accents from characters (e.g. á, â, â, ã, ä, å, ç)
4. Replace any other special characters with spaces
5. Replace multiple spaces or dashes (hyphens) with a single dash

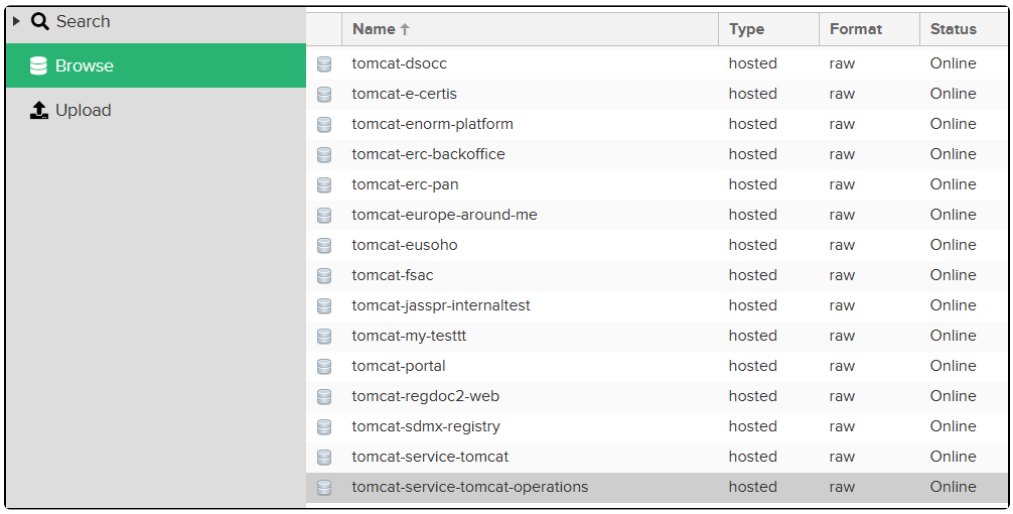
Artefacts

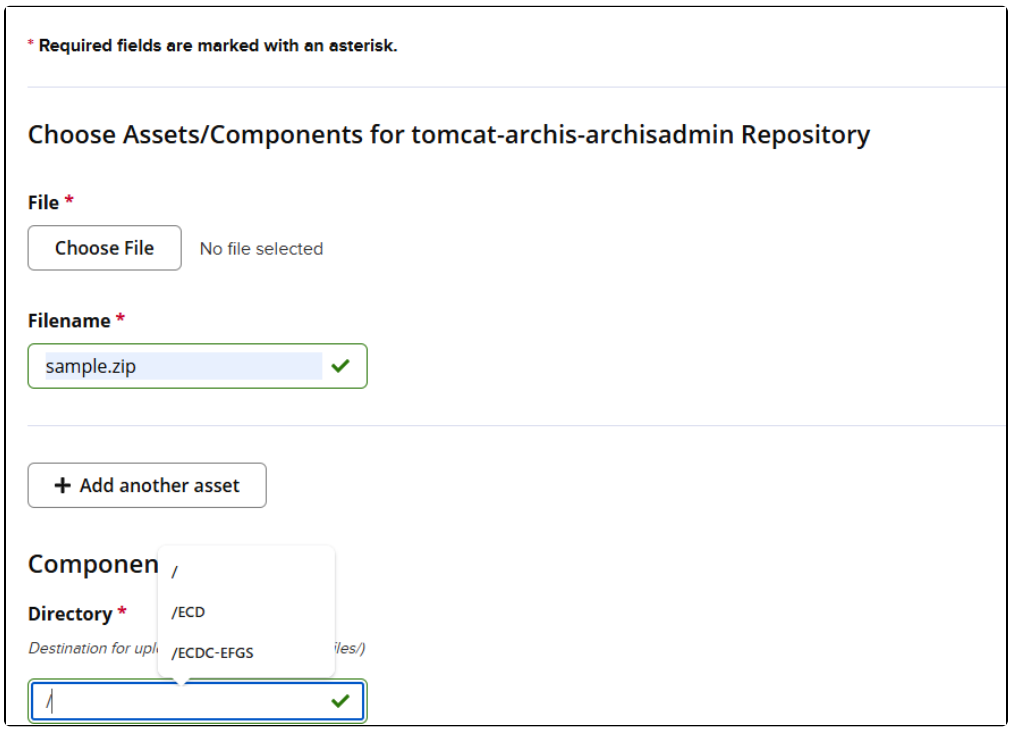
- A ZIP file must contain all WAR files at the root of the archive (If not they won't be deployed)
- This ZIP file will be uploaded in Nexus
  - **NB:** The Nexus repo URL is in the email sent to your FMB when you did the subscription to the service
    - e.g. https://tc-nexus.devops.tech.ec.europa.eu/#browse/browse:tomcat-service-tomcat-design
- Example:
  - **tc-demo-0.0.5-SNAPSHOT.zip** containing directly at the root of the file :
    - sample.war
    - monitor.war
    - ...
- This ZIP file can be manually uploaded in Nexus with the GUI or it can be automated with Bamboo, Maven, etc ...

Here is an example of a simple application that can easily be used



and here is the source code of the project: https://citnet.tech.ec.europa.eu/CITnet/stash/projects/DC2CLOSERV/repos/example-tomcat-war/browse





Basic minimal example

Here is the minimal information needed in the config.yaml properties file (more advanced options are detailed here : Tomcat in Cloud: Create Custom Deployment):

```
informationSystemName: "ams-api"
applicationName: "ams-api-app"
ingressName: "ams"
ingressPath: "/"
replicas: 1
customerImageName: "ams-api"
customerImageTag: "testimg85"
tomcatVersion: "9.0"
jvmVersion: "17"
artifactName: "tc-demo-0.0.5-SNAPSHOT.zip"
buildEnabled: true
```

Tomcat in Cloud: Nexus Repository Structure

Homepage

Page Topics

Related Links

Base Image

Build Customer Image

Version & Patches

Organisation of Nexus repository

Only the **5** last bundles will be kept

Depending on the number of environments in your IS, placing all artifacts in the root directory could be problematic.

We have improved the situation by supporting a directory structure with **one** or **two** levels.

You can organize your artifacts in ZIP bundles like this :

Root-only Repository Structure

```
Root Nexus Repository
├── bundle_1.zip
├── ...
├── bundle_5.zip
```

One-Level Repository Structure

```
Root Nexus Repository
└ <HOSTING_ENVIRONMENT_1>
    ├ bundle_1.zip
    ├ ...
    └ bundle_5.zip

└ <HOSTING_ENVIRONMENT_2>
    ├ bundle_1.zip
    ├ ...
    └ bundle_5.zip

└ <HOSTING_ENVIRONMENT_x>
    ├ bundle_1.zip
    ├ ...
    └ bundle_5.zip
```

Two-Level Repository Structure

```
Nexus Root Repository
└ <MODULE_1>
    └ <HOSTING_ENVIRONMENT_1>
        ├ bundle_1.zip
        ├ ...
        └ bundle_5.zip
    └ <HOSTING_ENVIRONMENT_2>
        ├ bundle_1.zip
        ├ ...
        └ bundle_5.zip
    └ <HOSTING_ENVIRONMENT_x>
        ├ bundle_1.zip
        ├ ...
        └ bundle_5.zip
└ <MODULE_2>
    └ <HOSTING_ENVIRONMENT_1>
        ├ bundle_1.zip
        ├ ...
        └ bundle_5.zip
    └ <HOSTING_ENVIRONMENT_2>
        ├ bundle_1.zip
        ├ ...
        └ bundle_5.zip
    └ <HOSTING_ENVIRONMENT_x>
        ├ bundle_1.zip
        ├ ...
        └ bundle_5.zip
└ <MODULE_x>
    └ <HOSTING_ENVIRONMENT_1>
        ├ bundle_1.zip
        ├ ...
        └ bundle_5.zip
    └ <HOSTING_ENVIRONMENT_2>
        ├ bundle_1.zip
        ├ ...
        └ bundle_5.zip
    └ <HOSTING_ENVIRONMENT_x>
        ├ bundle_1.zip
        ├ ...
        └ bundle_5.zip
```

Three-Level Directory Structure or more

The use of a **third-level** directory will be blocked during pipeline execution, and the directory will be deleted during the cleaning job.

Usage of artifactName Attribute in config.yaml File

The attribute **artifactName** should reflect the location of bundle file in the config.yaml file of your git repo

```
# root directory location
artifactName: "hello.zip"

# One-directory level location
artifactName: "dir1/hello.zip"

# Two-directory level location
artifactName: "dir1/dir2/hello.zip"
```

- Since we keep only a limited number of artifacts, we delete images that are not associated with an existing bundle file.

Tomcat in Cloud: Version & Patches

Homepage

Page Topics

Related Links

Base Image

Build Customer Image

Nexus Repository Structure

What patches are installed in the base images?

Installation of security or technical patches is automatically done during base images creation.

Base images are released when new versions or security patches are released

Is the customer required to invest resources ?

Migrating to a new version of Tomcat can require the customer to invest resources on modifying existing code or writing new code, depending on:

- How the application was written.
- Libraries being used.
- Whether the JDK version has been upgraded.

What software versions are supported and included in the base image ?

Product	Tomcat 9.0.x	Tomcat 10.1.x	Tomcat 11.0.x	TomEE 10.1.x
Latest Version	9.0.113	10.1.50	11.0.15	10.1.3
JDK	Temurin 8/11/17/21/25	Temurin 11/17/21/25	Temurin 17/21/25	Temurin 17/21/25
Apache HTTP Server	2.4.x	2.4.x	2.4.x	2.4.x
EULogin Client for Tomcat	9.15.x	9.15.x	9.15.x	9.15.x
Artefact store	Nexus Repository 3.x.x	Nexus Repository 3.x.x	Nexus Repository 3.x.x	Nexus Repository 3.x.x

NB:

- EULogin JAR file is provided in the base image so that you don't have to provide it
- Operating System is currently Ubuntu 24.04 "Noble Numbat"

What specifications are valid for Apache Tomcat versions ?

	Apache Tomcat 9.0.113	**Apache Tomcat 10.1.50**	**Apache Tomcat 11.0.15**	**Apache TomEE 10.1.3 (Plume)**
Available JDKs	Temurin 8/11/17/21/25	Temurin 11/17/21/25	Temurin 17/21/25	Temurin 17/21/25
Enterprise Edition Namespace	javax.*	jakarta.*	jakarta.*	jakarta.*
Based on Tomcat version	9.0.113	10.1.50	11.0.15	10.1.50
Network Type Availability	BHS/SHS	BHS/SHS	BHS/SHS	BHS/SHS
Jakarta Servlet	4.0	6.0	6.1	6.0
Jakarta Server Pages (JSP)	2.3	3.1	4.0	3.1
Jakarta Expression Language (EL)	3.0	5.0	6.0	5.0
Jakarta WebSocket	1.1	2.1	2.2	2.1
Jakarta Authentication (JASPIC)	1.1	3.0	3.1	3.0
Jakarta EE specifications	n/a	n/a	n/a	10.0
Jakarta Bean Validation	n/a	n/a	n/a	3.0
Jakarta Annotations	n/a	n/a	n/a	2.1
Jakarta Contexts and Dependency Injection (CDI)	n/a	n/a	n/a	4.0
Jakarta Dependency Injection (@Inject)	n/a	n/a	n/a	2.0
Jakarta Enterprise Beans (EJB)	n/a	n/a	n/a	4.0
Jakarta Faces (JSF)	n/a	n/a	n/a	4.0
Jakarta Interceptors	n/a	n/a	n/a	2.1
Jakarta JSON Binding (JSON-B)	n/a	n/a	n/a	3.0
Jakarta JSON Processing (JSON-P)	n/a	n/a	n/a	2.1
Jakarta Managed Beans	n/a	n/a	n/a	2.0
Jakarta Persistence (JPA)	n/a	n/a	n/a	3.1
Jakarta RESTful Web Services (JAX-RS)	n/a	n/a	n/a	3.1
Jakarta Security (Enterprise Security)	n/a	n/a	n/a	3.0
Jakarta Transactions (JTA)	n/a	n/a	n/a	2.0
MicroProfile Specifications	n/a	n/a	n/a	6.1
Jakarta Batch (JBatch)	n/a	n/a	n/a	2.1
Jakarta Messaging (JMS)	n/a	n/a	n/a	3.1
Jakarta Faces (JSF) Implementation	n/a	n/a	n/a	TomEE Mojarra 10.1.x
Jakarta Messaging Implementation	n/a	n/a	n/a	ActiveMQ 6.1.x

Jakarta Persistence (JPA) Implementation (s)	n/a	n/a	n/a	OpenJPA 4.1.x EclipseLink 4.0.x (formerly TopLink)

What is Apache TomEE ?

Apache TomEE (pronounced “Tommy”) can be summarized as:
Tomcat + Jakarta EE = TomEE

Apache TomEE is a **Jakarta EE-certified** application server built entirely from Apache projects. It extends Apache Tomcat by starting with a vanilla Tomcat ZIP distribution, adding the required Java EE libraries (JARs), and packaging the result - providing Tomcat with full enterprise features.

Key included components:

- **Apache CXF** – Web services framework (JAX-RS / JAX-WS)
- **ActiveMQ** – Messaging system (JMS)
- **OpenWebBeans** – Dependency injection (CDI)
- **OpenJPA** – Object-relational mapping (JPA)

If you're already using Tomcat and require Jakarta EE features, **TomEE is a natural upgrade path**. It integrates seamlessly and reduces the effort required to manually add enterprise libraries.

As a **complete and official Jakarta EE implementation**, TomEE makes porting applications from WebLogic significantly easier.

More information: https://tomee.apache.org/comparison.html

What versions of Tomcat are supported by DIGIT ?

Check the Services Roadmap Dashboard for the service lifecycle.

- Select **Service Roadmap Dashboard**.
- Select **General Roadmap**

View information about:

- End of support
- Deprecated dates
- Working duration
- Availability of new versions

> Refer to the Services Roadmap Dashboard User Guide.
>
> The upgrade must be on the official list supported by DIGIT.

What are the version compatibility issues ?

Tomcat Version	Supported Java Versions
Tomcat 9.0.x	Java 8, 11, 17, 21, 25

Tomcat 10.1.x	Java 11, 17, 21, 25
Tomcat 11.0.x	Java 17, 21, 25
TomEE 10.1.x	Java 17, 21, 25

What are the version enhancements ?

Refer to the Tomcat documentation for more information.

Tomcat in Cloud: Gitlab Pipelines for Artifact Creation

Homepage

Page Topics	Related Links
	Base Image Build Customer Image Nexus Repository Structure

Maven Pipeline

In order to rely on DIGIT best practices and take advantage of Gitlab pipeline templates, please refer to this page

- Managed Java Maven Pipeline Tomcat CDM Sample

Artifact ZIP Packaging

Here how your ZIP file must be created :

- Tomcat in Cloud: Simple User Tutorial#PackagetheApplicationFiles

Tomcat in Cloud: Infrastructure as Code (IaC) Blueprints

In the context of Tomcat in Cloud service, Infrastructure as Code (IaC) is made up of **a number of .yaml files**, each of which can be thought of as a member of your team; each performing a specific task(s), but collectively responsible for constructing your Tomcat architecture.

- Tomcat in Cloud: Configuration of the GitOps Repo
- Tomcat in Cloud: Reverse Proxy Mapping As Code (RPMaC)
- Tomcat in Cloud: Monitoring as Code (MONaCo)
- Tomcat in Cloud: Flow as Code (FaC)

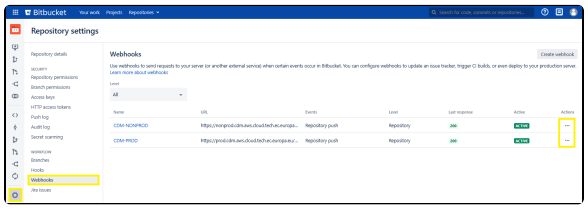
Tomcat in Cloud: Configuration of the GitOps Repo

Homepage

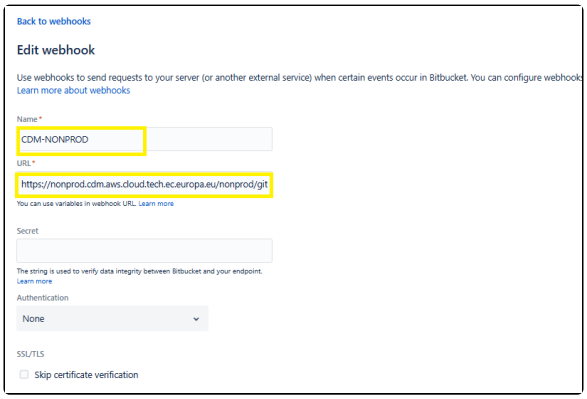
How to proceed with the configuration of the GitOps Repo ?

BitBucket

1. Configure the webhook for your Git repository.
2. In **Bitbucket**, go to settings and webhooks.



3. The following URLs must be used for each specific environment :
    a. https://**devops**.cdm.aws.cloud.tech.ec.europa.eu/**devops**/git-webhook
    b. https://**nonprod**.cdm.aws.cloud.tech.ec.europa.eu/**nonprod**/git-webhook
    c. https://**test**.cdm.aws.cloud.tech.ec.europa.eu/**test**/git-webhook
    d. https://**acc**.cdm.aws.cloud.tech.ec.europa.eu/**acc**/git-webhook
    e. https://**prod**.cdm.aws.cloud.tech.ec.europa.eu/**prod**/git-webhook
4. Use the correct branch: The branch name must be equal to the hosting environment value (example: dev/acc/prod).



5. Execute a Git commit and push the change.
6. Check your mail. You will receive a mail notification when the request is completed.

Related Links

Prerequisites

Request Subscription to the Service

What requests can you make?

Access & Connection to the Environment

Tomcat in Cloud: Reverse Proxy Mapping As Code (RPMaC)

Homepage

Refer to the Reverse Proxy Mapping as Code section in the above User Guide for detailed information.

For specific RPMaC information related to Tomcat, visit the following page to watch a short tutorial video and review a basic code sample

Related Links

Infrastructure as Code (IaC) User Guide

Video and Code Sample: (RPMaC)

Video and Code: Sample Reverse Proxy Mapping As Code (RPMaC)

Homepage

Watch the following video to learn more about using the rpm.yaml blueprint file, provided by DIGIT.

Your browser does not support the HTML5 video element

Related Links

IaC: RPMaC Use Cases

Infrastructure as Code (IaC) User Guide

Monitoring as Code (MONaCo)

Please consult the Reverse Proxy Mapping As Code (RPMaC) User Guide for more information on this platform feature.

Here is a simple example for a basic RPM definition.

The following content must be written in a **rpm.yaml** file and be positioned at the **root** of your git repository.

```
#@data/values
---
rpms:
  - domain: intragate.development.ec.europa.eu
    targetResource: internal-test-api
    sourcecontextRoot: test
    targetcontextRoot: test
    rewrites: true
    ecas: true
    timeout: 180
    compression: true
    caching: true
    extendHttpMethod: false
    testPage: /sample
```

URL to access will then be https://<domain>/<sourceContextRoot>

Tomcat in Cloud: Monitoring as Code (MONaCo)

Homepage

Monaco (Monitoring as a Code) is a CLI tool that automates deployment of Dynatrace Monitoring Configuration to one or multiple Dynatrace environments.

With **Monaco**, a developer can:

- Set up monitoring and observability easily and efficiently by utilizing configuration files instead of a Graphical User Interface.
- Use configuration files to enable you to create, update, and manage your monitoring configurations safely, consistently, and repetitively. They can be reused, versioned, and shared within your team.
- Enable their development teams to define monitoring configuration as code that is checked into version control alongside application source code.

Refer to the Monitoring as a Code section in the Infrastructure as Code (IaC) User Guide for more information.

Related Links

Infrastructure as Code (IaC) User Guide

Flow-as-Code (FaC)

Reverse Proxy Mapping As Code (RPMaC)

Video: Reverse Proxy Mapping As Code (RPMaC)

Tomcat in Cloud: Flow as Code (FaC)

Homepage

Introduction to Flow-as-Code (FaC)

Flow-as-Code (FaC) is an **Infrastructure as Code (IaC)** tool that automates the **opening and closing of network flows**, which define communication rules between applications and services.

A **flow** specifies allowed connections between resources, such as an application in the cloud and a service in a data center.

Key Benefits of FaC

By leveraging **GitOps**, FaC enables teams to:

- **Streamline configuration management** by managing flows through version-controlled `flow.yaml` files.
- **Enhance operational efficiency** by ensuring consistency, traceability, and security compliance.
- **Reduce manual intervention** and minimize configuration errors.

FaC applies exclusively to services **deployed in Containers**, running in **Cloud on Premises (COP)**.

Refer to Flow-as-Code (FaC) section in the above User Guide for more information.

Related Links

Infrastructure as Code (IaC) User Guide

Reverse Proxy Mapping As Code (RPMaC)

Video: Reverse Proxy Mapping As Code (RPMaC)

Monitoring as Code (MONaCo)

Tomcat in Cloud: Image Deployment

Tomcat in Cloud: Deployment Chain

Homepage

Page Topics

Related Links

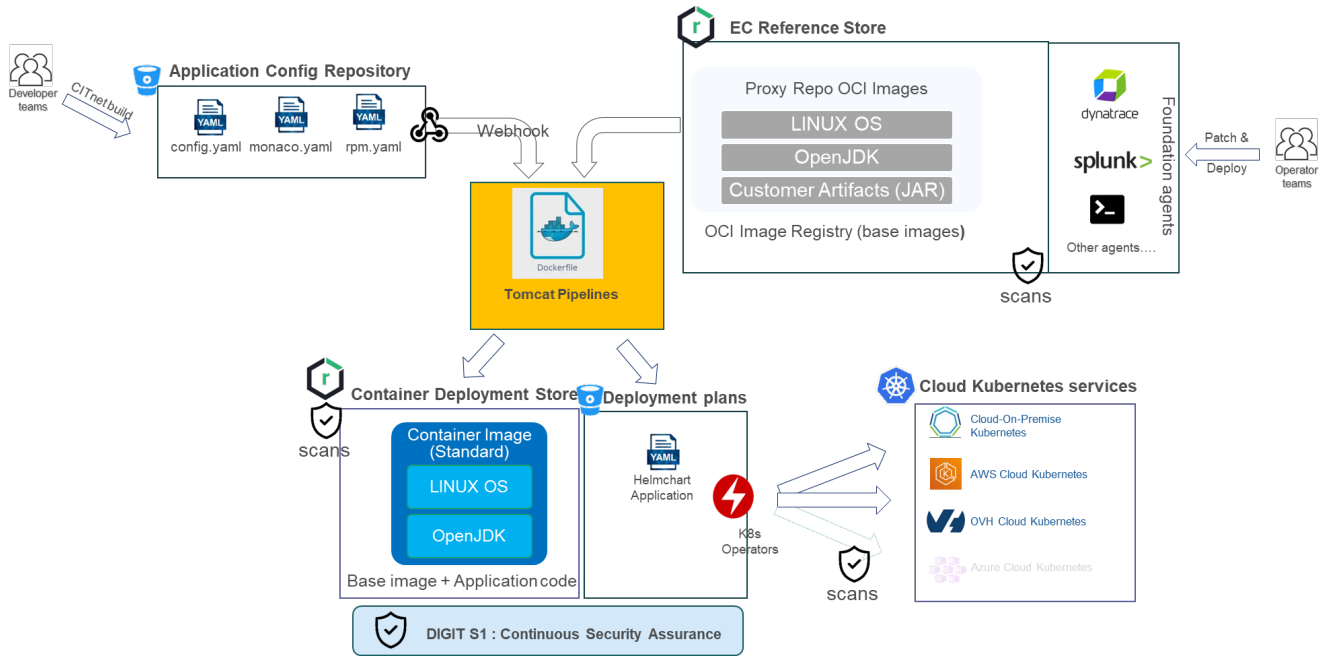
Create Custom Deployment

Management of Secrets

Pod Restart

Maintenance Mode

What is the Tomcat Cloud deployment flow?



GitOps Paradigm

GitOps is a set of practices that utilize Git as the single source of truth for managing infrastructure and application configurations, enabling the efficient delivery of infrastructure as code.

The Continuous Integration (CI) process validates submitted code, ensuring it meets quality standards, while the Continuous Deployment (CD) process enforces and implements requirements such as security protocols, infrastructure as code standards, and application framework constraints. By tracking all code changes, GitOps simplifies updates and provides robust version control, allowing for seamless rollbacks when necessary.

GitOps delivers:

- A standard workflow for application development
- Increased security for setting application requirements upfront
- Improved reliability with visibility and version control through Git
- Consistency across any cluster, any cloud, and any on-premise environment

The following tools are used together to build a GitOps framework :

- Deliverable repositories (Nexus)
- Distributed version control system repository (Git)
- Container orchestration system for automating software deployment, scaling, and management (Kubernetes)
- Continuous Integration/Continuous Delivery (CI/CD) tools (ArgoCD)
- Configuration management tools ( )

In simpler terms, it means that a configuration of the application is stored in the Git repository. The CD process should react to changes in this configuration, and then apply them to the Kubernetes cluster

- Atlassian BitBucket will be used as the Git-based source code repository
- Infrastructure as code (IaC) is the process of managing and provisioning environments through YAML machine-readable manifesto files.

Pipelines

Pipelines will be developed in AWS utilizing the following components:

- **Lambdas**
- **Step Functions**
- **CodeBuild**

Development will leverage **Serverless Framework** and **Terraform** to ensure cloud-provider agnosticism.

The solution requires the creation of two primary pipelines:

1. **Build Pipeline**: Responsible for creating Docker images.
2. **Deploy Pipeline**: Handles the deployment of these images to Kubernetes (K8S).

Tomcat in Cloud: Create Custom Deployment

Homepage

Page Topics

Related Links

Deployment Chain

Management of Secrets

Pod Restart

Maintenance Mode

Watch the following video to learn more about deploying your Tomcat apps to the Cloud.

Your browser does not support the HTML5 video element

How to create a custom deployment using **config.yaml** file

- Tomcat provides the capability to create a custom deployment using the `config.yaml` file. This file specifies the deployment details required by the customer and must adhere to the following requirements:
  - **Location**: The `config.yaml` file must be placed at the root of the customer's Git repository.
  - **Webhook Configuration**: A webhook must be configured for the Git repository.
    - GitLab Webhook creation
  - **Case Sensitivity**: The file name must be in lowercase (`config.yaml`). If not, it will be ignored.

For more information, watch the deployment explanation video, available here.

What meta data is required ?

Attribute Path	Description	Mandatory	Default	Example /Comments
k8sApplicationKind	Kind of Kubernetes application. It can be one of the following kind: ■ **Deployment** : stateless application ■ StatefulSet: stateful application	N	Deployment	
applicationName	Name of the application **(This must be in slug-format)**	Y	-	ams-api-app

ingressName	Base name of the ingress **(This must be in slug-format)** **NB: cannot be more than 12 characters**	Y	-	ams-api The application will be accessible through : https://**<ingressName>**.tc-app.**<k8sClusterFQDN>/<ingressPath>**
ingressPath	Represents the path of the deployed application **NB: cannot be more than 20 characters**	Y	-	/
replicas	Represents the number of instances/pods that will manage the application **NB**: It can be 0 to request Maintenance Mode	Y	-	4

customerImageName	• Represents the name of the built image.
	  ○ This image should not be updated or overwritten
	  ○ Only the previous 10 images are saved.
	
	NB: cannot be more than 20 characters
		Y	-	curex-app

customerImageTag	• Represents the tag of the built image.
	  ○ This image should not be updated or overwritten
	  ○ Only the previous 10 images are saved.
		Y	-	testimg85

artifactName	The name of the artifact containing the WAR files • Stored in Nexus	Y	-	projectProperties.zip
applicationPropertiesZip	The name of the artifact containing the properties files • It contains files for each environment (DEV/ACC/PROD...) • Stored in Nexus	N	-	

jvmVersion	Version of JVM to use for the runtime.	N	11	**NB**: the latest minor version available at the time of the build will be used Possible values for Tomcat 9.0: • 8, **11,** 17, 21, 25 Possible values for Tomcat 10.1: ▪ **11,** 17, 21, 25 Possible values for Tomcat 11.0: ▪ **17**, 21, 25 Possible values for Tomee 10.1: ▪ **17**, 21, 25
productName	Name of the flavour of Tomcat wanted (tomcat or tomee)	N	tomcat	Possible values are: • **tomcat,** tomee
tomcatVersion	Version of Tomcat to use for the runtime.	N	9.0	**NB**: the latest minor version available at the time of the build will be used Possible values are: ▪ **9.0,** 10.1, 11.0
tomeeVersion	Version of TomEE to use for the runtime.	N	10.1	**NB**: the latest minor version available at the time of the build will be used Possible values are: • **10.1**

clientParamsKey	Key under which the customer parameters will be defined in the HashiCorp Vault. NB: if the key is '**clientParams**', the parameters will be stored in different subdirs for each environment: ■ DEV: kv_customer/dev/**clientParams** ■ ACC: kv_customer/acc/**clientParams** ■ PROD: kv_customer/prod/**clientParams** ■ XXX: kv_customer/xxx/**clientParams** **NB:** • **Key must NOT start or finish with a slash '/'** • **Key must NOT start with 'kv_customer/<environment>'. This prefix will be prepended accordingly** • **Use this property only if you have parameters in the vault.** • **If you do not provide parameters in the vault, don't use this parameter or deployment will fail !**	N	if no value, the vault is not used	clientParams
clientFilesKey	Key under which the customer files will be defined in the HashiCorp Vault. NB: if the key is '**clientFiles**', the files will be stored in different subdirs for each environment: • DEV: kv_customer/dev/**clientFiles** • ACC: kv_customer/acc/**clientFiles** • PROD: kv_customer/prod/**clientFiles** • XXX: kv_customer/xxx/**clientFiles** **NB:** • **Key must NOT start or finish with a slash '/'** • **Key must NOT start with 'kv_customer/<environment>'. This prefix will be prepended accordingly** • **Use this property only if you have files in the vault.** • **If you do not provide files in the vault, don't use this parameter or deployment will fail !**	N	if no value, the vault is not used	clientFiles
autoscalingEnabled	When enabled, the number of pods will be automatically adapted in function of the available resources for the environment. It starts Tomcat instance(s) if the total Memory or CPU of pod is used at 80% during more than 30 seconds The number of instances depends number of **replicas** you configured • if replicas is lower or equals to **3** then **1** extra pod could be started • if replicas is greater than **3** and lower or equals than **8** then **2** extra pods could be started • if replicas is greater than **8** and lower or equals than **12** then **3** extra pods could be started • if replicas equals **13** then **2** extra pods could be started • if replicas equals **14** then **1** extra pod could be started When the load is coming back to normal situation (less 65% during more than 30 seconds) then extra pods will be shutdown at one	N	false	false
autoscalingUpInterval	This setting controls how many seconds the autoscaler waits before scaling up when a need for resources is detected. Shorter windows make HPA more responsive, but may cause instability. Longer windows increase stability, but reduce agility.	N	30	
autoscalingDownInterval	This setting controls how many seconds the autoscaler waits before scaling down when the peak period is over. Shorter windows make HPA more responsive, but may cause instability. Longer windows increase stability, but reduce agility.	N	60	

logRetentionPeriod	This parameters enables to change retention period for logs which has an impact on the cost of the IS. • Implementation of the feature	N	small	• Allowed values: ○ **xs** (1 month) ○ **small** (3 months) ○ **medium** (6 months) ○ **large** (12 months)
debugLogEnabled	Logs are more verbose when this option is enabled	N	false	
buildEnabled	Indicates if the build must be made with CDM	N	true	false/true
deployEnabled	Indicates if the deploy must be made with CDM	N	true	false/true
workloadSchedulerMode	Customers can specify their application to be running during which available plans which has an impact on the cost of their IS as their will be no consumed memory from their application.	N	24/7	Available modes are the following : • 13/5 : stopped during the night (between 21:00 and 7:00) and during WE • 24/5 : stopped during the WE • 24/7 : always started (DEFAULT)
ingress:	==========	N		==========
stickySessionEnabled	Indicates if ingress must use sticky sessions or not	N	false	
affinity	Sets the affinity type in all Upstreams of an Ingress	N	cookie	
affinityMode	Defines the stickiness of a session.	N	persistent	
sessionCookieName	Specifes the name of the cookie that will be used to route the requests	N	INGRESSCOOKIE	
sessionCookiePath	Controls the cookie path	N	/	

sessionCookieMaxAge	Time until the cookie expires, corresponds to the `Max-Age` cookie directive	N	172800 (corresponds to 48 hours)	
sessionCookieExpires	Legacy version of the previous annotation for compatibility with older browsers, generates an `Expires` cookie directive by adding the seconds to the current date	N	172800 (corresponds to 48 hours)	
proxySendTimeout	Sets a timeout for transmitting a request to the proxied server. The timeout is set only between two successive write operations, not for the transmission of the whole request. If the proxied server does not receive anything within this time, the connection is closed.	N	300	
proxyReadTimeout	Sets a timeout for reading a response from the proxied server. The timeout is set only between two successive read operations, not for the transmission of the whole response. If the proxied server does not transmit anything within this time, the connection is closed.	N	300	
proxyBodySize	Size of the transmitted body	N	0	
proxyBuffering	Enables or disables buffering of responses from the proxied server.	N	'on'	
proxyBufferSize	Sets the size of the buffer used for reading the first part of the response received from the proxied server. This part usually contains a small response header.	N	'16k'	
proxyBuffersNumber	Sets the number of the buffers used for reading a response from the proxied server, for a single connection.	N	8	
proxyBusyBuffersSize	When buffering of responses from the proxied server is enabled, limits the total *size* of buffers that can be busy sending a response to the client while the response is not yet fully read	N	'32k'	
sslPassthrough	Instructs the controller to send TLS connections directly to the backend instead of letting NGINX decrypt the communication. It allows to forward the SSL stream to the K8S service, so that the SSL transaction is managed directly with the service instead of the ingress-controller. More information here : Enable passthrough on the ingress	N	'false'	
tomcat:jvm:	========== ==========	N		==========
heapMinSize	Minimum size of the Heap of the Java Virtual Machine for Tomcat runtime. **NB: Value is in megabytes. Maximum value is 16384 Mb**	N	1536	**Maximum value is 16384 Mb** 1536
heapMaxSize	Maximum size of the Heap of the Java Virtual Machine for Tomcat runtime **NB: Value is in megabytes. Maximum value is 16384 Mb**	N	1536	**Maximum value is 16384 Mb** 1536
metaMinSize	Minimum size of the Meta of the Java Virtual Machine for Tomcat runtime **NB: Value is in megabytes. Maximum value is 2560 Mb**	N	768	**Maximum value is 2560 Mb** 768
metaMaxSize	Maximum size of the Meta of the Java Virtual Machine for Tomcat runtime **NB: Value is in megabytes. Maximum value is 2560 Mb**	N	768	**Maximum value is 2560 Mb** 768
directMaxSize	Maximum size of the Direct of the Java Virtual Machine for Tomcat runtime **NB: Value is in megabytes. Maximum value is 8192 Mb**	N	2048	**Maximum value is 8192 Mb** 2048
debugGC	Option used to have more logs regarding the garbage collector	N	true	true
debugSSL	Option used to have more information about SSL problems	N	false	false
debugJavaLauncher	Option used to have more information during startup of the JVM	N	false	
hideArgsInLog	Sensible arguments as passwords and secrets will be hidden in the logs.	N	false	false
gcLogPathSeparated	Garbage collector logs will be separated in a specific directory	N	true	true

hideVersions	All Tomcat version will be removed from pages and logs, so that hacking is more difficult	N	false	false
gcUsed	Garbage collector used	N	parallel	parallel
environment Variables	Environments variables passed to the Tomcat JVM	N	''	
logArgs	If `true`, the command line arguments passed to Java when Tomcat started will be logged. If not specified, the default value of `true` will be used.	N	true	
logEnv	If `true`, the current environment variables when Tomcat starts will be logged. If not specified, the default value of `false` will be used.	N	false	
logProps	If `true`, the current Java system properties will be logged. If not specified, the default value of `false` will be used.	N	false	
tomcat: catalinaProperties:	`==========`	N		`==========`
sharedLoader	Name of the shared loader to use	N	''	
tomcat: context:	`==========`	N		`==========`
cookieProcessor	The CookieProcessor element represents the component that parses received cookie headers into javax.servlet.http.Cookie objects accessible through HttpServletRequest.getCookies() and converts javax.servlet.http.Cookie objects added to the response through HttpServletResponse. addCookie() to the HTTP headers returned to the client.	N	org.apache. tomcat.util. http. Rfc6265Cooki eProcessor	org.apache. tomcat.util. http. Rfc6265Cooki eProcessor
cookieUseHttpOnly	Cookies will be used only for HTTP	N	true	true
cachingAllowed	If the value of this flag is `true`, the cache for static resources will be used. If not specified, the default value of the flag is `true`. This value may be changed while the web application is running (e. g. via JMX). When the cache is disabled any resources currently in the cache are cleared from the cache.	N	true	
cacheMaxSize	The maximum size of the static resource cache in kilobytes. If not specified, the default value is `10240` (10 MiB).	N	10240	
tomcat: connector:	`==========`	N		`==========`
httpPort	The TCP port number on which this Connector will create a server socket and await incoming connections. Your operating system will allow only one server application to listen to a particular port number on a particular IP address. If the special value of 0 (zero) is used, then Tomcat will select a free port at random to use for this connector. This is typically only useful in embedded and testing applications. **NB: If unsure, keep the default value.**	N	8080	8080
httpProxyName	If this Connector is being used in a proxy configuration, configure this attribute to specify the server name to be returned for calls to request.getServerName(). See Proxy Support for more information. **NB: If unsure, keep the default value.**	N	''	
httpProxyPort	If this Connector is being used in a proxy configuration, configure this attribute to specify the server port to be returned for calls to request.getServerPort(). See Proxy Support for more information. **NB: If unsure, keep the default value.**	N	''	
httpUriEncoding	This specifies the character encoding used to decode the URI bytes, after %xx decoding the URL. The default value is UTF-8. **NB: If unsure, keep the default value.**	N	''	
httpUpgradeProtocol	HTTP/2 connectors use non-blocking I/O, only utilising a container thread from the thread pool when there is data to read and write. However, because the Servlet API is fundamentally blocking, each HTTP/2 stream requires a dedicated container thread for the duration of that stream. Requests processed using HTTP/2 will have the following additional request attributes available: • `org.apache.coyote.connectionID` will return the HTTP/2 connection ID • `org.apache.coyote.streamID` will return the HTTP/2 stream ID **NB: If unsure, keep the default value.**	N	false	

httpsPort	The TCP port number on which this Connector will create a server socket and await incoming connections. Your operating system will allow only one server application to listen to a particular port number on a particular IP address. If the special value of 0 (zero) is used, then Tomcat will select a free port at random to use for this connector. This is typically only useful in embedded and testing applications. **NB: If unsure, keep the default value.**	N	8443	8080
httpsProxyName	If this Connector is being used in a proxy configuration, configure this attribute to specify the server name to be returned for calls to request.getServerName(). See Proxy Support for more information. **NB: If unsure, keep the default value.**	N	''	
httpsProxyPort	If this Connector is being used in a proxy configuration, configure this attribute to specify the server port to be returned for calls to request.getServerPort(). See Proxy Support for more information. **NB: If unsure, keep the default value.**	N	''	
httpsUriEncoding	This specifies the character encoding used to decode the URI bytes, after %xx decoding the URL. The default value is UTF-8. **NB: If unsure, keep the default value.**	N	''	
httpsUpgradeProtocol	HTTP/2 connectors use non-blocking I/O, only utilising a container thread from the thread pool when there is data to read and write. However, because the Servlet API is fundamentally blocking, each HTTP/2 stream requires a dedicated container thread for the duration of that stream. Requests processed using HTTP/2 will have the following additional request attributes available: • `org.apache.coyote.connectionID` will return the HTTP/2 connection ID • `org.apache.coyote.streamID` will return the HTTP/2 stream ID **NB: If unsure, keep the default value.**	N	false	
relaxedPathChars	The HTTP/1.1 specification requires that certain characters are %nn encoded when used in URI paths. Unfortunately, many user agents including all the major browsers are not compliant with this specification and use these characters in unencoded form. To prevent Tomcat rejecting such requests, this attribute may be used to specify the additional characters to allow. If not specified, no additional characters will be allowed. The value may be any combination of the following characters: " < > [\ ] ^ ` { | } . Any other characters present in the value will be ignored.	N	''	
relaxedQueryChars	The HTTP/1.1 specification requires that certain characters are %nn encoded when used in URI query strings. Unfortunately, many user agents including all the major browsers are not compliant with this specification and use these characters in unencoded form. To prevent Tomcat rejecting such requests, this attribute may be used to specify the additional characters to allow. If not specified, no additional characters will be allowed. The value may be any combination of the following characters: " < > [\ ] ^ ` { | } . Any other characters present in the value will be ignored.	N	''	
compressionEnabled	The Connector may use HTTP/1.1 GZIP compression in an attempt to save server bandwidth. The acceptable values for the parameter is "off" (disable compression), "on" (allow compression, which causes text data to be compressed), "force" (forces compression in all cases), or a numerical integer value (which is equivalent to "on", but specifies the minimum amount of data before the output is compressed). If the content-length is not known and compression is set to "on" or more aggressive, the output will also be compressed. If not specified, this attribute is set to "off". Note: There is a tradeoff between using compression (saving your bandwidth) and using the sendfile feature (saving your CPU cycles). If the connector supports the sendfile feature, e.g. the NIO connector, using sendfile will take precedence over compression. The symptoms will be that static files greater that 48 KiB will be sent uncompressed. You can turn off sendfile by setting useSendfile attribute of the connector, as documented below, or change the sendfile usage threshold in the configuration of the DefaultServlet in the default conf/web.xml or in the web.xml of your web application.	N	''	
compressionMinSize	If compression is set to "on" then this attribute may be used to specify the minimum amount of data before the output is compressed. If not specified, this attribute is defaults to "2048". Units are in bytes.	N	2048	2048

compressibleMimeType	The value is a comma separated list of MIME types for which HTTP compression may be used. The default value is text/html,text/xml,text/plain,text/css,text/javascript,application/javascript, application/json,application/xml . If you specify a type explicitly, the default is over-ridden.	N	text/html,text/xml,text/plain,text/css,text/javascript, application/javascript, application/json, application/xml	text/html,text/xml,text/plain,text/css, text/javascript, application/javascript, application/json, application/xml
allowTrace	A boolean value which can be used to enable or disable the TRACE HTTP method. If not specified, this attribute is set to false. As per RFC 7231 section 4.3.8, cookie and authorization headers will be excluded from the response to the TRACE request. If you wish to include these, you can implement the `doTrace()` method for the target Servlet and gain full control over the response.	N	false	
xpoweredBy	Set this attribute to true to cause Tomcat to advertise support for the Servlet specification using the header recommended in the specification. The default value is false.	N	false	
minSpareThreads	The minimum number of threads always kept running. This includes both active and idle threads. If not specified, the default of 10 is used. If an executor is associated with this connector, this attribute is ignored as the connector will execute tasks using the executor rather than an internal thread pool. Note that if an executor is configured any value set for this attribute will be recorded correctly but it will be reported (e.g. via JMX) as -1 to make clear that it is not used.	N	''	
maxSpareThreads	The maximum number of threads always kept running. This includes both active and idle threads. If not specified, the default of 10 is used. If an executor is associated with this connector, this attribute is ignored as the connector will execute tasks using the executor rather than an internal thread pool.	N	''	
maxPostSize	The maximum size in bytes of the POST which will be handled by the container FORM URL parameter parsing. The limit can be disabled by setting this attribute to a value less than zero. If not specified, this attribute is set to 2097152 (2 MiB). Note that the FailedRequestFilter can be used to reject requests that exceed this limit.	N	''	
maxThreads	The maximum number of request processing threads to be created by this Connector, which therefore determines the maximum number of simultaneous requests that can be handled. If not specified, this attribute is set to 200. If an executor is associated with this connector, this attribute is ignored as the connector will execute tasks using the executor rather than an internal thread pool. Note that if an executor is configured any value set for this attribute will be recorded correctly but it will be reported (e.g. via JMX) as -1 to make clear that it is not used.	N	''	
maxParameterCount	The maximum total number of request parameters (including uploaded files) obtained from the query string and, for POST requests, the request body if the content type is application/x-www-form-urlencoded or multipart/form-data. Request parameters beyond this limit will be ignored. A value of less than 0 means no limit. If not specified, a default of 10000 is used. Note that FailedRequestFilter filter can be used to reject requests that exceed the limit.	N	''	
maxHttpHeaderSize	Provides the default value for maxHttpRequestHeaderSize and maxHttpResponseHeaderSize. If not specified, this attribute is set to 8192 (8 KiB).	N	''	
maxSwallowSize	The maximum number of request body bytes (excluding transfer encoding overhead) that will be swallowed by Tomcat for an aborted upload. An aborted upload is when Tomcat knows that the request body is going to be ignored but the client still sends it. If Tomcat does not swallow the body the client is unlikely to see the response. If not specified the default of 2097152 (2 MiB) will be used. A value of less than zero indicates that no limit should be enforced.	N	''	
enableLookups	Set to true if you want calls to request.getRemoteHost() to perform DNS lookups in order to return the actual host name of the remote client. Set to false to skip the DNS lookup and return the IP address in String form instead (thereby improving performance). By default, DNS lookups are disabled.	N	''	
acceptCount	The maximum length of the operating system provided queue for incoming connection requests when maxConnections has been reached. The operating system may ignore this setting and use a different size for the queue. When this queue is full, the operating system may actively refuse additional connections or those connections may time out. The default value is 100.	N	''	

connectionTimeout	The number of milliseconds this Connector will wait, after accepting a connection, for the request URI line to be presented. Use a value of -1 to indicate no (i.e. infinite) timeout. The default value is 60000 (i.e. 60 seconds) but note that the standard server.xml that ships with Tomcat sets this to 20000 (i.e. 20 seconds). Unless disableUploadTimeout is set to false, this timeout will also be used when reading the request body (if any).	N	''	
keepAliveTimeout	The number of milliseconds this Connector will wait for another HTTP request before closing the connection. The default value is to use the value that has been set for the connectionTimeout attribute. Use a value of -1 to indicate no (i.e. infinite) timeout.	N	''	
disableUploadTimeout	This flag allows the servlet container to use a different, usually longer connection timeout during data upload. If not specified, this attribute is set to true which disables this longer timeout.	N	''	
tomcat:host:	==========	N		==========
deployXml	Set to false if you want to disable parsing the context XML descriptor embedded inside the application (located at /META-INF/context.xml). Security conscious environments should set this to false to prevent applications from interacting with the container's configuration. The administrator will then be responsible for providing an external context configuration file, and putting it in the location defined by the xmlBase attribute. If this flag is false, a descriptor is located at /META-INF/context.xml and no descriptor is present in xmlBase then the context will fail to start in case the descriptor contains necessary configuration for secure deployment (such as a RemoteAddrValve) which should not be ignored. The default is true unless a security manager is enabled when the default is false. When running under a security manager this may be enabled on a per web application basis by granting the org.apache.catalina.security.DeployXmlPermission to the web application. The Manager and Host Manager applications are granted this permission by default so that they continue to work when running under a security manager.	N	''	
copyXml	Set to true if you want a context XML descriptor embedded inside the application (located at /META-INF/context.xml) to be copied to xmlBase when the application is deployed. On subsequent starts, the copied context XML descriptor will be used in preference to any context XML descriptor embedded inside the application even if the descriptor embedded inside the application is more recent. The default is false. Note if deployXML is false, this attribute will have no effect.	N	''	
tomcat:log:	==========	N		==========
logRotate	Enables rotation of the logs	N	''	
logMaxAge	Maximum age of rotating logs. If the file is older than this age, the log is purged.	N	''	
logMaxSize	Maximum size of rotating logs. If the file is bigger than this size, the log is purged.	N	''	
tomcat:realm:	==========	N		==========
addAllRolesMode	Adds the **allRolesMode="authOnly"** property in the Realm of context.xml as required for ECAS /EULogin configuration See explanation here from ECAS/EULogin documentation : https://citnet.tech.ec.europa.eu/CITnet/confluence/spaces/IAM/pages/24641879/EU+Login+Client+for+Apache+Tomcat	N	false	
tomcat:valve:	==========	N		==========
rewriteValveEnabled	The rewrite valve implements URL rewrite functionality in a way that is very similar to mod_rewrite from Apache HTTP Server.	N	''	
ecasAuthenticatorValveEnabled	Valve that calls the ECAS/EULogin Authenticator.	N	''	
remoteIpValveEnabled	Tomcat port of mod_remoteip, this valve replaces the apparent client remote IP address and hostname for the request with the IP address list presented by a proxy or a load balancer via a request headers (e.g. "X-Forwarded-For").	N	''	
accessLogValveEnabled	Abstract implementation of the Valve interface that generates a web server access log with the detailed line contents matching a configurable pattern. The syntax of the available patterns is similar to that supported by the Apache HTTP Server mod_log_config module.	N	''	

errorReportValveEnabled	Implementation of a Valve that outputs HTML error pages. NB: possibly contains server version information.	N	''	
errorReportValveError404	Location of the page to use in case of 404 errors	N	''	
tomcat:web:	==========	N		==========
defaultSessionTimeout	Timeout in seconds after which the session automatically expires	N	''	
jspServletEnablePooling	Determines whether tag handler pooling is enabled. This is a compilation option. It will not alter the behaviour of JSPs that have already been compiled.	N	''	
requestCharacterEncoding	Character encoding used for the Web connector requests	N	''	
responseCharacterEncoding	Character encoding used for the Web connector responses	N	''	
hstsEnabled	HTTP Strict Transport Security (HSTS) is a web security policy mechanism, which helps protect web application users against some passive (eavesdropping) and active network attacks.	N	''	
secureCookies	Enable the HTTPOnly and Secure attributes for cookies as sent by Apache Tomcat.	N	''	
securityFilterEnabled	Enables an extra Security Servlet Filter developed for the EC	N	''	
httpsRedirectionEnabled	Automatically redirects all unsecured HTTP traffic to HTTPS	N	''	
corsFilterEnabled	Enables CORS (Cross-Origin Resource Sharing) filter, which enables cross-origin requests.	N	-	
corsAllowedOrigins	A list of origins that are allowed to access the resource. A * can be specified to enable access to resource from any origin. Otherwise, an allow list of comma separated origins can be provided. Eg: `https://www.w3.org, https://www.apache.org`. **Defaults:** The empty String. (No origin is allowed to access the resource).	N	'*'	
corsAllowedMethods	A comma separated list of HTTP methods that can be used to access the resource, using cross-origin requests. These are the methods which will also be included as part of `Access-Control-Allow-Methods` header in pre-flight response.	N	'GET,POST, HEAD, OPTIONS, TRACE,PUT, DELETE, PATCH, CONNECT'	
corsAllowedHeaders	A comma separated list of request headers that can be used when making an actual request. These headers will also be returned as part of `Access-Control-Allow-Headers` header in a pre-flight response. Eg: `Origin,Accept`. **Defaults:** `Origin, Accept, X-Requested-With, Content-Type, Access-Control-Request-Method, Access-Control-Request-Headers`	N	'Content-Type,X-Requested-With,Accept, Origin,Access-Control-Request-Method, Access-Control-Request-Headers, Authorization'	
corsExposedHeaders	A comma separated list of headers other than simple response headers that browsers are allowed to access. These are the headers which will also be included as part of `Access-Control-Expose-Headers` header in the pre-flight response. Eg: `X-CUSTOM-HEADER-PING,X-CUSTOM-HEADER-PONG`.	N	'Access-Control-Allow-Origin,Access-Control-Allow-Credentials'	
corsPreflightMaxage	The amount of seconds, browser is allowed to cache the result of the pre-flight request. This will be included as part of `Access-Control-Max-Age` header in the pre-flight response. A negative value will prevent CORS Filter from adding this response header to pre-flight response.	N	1800	
tomcat:datasources:	==========	N		==========
name	The name of the JNDI JDBC DataSource for this UserDatabase.	Y	''	

provider	The database provider to use for the datasource	Y	'oracle'	possible values are: ■ **oracle** ■ mssql ■ mysql ■ postgresql
url	The URL to use for the database connected to this datasource	Y	''	Please provide a valid jdbc URL for the chosen provider for Oracle : jdbc:oracle:thin:@//olrdev99.cc.cec.eu.int:1597:XXX_YYY_01_D for PostgreSQL : jdbc:postgresql://pgdev99.cc.cec.eu.int:1597/database for MySQL : jdbc:mysql://mysqlhost:3306/test
username	Username used for authentication NB: Associated password must be defined in the vault. More information here : Tomcat in Cloud: Manage Secrets	Y	''	
factory	Name of the factory used to create connections. This is required, and the value should be `org.apache.tomcat.jdbc.pool.DataSourceFactory !`	N	'org.apache.tomcat.jdbc.pool.DataSourceFactory'	
auth	Optional authentication method of the datasource (NB: if not sure, keep the default value)	N	'Container'	
type	Optional type of the datasource (NB: if not sure, keep the default value). Type should always be `javax.sql.DataSource` or `javax.sql.XADataSource` Depending on the type, a `org.apache.tomcat.jdbc.pool.DataSource` or a `org.apache.tomcat.jdbc.pool.XADataSource` will be created.	N	'javax.sql.DataSource'	
initialSize	Number of connections present in the pool after creation.	N	10	
minIdle	Minimum number of idle database connections to retain in pool. Set to -1 for no limit. See also the DBCP 2 documentation on this and the minEvictableIdleTimeMillis configuration parameter.	N	10	
maxIdle	Maximum number of idle database connections to retain in pool. Set to -1 for no limit.	N	100	
maxTotal	Maximum number of database connections in pool. Set to -1 for no limit.	N	100	
removeAbandonedTimeout	Timeout in seconds before an abandoned(in use) connection can be removed. The default value is `60` (60 seconds). The value should be set to the longest running query your applications might have.	N	false	
removeAbandonedOnMaintenance	The indication of whether to remove abandoned connections from the pool during pool maintenance. Default: false	N	false	
testOnBorrow	The indication of whether objects will be validated before being borrowed from the pool. If the object fails to validate, it will be dropped from the pool, and we will attempt to borrow another. In order to have a more efficient validation, see `validationInterval`. Default value is `false`	N	false	

logAbandoned	Flag to log stack traces for application code which abandoned a Connection. Logging of abandoned Connections adds overhead for every Connection borrow because a stack trace has to be generated. The default value is `false`.	N	false	
testWhileIdle	The indication of whether objects will be validated by the idle object evictor (if any). If an object fails to validate, it will be dropped from the pool. The default value is `false` and this property has to be set in order for the pool cleaner/test thread is to run (also see `timeBetweenEvictionRunsMillis`)	N	false	
testOnReturn	The indication of whether objects will be validated before being returned to the pool. The default value is `false`.	N	false	
removeAbandonedOnBorrow	The indication of whether to remove abandoned connections from the pool when a connection is borrowed. Default: false	N	false	
timeBetweenEvictionRunsMillis	The number of milliseconds to sleep between runs of the idle connection validation/cleaner thread. This value should not be set under 1 second. It dictates how often we check for idle, abandoned connections, and how often we validate idle connections. This value will be overridden by `maxAge` if the latter is non-zero and lower. The default value is `5000` (5 seconds).	N	5000	
validationInterval	avoid excess validation, only run validation at most at this frequency - time in milliseconds. If a connection is due for validation, but has been validated previously within this interval, it will not be validated again. The default value is `3000` (3 seconds).	N	3000	
minEvictableIdleTimeMillis	The minimum amount of time an object may sit idle in the pool before it is eligible for eviction. The default value is `60000` (60 seconds).	N	60000	
maxAge	Time in milliseconds to keep a connection before recreating it. When a connection is borrowed from the pool, the pool will check to see if the `now - time-when-connected > maxAge` has been reached , and if so, it reconnects before borrow it. When a connection is returned to the pool, the pool will check to see if the `now - time-when-connected > maxAge` has been reached, and if so, it tries to reconnect. When a connection is idle and `timeBetweenEvictionRunsMillis` is greater than zero, the pool will periodically check to see if the `now - time-when-connected > maxAge` has been reached, and if so, it tries to reconnect. Setting `maxAge` to a value lower than `timeBetweenEvictionRunsMillis` will override it (so idle connection validation/cleaning will run more frequently). The default value is `0`, which implies that connections will be left open and no age check will be done upon borrowing from the pool, returning the connection to the pool or when checking idle connections.	N	0	
maxWait	The maximum number of milliseconds that the pool will wait (when there are no available connections) for a connection to be returned before throwing an exception. Default value is `30000` (30 seconds)	N	30000	
defaultTransactionIsolation	The default TransactionIsolation state of connections created by this pool. One of the following: (see javadoc) • `NONE` • `READ_COMMITTED` • `READ_UNCOMMITTED` • `REPEATABLE_READ` • `SERIALIZABLE` If not set, the method will not be called and it defaults to the JDBC driver.	N	''	
validationQuery	The SQL query that will be used to validate connections from this pool before returning them to the caller. If specified, this query does not have to return any data, it just can't throw a `SQLException`. The default value is `null`. If not specified, connections will be validation by the isValid() method. Example values are `SELECT 1`(mysql), `select 1 from dual`(oracle), `SELECT 1`(MS Sql Server)	N	''	
connectionProperties	The connection properties that will be sent to our JDBC driver when establishing new connections. Format of the string must be [propertyName=property;]* NOTE - The "user" and "password" properties will be passed explicitly, so they do not need to be included here. The default value is `null`.	N	''	
tomcat: mailsessions:	==========	N		==========
name	JNDI name under which you will look up preconfigured sessions	N	''	
from	From field used in the sent mail	N	''	
auth	Optional authentication method of the mail session (NB: if not sure, keep the default value)	N	'Container'	

type	Optional type of the mail session (NB: if not sure, keep the default value)	N	'javax.mail.Session'	
transportProtocol	Transport protocol used by mail server. 'smtp' is the defautl	N	smtp	
smtpHost	Host used as mail server	N		
smtpPort	Port used on the mail server	N		
smtpAuth	Indicates if authentication is active or not	N	false	
smtpUser	If authentication is active, user name to use for authentication	N	''	
smtpPassword	If authentication is active, password to use for authentication	N	''	Passwords must be defined in the vault. More information here : Tomcat in Cloud: Manage Secrets
tomcat: persistentStorages:	==========	N		==========
mountPath	Access path for the persistent storage	N	''	
size	Size **in gigabytes** (**GB**) of the persistent storage. **NB: the size cannot exceed 1024 GB !**	N	''	

accessMode	The different access modes for the persistent storages are:
	• ReadWriteOnce
	  ◦ the volume can be mounted as read-write by a single node. ReadWriteOnce access mode still can allow multiple pods to access the volume when the pods are running on the same node. For single pod access, please see ReadWriteOncePod.
	• ReadOnlyMany
	  ◦ the volume can be mounted as read-only by many nodes.
	• **ReadWriteMany**
	  ◦ the volume can be mounted as read-write by many nodes.
	• ReadWriteOncePod
	  ◦ the volume can be mounted as read-write by a single Pod. Use ReadWriteOncePod access mode if you want to ensure that only one pod across the whole cluster can read that PVC or write to it.
		N	'ReadWriteMany'	

storageClass	Possible values are for COP : ▪ **kube-repl-0d** ▪ kube-repl-7d ▪ kube-repl-35d ▪ kube-repl-1y Possible values for AWS : ▪ 0d	N	'kube-repl-0d'	Tomcat in Cloud: Persistent and Ephemeral Volumes
tomcat: ephemeralStorages:	==========	N		
mountPath	Access path for the ephemeral storage (**NB**: Some applications need additional storage but don't care whether that data is stored persistently across restarts. For example, caching services are often limited by memory size and can move infrequently used data into storage that is slower than memory with little impact on overall performance.)	N	''	
size	Size **in gigabytes** (**GB**) of the ephemeral storage. **NB: the size cannot exceed 5 GB !**	N	''	
jfm	==========			
ldapGroups		N		
persistentStorageGroups		N		
livenessProbe	==========			

probeEnabled	Indicates if probe is enabled or not			
probeType	• **TCP_SOCKET**: The probe attempts to open a connection to a TCP port on the container. A successful connection means the probe succeeds. • **HTTP_GET**: The command issues an HTTP request to a URL inside the container. A response that includes a status code in the range of 200-399 indicates that the request was successful. • **GRPC**: Uses the gRPC health checking protocol to issue a Kubernetes health check using gRPC. To use this type of liveness probe, your application must support the health checking protocol, and you must be using Kubernetes version 1.23 or later.			
port	Port on which the probe will run			
path	Path on which the probe will run (**HTTP_GET** only)			
initialDelaySeconds	Number of seconds after the container has started before startup, liveness or readiness probes are initiated. If a startup probe is defined, liveness and readiness probe delays do not begin until the startup probe has succeeded. If the value of `periodSeconds` is greater than `initialDelaySeconds` then the `initialDelaySeconds` will be ignored. Defaults to 0 seconds. Minimum value is 0.			
periodSeconds	How often (in seconds) to perform the probe. Default to 10 seconds. The minimum value is 1. While a container is not Ready, the `ReadinessProbe` may be executed at times other than the configured `periodSeconds` interval. This is to make the Pod ready faster			
timeoutSeconds	Number of seconds after which the probe times out. Defaults to 1 second. Minimum value is 1.			
readinessProbe	==========			
probeEnabled	Indicates if probe is enabled or not			
probeType	• **TCP_SOCKET**: The probe attempts to open a connection to a TCP port on the container. A successful connection means the probe succeeds. • **HTTP_GET**: The command issues an HTTP request to a URL inside the container. A response that includes a status code in the range of 200-399 indicates that the request was successful. • **GRPC**: Uses the gRPC health checking protocol to issue a Kubernetes health check using gRPC. To use this type of liveness probe, your application must support the health checking protocol, and you must be using Kubernetes version 1.23 or later.			
port	Port on which the probe will run			
path	Path on which the probe will run (**HTTP_GET** only)			
initialDelaySeconds	Number of seconds after the container has started before startup, liveness or readiness probes are initiated. If a startup probe is defined, liveness and readiness probe delays do not begin until the startup probe has succeeded. If the value of `periodSeconds` is greater than `initialDelaySeconds` then the `initialDelaySeconds` will be ignored. Defaults to 0 seconds. Minimum value is 0.			
periodSeconds	How often (in seconds) to perform the probe. Default to 10 seconds. The minimum value is 1. While a container is not Ready, the `ReadinessProbe` may be executed at times other than the configured `periodSeconds` interval. This is to make the Pod ready faster			
timeoutSeconds	Number of seconds after which the probe times out. Defaults to 1 second. Minimum value is 1.			
startupProbe	==========			
probeEnabled	Indicates if probe is enabled or not			
probeType	• **TCP_SOCKET**: The probe attempts to open a connection to a TCP port on the container. A successful connection means the probe succeeds. • **HTTP_GET**: The command issues an HTTP request to a URL inside the container. A response that includes a status code in the range of 200-399 indicates that the request was successful. • **GRPC**: Uses the gRPC health checking protocol to issue a Kubernetes health check using gRPC. To use this type of liveness probe, your application must support the health checking protocol, and you must be using Kubernetes version 1.23 or later.			
port	Port on which the probe will run			
path	Path on which the probe will run (**HTTP_GET** only)			
initialDelaySeconds	Number of seconds after the container has started before startup, liveness or readiness probes are initiated. If a startup probe is defined, liveness and readiness probe delays do not begin until the startup probe has succeeded. If the value of `periodSeconds` is greater than `initialDelaySeconds` then the `initialDelaySeconds` will be ignored. Defaults to 0 seconds. Minimum value is 0.			

periodSeconds	How often (in seconds) to perform the probe. Default to 10 seconds. The minimum value is 1. While a container is not Ready, the `ReadinessProbe` may be executed at times other than the configured `periodSeconds` interval. This is to make the Pod ready faster			
timeoutSeconds	Number of seconds after which the probe times out. Defaults to 1 second. Minimum value is 1.			

How to convert a string to **slug-format** ?

1. Convert to lower case
2. Trim any leading or trailing spaces
3. Remove any accents from characters (e.g. á, â, â, ã, ä, å, ç)
4. Replace any other special characters with spaces
5. Replace multiple spaces or dashes (hyphens) with a single dash

Can we see an example ?

```
applicationName: "tccop-api-app"
ingressName: "tccop-api"
ingressPath: "/"
replicas: 1
customerImageName: "tccop-api-app"
customerImageTag: "testimg216"
jvmVersion: "11"
tomcatVersion: "9.0"
artifactName: "sample.zip"

tomcat:
  jvm:
    heapMinSize: 1536
    heapMaxSize: 1536
    metaMinSize: 768
    metaMaxSize: 768
    directMaxSize: 2048
    debugGC: true
    debugSSL: false
    debugJPDA: false
    hideArgsInLog: false
    gcLogPathSeparated: true
    hideVersions: false
    gcUsed: 'parallel'
    environmentVariables: ''
  catalinaProperties:
    sharedLoader: ''
  context:
    cookieProcessor: 'org.apache.tomcat.util.http.Rfc6265CookieProcessor'
    cookieUseHttpOnly: true
    cachingAllowed: true
  connector:
    httpPort: 8080
    httpProxyName: ''
    httpProxyPort: ''
    httpUriEncoding: ''
    httpsPort: 8443
    httpsProxyName: ''
    httpsProxyPort: ''
    httpsUriEncoding: ''
    relaxedPathChars: ''
    relaxedQueryChars: ''
    compressionEnabled: false
    compressionMinSize: 2048
    compressibleMimeType: 'text/html,text/xml,text/plain,text/css,text/javascript,application/javascript,
application/json,application/xml'
    allowTrace: false
    xpoweredBy: false
```

```
    minSpareThreads: 25
    maxSpareThreads: 75
    maxPostSize: 2097152
    maxThreads: 160
    maxParameterCount: 10000
    maxHttpHeaderSize: 8192
    maxSwallowSize: 104857600
    enableLookups: false
    acceptCount: 100
    connectionTimeout: 60000
    keepAliveTimeout: 60000
    disableUploadTimeout: true
  host:
    deployXml: true
    copyXml: false
  log:
    logRotate: 30
    logMaxAge: 60
    logMaxSize: 200
  realm:
    addAllRolesMode: false
  valve:
    rewriteValveEnabled: false
    ecasAuthenticatorValveEnabled: false
    remoteIpValveEnabled: true
    accessLogValveEnabled: true
    accessLogValveExtraFields: ''
    errorReportValveEnabled: false
    errorReportValveError404: ''
  web:
    defaultSessionTimeout: 30
    jspServletEnablePooling: true
    requestCharacterEncoding: 'UTF-8'
    responseCharacterEncoding: 'UTF-8'
    hstsEnabled: false
    secureCookies: false
    securityFilterEnabled: false
    httpsRedirectionEnabled: false
  datasources:
    - name: jdbc/mytest
      provider: oracle
      url: jdbc:oracle:thin:@localhost:1521:sid
      username: "username"
      # For the password, a secret must be created in the Vault under kv_customer/<env>/clientParams
/SECRET_PWD_JDBC_MYTEST
      initialSize: 10
      minIdle: 10
      maxIdle: 10
      maxTotal: 10
    - name: jdbc/another_test-Datasource
      provider: oracle
      url: jdbc:oracle:thin:@localhost:1521:sid
      username: "username"
      # For the password, a secret must be created in the Vault under kv_customer/<env>/clientParams
/SECRET_PWD_ANOTHER_TEST_DATASOURCE
      initialSize: 10
      minIdle: 10
      maxIdle: 10
      maxTotal: 10
  mailsessions:
    - name: mail/session1
      host: smtpmail.cec.eu.int
      from: noreply@ec.europa.eu
    - name: mail/session2
      host: smtpmail.cec.eu.int
      from: noreply@ec.europa.eu
  persistentStorages:
    - mountPath: /share
      size: 1
      accessMode: ReadWriteMany
      storageClass: kube-repl-0d
```

```
  - mountPath: /share2
    size: 2
    accessMode: ReadWriteMany
    storageClass: kube-repl-0d
```

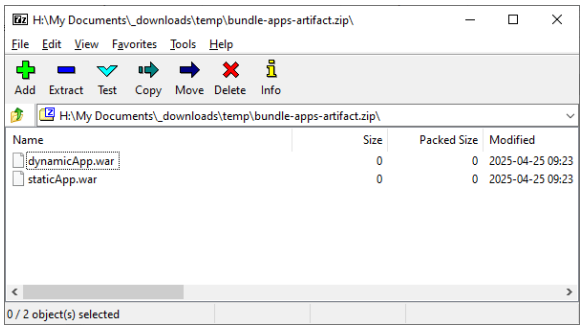
Tomcat in Cloud: Static Content Deployment

Homepage

Static content can be deployed as a **side application**.

It can be packaged as a standard **WAR file** within a **bundle ZIP archive**, and will be deployed in the same way as a dynamic application.

> A ZIP file can contain multiple WAR files, allowing you to colocate a static WAR alongside a dynamic WAR if needed.
>
> Below is an example of a bundle containing two applications, which can be uploaded to **Nexus** and deployed.



Related Links

- Deployment Chain
- Create Custom Deployment
- Pod Restart
- Maintenance Mode

Tomcat in Cloud: Management of Secrets

Homepage

Page Topics

Related Links

Deployment Chain

Create Custom Deployment

Pod Restart

Maintenance Mode

What is the purpose of the HashiCorpVault ?

Vault provides organizations with identity-based security to automatically authenticate and authorize access to secrets and other sensitive data.

How to access the HashiCorpVault ?

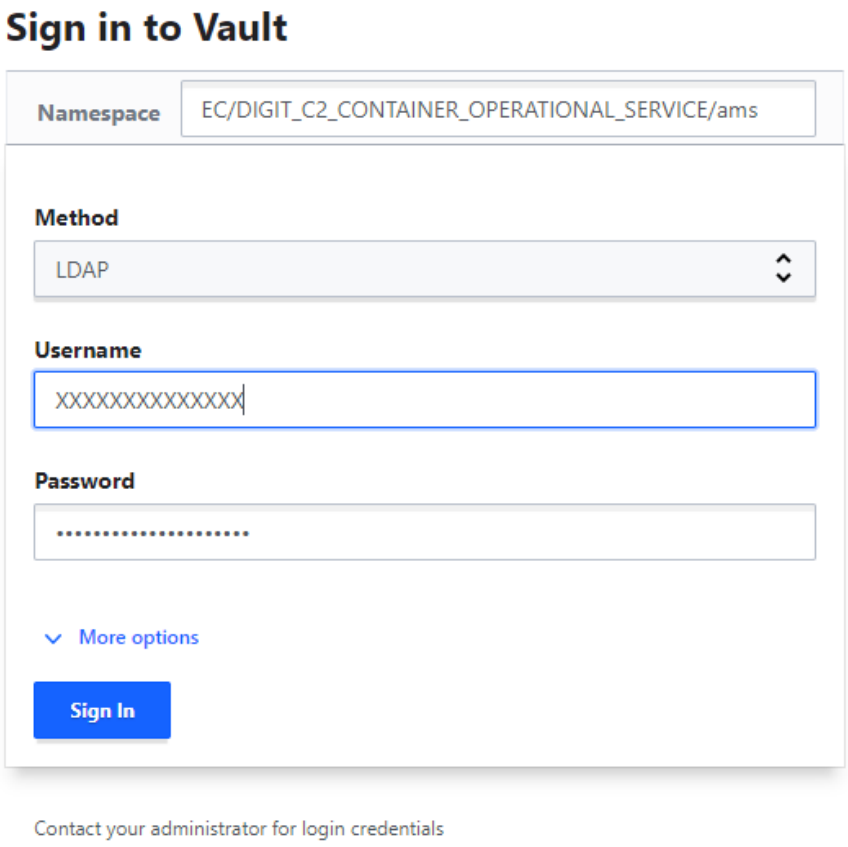
When the customer wants to consume the Tomcat service, DIGIT will create a dedicated namespace in HashiCorpVault

Access your namespace:

1. Sign in to the Vault.
   a. Enter the namespace URL provided to you in resolution email of the Tomcat service request.
   b. Select LDAP Method.
   c. Enter the relevant Username and password

Example:

```
Namespace:EC/DIGIT_C2_CONTAINER_OPERATIONAL_SERVICE/ams
Method:LDAP
Username:<YOUR_LDAP_USERNAME>
Password:<YOUR_LDAP_PASSWORD>
```



The customer secrets must be located in "**kv_customer**"

- "**kv_customer**" is only accessible in **READ/WRITE** mode to customer
- DIGIT ISHS Tomcat team doesn't have **WRITE** access
- DIGIT ISHS Tomcat team doesn't have **READ** access so we cannot see sensitive information
- DIGIT ISHS Tomcat can only **LIST** (useful to analyze typo error for example)

Define Client Parameters in the Vault

To use Client Parameters in the vault, the property **clientParamsKey** must be defined in the **config.yaml** file

- All parameters will have to be located under this path
- All secrets stored under this path will be automatically defined as environment variables for all pods
- Parameters can be different for each environment (dev/acc/prod/...). A specific path is used for each environment

For instance, if property is set to "clientParams", the following paths will be used

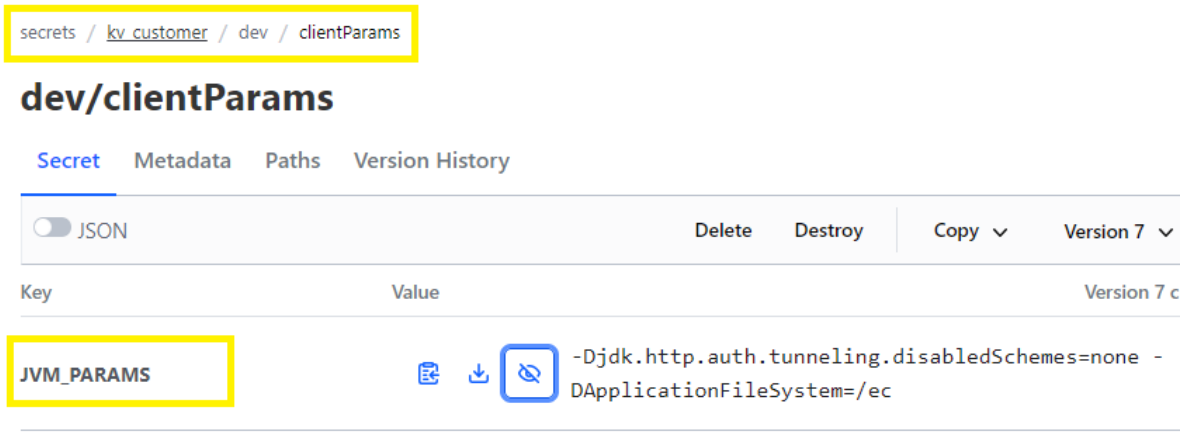
Environment	Path
dev	kv_customer/**dev**/clientParams/
acc	kv_customer/**acc**/clientParams/
prod	kv_customer/**prod**/clientParams/
any-env-name	kv_customer/**any-env-name**/clientParams/

The parts in bold must be identical to the Tomcat instance name (or IaC GIT repository branch you are using for deploying).

Example: If your Tomcat instance/IaC GIT branch is named "**dev-tomcat**", for example, you must make sure that after setting clientParamsKey: "clientParams" in your config.yaml file, you also put your secrets in the Hashicorp Vault here:

Environment	Path
dev	kv_customer/**dev-tomcat**/clientParams/

Here is an example screenshot from the HashiCorp Vault GUI



Define Client Files in the Vault

To use Client Files in the vault, the property **clientFilesKey** must be defined in the **config.yaml** file

- All parameters must be located under this path
- All files stored under this path will be automatically mounted as files for all pods
- Parameters can be different for each environment (dev/acc/prod/...). A specific path is used for each environment

For instance, if a property is set to "clientFiles", the following paths will be used.

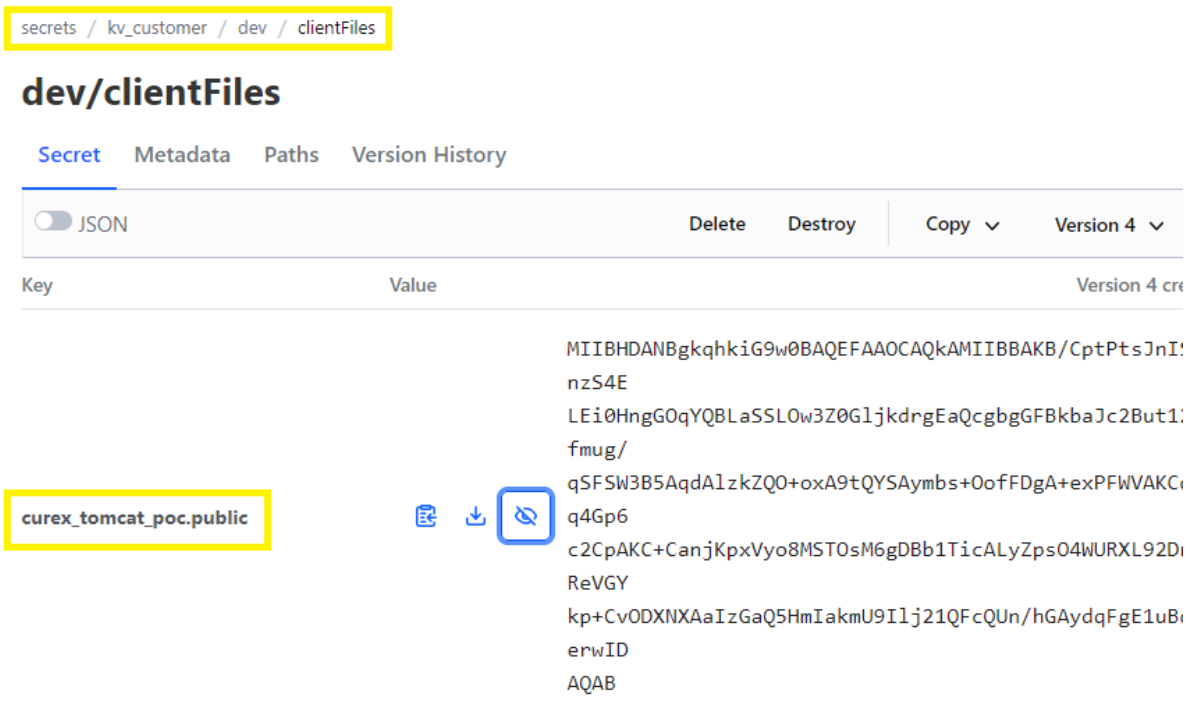
Environment	Path
dev	kv_customer/**dev**/clientFiles/
acc	kv_customer/**acc**/clientFiles/
prod	kv_customer/**prod**/clientFiles/
any-env-name	kv_customer/**any-env-name**/clientFiles/

The parts in bold must be identical to the Tomcat instance name (or IaC GIT repository branch you are using for deploying).

Example: If your Tomcat instance/IaC GIT branch is named "**dev-tomcat**", for example, you must make sure that after setting clientFilesKey: "clientFiles" in your config.yaml file, you also put your secrets in the Hashicorp Vault here:

Environment	Path
dev	kv_customer/**dev-tomcat**/clientFiles/

Here is an example screenshot from the HashiCorp Vault GUI



Are secrets synchronized and accessible from my application ?

- The HashiCorpVault secrets are synchronized with Tomcat secrets on Kubernetes cluster.
    - NB: secrets can be accessed directly in the application without using any API
        - **clientFiles** will be directly mounted in the image and are accessible by your application under **/usr/local/tomcat/<clientFilesKey>**
        - **clientParams** will be stored as OS **environment variables** that are accessible by your application
- If you change a secret in HashiCorpVault for a secret defined in a Tomcat model file (new password in case of DB migration for example), the **Tomcat instance must be at least restarted** (or a new image generated).

This is required to "refresh" the value of secret from the Tomcat instance point of view.

Tomcat in Cloud: Pod Restart

Homepage

Related Links

Deployment Chain

Create Custom Deployment

Management of Secrets

Maintenance Mode

> The following operations may be performed only if the Tomcat server configuration remains unchanged and no resources are added or removed. If any Tomcat resources need to be added, removed, or modified, it is recommended to **build a new image** instead.

How to execute a pod restart ?

A restart can be executed by deploying or redeploying an existing image or changing parameters in the config.yaml file.

The entire Tomcat instance is then restarted following these rules:

- Each Tomcat replica is restarted using a Zero-downtime strategy

> If you have at least 2 Tomcat replicas, your application will be accessible all the time (degradation of performance to be expected)

Tomcat in Cloud: Enable auto-scaling

Homepage

Related Links

Deployment Chain

Create Custom Deployment

Management of Secrets

Maintenance Mode

Enable auto-scaling

To enable autoscaling, the following properties has to be added in the **config.yaml** file.

Attribute	Values	Description
`autoscalingEnabled` (Optional)	true/**false**	• Start Tomcat instance(s) if the total Memory or CPU of pod is used at 70% during more than 30 seconds • The default value is **false** The number of instances depends number of **replicas** you configured • if replicas is lower or equals to **3** then **1** extra pod could be started • if replicas is greater than **3** and lower or equals than **8** then **2** extra pods could be started • if replicas is greater than **8** and lower or equals than **12** then **3** extra pods could be started • if replicas equals **13** then **2** extra pods could be started • if replicas equals **14** then **1** extra pod could be started When the load is coming back to normal situation (less 65% during more than 60 seconds) then extra pods will be shutdown at one ⚠ Pods are created and destroyed when activity is going up and down • Scaling up stabilization window: 30 secs • Scaling down stabilization window: 60 secs

Tomcat in Cloud: Maintenance Mode

Homepage

> The following operations can be executed **if and only if** the structure or resources of the Tomcat instances are not updated.
>
> Any change such as adding/removing/altering Tomcat resources implies the generation of new base images.

Related Links

- Deployment Chain
- Create Custom Deployment
- Management of Secrets
- Pod Restart

What is Maintenance Mode ?

Maintenance Mode can be used when the application needs to be stopped for administrative purposes.
In this mode, the application remains **deployed**, but it is **not started** and **cannot be accessed**.

For example, Maintenance Mode is appropriate when the application's database schema is being updated, during which access to the application must be temporarily disabled.

How to request Maintenance Mode ?

Maintenance Mode can be requested by changing the number of replicas to 0 in the parameters of the **config.yaml** file. All replicas will then be stopped.

> In order to exit Maintenance Mode, just add replicas as before in the **config.yaml** fil e.

Tomcat in Cloud: Workload Scheduler

Homepage

Related Links

Deployment Chain

Create Custom Deployment

Management of Secrets

Pod Restart

What is Workload Scheduler ?

Customers can specify their application to be running during which available plans which has an impact on the cost of their IS as their will be no consumed memory from their application.

Available modes are the following :

- 13/5 : stopped during the night (between 20:00 and 7:00) and during weekends
- 24/5 : stopped during the weekends
- **24/7 : always started (DEFAULT)**

How to request Workload Scheduler ?

It can be requested by changing the **workloadSchedulerMode** parameter of the **config.yaml** file.

eg:

workloadSchedulerMode: 13/5

Tomcat in Cloud: Defining Probes

Homepage

Page Topics

Introduction

Kubernetes uses Probes to monitor the health of containers running inside Pods. Probes are essential for ensuring that your containers are operating correctly and can handle traffic. They are a key part of Kubernetes' self-healing capabilities, enabling automatic restarts, load balancing, and health checks.

Types of Probes

Kubernetes has various types of probes:

- Startup probe
- Liveness probe
- Readiness probe

Startup probe

A startup probe **verifies if the application within a container is started**. This can be used to adopt liveness checks on slow starting containers or containers that require some time to start, avoiding them getting killed by the kubelet before they are up and running.

If such a probe is configured, it disables liveness and readiness checks until it succeeds.

This type of probe is only executed at startup, unlike liveness and readiness probes, which are run periodically.

Use case examples :

- A database that loads a huge amount of data in memory at startup.
- A Java application that compiles or loads a lot of dependencies at startup.

Liveness probe

A Liveness probe **determines if application is still running**. If not, then the container is restarted. For example, liveness probes could catch a deadlock when an application is running but unable to make progress.

If a container fails its liveness probe repeatedly, the kubelet restarts the container.

Liveness probes do not wait for readiness probes to succeed. If you want to wait before executing a liveness probe, you can either define initialDelaySeconds or use a startup probe.

Use case examples :

- An application that is locked in an endless loop.
- A suspended thread, preventing the application from responding to requests.
- A service that looses an esential connection and cannot continue to operate.

Readiness probe

A Readiness probe **determines when a container is ready to accept traffic**. This is useful when waiting for an application to perform time-consuming initial tasks that depend on its backing services; for example: establishing network connections, loading files, and warming caches. Readiness probes can also be useful later in the container's lifecycle, for example, when recovering from temporary faults or overloads.

If the readiness probe returns a failed state, Kubernetes removes the pod from all matching service endpoints.

Readiness probes run on the container during its whole lifecycle.

Use case examples :

- An API which must connect to a database before initializing.
- An application that loads configuration files before accepting requests.
- A mail service waiting external connections.

Type of actions triggered by probes

Kubernetes supports four types of action launched by probes:

1. httpGet
2. tcpSocket
3. gRPC
4. exec

Each type is suitable for different kinds of applications. Let's explore hands-on examples for each probe, explaining what they are doing and what exactly is being checked.

httpGet

What it is doing:

- The httpGet probe sends an HTTP GET request to a specific URL and port inside the container.
- The probe checks if the container responds with an HTTP status code in the 200–399 range, which indicates that the container is healthy.

What it is checking:

- It checks if the container is running and responding to HTTP requests correctly.
- It expects a valid response (e.g., a 200 OK status) from the specified URL path.

Example: For an NGINX-based application

```
livenessProbe:
  httpGet:
    path: /
    port: 80
  initialDelaySeconds: 5
  periodSeconds: 10
```

What it checks:

- This `httpGet` probe checks whether the NGINX server responds to requests on the `/` path.
- After an initial delay of 5 seconds, it sends an HTTP request every 10 seconds to check the health of the NGINX server.
- If the NGINX server returns a 200-series status code (indicating a successful response), the probe passes. Otherwise, it fails.

Best Practices:

- Health Endpoint: Set up a lightweight `/healthz` endpoint for health checks to avoid performance overhead.
- Time-sensitive: Adjust `timeoutSeconds` to ensure that probes do not wait too long before determining failure.

tcpSocket

What it is doing:

- The tcpSocket probe attempts to establish a TCP connection to the specified port inside the container.
- If the connection succeeds, the probe passes. If the connection fails (i.e., no service is listening on the port), the probe fails.

What it is checking:

- It checks whether the application is actively listening on the specified TCP port.
- Unlike `httpGet`, it does not check the content of the response, just whether the port is open and accepting connections.

Example: For an NGINX application using TCP probe

```
livenessProbe:
  tcpSocket:
    port: 80
  initialDelaySeconds: 5
  periodSeconds: 10
```

What it checks:

- The probe checks if the NGINX application is listening on port `80` by attempting to open a TCP connection to that port.
- If the port is open and the application is accepting connections, the probe will pass.
- If no service is listening on the port, the probe fails, triggering a container restart.

Best Practices:

- Use TCP probes for simple applications that don’t provide HTTP APIs but require ports to be open.
- Ensure the application is truly ready and not just listening on the port (i.e., avoid false positives).

grpc

What it is doing:

- The grpc probe sends a gRPC health-check request to a specified gRPC service endpoint.
- It uses the gRPC protocol to check if the service responds correctly to health-check messages.

What it is checking:

- It checks whether the gRPC service is running and responds with a success code (typically a “Serving” status in gRPC health checks).
- This probe is typically used in microservices built with gRPC.

Example: For an `etcd` service

```yaml
livenessProbe:
  grpc:
    port: 2379
  initialDelaySeconds: 10
```

What it checks:

- The probe checks whether the `etcd` service is responsive and healthy by connecting to the gRPC endpoint at port `2379`.
- If the gRPC service returns a healthy status, the probe passes. If it does not, the probe fails, and Kubernetes will attempt to restart the container.

Best Practices:

- Use for microservices or applications that expose health checks via gRPC.
- Ensure that the gRPC service correctly implements the health-check protocol for the probe to work.

exec

What it is doing:

- The `exec` probe runs a command inside the container.
- If the command exits with a status code of `0` (indicating success), the probe passes. If the command exits with any other code, the probe fails.

What it is checking:

- It checks if the command (e.g., `cat /tmp/healthy`) runs successfully. This can be used for more custom checks inside the container.

Example: For a simple BusyBox application

```yaml
livenessProbe:
  exec:
    command:
    - cat
    - /tmp/healthy
  initialDelaySeconds: 5
  periodSeconds: 10
```

What it checks:

- The `exec` probe checks whether the `cat /tmp/healthy` command inside the container can run successfully.
- The command attempts to read the `/tmp/healthy` file. If the file exists, the probe succeeds; if it doesn't, it fails.
- The `/tmp/healthy` file is created and deleted as part of a basic health-check mechanism in this example.

Best Practices:

- Keep `exec` commands lightweight to avoid unnecessary resource consumption.
- Use for custom health-check logic or simple file existence checks that don’t require HTTP or TCP endpoints.

Best Practices for Probes

- Start Simple: Begin with basic health checks and then expand as your application’s needs grow.
- Avoid Heavy Checks: Liveness probes should be lightweight to avoid consuming excessive CPU or memory.
- Set Sensible Intervals and Thresholds: Adjust `initialDelaySeconds`, `timeoutSeconds`, `failureThreshold`, and `periodSeconds` according to your application’s performance and stability characteristics.

- Monitor and Refine: Continuously monitor probe failures and adjust settings if the application is being restarted too frequently.

Each type of probe is intended for specific scenarios, so choose the one that best matches your application’s needs. Proper configuration and usage of liveness probes can significantly enhance the stability and availability of your applications on Kubernetes.

Probe Execution Order

Probes are executed in a specific order described by the following diagram :

Tomcat in Cloud: Monitoring & Reporting

Tomcat in Cloud: Monitoring

Homepage

Page Topics

Related Links

Configure Application Log Files

Monitoring as Code

Auditing & Retention

What are the types of monitoring ?

For example:

- Availability monitoring
- E2E monitoring
- Performance monitoring

How is monitoring performed ?

Monitoring is performed using Dynatrace, which allows application administrators to monitor their application performance and and end-user experience.

Dynatrace performs the following:

- Collects and stores information.
- Acts as a service provider for event and data management for notification infrastructure.
- Interprets data by using defined rules.

What is monitored?

Tomcat instance

- When a Tomcat instance is created, monitoring is put in place automatically as part of the instance creation process.
- When a Tomcat instance is decommissioned, monitoring is removed automatically as part of the instance removal process.

The Oracle databases (data layer) are monitored based on guidelines defined by the Oracle service.

What metrics are monitored by DIGIT ?

The following metrics are monitored proactively by DIGIT.

- CPU consumption
- Memory consumption

- Tomcat instance availability (processes running and responding)
- Application availability
- Data Source availability
- Memory usage

How can I monitor my environment ?

A dedicated Dynatrace monitoring dashboard for Tomcat environments in can be found here.

NB: Full documentation on how to use this Dynatrace Dashboard can be found here : Dynatrace: Tomcat

If you don't have access to the dashboard, your official needs to raise a ticket to DIGIT ISHS MONITORING team to add your LDAP group to your DG's Management Zone in Dynatrace. Once permission is granted, members of the LDAP group would be able to access the dashboard.

How is monitoring being used ?

- Errors on the above metrics are detected and handled re-actively in collaboration with the customer.
- DIGIT runs an environment health check daily to verify the integrity of the different configuration settings:
    - check that every production environment host is monitored.
- In case of problems, availability monitoring always leads to alerts adapted to the severity of the problem detected:
    - an SMS to the on-call Administrator in case of an important problem
    - an e-mail in case of a less urgent problem

How is E2E monitoring used ?

- End-to-End monitoring (E2E) monitors availability and response times from the point of view of the end user and detects potential problems:
    - Bottlenecks in the system

Tomcat in Cloud: Configure Application Log Files

Homepage

Page Topics

Related Links

Monitoring

Monitoring as Code

Auditing & Retention

Using Application Log Files

Direct access to logs is not possible. Application logs are forwarded to and stored by the Splunk service.

Log Files	Description
Application log sentences	• Application log sentences **cannot be** written on STDOUT or the Console appender
Log file generation	• The log file generated must be located in directory: "**/ec/logs**" ○ **NB:** catalina.out and other Tomcat log files are also located in **/ec/logs** directory

Log File Naming	• The filename must be "**customer_application*.log**".
	  ○ Valid examples:
	  ■ /ec/logs/customer_application_**my_log_name**.log
	  ■ /ec/logs/customer_application_**XXXXX**.log
	  ■ ...
	  ○ **All files not following this naming will be ignored !**
	• Logfile usage practices:
	  ○ Use generated filenames
	  ○ Limit logfile size (**1MB**)
	  ○ Do not store backup files in /ec/logs

Log File Format	• The format of lines must be as follows : "**[<LEVEL>] [<TIMESTAMP>] [<BLOCK_1>] [<BLOCK_N>]%n**"
	  ○ Each block must be between "**[]**"
	  ○ The first block must contain a maximum of **7** characters in length
	  ○ The second block must be the timestamp with the format "**yyyy/mm/dd HH:MM:SS**"
	  ○ The line must end with a carriage return
	  ○ **All files not following this format will be ignored !**

⚠ For example:

- Log4J2 : "[%-5level] [%d{yyyy/MM/dd HH:mm:ss}] [%F:%L] [%msg]%n"
- Log4J : "[%-5p] [%d{yyyy/MM/dd HH:mm:ss}] [%F:%L] [%msg]%n"

How do I change the log retention period ?

The **logRetentionPeriod** parameter enables to change retention period for logs which has an impact on the cost of the IS

Parameter values:

- **xs** (1 month)
- **small** (3 months)
- **medium** (6 months)
- **large** (12 months)

What is the list of indexes you can use to read Tomcat logs in Splunk?

Type	Indexe	Example
Tomcat application log file	index=<DG_NAME>_<INDEX_RETENTION>_ops sourcetype="tomcat:runtime:log" source="/ec/logs/*"	index=digit_c_ops sourcetype="tomcat:runtime:log" source="/ec/logs/*" host="*automation*"
STDOUT of Tomcat application	index=<DG_NAME>_<INDEX_RETENTION>_ops sourcetype="tomcat:runtime:out" source="/ec/logs/*"	index=digit_c_ops sourcetype="tomcat:runtime:log" source="/ec/logs/*"
HTTP access log files	index=<DG_NAME>_<INDEX_RETENTION>_ops sourcetype="tomcat:runtime:access" source="/ec/logs/*"	index=digit_c_ops sourcetype="tomcat:runtime:access" source="/ec/logs/*"
Application log files	index=<DG_NAME>_<INDEX_RETENTION>_ops sourcetype="${DG_NAME}:${IS_NAME}:${INSTANCE_NAME}" source="/ec/logs/*"	index=digit_c_ops sourcetype="digit_c::msp2-design-tests:automation" source="/ec/logs/*"

⚠

- The DG name is in lowercase
- The IS name is the slugify representation of IS
- The INSTANCE_NAME is the value of "**name**" property defined in yaml files
- Depending on the value of `splunk_retention_log`, the value for **INDEX_RETENTION** must be set:
  - `xs` : the index to use will be **<DG_NAME>_xs_ops**
  - `small` : the index to use will be **<DG_NAME>_ops** (no change)
  - `medium` : the index to use will be **<DG_NAME>_medium_ops**
  - `large` : the index to use will be **<DG_NAME>_large_ops**
- **NB : If the filenames and locations of the logs are not respected, data will not be sent to Splunk**

How to find more information about Splunk ?

You can find more information about Log Correlation service and how to request access to Splunk by consulting the following links:

- Service Catalog: Log Correlation Service
- Technical User Guide : How To Use Splunk

Tomcat in Cloud: Auditing & Retention

Homepage

Page Topics	Related Links
- Why is data audited ? - What data is audited ? - How long does DIGIT store data ?	Monitoring Configure Application Log Files Monitoring as Code

Why is data audited ?

Auditing is performed for DIGIT internal purposes.

What data is audited ?

All administrator actions are audited.

How long does DIGIT store data ?

Tomcat and application logs are pushed in Splunk where log retention by default is 3 months irrespective of environment type.

Tomcat in Cloud: Monitoring as Code

Homepage

What is Monitoring as Code (Monaco)?

> Before requesting Monitoring as Code for an end to end synthetic monitoring probe, make sure that your RPM has been successfully created and is fully operational.

Monaco (Monitoring as code) is a CLI tool that automates deployment of Dynatrace Monitoring Configuration to one or multiple Dynatrace environments. Using Monaco, only Synthetic HTTP monitors can be created.

Refer to the Infrastructure as Code (IaC) User Guide for information about setting up Monitoring as Code.

Related Links

- Monitoring
- Configure Application Log Files
- Auditing & Retention

Tomcat in Cloud: Git Feedback Notifications

Homepage

Page Topics

Related Links

Monitoring

Configure Application Log Files

Auditing & Retention

Monitoring as Code

All notifications will be available in the "tomcat-feedback" branch of your git repository

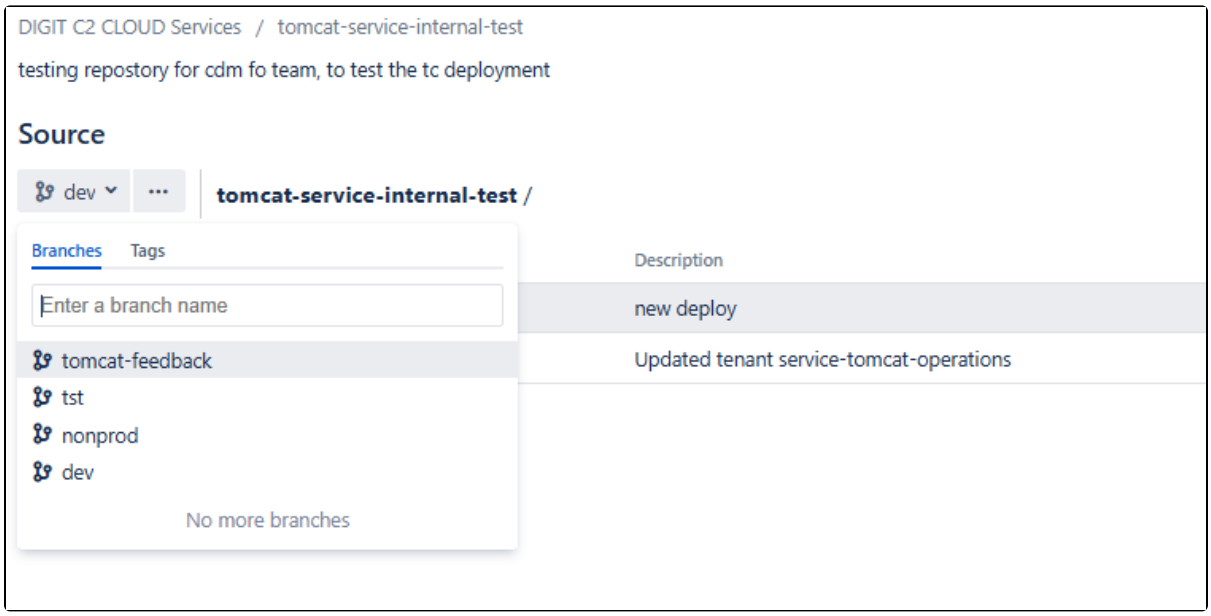
Mail notifications are also sent to FMB mailboxes defined during provisioning

Selection of the branch in your Git repository

All notification information is available in the **`tomcat-feedback`** branch.
You must create this branch in your repository and grant **write access** to the technical account **`emfortccdm`**.

Feedback will be automatically generated and pushed to this branch.

The full history of messages is retained, allowing you to browse and retrieve past notifications at any time.



Segregated Environment Files

Each environment has a specific file that contains all information.

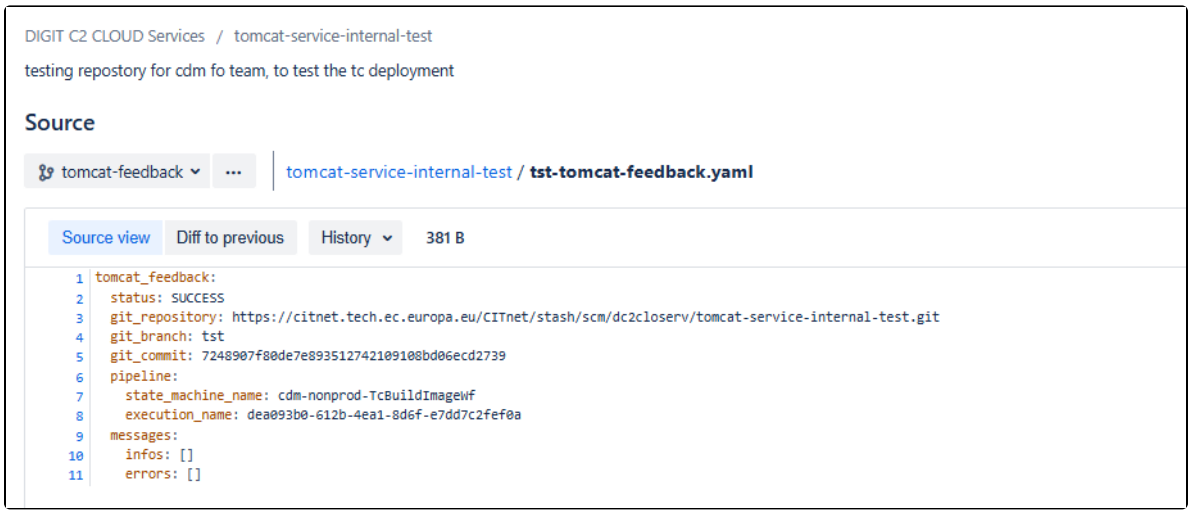
> ⚠ History is stored and you can retrieve previous messages/commits in the history.

For instance, in the following screenshot, the file for TST environment is available:

SUCCESS Example Feedback Messages

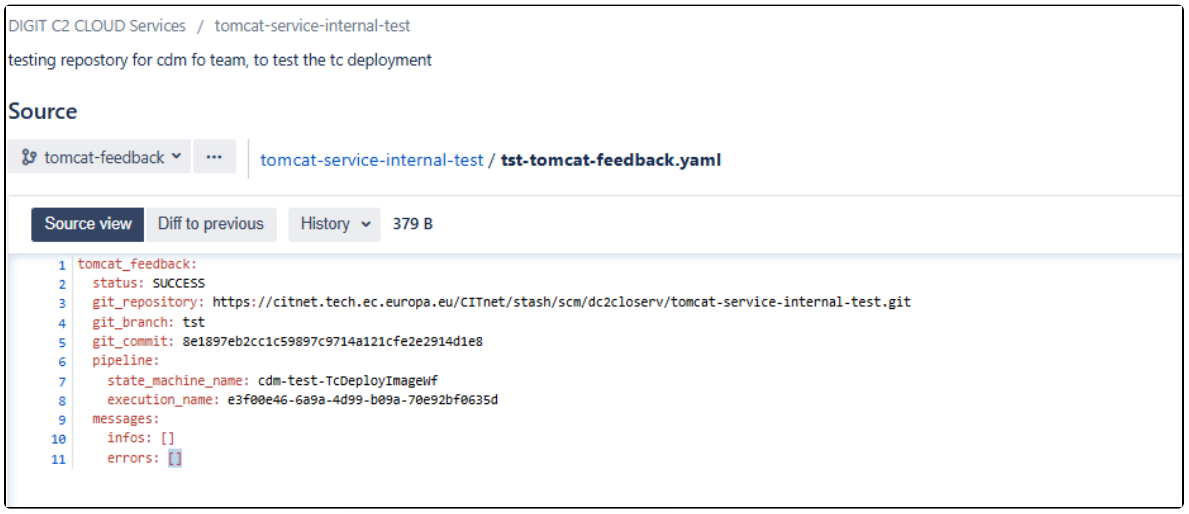
Successful Build

After a successful Build, this update can be seen in the feedback branch (with no error explanation given that everything was OK) :



Successful Deploy

After a successful deployment, this update can be seen in the feedback branch (with no error explanation given that everything was OK) :



ERROR Example Feedback Messages

Invalid Input Parameters

If some parameters are wrong or missing, this message can be seen in the feedback branch (with an error message explaining what is wrong) :

DIGIT C2 CLOUD Services / tomcat-service-internal-test

testing repostory for cdm fo team, to test the tc deployment

Source

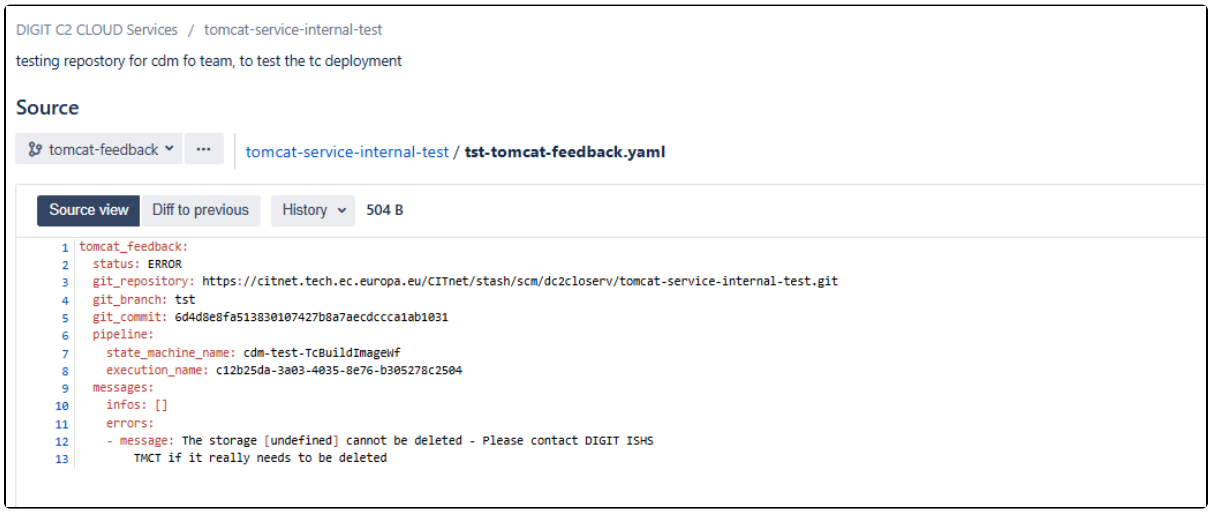
tomcat-feedback ··· | tomcat-service-internal-test / **tst-tomcat-feedback.yaml**

Source view | Diff to previous | History | 461 B

```yaml
tomcat_feedback:
  status: ERROR
  git_repository: https://citnet.tech.ec.europa.eu/CITnet/stash/scm/dc2closerv/tomcat-service-internal-test.git
  git_branch: tst
  git_commit: 96e26d5a76a72b6737f1a8df7956e126e445752d
  pipeline:
    state_machine_name: cdm-nonprod-TcBuildImageWf
    execution_name: efa06866-393a-4cd0-ba05-a256315cd252
  messages:
    infos: []
    errors:
    - message: 'Validation error: Unrecognized key(s) in object: ''managerEnabled'''
```

Failed Build because of Storage Deletion forbidden

After a failed Build, this message can be seen in the feedback branch (with an error message explaining what is wrong) :



Tomcat in Cloud: FAQ & Supporting Links

- Tomcat in Cloud: FAQ
- Tomcat in Cloud: Support & Useful Links
- Tomcat: Glossary

Tomcat in Cloud: FAQ

Homepage

Page Topics

Related Links

Support & Useful Links

SNC Ready

Sensitive Non-Classified (SNC) refers to information that is not formally classified as *EU Confidential*, *Secret*, or *Top Secret*, but still requires enhanced protection.

Full SNC compliance requires both the use of SNC Ready services and the implementation of application-level data protection.

Services designated as SNC Ready implement a number of security controls, defined by DIGIT.

⚠ SNC Ready is only available if chosen when you request the service. If you do not select this option, the protection of your data will not be SNC compliant.

Refer to the SNC User Guide for information.

Can this service be used with API Gateway?

You can use the API Gateway with PaaS services. Since RPM usage is no longer supported, you should rely on Kubernetes Ingress within the K8s cluster as your API endpoint.

Once your application is deployed, please open a ticket with the DIGIT ISHS TOMCAT team to get the full Ingress path details.

Action list:

- Register your project with the API Gateway team: Register your project in API GTW.
- Enable the following Security Groups in your flow.yaml.

DIGIT Services Security group rule identifier	Description	Service Name
ec_apigw-in-accext	flow coverage / EC APIGATEWAY Acceptance External	tc-service
ec_apigw-in-accint	flow coverage / EC APIGATEWAY Acceptance Internal	tc-service
ec_apigw-in-prodext	flow coverage / EC APIGATEWAY Production External	tc-service
ec_apigw-in-prodint	flow coverage / EC APIGATEWAY Production Internal	tc-service

- Deploy your application.
- Contact us to obtain the full ingress path to use as your API endpoint.

Refer to the Flow-as-Code (FaC) Security Groups Catalogue for more information.

What are the Tomcat components ?

Any type of application developed under the open-source implementation is eligible to be deployed on Tomcat.

- Java Servlet
- Java Server Pages
- Java Expression Language
- Java WebSocket specification

ⓘ Bundled with an optional Tomcat Native component, Tomcat can use Apache Portable Runtime (APR) to provide improved performance and scalability through tighter integration with native server implementations used as the hosting platform for the service.

What are the knowledge pre-requisites ?

You must have a good knowledge of:

- J2EE technology
- Tomcat architecture
- Experience of Application:
  - Development;
  - Tuning;
  - Deployment;
  - Back-up (Code, Configuration files and dependencies)
  - Debugging.

Hosting Environments

What is a Hosting Environment ?

- A Hosting Environment (Previously known as a Business Group) is used to logically organise, configure and allocate a set of services and resources to a set of users.
- It is also a level of granularity for chargeback and maintenances. It is the equivalent of an "environment" in the DIGIT datacentre.

What environment types are applicable in a Hosting Environment ?

- Production (PRD)
- Non-Production (NON-PRD). The Non-Production environment is updated first, in case of maintenance.

> In the case of a Critical IS, all the Hosting Environments created in the Production environment will be identified as **Critical**.

What is DIGIT's recommendation in relation to environments ?

DIGIT recommends using a Non-Production environment to validate changes before applying them in Production.

> The name of the branch in the Git repository **must** be identical to the Hosting Environment name.

How to create a Hosting Environment ?

The Hosting Environment will be created by DIGIT onboarding team following your request to be onboarded.

How is a Hosting Environment organized ?

- A Hosting Environment belongs to one Information System (IS) only.
- All allocated resources of a specific Hosting Environment are held within one environment's type (PRD or NON-PRD).

How should a Hosting Environment be named ?

- The Hosting Environment must have the same name as the branch in your Git repository.
- DIGIT recommends a clear naming convention, to help the customer understand the following:
  - Function of the Hosting Environment
  - Meaning and usage of any given resource and to avoid errors.
- Name examples:
  - *production-ec*: Production for Commission users
  - *production-public*: Production for the Public
  - *integration-nightly*: CI environment for nightly builds
  - *training-gold*: Training environment with a Gold service level
  - etc

Upgrades

What is the frequency of upgrades ?

- DIGIT organises and communicates planned system upgrades in advance to the customer.
- These are normally performed four times per year.

Why is a pod upgrade performed ?

- Higher version improves performance and/or functionality compared to the current version.
- Higher version fixes a problem in the current version.
- Higher version reduces security vulnerability.
- The current version is no longer supported.
- A request from the customer is received to upgrade.

How to request an upgrade ?

Upgrade can be requested by updating the config.yaml file.

See here for more explanation : Tomcat in Cloud: Create Custom Deployment

How does DIGIT perform an upgrade ?

Phase	Description
Plan and Prepare	DIGIT plans the upgrade and ensures the defined supported standards are not affected by compatibility issues.
DIGIT Validation	All non-production environments are upgraded to the target version by DIGIT.
Customer Validation	The customer validates the environments.
Production Environment	The PRD environment is upgraded (Performed by DIGIT)

Maintenance

What are the maintenance activities ?

- Minor releases
- Security patches

When are maintenance activities performed ?

Quarterly maintenance windows automatically run in the background to ensure an optimal service.

Docker Images

How are Docker images saved ?

- Docker images are located inside the internal Docker registry and this instance is backed-up.
- The Tomcat service runs on a **stateless** container, which means Tomcat instance data is not saved.
- The customer must save any required instance data to an Oracle database (cloud or legacy) or to a persistent volume

How many Docker images can be retrieved from docker repository ?

15 last images are saved.

Why is a restore performed ?

This is done to return to a previous stable state when something is wrong

What can be restored ?

The previous 15 Docker images are saved and thus can be retrieved.

Decommissioning

What is decommission ?

- Decommissioning removes all allocated resources.
- The Nexus repository is also decommissioned.

How to request a decommission ?

1. Make an Non-Standard request in JASSPR
    a. Complete the fields in the form.
    b. Submit the request.
    c. Monitor the request status on the My requests list.
2. For information and examples how to use JASSPR - consult the JASSPR User Guide

Tomcat in Cloud: Support & Useful Links

Homepage

Page Topics

Related Links
FAQ

Support

Link	Description
Tomcat Service Catalogue	Tomcat Service Catalogue Page
DIGIT Price List	DIGIT Price List for Infrastructure Services
Subscription to the Service Form	Requesting subscription to the service
Support Enquiry Form	Requesting support from the Tomcat service Centre of Excellence
Incident Management	Resolve issues efficiently and in a way that minimizes the impact from incidents
Making requests	What requests can you make?
JASSPR Tutorial Videos	• Incidents • Request for Change (RFC) • Non Standard Request for Change (RFC) • Request for Information (RFI) • Request for Service (RFC)
JASSPR	**Access to JASSPR** • Access from EC networks (European Commission and Executive Agencies): https://intragate.ec.europa.eu/jasspr • Access from TESTA network (EU Institutions and Agencies): https://webgate.ec.testa.eu/jasspr
JASSPR User Guide	Homepage for JASSPR User Guide
EU Login	Authenticate using EULogin

Packaged Libraries

Component	Link
Apache Tomcat	https://tomcat.apache.org/
Apache TomEE	https://tomee.apache.org/
OpenSSL	https://openssl-library.org/source/
OpenLDAP	https://www.openldap.org/
Oracle JDBC	https://www.oracle.com/europe/database/technologies/appdev/jdbc-downloads.html
PostgreSQL JDBC	https://jdbc.postgresql.org/
MSSQL JDBC	https://learn.microsoft.com/en-us/sql/connect/jdbc/download-microsoft-jdbc-driver-for-sql-server?view=sql-server-ver17
MySQL JDBC	https://dev.mysql.com/downloads/connector/j/
Java SDK releases	https://www.java.com/releases/
Reload4J	https://reload4j.qos.ch/

Roles & Responsibilities

DIGIT Responsibilities

DIGIT RESPONSIBILITIES	DESCRIPTION
INFRASTRUCTURE	
Datacentre	• Protecting datacentre and Kubernetes infrastructure that runs on premise cloud services. • DIGIT.B4 offers Tomcat servers on TKGI infrastructure (managed by DIGIT.C3) for each environment (Production & Non-Production) • Provide support and advice to the customer, when an infrastructure provisioning request is made.

Kubernetes	• Provisioning and maintaining Kubernetes infrastructure. For example (Non-exhaustive list):
	  ○ Cluster Namespace creation and size configuration
	  ○ Cluster ingress creation
	  ○ Cluster persistent storage
	  ○ ArgoCD Vault access configuration
	  ○ Backup of the cluster state
	  ○ Middleware container platform GIT content maintenance

CDM	For example (Non-exhaustive list): • Updating the CDM Inventory. • Offering advice when creating the CMDB Hosting Environment. • Creating the RPM miniArgoCD. • Maintenance of the DIGIT Container deployment store. • Maintenance of the DIGIT Reference configuration store. • Support and maintenance of CDM pipelines.

UPGRADES, MIGRATIONS, PATCHING & MAINTENANCE	
Upgrades	DIGIT.B4 organizes and plans patchings with the customer. These are normally performed quarterly.
Migration	• Transferring data from one system to another when upgrading the hosting service or its dependencies. • Data migration is usually performed when a higher version of software/hardware is introduced. • Coordinated between DIGIT B4 and the customer.
Patching & maintenance	**Patching** • Installation of security or technical patches. **Maintenance** • Quarterly maintenance windows automatically run in the background to ensure an optimal service. • Performing maintenance on cluster infrastructure. • Tomcat: Base image maintenance
BACKUPS, PATCHING	
Backup & Restore	The Tomcat service **is NOT** responsible for: • Creating backups • Scheduling backup and restore
Persistent storage	There is NO persistent storage backup. • DIGIT provide only Trident snapshots according to the trident StorageClass. • If the PVC gets deleted then data is lost.
Patching & maintenance	The Tomcat team and Kubernetes Infrastructure teams perform regular patching of both Production and Non-Production environments.
SUPPORT & ADMINISTRATION	
Providing the Required Tomcat Components	1. 1 JDK running the Tomcat server 2. 1 Tomcat software namespace for hosting the customer applications 3. Monitoring of the Tomcat instance from Dynatrace
Support	1. 24/7 support for Production service instances 2. 12/5 support for Non-Production service instances 3. Granting access to a release repository via LDAP groups
Fulfilling customer requests and offering support	**For example**: 1. Subscribe to this service. 2. Create a Cloud On-Premise Hosting Environment. 3. Provide customer access to namespaces. 4. Providing the customer with the means to logically organize, configure and allocate resources using Hosting Environments.

Management of Tomcat secrets	• Vault configuration, for example, configuration of: ○ Tomcat secrets • Customer namespace creation
DEPLOYMENT & SECURITY	
Deployment and Tooling	**Deployments** • Uploading a Tomcat release bundle to the Nexus artifact repository. • Tomcat: deployment support and issues troubleshooting **Repositories** • Creation and maintenance of the GIT repository • Grant access to Nexus artifact repository via LDAP groups. • Release a Nexus repository creation for a specific application.
Security	• Applications rely on EULogin for Single Sign-On. • Application developers can use EULogin as a basis for providing authentication and authorization to parts of their applications. • There is a specific client developed for Tomcat available here. • Performing image security scanning
DATA MANAGEMENT & MONITORING	
Auditing & Retention	• Enabling access to log management using Splunk that enables you to see the application and access logs. • All administrator actions are audited.
Monitoring Namespaces	• Monitoring of a Tomcat namespace • When a Tomcat namespace is created, monitoring is put in place automatically as part of the instance creation process. ⓘ Monitoring is performed using Dynatrace, which is designed to monitor and optimize application performance and development, IT infrastructure, and user experience.

Customer Responsibilities

CUSTOMER RESPONSIBILITIES	DESCRIPTION
INFRASTRUCTURE	

Requesting	Requesting infrastructure provisioning The most common customer requests to DIGIT are: 1. Create a Cloud On-Premise Hosting Environment. 2. Provide customer access to an instance(s). 3. Decommission this service. ⓘ • The customer is responsible for their application data after decommissioning. • The customer must request decommissioning.
CDM Creation & Configuration	For example: • Create a Hosting Environment • Creation of RPM ingress • Configuration of IaC customer GIT • Configuration of IaC Customer GIT: WebHook **Hosting Environment(s)** The customer is responsible for assigning the following to a Hosting Environment: • Environment you want your Hosting Environment to be part of (Prod data, or non-prod data) • A name to a Hosting Environment • NB: Strings must be slugified (all lowercases, no exotic characters, no spaces)
CUSTOMER APPLICATIONS	
Performance	• Issues related to application performance and generic support. • Modifying application data and associated debugging • Application data management • Application debugging is not part of DIGIT responsibilities. ⓘ Refer to the **Support** section of Information System Hosting Services to find out how to get help and the point of contact. The Hosting Services follow a common model to manage incidents. Consult Incident Management for more information.
Upgrades & Migration	• Working with DIGIT to organize and validate upgrades and migration. • Migration from one Tomcat version to another is handled by the customer.

Support	Liaise and work with DIGIT during (Non-exhaustive list): • Tomcat: deployment support and issues troubleshooting • Cluster provisioning (only on COP) **Management of customer secrets** Vault configuration: • KV Customer secrets configuration • Secrets lifecycle
SECURITY	

Application Security	The customer is responsible for their specific application(s). For example:
	• Safeguarding the history of customer application artifacts:
	  ○ Application modules
	  ○ Application configuration files

DEPLOYMENT	
Application deployment	• Deployment of application code/artifacts. • Tomcat On-Demand scheduled deployments
New Versions	• Testing the application after deployment. • IaC Customer GIT repo creation • IaC Customer GIT repo content maintenance
STRESS TESTING	
Monaco	Monaco (monitoring) configuration. Refer to the Monitoring as a code section on the 'Service Features & Security' tab.
IS Stress Testing	• It is important to estimate the impact a new application will have when it goes in production and to verify that there are no known issues. • However a stress test phase is optional and at the discretion of the customer.
Testing an Application	If a stress test is requested, a specific Hosting Environment can be created for the customer to test their application(s) in collaboration with the DIGIT Test Centre, responsible for the tests.
RISK ASSESSMENT	
Performing an IS Risk Assessment	• It is the responsibility of the customer to perform a risk assessment of the Information System (IS) and to select suitable services. • DIGIT provides support to the customer in order to help them make optimal choices and identify which services are best suited for an information system.

Tomcat: Glossary

Homepage

Related Links
Support & Useful Links

Term (Abbreviation)	Description
AWS	Amazon Web Services (Public Cloud)
CoP	Cloud on Premises
Dynatrace	Dynatrace enables monitoring of your entire infrastructure including your hosts, processes, and network.
EU Login	EU Login is the European Commission's user authentication service. It enables authorised users to access a wide range of Commission web services, using a single email address and password.
Flow-as-Code	Infrastructure as Code (IaC) tool that automates the opening and closing of network flows, which define communication rules between applications and services.
GitOps	GitOps is an operational framework that takes DevOps best practices used for application development such as: • version control; • collaboration; • compliance; • CI/CD. and applies them to infrastructure automation.
HashiCorp Vault	Vault provides organizations with identity-based security to automatically authenticate and authorize access to secrets and other sensitive data
Hosting environment	• A Hosting Environment (Previously known as a Business Group) is used to logically organise, configure and allocate a set of services and resources to a set of users. • It is also a level of granularity for chargeback and maintenances. It is the equivalent of an "environment" in the DIGIT datacentre.
Information System	An Information System (IS) is a computer-based tool that helps people to transform data into information that supports the business.
Integration	TomEE integrates easily with Tomcat, reducing the effort and time required to add additional libraries.
K8S	Kubernetes
LDAP	Light-weight Directory Access Protocol
Monitoring as a Code	CLI tool that automates deployment of Dynatrace Monitoring Configuration to one or multiple Dynatrace environments.
Nexus	Sonatype Nexus Repository is a software repository manager, available under both an open-source license and a proprietary license.
Portability	TomEE, being a full and official JavaEE implementation, simplifies the process of porting applications from WebLogic.

PrivX	Passwordless Privileged Access Management (PAM) solution that enables just-in-time, role-based access to critical systems, enhancing security and eliminating static credentials.
RPMaC	Infrastructure as Code (IaC) tool designed to automate the creation, update, and deletion of Reverse Proxy Mappings (RPMs).
S3	S3 Storage
TomEE	TomEE stands for Tomcat + Java Enterprise Edition. It is an all-Apache Jakarta EE certified application server that extends Apache Tomcat.
TomEE Architecture	• Apache TomEE is built by starting with a vanilla Apache Tomcat zip file, adding necessary jars, and then zipping it up. • The result is Tomcat with EE features such as ActiveMQ, Apache CXF, OpenWebBeans, and OpenJPA.
Webhook	Data and executable commands sent from one app to another over HTTP instead of through the command line in your computer, formatted in XML, JSON, or form-encoded serialization

Tomcat in Cloud: TLDR

Homepage

Page Topics

Related Links

Support & Useful Links

How to define OS environment variables ?

Environment variables can be defined in the vault under the following path: kv_customer/<environment>/<clientParamsKey>/...

For instance, if you define the secret kv_customer/dev/clientParams/**SPRING_ACTIVE_PROFILES**, the environment variable **SPRING_ACTIVE_PROFILES** will be defined at OS level in the pods

Info

How to pass java parameters to JVM at startup ?

A specific environment variable must be created with the name **JVM_PARAMS**

NB: see previous paragraph for environment variables creation

I don't have a GitLab account. How can I get it?

Refer to this page in the SDLC-GitLab User Guide

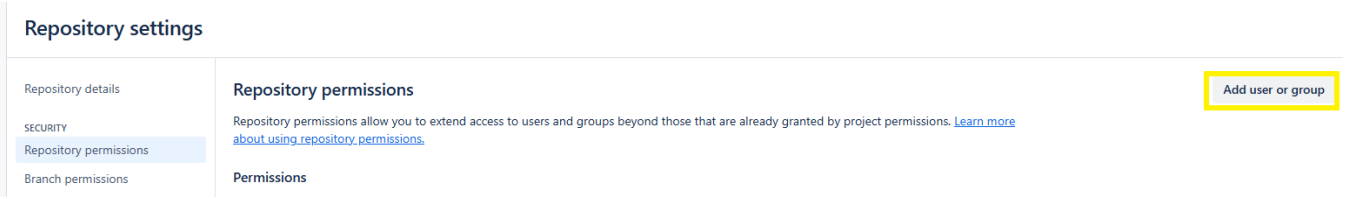
Tomcat in Cloud: Excerpts

BitBucket Permissions

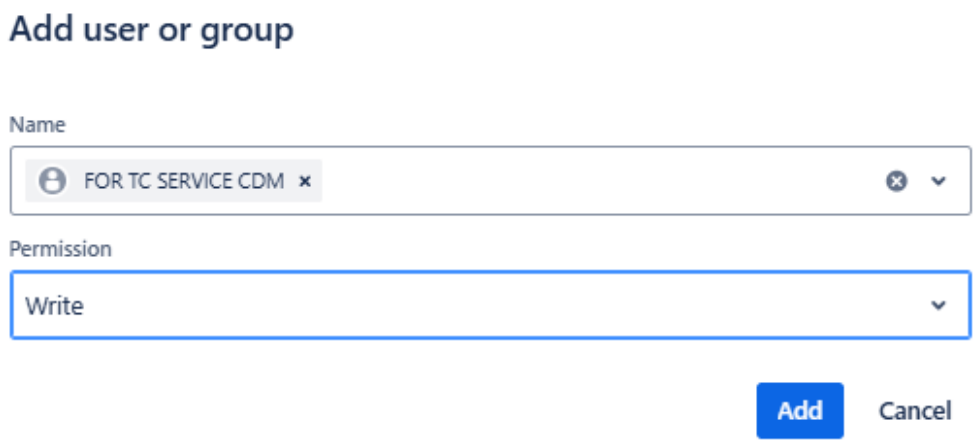
A **Tomcat functional user**, provided during onboarding, must be configured in the Git repository. This access is essential for enabling feedback to be triggered and delivered to the customer.

Perform the following steps in **Bitbucket** to add the functional user.

1. Click on the **Repository settings** button on the lower-left corner.
2. Select the **Repository permissions** entry.
3. Click on **Add user or group** button.



4. Add "FOR TC SERVICE CDM" with write access



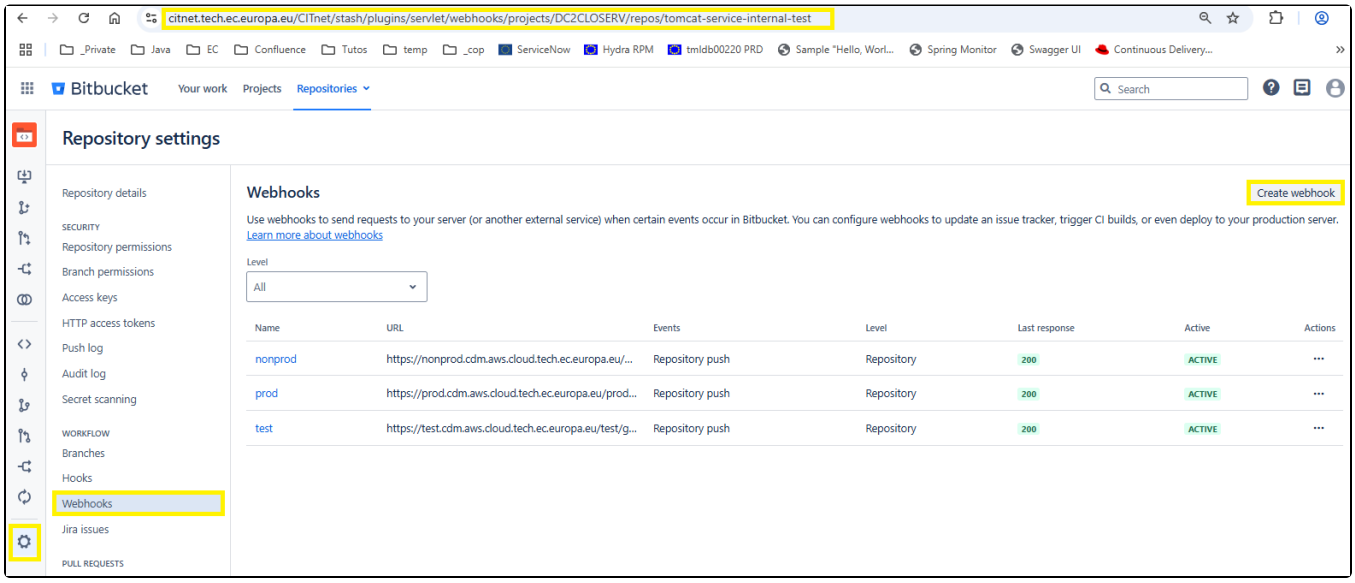
BitBucket Webhook

A webhook must first be configured and tested in the Git repository provided during onboarding. This webhook is essential for triggering pipeline executions and will be activated each time a configuration change is made.

- If the webhook is not correctly configured, the pipeline will not be triggered.
- If the Git repository name does not exactly match the URL provided during onboarding, the pipeline will not start, and no feedback will be returned to the customer.
- If the Git repository name is changed after onboarding (e.g., due to a migration from Bitbucket to GitLab), the onboarding process must be repeated; otherwise, the pipeline will not function.

To create and test the required webhook in Bitbucket, follow these steps:

1. Click on the **Settings** button on the lower-left corner.
2. Select the **Webhooks** entry.
3. Click on **Create webhook** button.

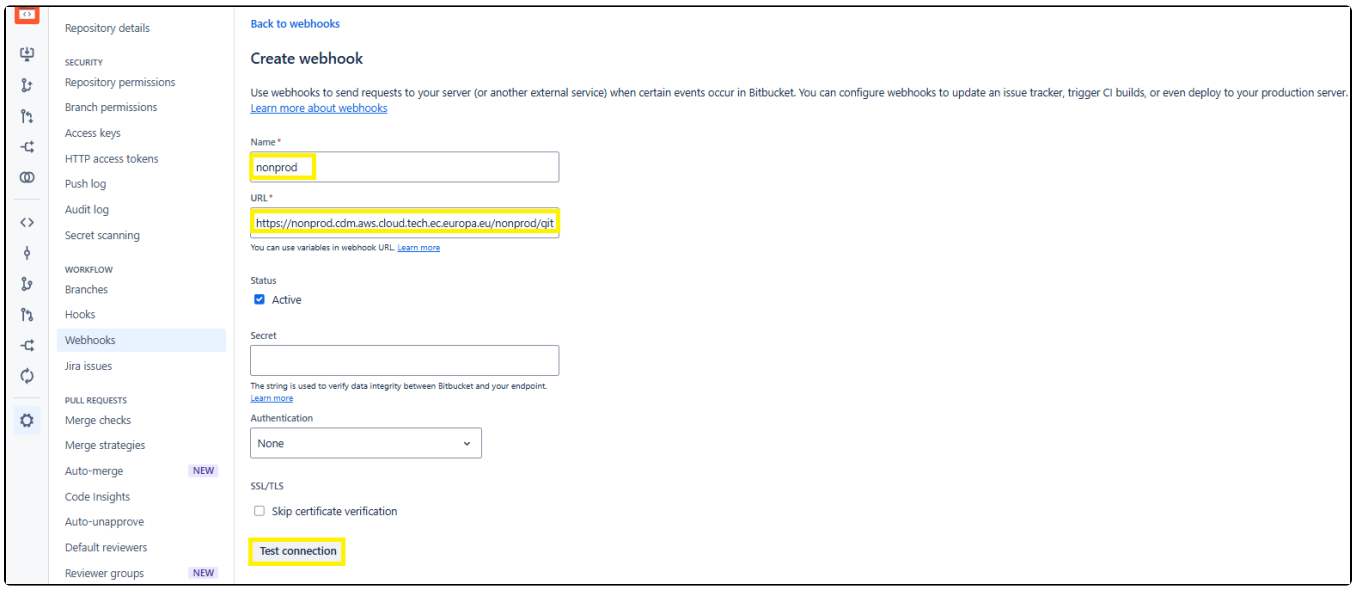


A second screen will be displayed:

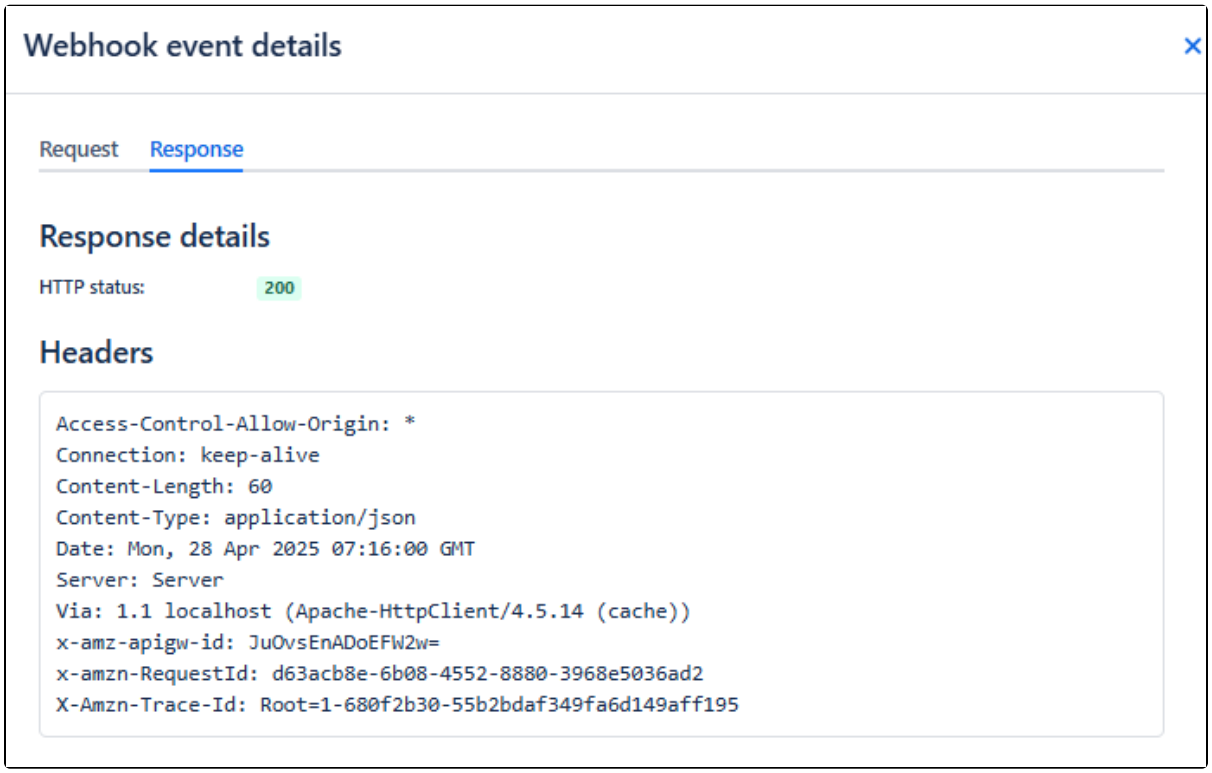
4. Enter a **name** for the webhook (free text)
5. Enter a valid **URL** pointing to the desired environment

   NONPROD environments : https://**nonprod**.cdm.aws.cloud.tech.ec.europa.eu/**nonprod**/git-webhook

   PROD environments : https://**prod**.cdm.aws.cloud.tech.ec.europa.eu/**prod**/git-webhook
6. Click on the **Save** button.



During the creation process, clicking the **Test Connection** button will display the following pop-up window.



If the HTTP response code is not **200**, this indicates an issue, and the webhook is not functioning correctly.
The configuration problem must be resolved **before proceeding**.

Enable auto-scaling

To enable autoscaling, the following properties has to be added in the **config.yaml** file.

Attribute	Values	Description
`autoscalingEnabled` (Optional)	true/**false**	• Start Tomcat instance(s) if the total Memory or CPU of pod is used at 70% during more than 30 seconds • The default value is **false** The number of instances depends number of **replicas** you configured • if replicas is lower or equals to **3** then **1** extra pod could be started • if replicas is greater than **3** and lower or equals than **8** then **2** extra pods could be started • if replicas is greater than **8** and lower or equals than **12** then **3** extra pods could be started • if replicas equals **13** then **2** extra pods could be started • if replicas equals **14** then **1** extra pod could be started When the load is coming back to normal situation (less 65% during more than 60 seconds) then extra pods will be shutdown at one ⚠ Pods are created and destroyed when activity is going up and down • Scaling up stabilization window: 30 secs • Scaling down stabilization window: 60 secs

Enable passthrough on the ingress

Page Topics

Related Links

Auto Scaling

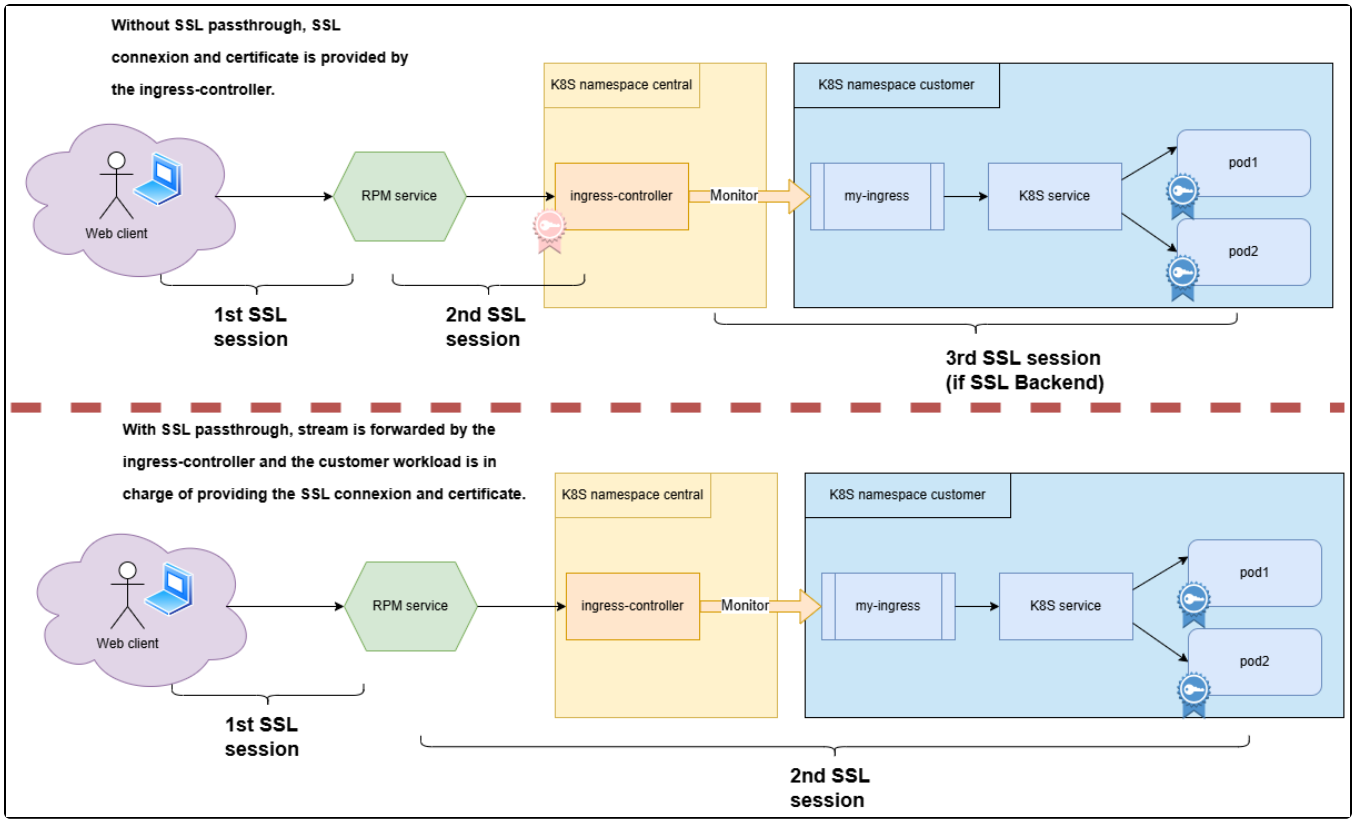
Applicative proxy

Introduction

In **Container-as-a-Service (CaaS)** environments, it is now possible to enable passthrough support allowing to forward the SSL stream to the K8S service. Therefore, the SSL transaction is managed directly with the service instead of the ingress-controller.

Current Behaviour

In a CaaS setup, when an ingress defines TLS attributes, the Kubernetes ingress controller manages the SSL connection with the client. Consequently, it supplies the SSL certificate specified in the TLS section of the ingress configuration.



Customizing the Behaviour

Add the annotation to enable the passthrough

To enable the passthrough for the related ingress, add the annotation nginx.ingress.kubernetes.io/ssl-passthrough with value "**true**"

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    nginx.ingress.kubernetes.io/ssl-passthrough: "true"
```

More details here

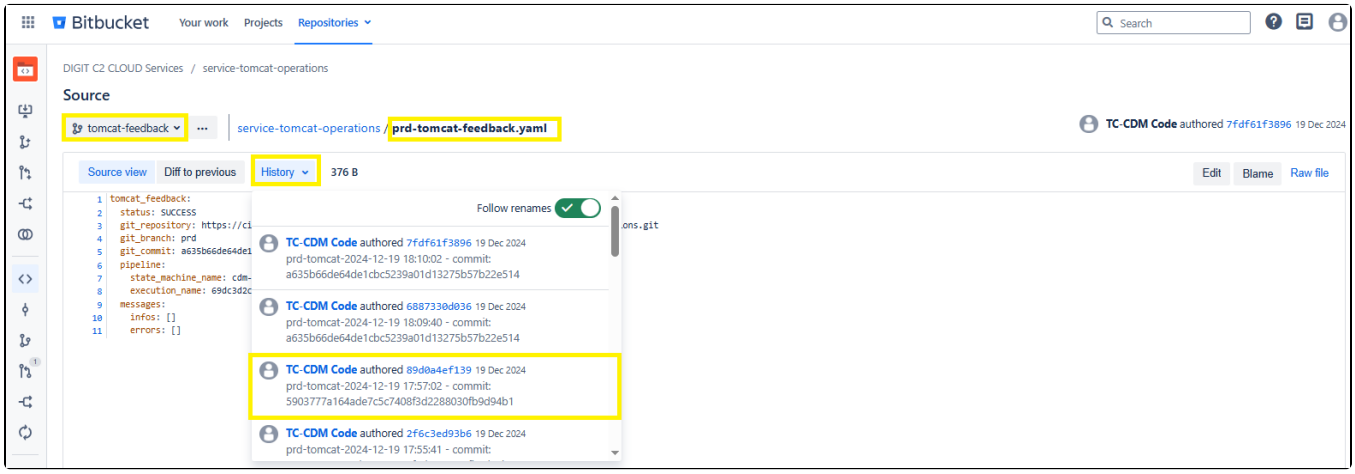
Git Feedback

A branch named **`tomcat-feedback`** is created in the configuration repository to store all notification-related information.

Each environment has a dedicated file following the naming convention: **`[ENVIRONMENT_NAME]-tomcat-feedback.yaml`**.

For example, the **production (prd)** environment will use the file: **`prd-tomcat-feedback.yaml`**.

⚠ Once an environment file is selected, its history can be parsed to retrieve previously received messages.

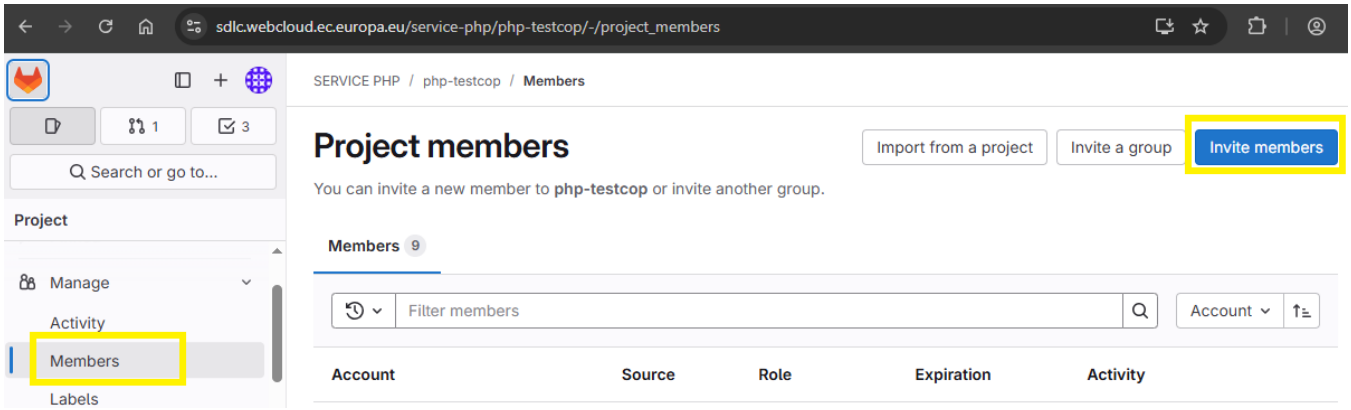


GitLab Permissions

A Tomcat functional user provided during onboarding must be configured in the git repository. This access is crucial for triggering feedb ack to the customer.

Perform the following steps in **Gitlab** to add the functional user.

1. Click on the **Manage → Members** button on the lower-left corner.
2. Click on **Invite members** button.



3. Add "TomcatGitlabFunctionalUser" with Developer role access (with no expiration date defined) and click on **Invite** button

Invite members

You're inviting members to the **gitlab-test** project.

Username, name or email address

TomcatGitlabFunctionalUser

Select from GitLab usernames or enter email addresses

Select maximum role

Developer

nvited members are assigned the sel cted role or the role they have in the group, whichever is lower. Learn more about roles.

Access expiration date (optional)

YYYY-MM-DD

From this date onward, the user can no longer access the group or project. Learn more about access.

Cancel Invite

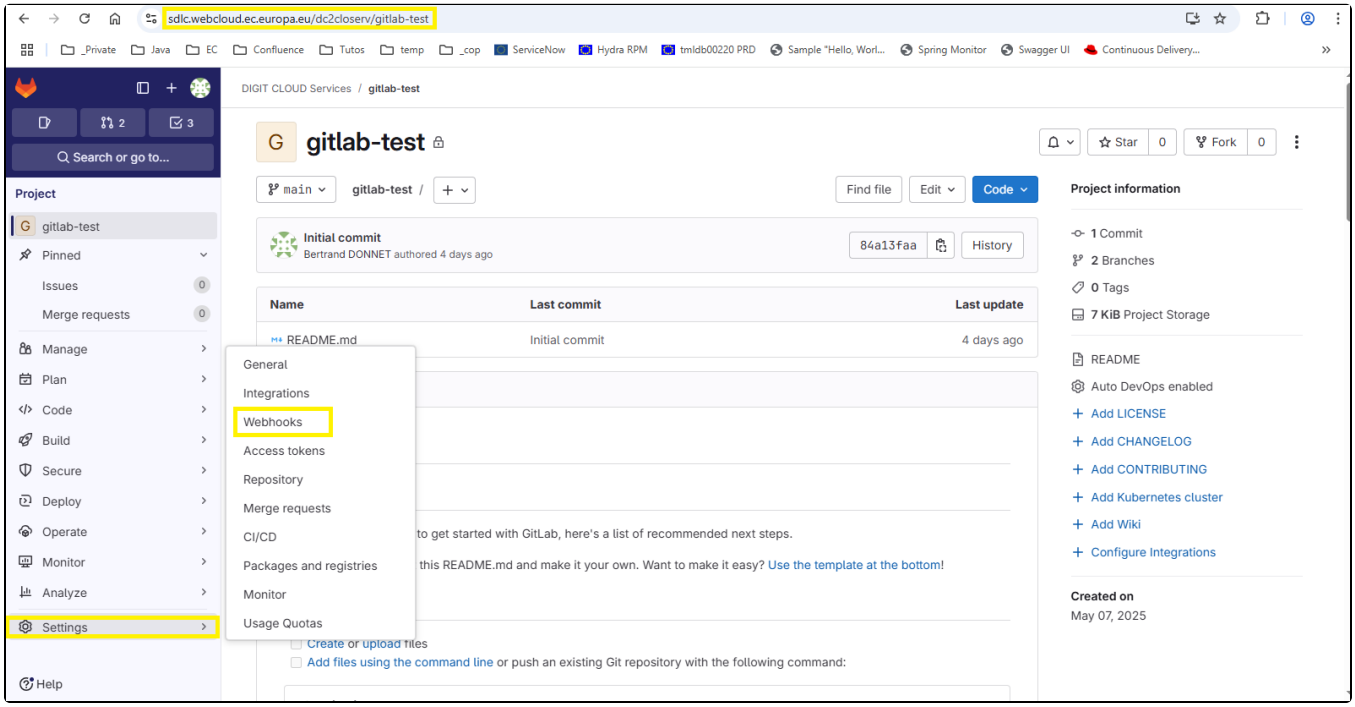
GitLab Webhook

A webhook must first be configured and tested in the Git repository provided during the onboarding process. This webhook is essential for triggering pipeline executions and is activated whenever a configuration change is made.

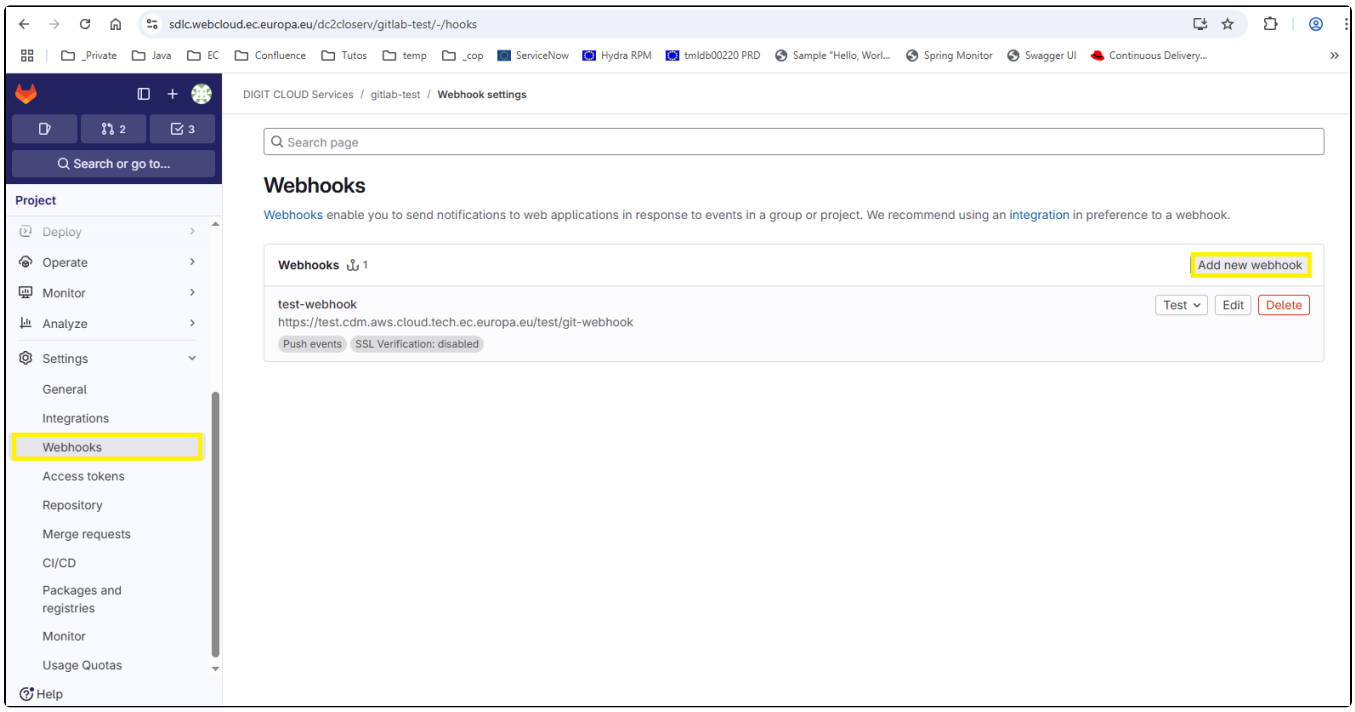
- If the webhook is not properly configured, the pipeline cannot be triggered.
- If the Git repository name does not exactly match the URL provided during onboarding, the pipeline will not start, and no feedback will be sent to the customer.
- If the Git repository name is changed after onboarding (e.g., due to a migration from Bitbucket to GitLab), the onboarding process must be repeated; otherwise, the pipeline will not function.

To create and test the required webhook in GitLab, follow these steps:

1. Click on the **Settings** icon in the lower-left corner.
2. Select **Webhooks** from the menu.
3. Click the **Create webhook** button.



4. Click on **Add new webhook**



A second screen will be displayed:

5. Enter a valid **URL** pointing to the desired environment

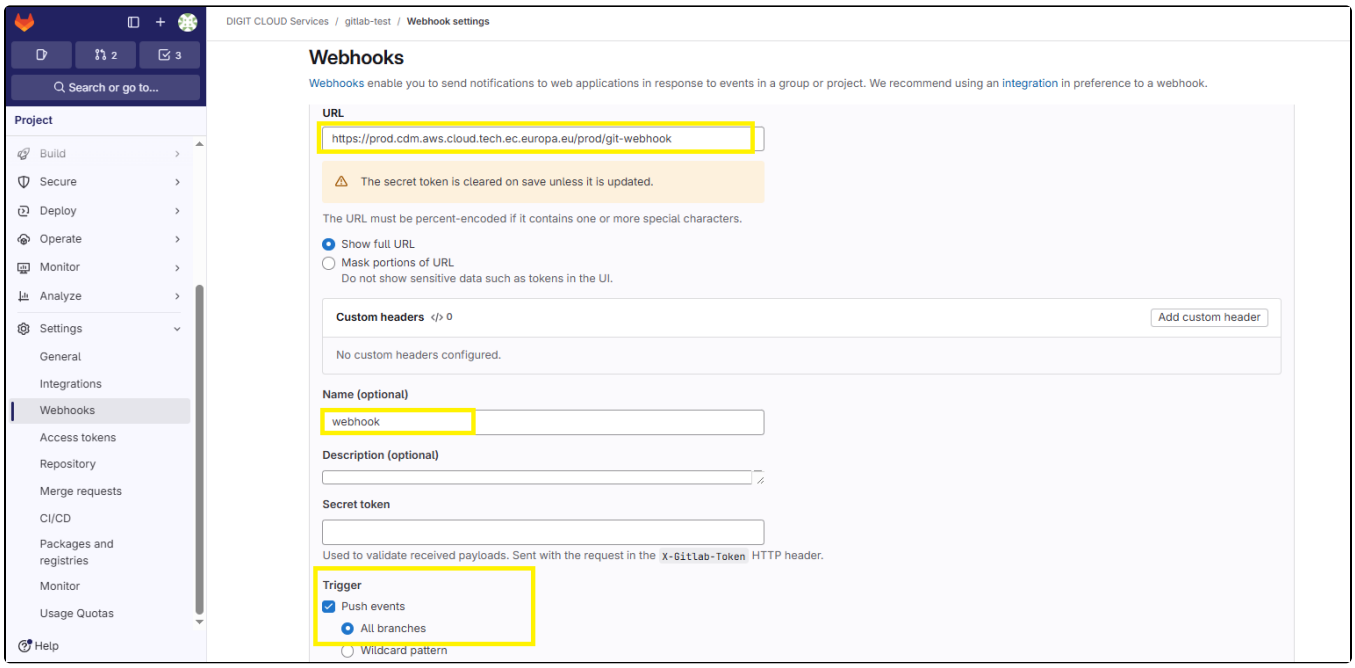
   NONPROD environment : https://**nonprod**.cdm.aws.cloud.tech.ec.europa.eu/**nonprod**/git-webhook

   PROD environment : https://**prod**.cdm.aws.cloud.tech.ec.europa.eu/**prod**/git-webhook

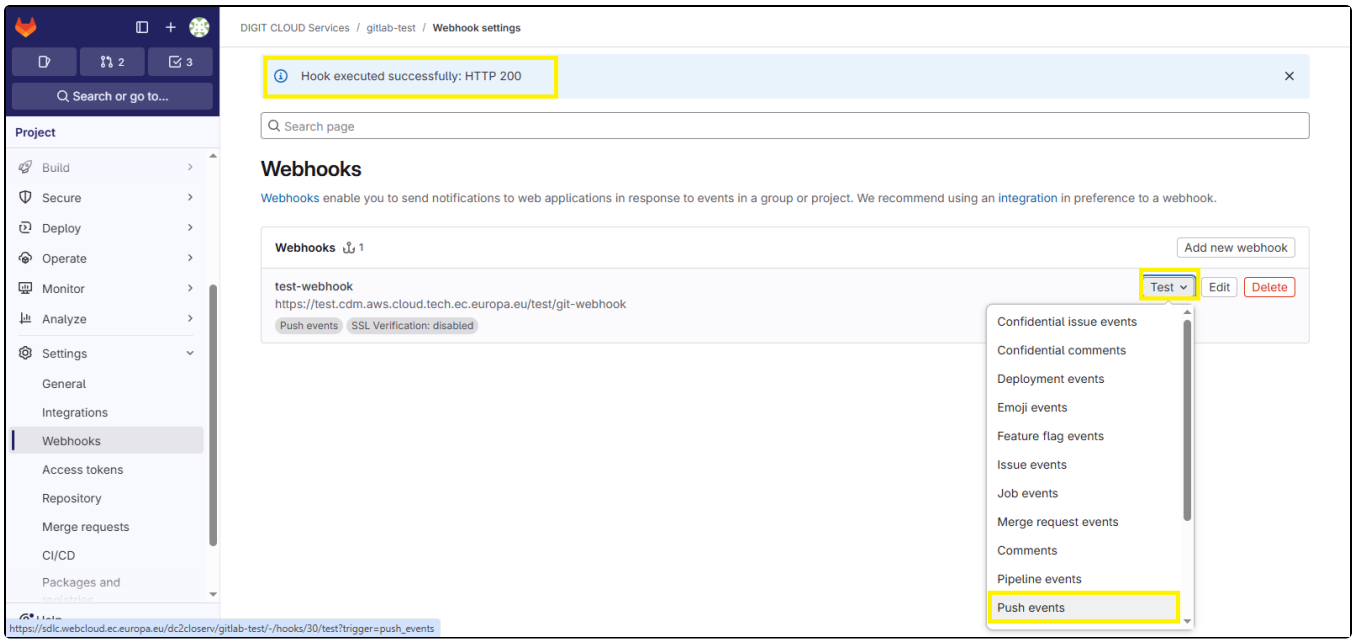
   ....

   XXX environment : https://xxx.cdm.aws.cloud.tech.ec.europa.eu/xxx/git-webhook

6. Enter a **name** for the webhook (optional free text)
7. Check the **Push events** trigger with option "**All Branches**"
8. Disable **SSL Verification** option
9. Click on **Add Webhook** button



During the creation process, clicking the **Test Connection** button will display the following pop-up.



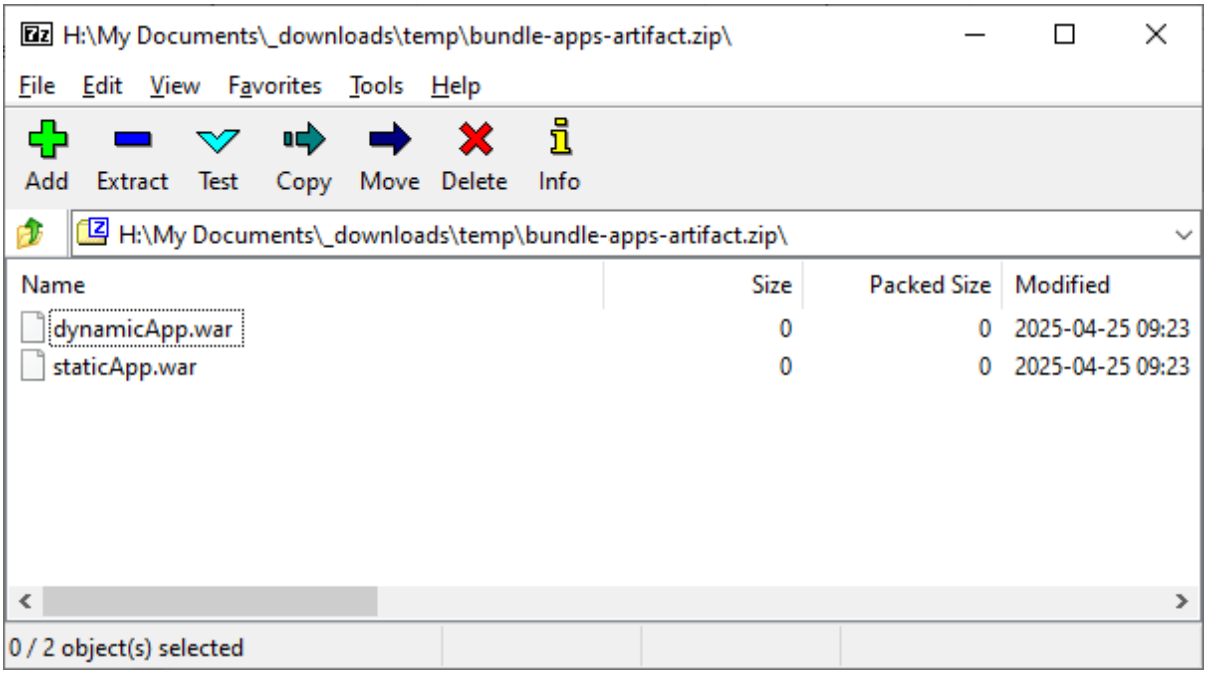
If the HTTP response code is anything other than **200**, it indicates an issue and the webhook is not functioning correctly.
This configuration problem must be resolved **before proceeding**.

Static Content Deployment

Static content can be deployed as a **side application**.

It can be packaged as a standard **WAR file** within a **bundle ZIP archive**, and will be deployed in the same way as a dynamic application.

> ⚠ A ZIP file can contain multiple WAR files, allowing you to colocate a static WAR alongside a dynamic WAR if needed.
>
> Below is an example of a bundle containing two applications, which can be uploaded to **Nexus** and deployed.



YAML configuration section

Attribute Path	Description	Mandatory	Example	
applicationName	Name of the application **(This must be in slug-format)**	Y	ams-api-app	
ingressName	Base name of the ingress **(This must be in slug-format)**	Y	ams-api	
ingressPath	Represents the path of the deployed application	Y	/	
replicas	Represents the number of instances/pods that will manage the application	Y	4	

customerImageName	• Represents the name of the built image.
	  ○ This image should not be updated or overwritten
	  ○ Only the previous 10 images are saved.
		Y	curex-app	

customerImageTag	• Represents the tag of the built image.
	  ○ This image should not be updated or overwritten
	  ○ Only the previous 10 images are saved.
		Y	testimg85	

artifactName	The name of the artifact containing the properties files • It contains files for each environment (DEV/ACC/PROD...) • Stored in Nexus during build phase	Y	projectProperties.zip	
applicationPropertiesZip	The name of the artifact containing the properties files • It contains files for each environment (DEV/ACC/PROD...) • Stored in Nexus	N	-	
jvmVersion	Version of JVM to use for the runtime.	Y	11	
productName	Name of the flavour of Tomcat wanted	N	tomcat	Possible values are: • **tomcat** • tomee
tomcatVersion	Version of Tomcat to use for the runtime.	Y	9.0	

tomeeVersion	Version of TomEE to use for the runtime.	N	8.0	**NB**: the latest minor version available at the time of the build will be used Possible values are: • **8.0**
clientParamsKey	Key under which the customer parameters will be defined in the HashiCorp Vault. NB: if the key is 'clientParams', the parameters will be stored in different subdirs for each environment: • DEV: kv_customer/dev/clientParams/ • ACC: kv_customer/acc/clientParams/ • PROD: kv_customer/prod/clientParams/ NB: More information on vault configuration can be found here : Tomcat in Cloud: HashiCorp Vault Configuration	N	clientParams	
clientFilesKey	Key under which the customer files will be defined in the HashiCorp Vault. NB: if the key is 'clientFiles', the files will be stored in different subdirs for each environment: • DEV: kv_customer/dev/clientFiles/ • ACC: kv_customer/acc/clientFiles/ • PROD: kv_customer/prod/clientFiles/ NB: More information on vault configuration can be found here : Tomcat in Cloud: HashiCorp Vault Configuration	N	clientFiles	
autoscalingEnabled	When enabled, the number of pods will be automatically adapted in function of the available resources for the environment.	Y	false	
buildEnabled	Indicates if the build must be made with CDM (disabled by default)	N	false	false /true
deployEnabled	Indicates if the deploy must be made with CDM (disabled by default)	N	false	false /true
tomcat:jvm:	==========	Y	==========	
heapMinSize	Minimum size of the Heap of the Java Virtual Machine for Tomcat runtime **NB**: Value is in megabytes.	Y	1536	
heapMaxSize	Maximum size of the Heap of the Java Virtual Machine for Tomcat runtime **NB**: Value is in megabytes.	Y	1536	
metaMinSize	Minimum size of the Meta of the Java Virtual Machine for Tomcat runtime **NB**: Value is in megabytes.	Y	768	
metaMaxSize	Maximum size of the Meta of the Java Virtual Machine for Tomcat runtime **NB**: Value is in megabytes.	Y	768	
directMaxSize	Maximum size of the Direct of the Java Virtual Machine for Tomcat runtime **NB**: Value is in megabytes.	Y	2048	
debugGC	Option used to have more logs regarding the garbage collector		true	
debugSSL	Option used to have more information about SSL problems		false	
debugJPDA	Gives more information about the Java Platform Debugger Architecture (JPDA) is a collection of APIs to debug Java code		false	

hideArgsInLog	Sensible arguments as passwords and secrets will be hidden in the logs.		false	
gcLogPathSeparated	Garbage collector logs will be separated in a specific directory		true	
hideVersions	All Tomcat version will be removed from pages and logs, so that hacking is more difficult		false	
gcUsed	Garbage collector used		parallel	
environmentVariables	Environments variables passed to the Tomcat JVM			
tomcat: catalinaProperties:	==		==========	
sharedLoader	Name of the shared load to use			
tomcat: context:	==		==========	
cookieProcessor	The CookieProcessor element represents the component that parses received cookie headers into javax.servlet.http.Cookie objects accessible through HttpServletRequest.getCookies() and converts javax.servlet.http.Cookie objects added to the response through HttpServletResponse.addCookie() to the HTTP headers returned to the client.		org.apache.tomcat.util.http.Rfc6265CookieProcessor	
cookieUseHttpOnly	Cookies will be used only for HTTP		true	
tomcat: connector:	==		==========	
httpPort	The TCP port number on which this Connector will create a server socket and await incoming connections. Your operating system will allow only one server application to listen to a particular port number on a particular IP address. If the special value of 0 (zero) is used, then Tomcat will select a free port at random to use for this connector. This is typically only useful in embedded and testing applications.		8080	
httpProxyName	If this Connector is being used in a proxy configuration, configure this attribute to specify the server name to be returned for calls to request.getServerName(). See Proxy Support for more information.			
httpProxyPort	If this Connector is being used in a proxy configuration, configure this attribute to specify the server port to be returned for calls to request.getServerPort(). See Proxy Support for more information.			
httpUriEncoding	This specifies the character encoding used to decode the URI bytes, after %xx decoding the URL. The default value is UTF-8.			
httpsPort	The TCP port number on which this Connector will create a server socket and await incoming connections. Your operating system will allow only one server application to listen to a particular port number on a particular IP address. If the special value of 0 (zero) is used, then Tomcat will select a free port at random to use for this connector. This is typically only useful in embedded and testing applications.		8443	
httpsProxyName	If this Connector is being used in a proxy configuration, configure this attribute to specify the server name to be returned for calls to request.getServerName(). See Proxy Support for more information.			
httpsProxyPort	If this Connector is being used in a proxy configuration, configure this attribute to specify the server port to be returned for calls to request.getServerPort(). See Proxy Support for more information.			
httpsUriEncoding	This specifies the character encoding used to decode the URI bytes, after %xx decoding the URL. The default value is UTF-8.			
relaxedPathChars	The HTTP/1.1 specification requires that certain characters are %nn encoded when used in URI paths. Unfortunately, many user agents including all the major browsers are not compliant with this specification and use these characters in unencoded form. To prevent Tomcat rejecting such requests, this attribute may be used to specify the additional characters to allow. If not specified, no additional characters will be allowed. The value may be any combination of the following characters: " < > [\ ] ^ ` { | } . Any other characters present in the value will be ignored.			
relaxedQueryChars	The HTTP/1.1 specification requires that certain characters are %nn encoded when used in URI query strings. Unfortunately, many user agents including all the major browsers are not compliant with this specification and use these characters in unencoded form. To prevent Tomcat rejecting such requests, this attribute may be used to specify the additional characters to allow. If not specified, no additional characters will be allowed. The value may be any combination of the following characters: " < > [\ ] ^ ` { | } . Any other characters present in the value will be ignored.			

compression Enabled	The Connector may use HTTP/1.1 GZIP compression in an attempt to save server bandwidth. The acceptable values for the parameter is "off" (disable compression), "on" (allow compression, which causes text data to be compressed), "force" (forces compression in all cases), or a numerical integer value (which is equivalent to "on", but specifies the minimum amount of data before the output is compressed). If the content-length is not known and compression is set to "on" or more aggressive, the output will also be compressed. If not specified, this attribute is set to "off". Note: There is a tradeoff between using compression (saving your bandwidth) and using the sendfile feature (saving your CPU cycles). If the connector supports the sendfile feature, e.g. the NIO connector, using sendfile will take precedence over compression. The symptoms will be that static files greater that 48 KiB will be sent uncompressed. You can turn off sendfile by setting useSendfile attribute of the connector, as documented below, or change the sendfile usage threshold in the configuration of the DefaultServlet in the default conf/web.xml or in the web.xml of your web application.			
compression MinSize	If compression is set to "on" then this attribute may be used to specify the minimum amount of data before the output is compressed. If not specified, this attribute is defaults to "2048". Units are in bytes.		2048	
compressibl eMimeType	The value is a comma separated list of MIME types for which HTTP compression may be used. The default value is text/html,text/xml,text/plain,text/css,text/javascript,application/javascript,application/json,application/xml . If you specify a type explicitly, the default is over-ridden.		text/html, text/xml, text/plain, text/css, text /javascript, application /javascript, application /json, application /xml	
xpoweredBy Enabled	Set this attribute to true to cause Tomcat to advertise support for the Servlet specification using the header recommended in the specification. The default value is false.			
minSpareThr eads	The minimum number of threads always kept running. This includes both active and idle threads. If not specified, the default of 10 is used. If an executor is associated with this connector, this attribute is ignored as the connector will execute tasks using the executor rather than an internal thread pool. Note that if an executor is configured any value set for this attribute will be recorded correctly but it will be reported (e.g. via JMX) as -1 to make clear that it is not used.			
maxSpareTh reads	The maximum number of threads always kept running. This includes both active and idle threads. If not specified, the default of 10 is used. If an executor is associated with this connector, this attribute is ignored as the connector will execute tasks using the executor rather than an internal thread pool.			
maxPostSize	The maximum size in bytes of the POST which will be handled by the container FORM URL parameter parsing. The limit can be disabled by setting this attribute to a value less than zero. If not specified, this attribute is set to 2097152 (2 MiB). Note that the FailedRequestFilter can be used to reject requests that exceed this limit.			
maxThreads	The maximum number of request processing threads to be created by this Connector, which therefore determines the maximum number of simultaneous requests that can be handled. If not specified, this attribute is set to 200. If an executor is associated with this connector, this attribute is ignored as the connector will execute tasks using the executor rather than an internal thread pool. Note that if an executor is configured any value set for this attribute will be recorded correctly but it will be reported (e.g. via JMX) as -1 to make clear that it is not used.			
maxParamet erCount	The maximum total number of request parameters (including uploaded files) obtained from the query string and, for POST requests, the request body if the content type is application/x-www-form-urlencoded or multipart/form-data. Request parameters beyond this limit will be ignored. A value of less than 0 means no limit. If not specified, a default of 10000 is used. Note that FailedRequestFilter filter can be used to reject requests that exceed the limit.			
maxHttpHea derSize	Provides the default value for maxHttpRequestHeaderSize and maxHttpResponseHeaderSize. If not specified, this attribute is set to 8192 (8 KiB).			
maxSwallow Size	The maximum number of request body bytes (excluding transfer encoding overhead) that will be swallowed by Tomcat for an aborted upload. An aborted upload is when Tomcat knows that the request body is going to be ignored but the client still sends it. If Tomcat does not swallow the body the client is unlikely to see the response. If not specified the default of 2097152 (2 MiB) will be used. A value of less than zero indicates that no limit should be enforced.			
enableLooku ps	Set to true if you want calls to request.getRemoteHost() to perform DNS lookups in order to return the actual host name of the remote client. Set to false to skip the DNS lookup and return the IP address in String form instead (thereby improving performance). By default, DNS lookups are disabled.			
acceptCount	The maximum length of the operating system provided queue for incoming connection requests when maxConnections has been reached. The operating system may ignore this setting and use a different size for the queue. When this queue is full, the operating system may actively refuse additional connections or those connections may time out. The default value is 100.			

connectionTimeout	The number of milliseconds this Connector will wait, after accepting a connection, for the request URI line to be presented. Use a value of -1 to indicate no (i.e. infinite) timeout. The default value is 60000 (i.e. 60 seconds) but note that the standard server.xml that ships with Tomcat sets this to 20000 (i.e. 20 seconds). Unless disableUploadTimeout is set to false, this timeout will also be used when reading the request body (if any).			
keepAliveTimeout	The number of milliseconds this Connector will wait for another HTTP request before closing the connection. The default value is to use the value that has been set for the connectionTimeout attribute. Use a value of -1 to indicate no (i.e. infinite) timeout.			
disableUploadTimeout	This flag allows the servlet container to use a different, usually longer connection timeout during data upload. If not specified, this attribute is set to true which disables this longer timeout.			
tomcat:host:	==========		==========	
deployXml	Set to false if you want to disable parsing the context XML descriptor embedded inside the application (located at /META-INF/context.xml). Security conscious environments should set this to false to prevent applications from interacting with the container's configuration. The administrator will then be responsible for providing an external context configuration file, and putting it in the location defined by the xmlBase attribute. If this flag is false, a descriptor is located at /META-INF/context.xml and no descriptor is present in xmlBase then the context will fail to start in case the descriptor contains necessary configuration for secure deployment (such as a RemoteAddrValve) which should not be ignored. The default is true unless a security manager is enabled when the default is false. When running under a security manager this may be enabled on a per web application basis by granting the org.apache.catalina.security.DeployXmlPermission to the web application. The Manager and Host Manager applications are granted this permission by default so that they continue to work when running under a security manager.			
copyXml	Set to true if you want a context XML descriptor embedded inside the application (located at /META-INF/context.xml) to be copied to xmlBase when the application is deployed. On subsequent starts, the copied context XML descriptor will be used in preference to any context XML descriptor embedded inside the application even if the descriptor embedded inside the application is more recent. The default is false. Note if deployXML is false, this attribute will have no effect.			
tomcat:log:	==========		==========	
logRotate	Enables rotation of the logs			
logMaxAge	Maximum age of rotating logs. If the file is older than this age, the log is purged.			
logMaxSize	Maximum size of rotating logs. If the file is bigger than this size, the log is purged.			
tomcat:realm:	==========		==========	
addAllRolesMode	Adds the **addAllRolesMode** property in the realm as required for EULogin			
tomcat:valve:	==========		==========	
rewriteValveEnabled	The rewrite valve implements URL rewrite functionality in a way that is very similar to mod_rewrite from Apache HTTP Server.			
ecasAuthenticatorValveEnabled	Valve that calls the ECAS/EULogin Authenticator.			
remoteIpValveEnabled	Tomcat port of mod_remoteip, this valve replaces the apparent client remote IP address and hostname for the request with the IP address list presented by a proxy or a load balancer via a request headers (e.g. "X-Forwarded-For").			
accessLogValveEnabled	Abstract implementation of the Valve interface that generates a web server access log with the detailed line contents matching a configurable pattern. The syntax of the available patterns is similar to that supported by the Apache HTTP Server mod_log_config module.			
errorReportValveEnabled	Implementation of a Valve that outputs HTML error pages. NB: possibly contains server version information.			
errorReportValveError404	Location of the page to use in case of 404 errors			
tomcat:web:	==========		==========	
defaultSessionTimeout	Timeout in seconds after which the session automatically expires			
jspServletEnablePooling	Determines whether tag handler pooling is enabled. This is a compilation option. It will not alter the behaviour of JSPs that have already been compiled.			

requestCharacterEncoding	Character encoding used for the Web connector requests			
responseCharacterEncoding	Character encoding used for the Web connector responses			
hstsEnabled	HTTP Strict Transport Security (HSTS) is a web security policy mechanism, which helps protect web application users against some passive (eavesdropping) and active network attacks.			
secureCookies	Enable the HTTPOnly and Secure attributes for cookies as sent by Apache Tomcat.			
securityFilterEnabled	Enables an extra Security Servlet Filter developed for the EC			
httpsRedirectionEnabled	Automatically redirects all unsecured HTTP traffic to HTTPS			
tomcat: datasources:	==========	N	==========	
name	The name of the JNDI JDBC DataSource for this UserDatabase.			
url	The URL to use for the database connected to this datasource			
username	Username used for authentication			
password	Password used for authentication			
driverClassName	Class name for the old mm.mysql JDBC driver is org.gjt.mm.mysql.Driver - we recommend using Connector/J though. Class name for the official MySQL Connector/J driver is com.mysql.jdbc.Driver.			
initialSize	Number of connections present in the pool after creation.			
minIdle	Minimum number of idle database connections to retain in pool. Set to -1 for no limit. See also the DBCP 2 documentation on this and the minEvictableIdleTimeMillis configuration parameter.			
maxIdle	Maximum number of idle database connections to retain in pool. Set to -1 for no limit.			
maxTotal	Maximum number of database connections in pool. Set to -1 for no limit.			
tomcat: mailsessions:	==========	N	==========	
name	JNDI name under which you will look up preconfigured sessions			
host	Host used for sending mails			
from	From field used in the sent mail			
auth		N		
type		N		
transportProtocol		N		
smtpHost		N		
smtpPort		N		
smtpAuth		N		
smtpUser		N		

smtpPassword		N		Passwords must be defined in the vault. More information here : Tomcat in Cloud: Manage Secrets
tomcat: persistentStorages:	==========	N	==========	
mountPath	Access path for the persistent storage			
size	Size in Gigabytes of the persistent storage			

accessMode	The different access modes for the persistent storages are:
	
	• **ReadWriteOnce**
	  ○ the volume can be mounted as read-write by a single node. ReadWriteOnce access mode still can allow multiple pods to access the volume when the pods are running on the same node. For single pod access, please see ReadWriteOncePod.
	• **ReadOnlyMany**
	  ○ the volume can be mounted as read-only by many nodes.
	• **ReadWriteMany**
	  ○ the volume can be mounted as read-write by many nodes.
	• **ReadWriteOncePod**
	  ○ the volume can be mounted as read-write by a single Pod. Use ReadWriteOncePod access mode if you want to ensure that only one pod across the whole cluster can read that PVC or write to it.			

storageClass	Possible values are : • "trident-iron" • "trident-bronze" • "trident-silver" • "trident-gold"			

How to convert a string to **slug-format** ?

1. Convert to lower case
2. Trim any leading or trailing spaces
3. Remove any accents from characters (e.g. á, â, â, ã, ä, å, ç)
4. Replace any other special characters with spaces
5. Replace multiple spaces or dashes (hyphens) with a single dash

The application will be accessible through :

https://**<ingressName>**.tc-app.**<k8sClusterFQDN><ingressPath>**

Can we see an example ?

```
applicationName: "tccop-api-app"
ingressName: "tccop-api"
ingressPath: "/"
replicas: 1
customerImageName: "tccop-api-app"
customerImageTag: "testimg216"
jvmVersion: "11"
tomcatVersion: "9.0.88"
clientFilesKey:  clientFiles
clientParamsKey: clientParams
deployEnabled: true
namespace: "tccop"
artifactName: "sample.zip"

tomcat:
  jvm:
    heapMinSize: 1536
    heapMaxSize: 1536
    metaMinSize: 768
    metaMaxSize: 768
    directMaxSize: 2048
    debugGC: true
    debugSSL: false
    debugJPDA: false
    hideArgsInLog: false
    gcLogPathSeparated: true
    hideVersions: false
    gcUsed: 'parallel'
    environmentVariables: ''
  catalinaProperties:
    sharedLoader: ''
  context:
    cookieProcessor: 'org.apache.tomcat.util.http.Rfc6265CookieProcessor'
    cookieUseHttpOnly: true
  connector:
    httpPort: 8080
    httpProxyName: ''
    httpProxyPort: ''
    httpUriEncoding: ''
    httpsPort: 8443
    httpsProxyName: ''
    httpsProxyPort: ''
    httpsUriEncoding: ''
    relaxedPathChars: ''
    relaxedQueryChars: ''
    compressionEnabled: false
    compressionMinSize: 2048
    compressibleMimeType: 'text/html,text/xml,text/plain,text/css,text/javascript,application/javascript,
application/json,application/xml'
    xpoweredByEnabled: false
    minSpareThreads: 25
    maxSpareThreads: 75
    maxPostSize: 2097152
    maxThreads: 160
    maxParameterCount: 10000
    maxHttpHeaderSize: 8192
    maxSwallowSize: 104857600
    enableLookups: false
    acceptCount: 100
    connectionTimeout: 60000
    keepAliveTimeout: 60000
    disableUploadTimeout: true
  host:
    deployXml: true
    copyXml: false
  log:
    logRotate: 30
    logMaxAge: 60
    logMaxSize: 200
  realm:
    addAllRolesMode: false
```

```
valve:
  rewriteValveEnabled: false
  ecasAuthenticatorValveEnabled: false
  remoteIpValveEnabled: true
  accessLogValveEnabled: true
  accessLogValveExtraFields: ''
  errorReportValveEnabled: false
  errorReportValveError404: ''
web:
  defaultSessionTimeout: 30
  jspServletEnablePooling: true
  requestCharacterEncoding: 'UTF-8'
  responseCharacterEncoding: 'UTF-8'
  hstsEnabled: false
  secureCookies: false
  securityFilterEnabled: false
  httpsRedirectionEnabled: false
datasources:
  - name: " jdbc/mytest "
    url: jdbc:oracle:thin:@localhost:1521:sid
    username: "   ceciEstUnTest  "
    password: pwd1
    driverClassName: oracle.jdbc.OracleDriver
    initialSize: 10
    minIdle: 10
    maxIdle: 10
    maxTotal: 10
  - name: jdbc/ds2
    url: jdbc:oracle:thin:@localhost:1521:sid
    username: uid2
    password: pwd2
    driverClassName: oracle.jdbc.OracleDriver
    initialSize: 10
    minIdle: 10
    maxIdle: 10
    maxTotal: 10
mailsessions:
  - name: mail/session1
    host: host1
    from: from1
  - name: mail/session2
    host: host2
    from: from2
persistentStorages:
  - name: ps-drive1
    size: 1
    mountPath: /share
    accessMode: ReadWriteMany
    storageClass: trident-silver
  - name: ps-drive2
    size: 2
    mountPath: /share2
    accessMode: ReadWriteMany
    storageClass: trident-silver
```