



Tungsten Dashboard 8.0 Manual

Continuent Ltd

Tungsten Dashboard 8.0 Manual

Continuent Ltd

Copyright © 2026 Continuent Ltd

Abstract

This manual documents Tungsten Dashboard, a simple graphical user interface allowing you to manage all of your Tungsten Clusters in one single place.

This manual includes information for 8.0, up to and including 8.0.4.

Build date: 2026-03-16 [ac746a88]

Up to date builds of this document: [Tungsten Dashboard 8.0 Manual \(Online\)](#), [Tungsten Dashboard 8.0 Manual \(PDF\)](#)

Table of Contents

1. Introduction	7
2. Prerequisites	8
2.1. Common Prerequisites	8
2.2. Prerequisites for a Docker Deployment	8
2.2.1. Docker	8
2.3. Prerequisites for a Kubernetes Deployment	8
2.3.1. kubectl	9
2.3.2. helm	9
2.3.3. Kubernetes Cluster	9
2.3.4. Upload tungsten-dashboard to Container Registry	9
3. Installation	11
3.1. Downloading Installation Packages	11
3.2. Installing via docker	11
3.2.1. Pre-Installation Steps	11
3.2.1.1. Self-Signed	11
3.2.1.2. Let's Encrypt	12
3.2.2. Install Steps for Docker	12
3.2.3. Uninstall	13
3.3. Installation with Kubernetes	13
3.3.1. Installation Quickstart	13
3.3.2. Overview	13
3.3.3. Helm Chart Details	14
3.3.3.1. General settings	14
3.3.3.2. Image settings	14
3.3.3.3. Secrets settings	14
3.3.3.4. Clusters settings	14
3.3.3.5. Application Configuration	15
3.3.3.6. User settings	15
3.3.3.7. Network settings	15
3.3.3.8. Resources settings	16
4. Updates & Upgrades	17
4.1. Updating an existing installation	17
4.2. Upgrading to new Tungsten Dashboard version	17
5. User Guide	18
5.1. The Front Page	18
5.2. Create your own user account	19
5.3. Add Clusters	20
5.4. Monitor Clusters	22
6. User Interface	23
6.1. Front Page Interface	23
6.2. Clusters Interface	24
6.3. User Management Interface	29
7. Command Line Tools	32
7.1. install.pl script	32
7.2. getcert.pl script	32
7.3. letsencrypt2jetty.pl script	33
7.4. collect-diag.pl script	34
8. Establishing Connectivity to Tungsten Clusters	35
8.1. Configuring SSL when Deploying with Kubernetes	35
8.1.1. Basic SSL Connection To REST API only	35
8.1.2. SSL with Self-Signed Certificates	35
9. Operations	37
9.1. Service Operations	37
9.2. Cluster Datasource and Replicator operations	39
9.3. Tungsten Dashboard Exclusive operations	41
10. Best Practices	43
11. Troubleshooting	44
11.1. Lost Password	44
11.2. 404 After Test Connection Succeeds	44
11.3. Troubleshooting problem installations and cleaning up	45
11.4. DASHBOARD_SECRET	46
11.4.1. Lost DASHBOARD_SECRET	46
11.5. Connecting to Docker	46
A. Dashboard Internals	47

A.1. Dashboard Base URL	47
A.2. Environment Variables	47
A.3. The <code>values.yaml</code> file	48
A.3.1. Application Configuration	48
A.3.2. Cluster Configuration	49
A.3.3. User Configuration	49
B. Release Notes	50
B.1. Tungsten Dashboard 8.0.4 GA (10 Mar 2026)	50
B.2. Tungsten Dashboard 8.0.3 GA (13 Jan 2026)	51
B.3. Tungsten Dashboard 8.0.2 GA (6 Oct 2025)	52
B.4. Tungsten Dashboard 8.0.1 GA (14 Jul 2025)	54
B.5. Tungsten Dashboard 8.0.0 GA (28 Apr 2025)	55

List of Figures

5.1. Dashboard Front Page Overview	18
5.2. Add User Form	19
5.3. Cluster Front Page - Add Cluster	21
5.4. Add Cluster Form	21
6.1. Front Page Interface	24
6.2. Populated Cluster Front Page	25
6.3. Cluster Tool Bar	25
6.4. Cluster Header Detail	25
6.5. Parent Service Actions/Operations	26
6.6. Edit Cluster Form	27
6.7. Cluster Host Table [CAA]	28
6.8. Primary Host Operations	29
6.9. Replica Host Operations	29
6.10. User Management Page	30
6.11. Add User Form	31
9.1. Service Policy Dropdown	37
9.2. Cluster Lock Dropdown	38
9.3. Composite Service Operations	38
9.4. Service Operartions	39
9.5. Primary Host Operations	40
9.6. Replica Host Operations	40
9.7. Blocked Cluster	41
9.8. Cluster Edit Form	42

List of Tables

7.1. <code>install.pl</code> Options	32
7.2. <code>getcert.pl</code> Options	32
7.3. <code>letsencrypt2jetty.pl</code> Options	33
A.1. Environment Variables	47
A.2. Environment Settings	48
A.3. Start-Up Settings	48
A.4. Webserver Settings	48
A.5. Cluster Connection Settings	49
A.6. Cluster Settings	49
A.7. User Settings	49

Chapter 1. Introduction

Tungsten Dashboard version 8.0.4 (v8) builds on the foundation of the first Tungsten Dashboard release (v1).

Tungsten Dashboard v8 is designed to be used with Tungsten Clustering v8 and Tungsten Kubernetes clusters. This new UI offers a fresh approach to operating and monitoring Tungsten Clusters.

Customers running v6/v7 of Tungsten should use Tungsten Dashboard v1. Please see the [Tungsten Dashboard v1 Documentation](#) for information on the installation and configuration of Dashboard v1.

The major differences in the v8 release versus v1 are as follows:

New Communication Protocol

The most significant change in the new Dashboard is the communication method between the Dashboard service and the clusters. Instead of continuously polling the cluster API, which could impact performance, the new Dashboard employs a publish/subscribe protocol over a persistent TCP connection. This change enhances cluster performance and overall efficiency. Initially, the user defines only one host, which serves as an entry point to the entire cluster. The Tungsten Dashboard then discovers the rest of the cluster. Load balancing, communication routing, and error handling between the Dashboard and the fully discovered cluster are handled on the Dashboard side, eliminating the need for third-party load balancers.

Enhanced Security

The new version delivers enhanced security features, including encryption of sensitive data, simple role-based user management, and token access authentication. It also fully supports SSL communication between the Dashboard and clusters, ensuring secure data transmission over API and TCP connections.

Improved Cluster Locking Mechanism

To prevent simultaneous cluster operations, we've improved the cluster locking mechanism and introduced a blocking operations feature. This enhancement ensures smoother and more controlled cluster operations.

Real-Time Monitoring and User-Friendly Operations

The Tungsten Dashboard is designed for real-time monitoring of clusters across various infrastructure hosting providers. It also provides user-friendly options for triggering cluster operations, along with features for filtering and searching among clusters.

Installation Methods

The new UI can be installed using Docker, which simplifies the process considerably. Alternatively, for Tungsten Clusters deployed in a Kubernetes environment, installation with [helm](#) is also available.

To get started, first ensure all prerequisites are in place - See [Chapter 2, Prerequisites](#)

When all of the prerequisites are complete, you can then install Dashboard v8 - See [Chapter 3, Installation](#)

Once the Dashboard is installed and running, consult [Chapter 5, User Guide](#) for more details.

Chapter 2. Prerequisites

There are two methods of deploying Tungsten Dashboard, and each will have slightly different requirements, as well as some common requirements. These are all outlined in the sections listed in the table of contents above.

2.1. Common Prerequisites

The following prerequisites are common across both deployment methods (docker-compose and helm):

- A staging/admin host for download and unpacking the various tools and for holding the Tungsten Dashboard installation packages. There are no specific OS version requirements, other than the host must be a Linux family host or VM.

If deploying Dashboard using docker-compose, then the staging host can also be used as the host for installation.

- Access to download the latest Tungsten Dashboard packages is via the [Continuent Download Portal](#). If you do not have access, please contact Continuent Support.
- Tungsten Clusters that are running v8.0.0 (or later).
 - Tungsten Dashboard v8 will NOT connect to Tungsten Clusters running older versions
 - If you are using an older release and are unable to upgrade at this time, you will need to follow the installation steps in the [Tungsten Dashboard v1 Documentation](#).
- A configured directory for storing downloaded Tungsten images. Further examples in this documentation use `/opt/continuent/software` as this location and aligns with the recommended default for all Tungsten software staging locations.
- A non-root OS user, we recommend configuring an OS user called `tungsten`
- OPTIONAL: `openssl` - used for fetching certificates
- OPTIONAL: `keytool` - used for generating jks files, included with Java

2.2. Prerequisites for a Docker Deployment

In order to install Tungsten Dashboard using the Docker Compose plugin you will require a host configured with the following minimum requirements:

- `Docker` - No specific version requirements.

2.2.1. Docker

To install Docker, follow the installation documentation per linux distribution on the [official docker documentation](#)

To confirm the installation run:

```
shell> docker --version
```

2.3. Prerequisites for a Kubernetes Deployment

In order to install Tungsten Dashboard in Kubernetes you will require a staging host to initiate the installation, with a minimum of the following requirements:

- `kubectl` client - Minimum version v1.29.3
- `kubectl` server - Minimum version v1.30.0
- `helm` client - Minimum version v3.15.0
- OPTIONAL: `kind` should be installed if you wish to host a local kubernetes cluster.
- CLI tools such as AWS or GCP cli tools, or equivalent for managing and accessing your cloud environment.

Additionally your cluster must have an access to the docker image file provided in the installation package. Our recommendation is for you to host a private registry of your own in order to manage the images in a centralized manner. This is accepted as the best practice for Kubernetes.

For steps on uploading the images to a Container Registry, see [Section 2.3.4, "Upload tungsten-dashboard to Container Registry"](#)

Alternatively you can upload the images as local images directly to the cluster you are using and rely on the local registry of the cluster, please bear in mind how you achieve this depends on the details of your cluster. In addition you should alter the image pull policies in the provided yaml files to match the cluster configuration.

Note

At this time Continuent is not hosting the docker image in any publicly-available registry. Please verify your access to the image before attempting an installation via Helm.

2.3.1. kubectl

For the [kubectl](#) installation steps, latest versions and up to date documentation please follow the [official kubectl documentation](#)

To check the installation:

```
shell> kubectl version --client
```

2.3.2. helm

For the [helm](#) installation steps, latest versions and up to date documentation please follow the [official helm documentation](#)

To check the installation:

```
shell> helm version
```

2.3.3. Kubernetes Cluster

Deploying Tungsten Dashboard via the Kubernetes deployment methods, will, naturally also require access to a Kubernetes Cluster. Tungsten Dashboard will deploy into any Kubernetes Cluster solution such as Amazon EKS or Google GKE. Additionally, you could also deploy into a local kubernetes cluster, such as [kind](#)

This documentation does not cover the installation and configuration of a Kubernetes Cluster - for further details should be referred to the consult the [Kubernetes Documentation](#)

Once access to a cluster is complete, you will need to set the [kubectl](#) context. For example, the following command can be used if you are installing into Amazon EKS using the AWS CLI tools:

```
shell> aws eks --region <region-code> update-kubeconfig --name <cluster-name>
```

For further information, consult the documentation appropriate for you Kubernetes provider. The links for Google and Amazon are below for convenience.

- [Amazon EKS](#)
- [Google GKE](#)

2.3.4. Upload tungsten-dashboard to Container Registry

In order to deploy the Tungsten Dashboard to any remote environment you must have a container registry available and the images that are provided in this installation package uploaded into it.

To upload an image to, for example, Amazon ECR (Elastic Container Registry) you can use a local Docker installation like this:

Load the image into your local Docker daemon

```
shell> docker load -i images/tungsten-dashboard_{{ version_postfix }}-linux_{{ arch }}.tar
```

Then, login to Amazon Elastic Container Registry:

```
shell> aws ecr get-login-password --region <region> | docker login --username AWS --password-stdin <registry ID>
```

Finally, tag and push local docker image to remote registry

```
shell> docker tag tungsten-dashboard:{{ version_postfix }} {{ registry ID }}/tungsten-dashboard:{{ version_postfix }}
shell> docker push {{ registry ID }}/tungsten-dashboard:{{ version_postfix }}
```

Bear in mind that the commands used to achieve this may change depending on the platform you're using. Once the cluster has access to the image via a registry or some other means, start by modifying the `values.yaml` file for your environment specific values. These include: clusters list, application configuration, users list and potentially ingress configuration.

As a docker container registry you can use any from the public service providers. The most popular are:

- [Docker Hub](#)
- [GitHub Container Registry](#)
- [Amazon ECR](#)
- [Google Container Registry](#)
- [Azure Container Registry \(ACR\)](#)
- [Harbor \[Open Source, self-hosted\]](#)

Chapter 3. Installation

This guide provides instructions on how to install Tungsten Dashboard using either Docker or Kubernetes.

The easiest way of configuring and installing Tungsten Dashboard is using the provided `install.pl` script. This script supports both Docker and Kubernetes as deployment methods.

This script can also be used when upgrading the Tungsten Dashboard version, or updating the configuration of an existing installation. For more details see [Chapter 4, Updates & Upgrades](#)

You can call the installer with an argument specifying the installation method `[-d|-k]`, or select the method when prompted by the script.

Following the interactive installer steps and instructions below should be sufficient to completely set up Tungsten Dashboard in your environment.

Note

Both Tungsten Dashboard installation methods store information into a persistent volume. This data includes user details with hashed passwords, cluster details with encrypted usernames/passwords and configuration json. Stored data can be transferred and used in other Dashboard installations, provided that the `DASHBOARD_SECRET` environment variable has the same value between the instances.

The following sections cover this in more detail.

Before continuing, ensure that you have completed all of the Prerequisites, outlined in [Chapter 2, Prerequisites](#)

3.1. Downloading Installation Packages

Download the Tungsten Dashboard TAR package from the [Continuent Download Portal](#) and place the file into the `/opt/continuent/software` directory on the staging host. From there, unpack the file:

```
shell> cd /opt/continuent/software
shell> tar zxvf tungsten-dashboard-8.0.4-67.tar.gz
shell> cd /opt/continuent/software/tungsten-dashboard-8.0.4-67
```

3.2. Installing via docker

3.2.1. Pre-Installation Steps

If you would like the Dashboard to run securely and support SSL/TLS internally, then a certificate must be provided at install time to enable HTTPS. Running securely is strongly recommended for production environments.

When internal SSL (https) mode is enabled, Dashboard handles the SSL by itself without external services (i.e. proxy, ingress, ...). Dashboard will also automatically redirect any http request to https, using the provided certificate.

Placing a certificate under the alias `jetty` into `cert/jetty.jks` prior to installation will enable the Dashboard to run in SSL mode.

To generate an SSL certificate, you may create a self-signed cert or source one from a commercial vendor. We have provided examples below for self-signed and Let's Encrypt.

Important

Keep your certificate password readily available, because you will be prompted for it during installation process.

3.2.1.1. Self-Signed

Below is an example of generating a self-signed certificate using the Java keytool command:

```
shell> cd /opt/continuent/software/tungsten-dashboard-8.0.4-67
shell> mkdir -p cert
shell> keytool -keysize 2048 -genkey
           -alias jetty
           -keyalg RSA
           -keystore cert/jetty.jks
           -storepass $DASHBOARD_KEYSTORE_PASSWORD
           -dname "CN=localhost, OU=Test, O=MyOrg, L=MyCity, ST=MyState, C=US"
           -ext "SAN=dns:localhost,ip:127.0.0.1"
```

```
Generating 2048-bit RSA key pair and self-signed certificate (SHA384withRSA) with a validity of 90 days
for: CN=localhost, OU=Test, O=MyOrg, L=MyCity, ST=MyState, C=US
```

3.2.1.2. Let's Encrypt

Below is an example of converting an existing cert issued by Let's Encrypt using our provided tool `letsencrypt2jetty.pl`, which calls the `openssl` and Java `keytool` commands:

```
shell> sudo ./letsencrypt2jetty.pl -d dashdev.continuent.com

Keystore file's password: tungsten
Creating './cert/'
>>> ACTION: Converting the Let's Encrypt source files to P12 format:
SOURCE:
  /etc/letsencrypt/live/dashdev.continuent.com/fullchain.pem
  /etc/letsencrypt/live/dashdev.continuent.com/privkey.pem
TARGET
  cert/jetty.p12

SUCCESS: Generated the P12 file 'cert/jetty.p12'

>>> ACTION: Converting the P12 file to JKS format:
SOURCE
  cert/jetty.p12
TARGET
  cert/jetty.jks

Importing keystore cert/jetty.p12 to cert/jetty.jks...

SUCCESS: Generated the internal jetty cert file 'cert/jetty.jks'

shell> sudo chown -R dashboard: cert
```

Or:

```
shell> mkdir cert
shell> cp /etc/letsencrypt/live/dashdev.continuent.com/fullchain.pem cert/
shell> sudo cp /etc/letsencrypt/live/dashdev.continuent.com/privkey.pem cert/
shell> sudo chmod a+r cert/privkey.pem
shell> ./letsencrypt2jetty.pl -f cert/fullchain.pem -k cert/privkey.pem

Keystore file's password: tungsten
Creating './cert/'
>>> ACTION: Converting the Let's Encrypt source files to P12 format:
SOURCE:
  ./fullchain.pem
  ./privkey.pem
TARGET
  cert/jetty.p12

SUCCESS: Generated the P12 file 'cert/jetty.p12'

>>> ACTION: Converting the P12 file to JKS format:
SOURCE
  cert/jetty.p12
TARGET
  cert/jetty.jks

Importing keystore cert/jetty.p12 to cert/jetty.jks...

SUCCESS: Generated the internal jetty cert file 'cert/jetty.jks'
```

3.2.2. Install Steps for Docker

Below is an example of installing Tungsten Dashboard using the `install.pl` script:

```
shell> cd /opt/continuent/software/tungsten-dashboard-8.0.4-67
shell> ./install.pl -d
```

The installation will prompt for a number of configuration items, such as usernames, passwords, whether or not to use HTTPS or not. Each prompt will provide details along with the default values.

Warning

During installation you will be asked for the `DASHBOARD_SECRET`, or have it auto generated for you.

It is critical that this key is stored safely as it will be required for future upgrades since it serves as the encryption key. If it is lost, then there is the possibility of losing data after upgrades.

3.2.3. Uninstall

Important

Before running the uninstall command, you **MUST** ensure you are in the original installation directory and the `docker-compose.yml` file must also exist in that directory

To uninstall Tungsten Dashboard, simply execute the following:

```
shell> cd <ORIGINAL INSTALL PATH>
shell> docker compose down
```

or, to remove persistent volumes along with the uninstallation:

```
shell> cd <ORIGINAL INSTALL PATH>
shell> docker compose down -v
```

Warning

When removing persistent volumes you lose the application state: application settings, application users, and connected clusters.

3.3. Installation with Kubernetes

3.3.1. Installation Quickstart

To install Tungsten Dashboard into a Kubernetes cluster, please make sure that the prerequisites for Kubernetes and helm, defined in [Section 2.3, "Prerequisites for a Kubernetes Deployment"](#) are complete. Then run `./install.pl -k` script found in the installation package.

This interactive installation script automates the deployment of Tungsten Dashboard to your Kubernetes cluster using Helm charts. The installer guides you through a 6-step process to configure and deploy the dashboard with minimal manual configuration.

The installation process walks you through:

1. Prerequisites Validation - Verifies that required tools (kubectl, helm) are installed and checks for container registry availability
2. Configuration File Setup - Creates or updates the Helm chart values.yaml file with your deployment settings
3. Container Image Configuration - Configures your container registry details, repository path, and dashboard version for deployment
4. Network Setup - Configures traffic routing through either:
5. Ingress - Automatically installs/upgrades nginx ingress controller with optional HTTPS/TLS support
6. Port Forwarding - Sets up direct container port access for local or custom networking
7. Dashboard Settings - Sets up admin credentials and optionally discovers and configures Tungsten cluster connections by scanning your Kubernetes namespaces for compatible services
8. Deployment - deploys using Helm

The answers you give the install script are automatically added to the `values.yaml` file in the directory housing the install script. After the script is finished running it is recommended you backup this file securely. This is a standard helm chart yaml file and should follow syntax set by helm and the charts.

3.3.2. Overview

Dashboard is installed using a helm chart, this chart is configured by a helm `values.yaml` file. The file can be generated by the use of the `install.pl` script or by copying and renaming `values.yaml.sample` file.

Helm collects the variables set into the `values.yaml` file and posts them into kubernetes api with the right kubernetes syntax.

Note

If you encounter issues during installation. Start by inspecting the yaml posted to kubernetes and validating it against kubernetes documentation.

While we recommend using the `install.pl` script found in the installation package for a smooth installation experience a manual install using pure helm chart can be done by first filling in the details in the `values.yaml` file and then running:

```
shell> helm upgrade tungsten-dashboard charts/tungsten-dashboard-8.0.3.tgz --install --namespace tungsten-dashboard -f values.yaml
```

Manual installation can be useful with highly customised setups. Please familiarise yourself with the details of the helm chart outlined in the next section if you want to use it.

3.3.3. Helm Chart Details

3.3.3.1. General settings

The general settings house settings typical to Kubernetes deployments, in most common scenarios these settings can be left untouched.

```
### General settings ###
replicaCount: 1 # By default we run only one replica. Multiple replicas are NOT supported.
nameOverride: ""
fullnameOverride: ""
serviceAccount:
  # Specifies whether a service account should be created
  create: true
  annotations: {}
  # The name of the service account to use.
  # If not set and create is true, a name is generated using the fullname template
  name: ""
podAnnotations: {}
podSecurityContext: {}
securityContext: {}
serviceType: ClusterIP
nodeSelector: {}
tolerations: []
affinity: {}
```

3.3.3.2. Image settings

The Image settings house the pull policy as well as container registry and repository information along with version information. It is important to update the registry and repository details here to match your own environment. Tag should also follow the version number of the tungsten dashboard you want to deploy.

```
### Image settings ###
image:
  registry: "" # Your container registry hostname should be entered here
  repository: tungsten-dashboard
  pullPolicy: IfNotPresent
  # Overrides the image tag whose default is the chart appVersion.
  tag: "8.0.3"
```

3.3.3.3. Secrets settings

The secrets section specifies details related to the usage of secrets in the helm installation, it is recommended to not change these values.

```
### Secrets ###
imagePullSecrets: []
crypt:
  secretName: "tungsten-dashboard-secret"
  value: ""
```

3.3.3.4. Clusters settings

The clusters section allows you to predefine clusters you want the dashboard to connect to. As you define the clusters you may want to include their host details into the Network section to make sure dashboard is able to resolve the IP of your cluster.

```
### Clusters ###
# You can list connected clusters to dashboard here.
# When this is enabled, dashboard configuration is read-only, and only way to modify clusters is via Helm values.
clusters: []
# - host: "db1.example.com"
#   namespace: "dev"
#   tcpPort: 11999
#   apiPort: 8090
#   ssl: true
#   cert: "<base64-encoded-certificate>"
#   apiUser: "sample-user"
#   apiPassword: "<clear-text-password>"
#   apiSsl: true
#   apiCert: "<base64-encoded-certificate>"
#   hostnameValidation: true
```

Note

While in this file the passwords are clear text, they are encrypted using the tungsten-dashboard-secret either defined manually, or generated automatically during installation. Values within the deployed installation will be encrypted.

3.3.3.5. Application Configuration

The application configuration allows you to fine-tune the behaviour of the dashboard application. Important details are domain, path, port and browserPort fields.

Port can generally be left untouched, but browserPort should be updated to match the port of your ingress, gateway or other traffic routing solution.

```
### Application configuration ###
config:
  env: "production"
  version: "8.0.3"
  configPath: "/app/persistent/"
  logType: "console-edn"
  heln: true
  domain: ""
  path: ""
  port: 4090
  # internal ssl toggle (keep false for kubernetes; use ingress)
  ssl: false
  sslPort: 4091
  browserPort:
  clustersConnectOnStart: false
  topologyStartupDiscover: false
  topologyCleanOnExit: false
  defaultRestPort: 8090
  defaultTcpPort: 11999
  hostnameValidation: true
  restTimeout: 120000
```

3.3.3.6. User settings

Users allows you to predefine users available to use the dashboard itself. If you want more than one user by default, define them here.

```
### users ###
users:
- role: admin
  username: default-admin-username
  password: default-admin-password
  email: "admin@admin.com"
```

Passwords in here are provided as clear text but they will be encrypted during installation.

3.3.3.7. Network settings

The network section houses the settings to configure ingress and host alias details for the dashboard container. Please note that the ingress configuration here only deploys an ingress resource by default, if you do not use the install script you must install ingress manually for the details in this section to have effect. Notable fields for ingress are: enabled, classname and tls details.

```
### Network ###
ingress:
  # When true, creates dashboard ingress resource
  enabled: false
  # You must create the IngressClass and then add it here.
  # Example of a controller: nginx see https://github.com/kubernetes/ingress-nginx for more information
  className: "" # ingressClassName ie. nginx
  # use config field's domain, path and port for ingress.
  # to fully control your setup set this to false and specify extraHosts.
  hostFromConfig: true
  annotations: {}
  # kubernetes.io/ingress.class: nginx
  # kubernetes.io/tls-acme: "true"
  extraHosts: []
  # - host: chart-example.local
  #   paths:
  #     - path: /
  #       pathType: Prefix
  # TLS config for ingress.
  # To use this create a secret with any name and add it here.
  # The secret must contain both base64 encoded certificate and the key as base64 encoded values.
  # See https://kubernetes.io/docs/concepts/services-networking/ingress/#tls for more
  tls: []
  # - secretName: chart-example-tls
  #   hosts:
  #     - chart-example.local # This value must match config.domain field
hostAliases: []
# If you are using Tungsten node names that cannot be resolved, such as db1, db2 etc. in tungsten.ini,
# you can associate them with their corresponding IP addresses using hostAliases
# - ip: "10.0.0.1"
```

```
# hostnames:
# - "db1"
# - ip: "10.0.0.2"
# hostnames:
# - "db2"
# - ip: "10.0.0.3"
# hostnames:
# - "db3"
```

Important

Dashboard does not install Ingress-Controllers by default! You must install the ingress controller yourself before installing the Dashboard.

Dashboard only deploys an ingress resource to the cluster under its own namespace. Without a controller this resource does nothing.

3.3.3.7.1. Ingress - what is it?

Ingress is a Kubernetes resource that manages external access to services within a cluster, typically HTTP and HTTPS traffic. It provides a single entry point for routing external requests to the appropriate backend services based on rules you define.

When an Ingress resource is deployed, an Ingress Controller (such as NGINX, Traefik, or others) processes it and configures the actual routing logic. The controller watches for Ingress resources and automatically updates its configuration to handle incoming traffic according to the rules specified—such as host-based routing (routing `api.example.com` to one service and `web.example.com` to another) or path-based routing (routing `/api/*` and `/app/*` to different services).

Ingress also handles TLS/SSL termination, allowing you to manage certificates and secure connections at the edge rather than within each individual service. Once installed and configured, the Ingress Controller acts as a reverse proxy and load balancer, directing traffic from outside the cluster to the correct internal services seamlessly.

3.3.3.8. Resources settings

The resources section covers the amount of CPU and memory limits along with persistence configurations for the dashboard. If you do configure the limitations for resources please allocate enough to cover the JVM usage of the dashboard. Minimum of at least 1 gigabyte is recommended depending on the amount of clusters you connect to and the amount of users present.

```
### Resources ###
resources: {}
# We usually recommend not to specify default resources and to leave this as a conscious
# choice for the user. This also increases chances charts run on environments with little
# resources, such as Minikube. If you do want to specify resources, uncomment the following
# lines, adjust them as necessary, and remove the curly braces after 'resources:'.
# limits:
#   cpu: 100m
#   memory: 128Mi
# requests:
#   cpu: 100m
#   memory: 128Mi
# Whether to persist changes to disk or not. The dashboard writes some semi-persistent metadata such as discovered topologies of the Tungsten clusters to
# If you want to persist that over pod restarts, you can enable it here.
#
# In addition, if clusters configuration is not managed declaratively, and you are using edit features within the dashboard UI, only way to persist those
persistence:
# enabled: true ; persistence cannot be disabled as it is required by init containers.
size: 1Gi
# If defined, storageClassName: <storageClass>.
# If set to "-", storageClassName: "", which disables dynamic provisioning.
# If empty or set to null, no storageClassName spec is
# set, choosing the default provisioner (gp2 on AWS, standard on GKE, AWS, and OpenStack).
storageClass: null
# -- Selector for persistent disk
selector: null
# -- Annotations for volume claim
annotations: {}
```

Chapter 4. Updates & Upgrades

Updating and Upgrading Tungsten Dashboard is a simple process and handled by the same `install.pl` command that was used for the initial installation.

When executing `install.pl`, the command will automatically determine whether you are updating an existing installation with new configuration variables or upgrading to a new Tungsten Dashboard release.

Note

Updating your running environment from no TLS (http) to TLS (https) using the Docker installation method is now supported from v8.0.2 onwards. To do this, use the `--reset` flag with the `install.pl` script and specify the updated configuration.

4.1. Updating an existing installation

The `install.pl` command will update an existing configuration if it is executed from within the same directory that was previously used for the initial installation.

The update will detect modifications to the various configuration variables and during the process you will be prompted for the same configuration options as when the installation was performed.

Note

It is recommended to backup your modified config files (`docker-compose.yml` and `.env` file for Docker method and `values.yml` file for kubernetes).

This process could be thought of as the Tungsten Clustering equivalent of `tpm update`

4.2. Upgrading to new Tungsten Dashboard version

The `install.pl` command will upgrade an existing configuration if it is executed from within the new directory which was created by unpacking the software archive..

The upgrade will prompt for the same configuration options as when the initial installation was performed, and detect an existing Tungsten Dashboard version running. You will then be prompted to continue. An example of the process detecting the running dashboard is as follows:

Example of a detected Docker deployment:

```
Docker service 'tungsten-dashboard' is already running.
CONTAINER ID   IMAGE                                COMMAND                                CREATED        STATUS        PORTS        NAMES
52594391c843   tungsten-dashboard:8.0.0            "java -jar /app/dash..."           40 minutes ago Up 40 minutes (healthy)           tungsten-dashboard-se
Do you want to upgrade the existing deployment? [y/N]:
```

Example of a detected kubernetes deployment:

```
Kubernetes deployment 'tungsten-dashboard' already exists.
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
tungsten-dashboard  1/1     1             1           40m14s
Do you want to upgrade the existing deployment? [y/N]:
```

This process could be thought of as the Tungsten Clustering equivalent of `tools/tpm update --replace-release`

Chapter 5. User Guide

Tungsten Dashboard allows you to monitor and operate the clusters in real time.

This browser-based web application gets data from a purpose-built backend, which is designed to efficiently gather data from the cluster nodes independently of the browser requests.

The Dashboard backend serves HTML and React JavaScript to the browser via a built-in JVM-hosted Jetty Server.

The browser-based REACT JavaScript frontend then uses an API to authenticate the user. To provide a smooth user experience, the frontend forms a websocket connection with the backend server to get and display cluster information. For security, each websocket connection must have a valid user login token behind it.

The Dashboard requires you to login with a user account before you can see or act on the clusters themselves.

For first time installations, either the `install.pl` script prompts you for the credentials, or the credentials are included in your helm deployment configuration.

To create additional users, and to get started, see [Section 5.2, "Create your own user account"](#)

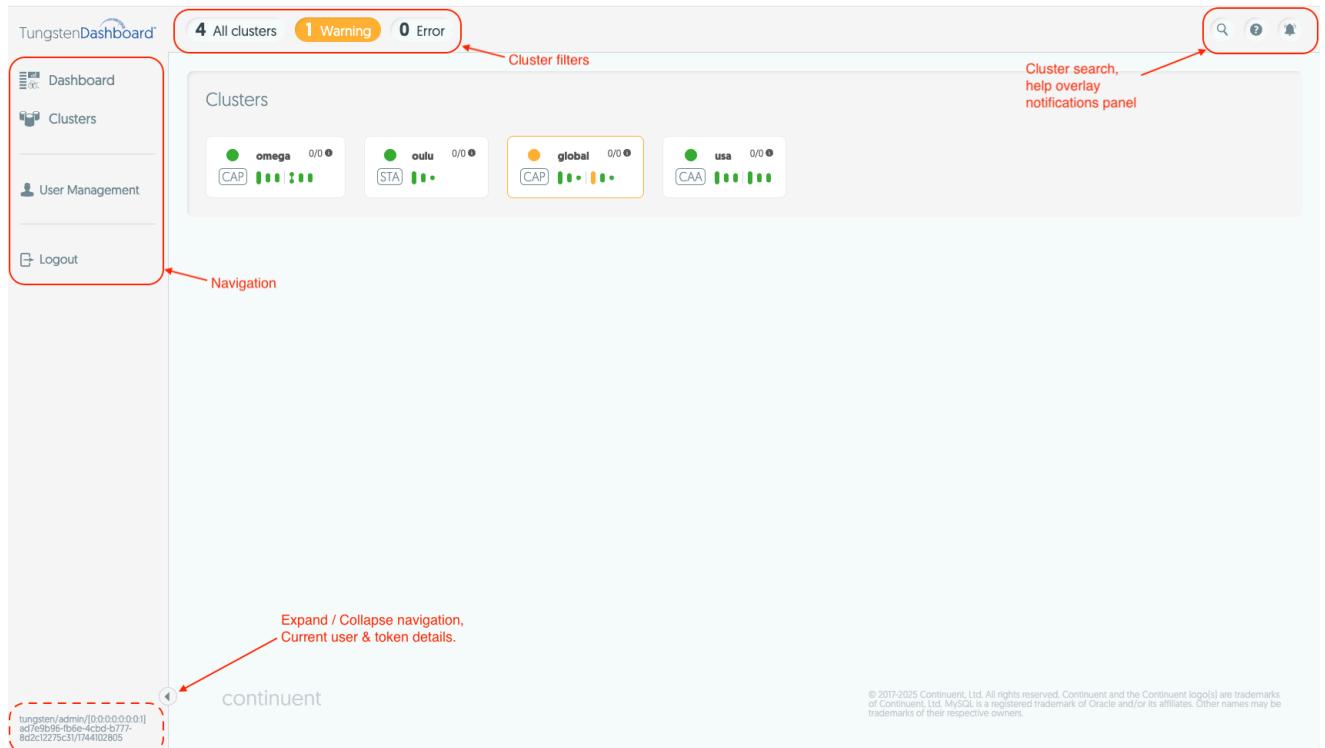
Note

The Dashboard uses the browser's Local Storage feature to save non-sensitive metadata (about expanded services, expanded sidebar, and temporary access token). Please make sure your browser allows storing data using Local Storage or Cookies.

5.1. The Front Page

Once logged in, you are presented with the following view:

Figure 5.1. Dashboard Front Page Overview



An explanation of each highlighted area is as follows:

Navigation

On the left you can see the primary navigation. Hover over it to expand it, and use the button at the bottom of the side bar to lock it in place.

- Dashboard page is the first page opened

- Clusters page is the main monitoring page and page that allows for cluster operations.
- User management allows you to see, create and delete users.
- At the bottom of the navigation you can see the toggle button and current user details.

Cluster Filters

At the top you have the cluster filters. Clicking any of them will take you to the Clusters page and show you the clusters that are reporting the filtered status.

Search & Notifications

On the far right of the top bar you have:

- Search - takes you to the Clusters page while filtering services by their name based on your input.
- Help - this opens an overlay that shows details about elements on the page. Hovering over any element will display more details about it.
- Notification - clicking this will open the notification messages sidebar on the right.

5.2. Create your own user account

To start using the Dashboard the first recommended action is to create your own user account. Login with the installation default account, then open the sidebar by hovering over it and click "User Management". On this page click the "Add" button and fill in the details of the new account. Next, click the logout button and log back in using the account you just created.

In Summary:

1. Login using the installation default account
2. Navigate to "User Management"
3. Click add
4. Fill in details
5. Click save
6. Logout, and Login again using your new account

Figure 5.2. Add User Form

The screenshot displays the TungstenDashboard interface. At the top, it shows '4 All clusters', '1 Warning', and '0 Error'. The main content area is titled 'Users' and contains a table with the following data:

Role	Username	Email	Actions
Admin	tungsten	admin@admin.com	[Delete]
Admin	jane	jane-doe@example.com	[Delete]

Below the table is a modal form titled 'Add new user' with the following fields:

- Username: my-admin-account
- Email: my-account@company.com
- User Password: [masked]
- Role: admin

Buttons for 'Cancel' and 'Save' are located at the bottom of the modal. The footer of the dashboard includes the Continuent logo and copyright information: © 2017-2025 Continuent, Ltd. All rights reserved. Continuent and the Continuent logo(s) are trademarks of Continuent, Ltd. MySQL is a registered trademark of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Note

Roles cannot be changed in v8.0.0 - this feature will be implemented in a future update.

User accounts are stored in a file on the Dashboard server. This file is intended to be stored in the persistent volume of the Dashboard. In that file you'll find the email, username and hash of the password. Future logins work by matching the incoming hash against the stored one.

Each login returns a JWT token that the browser uses to communicate with the backend. This signed token contains basic JWT details and the username. While the content of the token can be decoded with base64, altering the token in any way will cause the backend to reject it. When a token is rejected, the backend closes the websocket session from the browser.

Note

Every token also has a UUID value. This UUID changes every time a user logs out. Forcing a logout deletes all tokens for that user. This means that the logout effectively signs the user out of all devices.

5.3. Add Clusters

Once you've logged in, click the "Add cluster" button on the front page (see [Figure 5.3, "Cluster Front Page - Add Cluster"](#)) or the clusters page. This opens up a form which asks for the following details:

- Initial hostname - i.e. db101.continuent.com
- Namespace - optional tag for the cluster that can be used as a filter.
- Rest API Port - optional port number if different from the default Manager Rest API port [8090].
- TCP Port - optional TCP port for the Tungsten Router Gateway protocol if different from the default [11999].
- Rest API and TCP SSL - Check the desired boxes to enable or disable SSL for each.

If enabled, a certificate must be provided. You can fetch both needed certificates using the provided [getcert.pl](#) script included in the installation package, or perform the steps manually.

Warning

Tungsten Cluster v8.0.0 uses different certificates for the API and TCP. Please fetch the certificate separately for each.

- Credentials - add the RestAPI credentials of the v8 API for your cluster here.
- Test connection - once you've filled the needed details click the test connection button to see if the dashboard can connect to the cluster given. [See [Figure 5.4, "Add Cluster Form"](#)]
- Finally click save to add the cluster to the Dashboard.

Figure 5.3. Cluster Front Page - Add Cluster

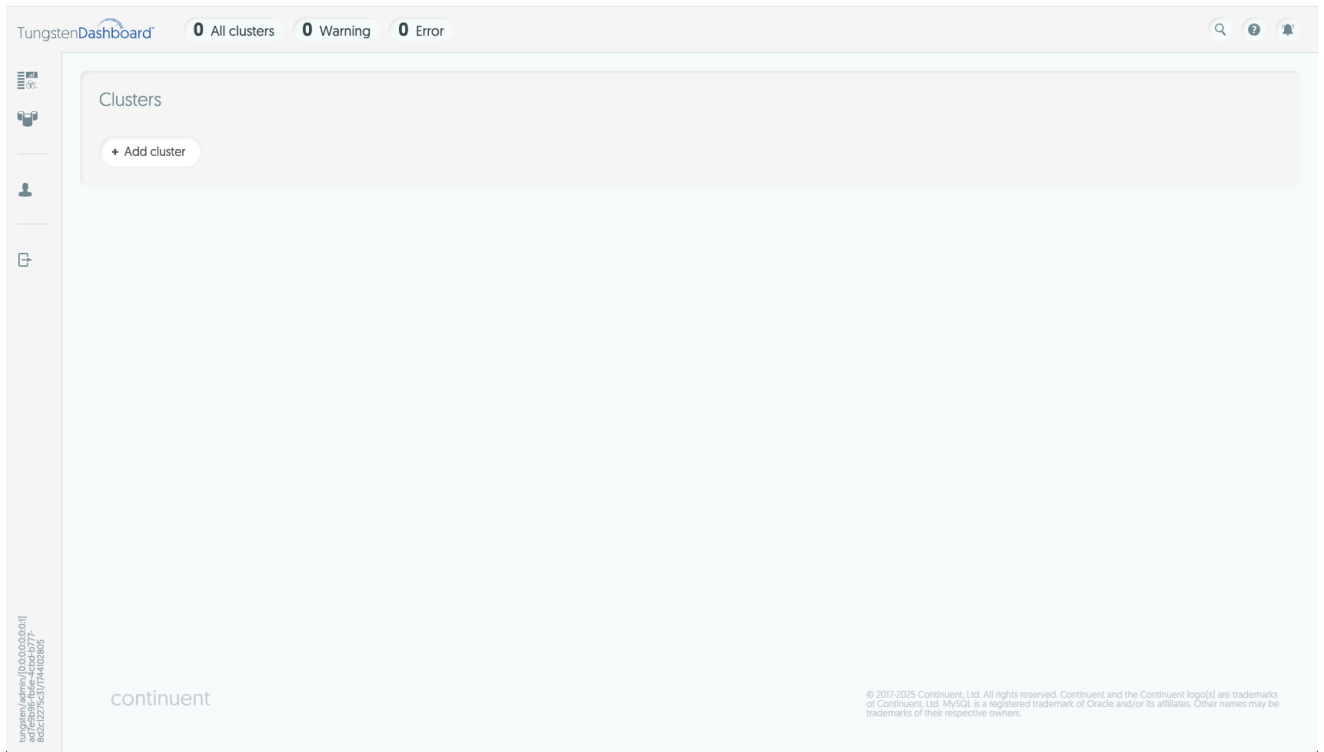
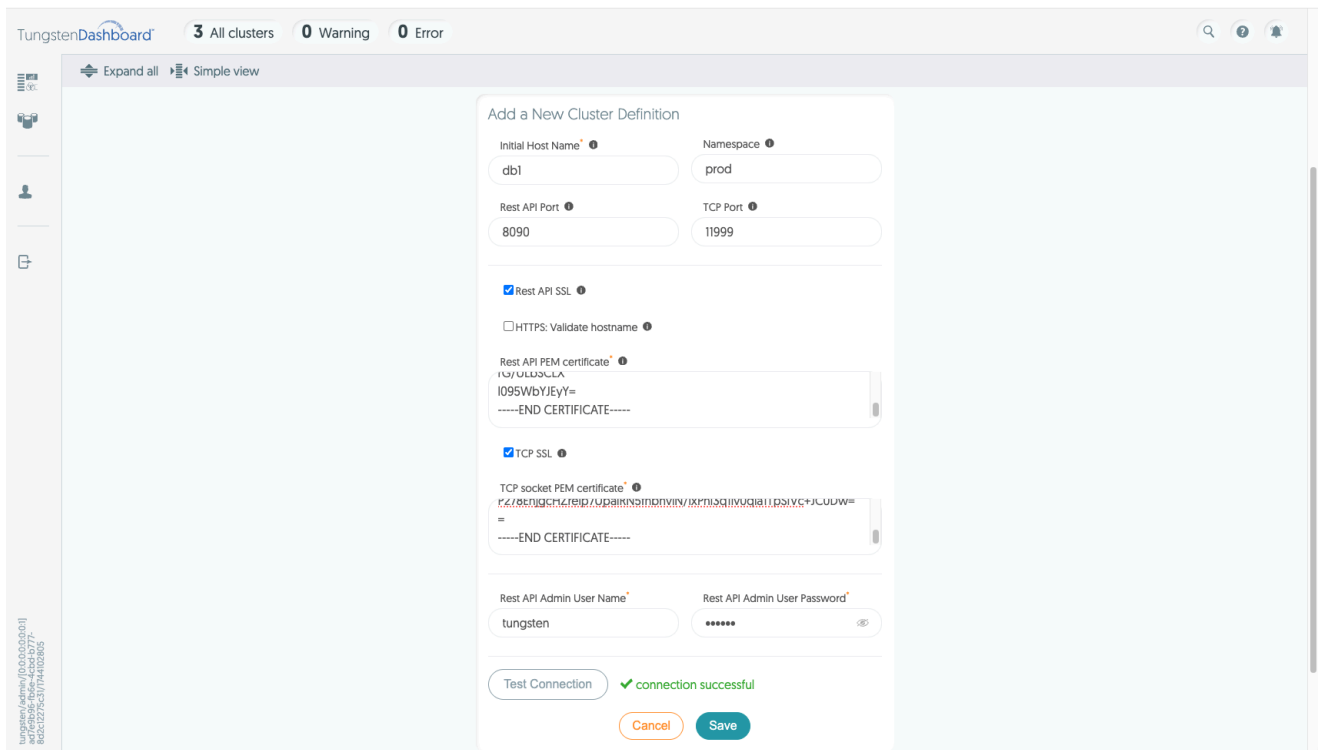


Figure 5.4. Add Cluster Form



Note

Troubleshooting Tip

Dashboard works by retrieving the cluster's topology from the `/api/v8/manager/cluster/topology` endpoint. This endpoint returns host names that the Dashboard attempts to connect to through its internal load balancer. These host names MUST be resolvable by the Dashboard instance. Otherwise the Dashboard will not be able to connect even if the first connection from the connection test works.

When cluster connection details are added to the Dashboard, the changes are written to both the `clusters.json` file and into the Dashboard's active memory. The Dashboard's Topology sub-system is aware when these changes occur, and will gather the new details from the cluster via the RestAPI, then store the response into the `/app/persistent/dynamic/clusters.edn` file, which keeps a map of the hosts by cluster.

When the Dashboard connects to a cluster, one host is selected for each service in the `clusters.edn` file, and that host is then called via the Rest API to get the initial data for the cluster. This is followed by a TCP connection to the cluster which updates the data in real time.

Each cluster connection is hosted in a virtual thread inside the JVM. If an error occurs, the Dashboard attempts to select a new host and restart the cluster connection unless the error is significant enough (i.e. "invalid certification" or credential data), at which point the Dashboard stops further connection attempts.

5.4. Monitor Clusters

Once you've added the desired clusters, you can use the Dashboard or Clusters pages to monitor the status of these clusters in real-time. While the Dashboard page provides limited data, the Clusters page allows you to view detailed information and execute corrective actions to a cluster as needed.

Important

While the Dashboard has automatic and manual lock functionality that prevents other Dashboard users from activating commands to the cluster, this functionality does NOT prevent users of the `cctrl` cli tool from executing actions in the cluster. It is critical that you always communicate clearly with your team about any actions you're about to take in a cluster with either the Dashboard or via `cctrl`.

Chapter 6. User Interface

The Dashboard user interface is designed for optimal readability and intuitive controls, all aimed at delivering a positive user experience.

In this chapter, you'll find a breakdown of the features available on each page of the Dashboard. For detailed instructions on how to perform specific actions, please refer to [Chapter 5, User Guide](#)

6.1. Front Page Interface

From this page you can find a quick status overview of any of the clusters you have connected the Dashboard to. Clusters are split into small "widgets" on the page. [See [Figure 6.1, "Front Page Interface"](#)]

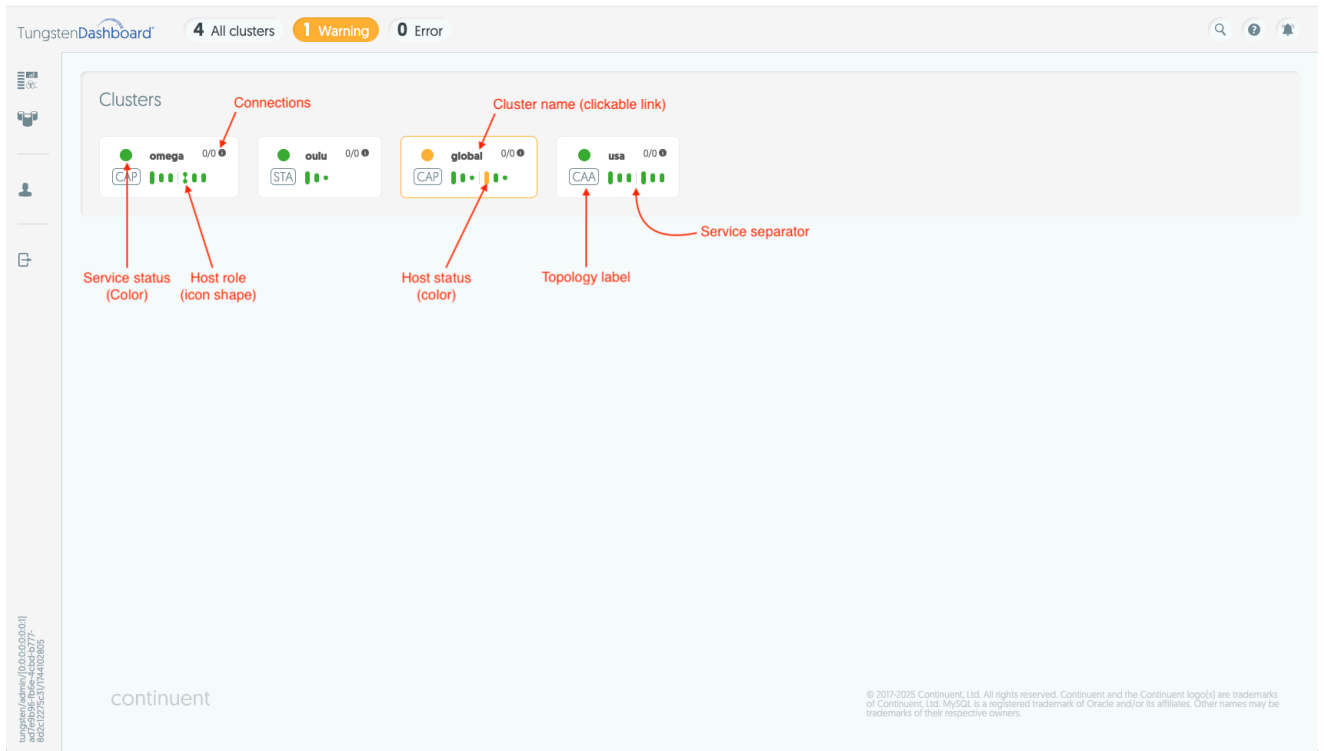
Each widget shows:

- Status of the service with in the border and status icon color
 - Green - ok
 - Yellow - warning
 - Red - error
- A clickable cluster name that takes you to the Clusters page to see more details about that cluster
- The number of connections to each cluster broken down by host
- The topology of the cluster:
 - STA - Standalone
 - CAP - Composite Active/Passive
 - CAA - Composite Active/Active (and Composite Dynamic Active/Active)
- The role of the nodes for each cluster are indicated by the icon size:
 - The full-size icon indicates a "Primary" node
 - The medium-sized icon is a "Replica" node
 - The small dot is a "Witness" node

Refer to the help overlay of the Dashboard for a break-down of the icon information or Tungsten Cluster documentation for a breakdown of the roles.

- Each service for a composite cluster is separated by a small vertical line in this widget.

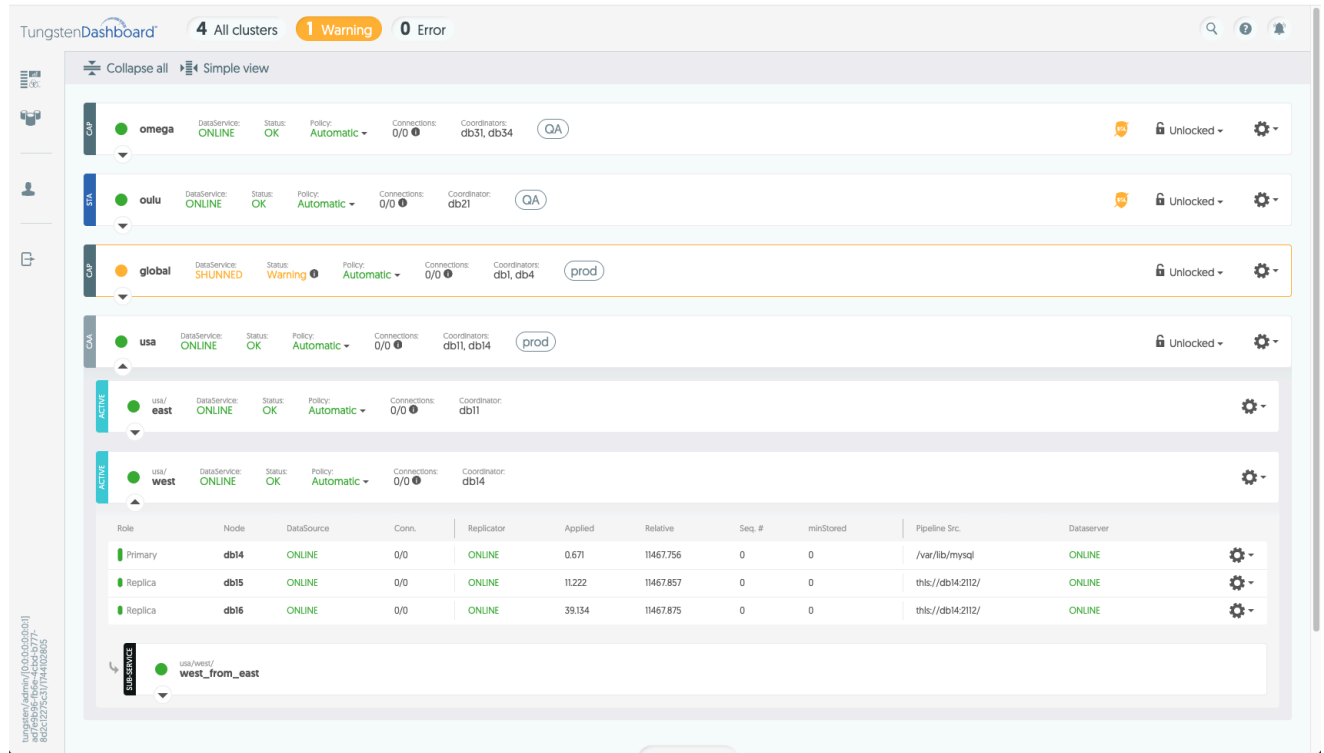
Figure 6.1. Front Page Interface



6.2. Clusters Interface

On the Clusters page you can review your cluster status details, execute cluster actions and add or remove clusters from the Dashboard. [See Figure 6.2, "Populated Cluster Front Page"]

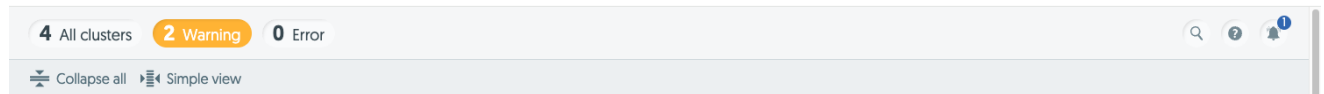
Figure 6.2. Populated Cluster Front Page



The first tools available to you on the Clusters page are the Collapse/Expand and Simple/Advanced View toggle buttons just below the top navigation bar. (See Figure 6.3, "Cluster Tool Bar")

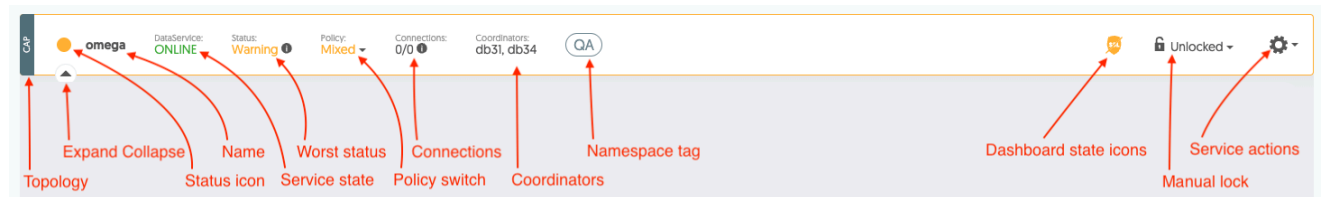
Collapse/Expand opens or closes all clusters visible on the page, and the Simple/Advanced View toggle button switches which columns are shown to you.

Figure 6.3. Cluster Tool Bar



Below the toolbar you can find the clusters themselves:

Figure 6.4. Cluster Header Detail



A description of each item is as follows:

- Topology label - the same three-letter abbreviations that the front page has (STA, CAP, CAA)
- Status icon - colored based on the worst status of any of the cluster's hosts (green, yellow or red)
- Cluster name - clickable link that hides other clusters
- Service state* - The state of the cluster's parent service

- Status - the Dashboard's calculated status for the cluster based on the status of the worst element. Placing your mouse over the info icon next to this status will give you the details why the cluster was given that status
- Policy - shows the current policy of the cluster, and clicking this element allows you to change it. There are three possible states: Automatic, Mixed and Maintenance. For up to date documentation about these states, please refer to the [Tungsten Clustering documentation](#)
- Connections - Number of connections to the cluster; hover to see connections broken down by host
- Coordinator[s] - the Cluster's current coordinator[s]
- Namespace label - Clickable link which limits the shown clusters to those clusters with the same namespace label.
- Dashboard state icons - These optionally visible elements alert you if there is anything that requires your attention in the connection from the Dashboard to the configured cluster.
- Manual Lock - Dropdown which allows you to lock the cluster, which prevents other users of the Dashboard from performing any actions on that cluster while the lock is active. Use this in combination with Maintenance mode to prevent other users from acting on a cluster that you're working on
- Action dropdown - The gear icon provides access to a variety of controls for controlling the cluster service. [See [Figure 6.5, "Parent Service Actions/Operations"](#)] Note that the content of this dropdown changes based on the type of cluster connected to. You can find more details in [Chapter 9, Operations](#) and for additional information about individual operations, please refer to the [Tungsten Clustering documentation](#).

Figure 6.5. Parent Service Actions/Operations

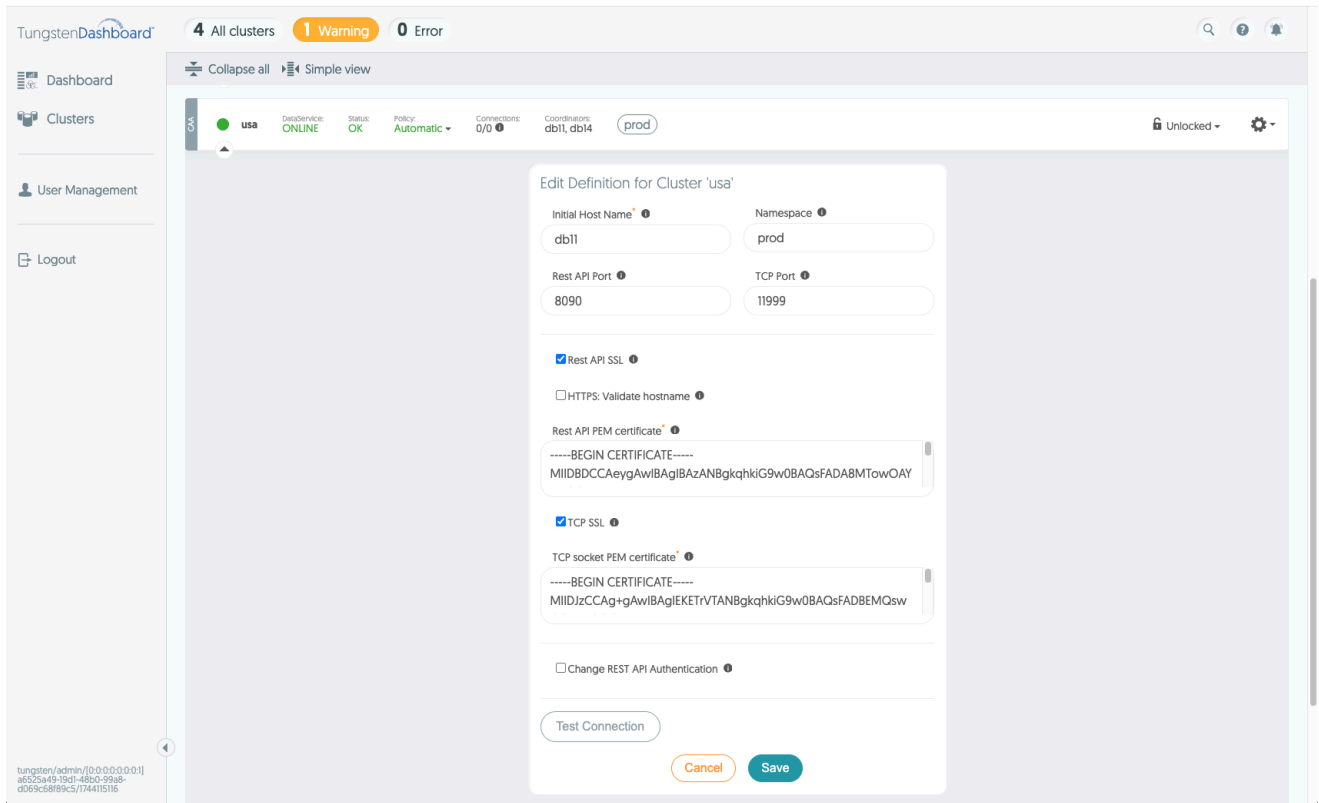
The screenshot displays the Tungsten Clustering Dashboard interface. At the top, there are three cluster cards: 'omega', 'alpha', and 'beta'. The 'alpha' card is highlighted as 'ACTIVE'. Below the 'alpha' card is a table with the following columns: Role, Node, DataSource, Conn., Replicator, Applied, Relative, Seq. #, minStored, Pipeline Src., and Dataserver. The table contains three rows: a Primary role on node db31, and two Replica roles on nodes db32 and db33. All roles and nodes are in an 'ONLINE' state. A dropdown menu is open over the 'alpha' card, showing 'Dataservice operations' (Heartbeat, Recover, Switch, Failover) and 'Configuration options' (Reload Topology, Edit Definition, Remove Definition).

Role	Node	DataSource	Conn.	Replicator	Applied	Relative	Seq. #	minStored	Pipeline Src.	Dataserver
Primary	db31	ONLINE	0/0	ONLINE	0.678	12711.462	0	0	/var/lib/mysql	ONLINE
Replica	db32	ONLINE	0/0	ONLINE	20.541	12711.096	0	0	thl://db31:2112/	ONLINE
Replica	db33	ONLINE	0/0	ONLINE	39.122	12711.153	0	0	thl://db31:2112/	ONLINE

Following the cluster service actions are the Dashboard-specific actions for each service:

- Reload topology - Clears cluster data from the Dashboard and fetches the topology from cluster's REST API then re-connects the TCP session with the cluster.
- Edit cluster definition - Opens the Edit Cluster form so that you can alter any of the cluster's configured connection details. [See [Figure 6.6, "Edit Cluster Form"](#)]
- Remove cluster - Deletes the cluster from the Dashboard

Figure 6.6. Edit Cluster Form



The cluster's host details are displayed as a table. (See [Figure 6.7, "Cluster Host Table \[CAA\]"](#)) It has fields such as:

- Role
- Name
- Datasource status
- Connection count
- Replicator status
- Replicator applied latency
- Replicator relative latency
- Current Sequence number
- Minimum Stored Sequence number
- Pipeline Source
- Dataserver status
- Action dropdown.

Figure 6.7. Cluster Host Table (CAA)

The screenshot shows the TungstenDashboard interface with the following data:

Dashboard Summary: 4 All clusters, 1 Warning, 0 Error

Cluster: usa (DataService: ONLINE, Status: OK, Policy: Automatic, Connections: 0/0, Coordinators: db11, db14, prod)

Role	Node	DataSource	Conn.	Replicator	Applied	Relative	Seq. #	minStored	Pipeline Src.	Dataserver
Primary	db11	ONLINE	0/0	ONLINE	1.27	12437.276	0	0	/var/lib/mysql	ONLINE
Replica	db12	ONLINE	0/0	ONLINE	15.615	12435.589	0	0	this://db12:2112/	ONLINE
Replica	db13	ONLINE	0/0	ONLINE	53.17	12435.607	0	0	this://db12:2112/	ONLINE

Cluster: east (DataService: ONLINE, Status: OK, Policy: Automatic, Connections: 0/0, Coordinator: db11)

Role	Node	DataSource	Conn.	Replicator	Applied	Relative	Seq. #	minStored	Pipeline Src.	Dataserver
Replica	db13	ONLINE	0/0	ONLINE	0.805	12349.607	0	0	this://db11:2113/	-
Relay	db11	ONLINE	0/0	ONLINE	0.802	12351.276	0	0	this://db14:2112/	-
Replica	db12	ONLINE	0/0	ONLINE	0.85	12349.589	0	0	this://db11:2113/	-

Cluster: west (DataService: ONLINE, Status: OK, Policy: Automatic, Connections: 0/0, Coordinator: db14)

Footer: tungsten/admin/[0.0.0.0:0.0.0.1] a6325a49-19d1-48b0-99a8-d069c88f89c5/1744115116 © 2017-2025 Continuent, Ltd. All rights reserved. Continuent and the Continuent logo(s) are trademarks

Note

Please note that the role of the host affects what data is available in any given row. Refer to Tungsten Cluster documentation for up-to-date information about the data contained in each of these fields.

In the host row actions dropdown (See [Figure 6.8, "Primary Host Operations"](#) & [Figure 6.9, "Replica Host Operations"](#)) you can find a set of operations the Dashboard can post to the cluster's REST API for each of the hosts. Note that the host's Role affects what actions are visible. To see more details about operations see [Chapter 9, Operations](#)

Figure 6.8. Primary Host Operations

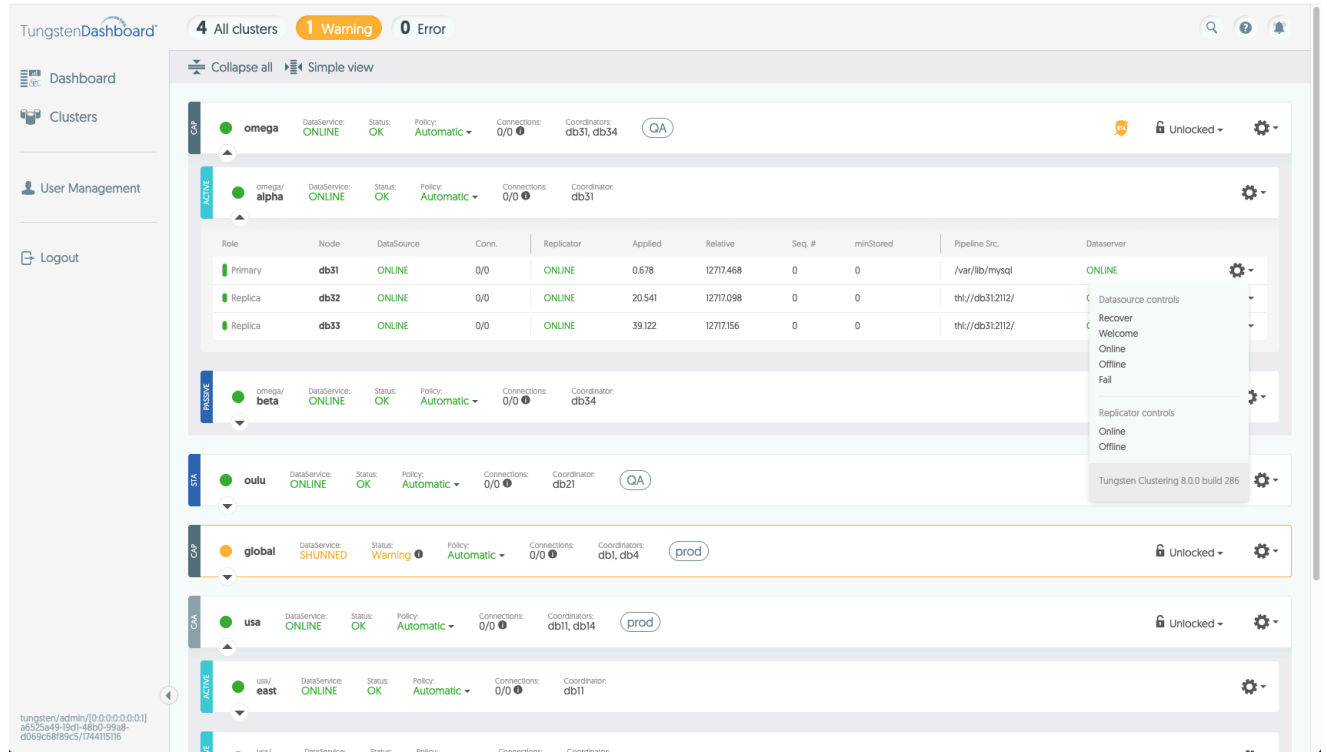
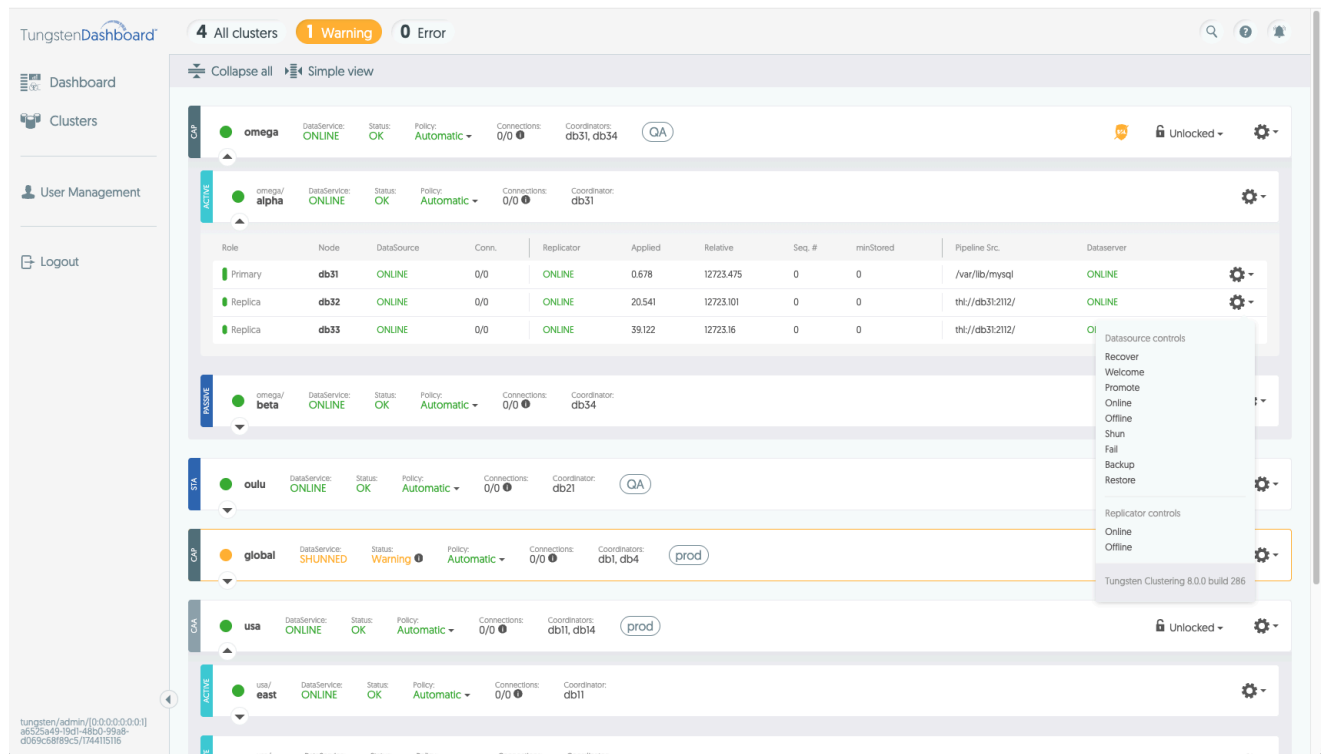


Figure 6.9. Replica Host Operations



6.3. User Management Interface

This page lists the current users stored in the Dashboard with their role, username and email. [See [Figure 6.10, "User Management Page"](#)]

From here you can add or remove users as you desire. [See [Figure 6.11, “Add User Form”](#)]

Note

In version 8.0.0 only the admin role is supported and it cannot be changed.

This feature will be introduced in a future update.

Figure 6.10. User Management Page

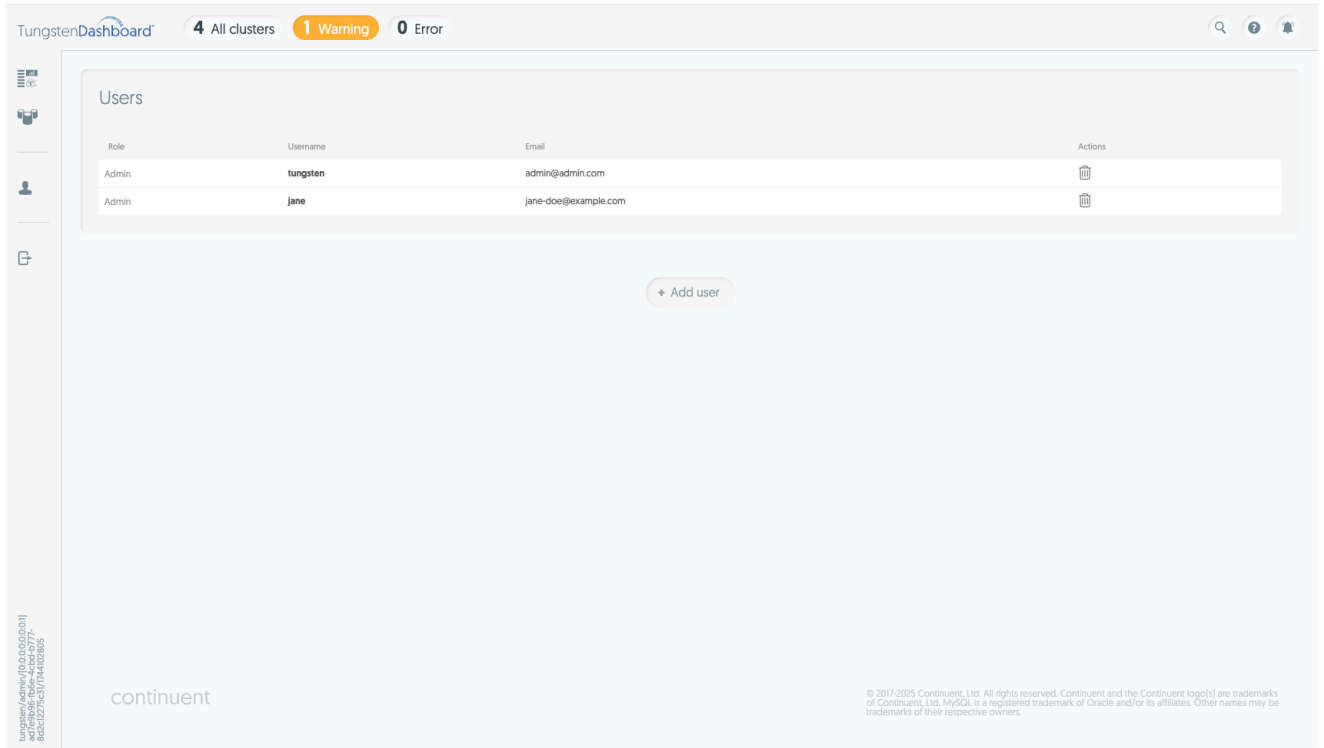
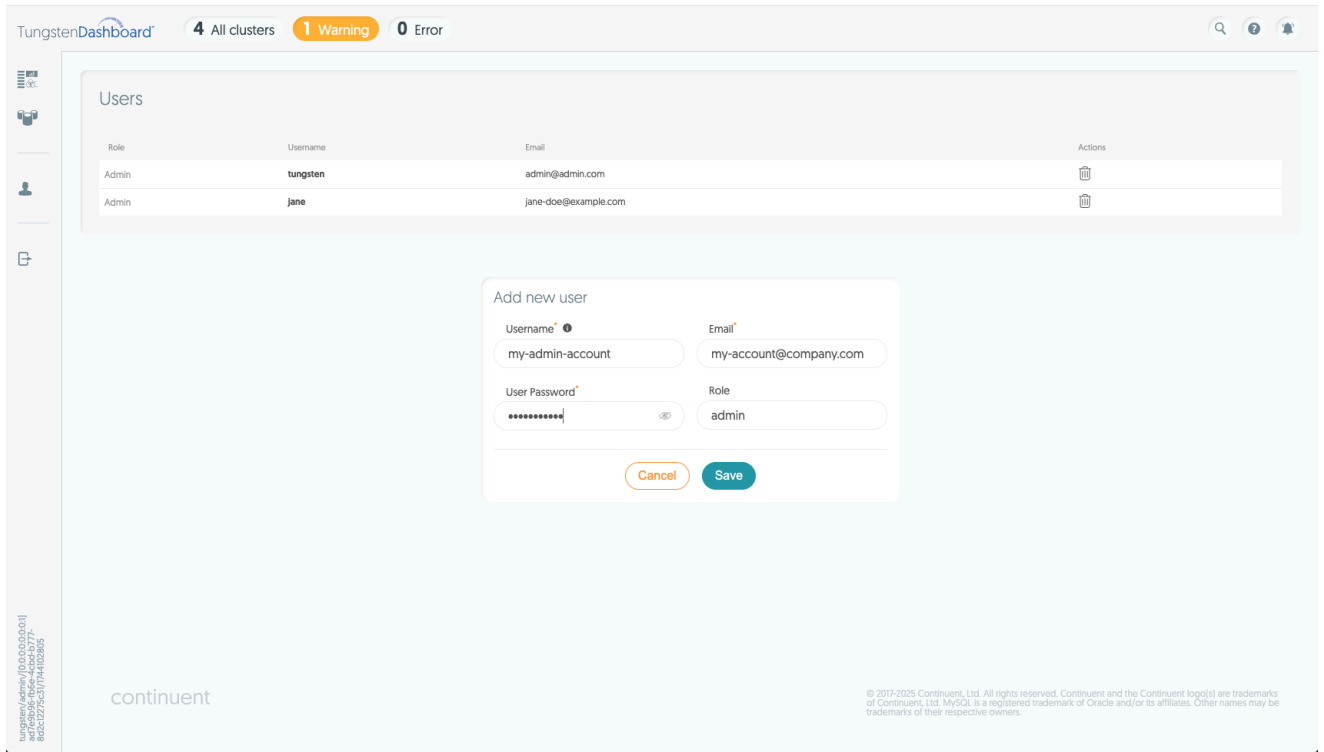


Figure 6.11. Add User Form



Important

In the first release of the Tungsten Dashboard (v8.0.0), users are not able to change or reset passwords. Please re-create the user if original password is lost or needs to be changed.

Chapter 7. Command Line Tools

7.1. install.pl script

The `install.pl` script is a command-line tool for installing, updating and upgrading the Tungsten Dashboard. This document provides a detailed guide on how to use this script.

Usage

```
install.pl [ --docker, --docker-compose, -d ] [ --help, -h ] [ --kubernetes, --k8, -k ] [ --method, -m ] [ --reset ]
```

Where:

Table 7.1. `install.pl` Options

Option	Description
<code>--docker, --docker-compose, -d</code>	Specify the docker-compose deployment method (The same as issuing <code>-m docker-compose</code>)
<code>--help, -h</code>	Use this option to display the help message.
<code>-k, --k8, --kubernetes</code>	Specify the kubernetes deployment method (The same as issuing <code>-m kubernetes</code>).
<code>--method, -m</code>	Specify the deployment method (docker-compose or kubernetes).
<code>--reset</code>	Reset config files from .sample versions before installation.

For example, to install using Docker, you would run:

```
shell> ./install.pl -m docker-compose
```

To install using Kubernetes, you would run:

```
shell> ./install.pl -m kubernetes
```

If no method is specified, the script will prompt you to choose a method interactively.

Prerequisites check

The script checks for prerequisites based on the selected installation method:

- Docker for the Docker method.
- Kubectl and Helm for the Kubernetes method.

If the required tools are not found, the script will display an error message and exit. For more information on prerequisites, see [Chapter 2, Prerequisites](#)

Interactive Prompts

During execution, the script will prompt you for various pieces of information, such as the admin username and password, the domain for your application, and the path for your application. Please follow the prompts and provide the necessary information.

For details on Updating and Upgrading an existing installation, see [Chapter 4, Updates & Upgrades](#)

7.2. getcert.pl script

`getcert.pl` is a script designed to fetch certificates from a specified remote host. Once fetched, these certificates can be copied and pasted into the dashboard form, or included in the `values.yaml` during installation.

Usage

```
./getcert.pl [--api|--tcp|--help] [--port PORT] [--base64] {HOSTNAME|HOSTNAME:PORT}
```

Where:

Table 7.2. `getcert.pl` Options

Option	Description
<code>--api, -a</code>	Use this option to fetch the Manager API certificate on the default port 8090

Option	Description
<code>-b, --base64</code>	Output the certificate encoded in base64.
<code>--help, -h</code>	Use this option to display the help message.
<code>--port, -p</code>	Use this option to specify the port - requires either <code>--api</code> or <code>--tcp</code> argument.
<code>--quiet, -q</code>	Use this option to suppress verbose output and display only the certificate content.
<code>--tcp, -t</code>	Use this option to fetch the TCP certificate on the default port 11999

Examples

Fetch the API certificate from a host:

```
shell> ./getcert.pl --api db1.continuent.com

Fetching API certificate from db1.continuent.com:8090 ...

-----BEGIN CERTIFICATE-----
MIIDBDCCAeygAwIBAgIBAzANBgkqhkiG9w0BAQsFADABMTowOAYDVQQDDDFNeVNR
... rest of certificate content ...
LwZn40Kuwnk=
-----END CERTIFICATE-----

Please copy this API certificate (including BEGIN and END markers) into the UI form
```

Fetch the TCP certificate from a host on a specific port:

```
shell> ./getcert.pl --tcp --port 11999 {HOSTNAME}
```

Redirecting Output to a File

The output of the script can be redirected to a file using the `>` operator. For example:

```
shell> ./getcert.pl --api {HOSTNAME} > cert.txt
```

This will save the certificate content into `cert.txt` without any additional verbose output.

Fetch base64 encoded API certificate from a host for a kubernetes `values.yaml` file:

```
shell> ./getcert.pl --api db1.continuent.com --base64

Fetching API certificate from db1.continuent.com:8090 ...

LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSB0tLS0tCk1JSURKekNDQWcrZ0F3SUJBZ01
FS0VUc1ZUQU5CZ2txaGtpRzl3MEJBUXNQRURCRU1Rc3dDUVlEVlFRR0V3S1YKVX
... rest of base64 certificate content ...
```

Please copy this API certificate encoded in base64 into the `values.yaml` configuration file

7.3. letsencrypt2jetty.pl script

`letsencrypt2jetty.pl` is a script that converts a Let's Encrypt-provided certificate into a Dashboard-ready JKS file.

To use this tool you will need to ensure you have the `fullchain.pem` and `privkey.pem` files from Let's Encrypt available.

Usage

```
letsencrypt2jetty.pl [ --domain, -d ] [ --fullchain, -f ] [ --help, -h ] [ --key, -k ] [ --password, -p ]
```

Where:

Table 7.3. `letsencrypt2jetty.pl` Options

Option	Description
<code>--domain, -d</code>	Specify the FQDN used for the certificate; if provided will define the <code>fullchain.pem</code> fullpath as: <code>/etc/letsencrypt/live/{domain}/fullchain.pem</code> and will define the <code>privkey.pem</code> fullpath as: <code>/etc/letsencrypt/live/{domain}/privkey.pem</code>
<code>--fullchain, -f</code>	Specify the Let's Encrypt <code>fullchain.pem</code> fullpath

Option	Description
--help, -h	Use this option to display the help message.
--key, -k	Specify the Let's Encrypt privkey.pem fullpath
--password, -p	Specify the keystore password

Below is an example of converting an existing cert issued by Let's Encrypt using our provided tool `letsencrypt2dashboard.pl`, which calls the `openssl` and Java `keytool` commands:

```
shell> sudo ./letsencrypt2jetty.pl -d dashdev.continuent.com
Keystore file's password: tungsten
Creating './cert/'
>>> ACTION: Converting the Let's Encrypt source files to P12 format:
SOURCE:
  /etc/letsencrypt/live/dashdev.continuent.com/fullchain.pem
  /etc/letsencrypt/live/dashdev.continuent.com/privkey.pem
TARGET
  cert/jetty.p12

SUCCESS: Generated the P12 file 'cert/jetty.p12'

>>> ACTION: Converting the P12 file to JKS format:
SOURCE
  cert/jetty.p12
TARGET
  cert/jetty.jks

Importing keystore cert/jetty.p12 to cert/jetty.jks...

SUCCESS: Generated the internal jetty cert file 'cert/jetty.jks'

shell> sudo chown -R dashboard: cert
```

Or:

```
shell> mkdir cert
shell> cp /etc/letsencrypt/live/dashdev.continuent.com/fullchain.pem cert/
shell> sudo cp /etc/letsencrypt/live/dashdev.continuent.com/privkey.pem cert/
shell> sudo chmod a+r cert/privkey.pem
shell> ./letsencrypt2jetty.pl -f cert/fullchain.pem -k cert/privkey.pem

Keystore file's password: tungsten
Creating './cert/'
>>> ACTION: Converting the Let's Encrypt source files to P12 format:
SOURCE:
  ./fullchain.pem
  ./privkey.pem
TARGET
  cert/jetty.p12

SUCCESS: Generated the P12 file 'cert/jetty.p12'

>>> ACTION: Converting the P12 file to JKS format:
SOURCE
  cert/jetty.p12
TARGET
  cert/jetty.jks

Importing keystore cert/jetty.p12 to cert/jetty.jks...

SUCCESS: Generated the internal jetty cert file 'cert/jetty.jks'
```

7.4. collect-diag.pl script

Note

This new script was introduced with v8.0.2

`collect-diag.pl` is a script designed to collect all the necessary information, such as logs and files about the running installation.

The resulting file can then be supplied to Continuent Support to assist with issues that you may be having.

Chapter 8. Establishing Connectivity to Tungsten Clusters

The Tungsten Dashboard can be connected to any IP address or hostname/Fully Qualified Domain Name (FQDN) of any Tungsten cluster node. Upon establishing the initial connection, the system performs a topology discovery process. This process generates a list of dataser-vices (clusters) and datasources (nodes) within the Tungsten cluster.

Subsequent connection attempts are made to the node names that were returned during the topology discovery process. These node names must be Fully Qualified Domain Names (FQDNs) or be otherwise resolvable by the Dashboard backend process.

Warning

Please make sure that you have hostname resolution for all nodes via `/etc/hosts` or DNS set up and tested before trying to add the cluster to the Dashboard. The initial host plus all other cluster nodes must be reachable from the server hosting the Dashboard. Make sure that `/etc/tungsten/tungsten.ini` is configured consistently, and if using short hostnames, then the `hostname -f` command should always return the FQDN.

In cases where the node names are not directly resolvable, a mechanism is provided to create a mapping between the node names and their corresponding IP addresses. This mechanism is included as part of the Helm chart or Docker installation processes.

To utilize this feature, please refer to the `values.yaml` file in the case of a Helm chart installation, or the `docker-compose.yml` file for a docker-compose installation.

The Dashboard uses a custom communication protocol via a TCP socket. Please be aware that launching and connecting a Dashboard server to a Tungsten Cluster will cause the Dashboard to show up as a router in the `ctrl` command, the APlv8 endpoint `/api/v8/manager/cluster/topology`, as well as the APlv2 endpoint.

The Dashboard will not behave like the other routers in this list, since it can only communicate via the TCP connection between the cluster and the Dashboard. While the Dashboard will acknowledge any commands sent to it via the TCP connection, it will only act on a very select few of them. Direct TCP connection to the Dashboard server is NOT supported.

Operations executed via the Dashboard are effectively the same as sending REST API requests directly to the cluster. For more information about how to best manage the impact of the Dashboard on the clusters it is monitoring, please see [Chapter 10, Best Practices](#)

8.1. Configuring SSL when Deploying with Kubernetes

When deploying through Kubernetes there are two options available, these are as follows:

8.1.1. Basic SSL Connection To REST API only

```
...
api-ssl: true
api-cert: "<base64-encoded-certificate>"
ssl: false
...
```

The above setup allows HTTPS connections to the socket without a certificate for the TCP connection to the cluster.

To enable TCP SSL you must provide a certificate for the TCP connection.

8.1.2. SSL with Self-Signed Certificates

```
...
api-ssl: true
api-cert: "<base64-encoded-certificate>"
ssl: true
cert: "<base64-encoded-certificate>"
hostname-validation: false
...
```

For a self-signed certificate setup, provide both certificates as base64-encoded values and disable the hostname validation for the HTTPS protocol. `hostname-validation` can be left on for certificates signed by a trusted authority.

To gather the certificates, you should use the [getcert.pl](#) tool.

The `hostname-validation` setting can be configured in three ways:

1. Per cluster in `clusters.json` using the `hostname-validation` field
2. Globally in `DASHBOARD_HOSTNAME_VALIDATION` environment variable.

3. Globally in `config.json` using the `hostname-validation` field
4. If none are specified, it defaults to `true`

Priority order:

1. Cluster-specific setting in `clusters.json` (highest priority)
2. `DASHBOARD_HOSTNAME_VALIDATION` environment variable.
3. Global setting in `config.json`
4. Default value of `true` (lowest priority)

When hostname validation is:

- Enabled: The certificate's hostname must match the server's hostname.
- Disabled: The certificate is still verified, but the hostname matching is skipped. This is mostly the scenario for self-signed certificates.

Important

When hostname validation is disabled (either per cluster or globally), a certificate must be provided:

- If `hostname-validation=false` is set for a specific cluster, that cluster must include a `cert` field
- If `hostname-validation=false` is set globally in `config.json`, all SSL-enabled clusters must include a `cert` field
- This requirement ensures secure connections even when hostname matching is skipped

Chapter 9. Operations

Tungsten Dashboard currently supports three different kinds of operations/actions with a Tungsten Cluster:

- Service-level
- Host-level
- Dashboard-specific

All actions from the Tungsten Dashboard use the service's REST API to execute and monitor the status of these operations.

9.1. Service Operations

These actions can be used on standalone or composite services. They include `ctrl` operations like `heartbeat`, `recover`, `switch` and `failover` as well as changing the cluster's policy.

To change the policy for a service, click on the current policy displayed in the service header and then click on the desired new policy in the dropdown. In a composite cluster, you can set a different policy for each service individually.

To execute service operations do the following:

1. Navigate to the clusters page
2. Find the cluster you want to execute an operation on
3. On the header row, click the "Policy" label and choose "Maintenance" [See [Figure 9.1, "Service Policy Dropdown"](#)]
4. Click the "unlocked" text and choose "lock" [See [Figure 9.2, "Cluster Lock Dropdown"](#)]
5. For composite clusters: expand the cluster using the down arrow on the left until you can see the service you are looking for
6. Click the gear icon for the service to open the cluster operations menu for that service [See [Figure 9.3, "Composite Service Operations"](#) & [Figure 9.4, "Service Operations"](#)]
7. Select the operation you want to perform

Figure 9.1. Service Policy Dropdown

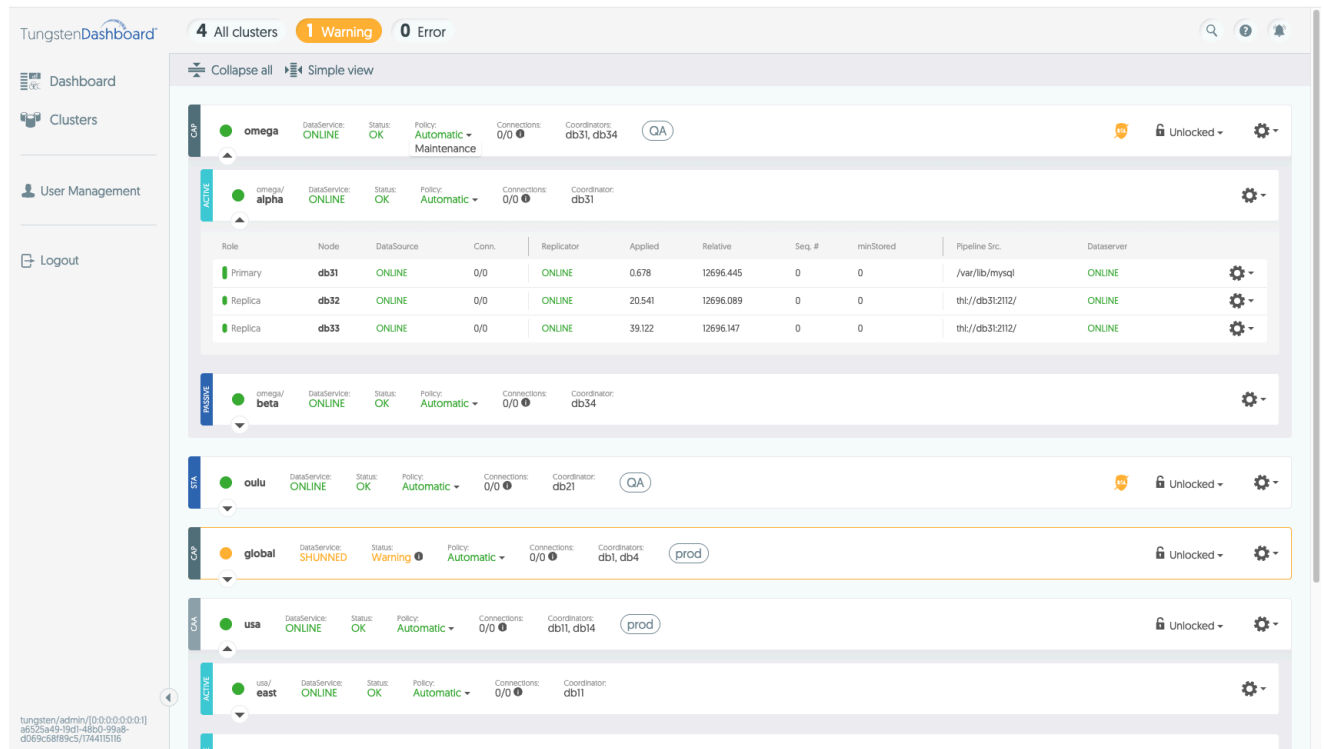


Figure 9.2. Cluster Lock Dropdown

The screenshot shows the TungstenDashboard interface with a sidebar on the left containing 'Dashboard', 'Clusters', 'User Management', and 'Logout'. The main area displays a list of clusters. The 'global' cluster is highlighted with a yellow border and has a 'Warning' icon. A dropdown menu is open for this cluster, showing options: 'Unlocked' (selected), 'Lock', and 'QA'. Below the cluster list, a table shows the details for the 'alpha' cluster.

Role	Node	DataSource	Conn.	Replicator	Applied	Relative	Seq. #	minStored	Pipeline Src.	Datasever
Primary	db31	ONLINE	0/0	ONLINE	0.678	12702.45	0	0	/var/lib/mysql	ONLINE
Replica	db32	ONLINE	0/0	ONLINE	20.541	12702.092	0	0	thl://db31:2112/	ONLINE
Replica	db33	ONLINE	0/0	ONLINE	39.122	12702.149	0	0	thl://db31:2112/	ONLINE

Figure 9.3. Composite Service Operations

This screenshot shows the 'alpha' cluster details from Figure 9.2. A dropdown menu is open for the 'alpha' cluster, displaying two sections of operations:

- Datasever operations:** Heartbeat, Recover, Switch, Failover.
- Configuration options:** Reload Topology, Edit Definition, Remove Definition.

The table below shows the details for the 'alpha' cluster:

Role	Node	DataSource	Conn.	Replicator	Applied	Relative	Seq. #	minStored	Pipeline Src.	Datasever
Primary	db31	ONLINE	0/0	ONLINE	0.678	12711.462	0	0	/var/lib/mysql	ONLINE
Replica	db32	ONLINE	0/0	ONLINE	20.541	12711.096	0	0	thl://db31:2112/	ONLINE
Replica	db33	ONLINE	0/0	ONLINE	39.122	12711.153	0	0	thl://db31:2112/	ONLINE

Figure 9.4. Service Operations

The screenshot shows the TungstenDashboard interface with the following details:

- Dashboard Summary:** 4 All clusters, 1 Warning, 0 Error.
- Cluster Overview:**
 - omega:** DataService: ONLINE, Status: OK, Policy: Automatic, Connections: 0/0, Coordinators: db31, db34.
 - alpha:** DataService: ONLINE, Status: OK, Policy: Automatic, Connections: 0/0, Coordinator: db31.
 - beta:** DataService: ONLINE, Status: OK, Policy: Automatic, Connections: 0/0, Coordinator: db34.
 - oulu:** DataService: ONLINE, Status: OK, Policy: Automatic, Connections: 0/0, Coordinator: db21.
 - global:** DataService: SHUNNED, Status: Warning, Policy: Automatic, Connections: 0/0, Coordinators: db1, db4.
 - usa:** DataService: ONLINE, Status: OK, Policy: Automatic, Connections: 0/0, Coordinators: db11, db14.
 - east:** DataService: ONLINE, Status: OK, Policy: Automatic, Connections: 0/0, Coordinator: db11.
- alpha Cluster Details Table:**

Role	Node	DataSource	Conn.	Replicator	Applied	Relative	Seq. #	minStored	Pipeline Src.	Datasever
Primary	db31	ONLINE	0/0	ONLINE	0.678	12714.465	0	0	/var/lib/mysql	ONLINE
Replica	db32	ONLINE	0/0	ONLINE	20.541	12714.097	0	0	th:/db31:2112/	ONLINE
Replica	db33	ONLINE	0/0	ONLINE	39.122	12714.155	0	0	th:/db31:2112/	ONLINE
- Operations Menu:** Datservice operations: Heartbeat, Recover, Switch, Failover.

9.2. Cluster Datasource and Replicator operations

To execute a host operation, please follow the following steps:

1. Navigate to clusters page
2. Find the cluster you wish to perform the operation on
3. Use the expand arrows on the left side of the service header to expand the services until you find the host row you are looking for
4. Click the gear icon at the far right of the Datasource row to open the operations menu (See [Figure 9.5, "Primary Host Operations"](#) & [Figure 9.6, "Replica Host Operations"](#))
5. Select the operation you want to run on the cluster

Note

The list of available operations depend on the Role of the host you are executing the operations on.

Note

It is strongly recommended to set the service to `MAINTENANCE` policy before running any cluster-level operations

Figure 9.5. Primary Host Operations

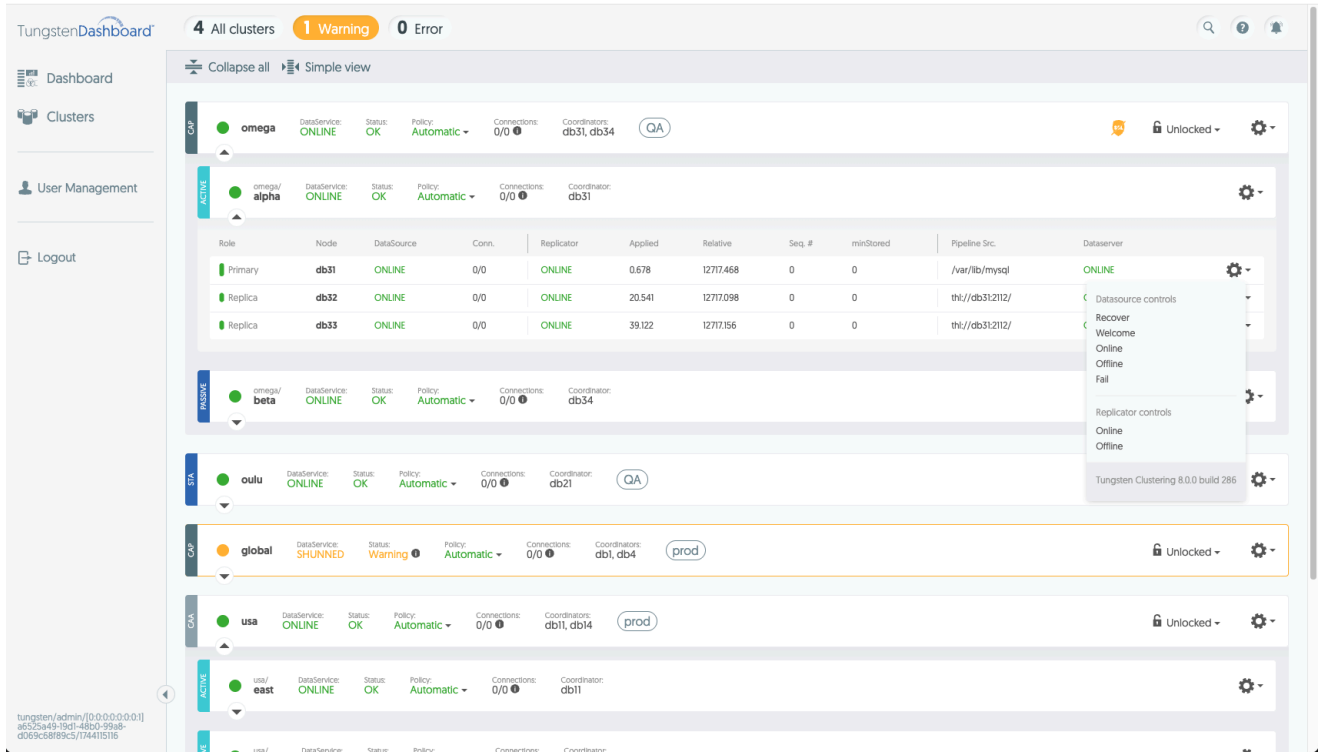
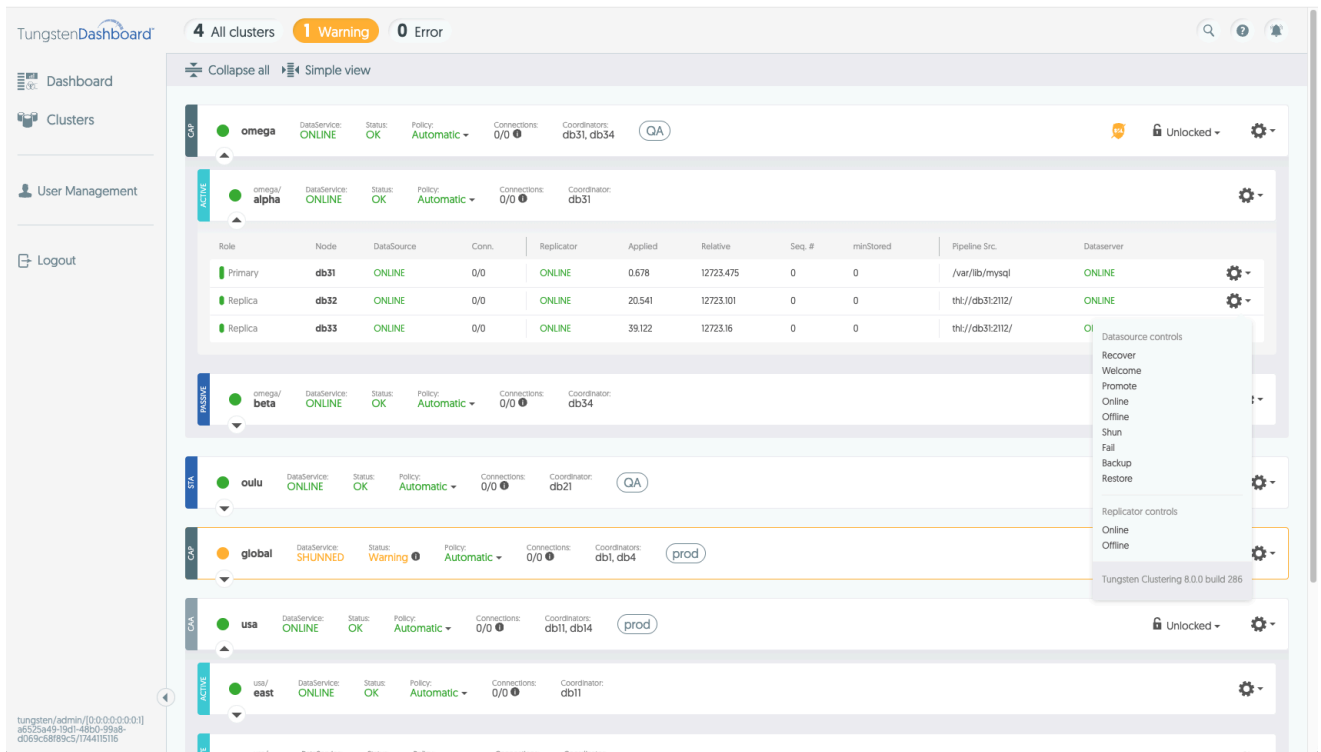


Figure 9.6. Replica Host Operations



Once an action is clicked on, the operation will lock the service and block it (See Figure 9.7, “Blocked Cluster”) from all Dashboard users for the duration of the operation. By default the Dashboard has a 120-second timeout for operations. Please bear in mind that some actions may

take much longer, especially with large clusters. For these use-cases it is recommended to adjust the `rest_timeout` property in the `config.json` file.

Figure 9.7. Blocked Cluster

Role	Node	DataSource	Conn.	Replicator	Applied	Relative	Seq. #	minStored	Pipeline Src.	Dataserver
Primary	db31	ONLINE	0/0	ONLINE	0.678	13231.038	0	0	/var/lib/mysql	ONLINE
Replica	db32	ONLINE	0/0	ONLINE	20.541	13230.339	0	0	thl://db31:2112/	ONLINE
Replica	db33	ONLINE	0/0	ONLINE	39.122		0	0	thl://db31:2112/	ONLINE
Replica	db34	OFFLINE	0/0	OFFLINE-NORMAL		-1	-1	-1	UNKNOWN	ONLINE
Replica	db35	ONLINE	0/0	GOING-ONLINE-SYNCHRONIZING	86.27	13232.701	0	0	thl://db34:2112/	ONLINE
Relay	db36	OFFLINE	0/0	ONLINE	102	13230.225	0	0	thl://db31:2112/	ONLINE

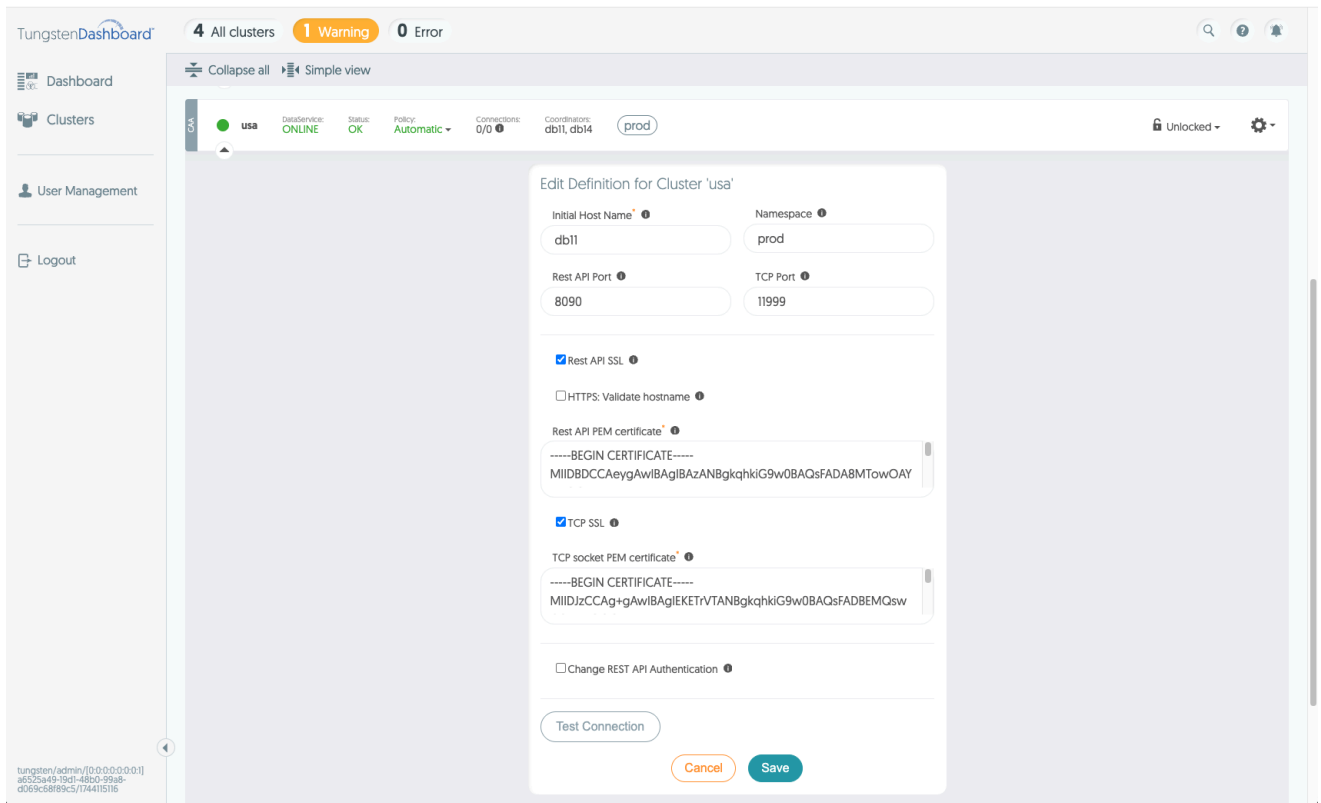
9.3. Tungsten Dashboard Exclusive operations

In addition to the control operations listed in the previous sections there are several operations unique to the Dashboard and its interaction with Tungsten Clusters.

- **Reload Topology** : Calls the cluster's topology endpoint again and updates the Dashboard cache file to match the new information fetched. Useful if your cluster was altered while the Dashboard was running.
- **Edit Definition** : Form with details about the initial cluster connection host. This includes details like domain, port, api-credentials, certificates and more. [See [Figure 9.8, "Cluster Edit Form"](#)]
- **Remove Definition** : Removes the cluster connection from the Dashboard.

Additionally for dataservices you can use "Lock" & "Unlock" operations from the lock dropdown next to the service actions. [See [Figure 9.3, "Composite Service Operations"](#)] This prevents other Dashboard users from executing operations on the service without first unlocking the cluster by hand.

Figure 9.8. Cluster Edit Form



Chapter 10. Best Practices

The following tips will help ensure proper Tungsten Cluster operations and security, along with a smooth Dashboard user experience:

1. Do not share user accounts.
2. Use namespaces for your clusters.
3. When performing actions or investigating a cluster, use the manual lock to prevent other users from changing the cluster at the same time.
4. Install the Dashboard behind an SSL proxy and only use the Dashboard with SSL.
5. If you experience connection difficulties with the Dashboard and a given cluster:
 - Try to be patient since the Dashboard has a 60-second timeout by default.
 - Adding the same cluster multiple times or running the same action multiple times can have unpredictable consequences
6. Prefer SSL connections to clusters whenever reasonable and possible.
7. To retain cluster performance:
 - Avoid multiple Dashboard instances being connected to the same cluster or rapid startup and shut down cycles to the same Dashboard instance.
 - Clusters clean connections periodically but not instantly, avoid situations where you generate several connections from the Dashboard(s) to the cluster in a short period of time.
 - Prefer single long-running instances over rapidly-cycling ones.
8. If you persist data for version updates or restart the cluster, make sure that you use the same `DASHBOARD_SECRET` in the new instance of the Dashboard, otherwise you might lose access to your encrypted data.

Chapter 11. Troubleshooting

This page contains some commonly-encountered situations with the Dashboard and advice on how to resolve them.

11.1. Lost Password

Question: I lost my password and got locked out of the Dashboard

Answer:

To recover access to a single account, and you still have Dashboard access via another user, resolve by recreating the Dashboard account.

If you no longer have access to the Dashboard from any account, you can use the fact that if the `users.json` file does not exist at startup, the Dashboard will recreate the default user for you as long as the default admin user and password environment variables exist.

Perform the following steps to remove all user accounts and revert to the default admin user:

1. Verify that you've set default admin user and default admin passwords environment variables into the docker-compose or `.env` file available to it.
2. SSH to the dashboard container.
3. Navigate to config directory (by default `app/persistent/`)
4. Delete the `users.json` file.
5. Exit from the container
6. Restart the container

For Docker:

- Shutdown the docker container by running `docker compose down`
- Restart the container with `docker compose up -d`

For kubernetes / helm:

- `kubectl delete pod -n tungsten-dashboard {pod-name}`

When the user file does not exist the Dashboard will recreate the default user for you when it restarts, use this to recover access

11.2. 404 After Test Connection Succeeds

Problem : You receive a 404 after "test-connection" was successful

This situation happens when the Dashboard cannot resolve the domain names from returned by the API endpoint `api/v8/manager/cluster/topology`

For example:

```
{
  "payloadType": "ClusterTopologyPayload",
  "payloadVersion": "1",
  "payload": {
    "topology": "CLUSTERED_PRIMARY",
    "name": "oulu",
    "datasources": [
      {
        "name": "db21",
        /* the value of "host" must be resolvable by Dashboard's dns */
        "host": "db21",
        "role": "master"
      },
      {
        "name": "db22",
        /* the value of "host" must be resolvable by Dashboard's dns */
        "host": "db22",
        "role": "slave"
      },
      {
        "name": "db23",
        /* the value of "host" must be resolvable by Dashboard's dns */

```

```

        "host": "db23",
        "role": "witness"
    },
    "routers": [...]
}
}

```

If the Dashboard cannot obtain an IP address for the node name located above, the load balancing functionality breaks down. To resolve the issue:

1. Identify all hostnames returned by the REST API endpoint `api/v8/manager/cluster/topology`
2. SSH to the Dashboard container
3. Install the DNS Tools to get access to the dig command:

```

shell> apt upgrade;
shell> apt install dnsutils;

```

4. Test the hostname resolution for each hostname you collected in step 1. If successful you will get ';; ANSWER SECTION:'

```

shell> dig db21;
"
; <<>> DiG 9.18.30-0ubuntu0.24.04.2-Ubuntu <<>> db21
;; global options: +cmd
;; Got answer:
;; ->HEADER<<- opcode: QUERY, status: NOERROR, id: 30152
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: ab4ead051ab4fa5d (echoed)
;; QUESTION SECTION:
;db21.      IN  A

;; ANSWER SECTION:
db21.      5 IN  A  XX.XXX.XX.XXX" # <---- note valid ip in the answer.
"

;; Query time: 4 msec
;; SERVER: XX.XX.X.XXX#53(XX.XX.X.XX) (UDP)
;; WHEN: Wed Apr 09 12:13:03 UTC 2025
;; MSG SIZE rcvd: 65
"

```

Example of a failed dns resolution:

```

shell> dig db20;
"
; <<>> DiG 9.18.30-0ubuntu0.24.04.2-Ubuntu <<>> db20
;; global options: +cmd
;; Got answer:
;; ->HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 12636
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: ad741d4ba38cc0a7 (echoed)
;; QUESTION SECTION:
;db20.      IN  A"

```

Note how the dig never got a valid IP address in the ';; ANSWER SECTION:' like in the success example.

Make sure that all of the hosts returned by the API topology endpoint can be resolved before proceeding further. Once all hostnames are resolvable, re-add the cluster to the Dashboard or reload the topology using the reload action or by making a small change to it, such as changing the first host.

11.3. Troubleshooting problem installations and cleaning up

Example Scenario

The Dashboard is running but you cannot stop it using docker compose down because you inadvertently removed the original installation directory.

In this scenario you will need to clean up and re-install.

Is everything removed from a docker?

```
shell> docker image ls | grep dashboard
shell> docker container ls -a | grep dashboard
shell> docker volume ls | grep dashboard
shell> docker network ls | grep dashboard
```

Depending upon the outputs from above, you can then user the following command to clean up:

```
shell> docker [component] rm [component-id]
```

Check if there are no environment leftovers:

```
shell> env | grep DASHBOARD
```

See [Chapter 3, Installation](#) for more details on installation.

11.4. DASHBOARD_SECRET

During the very first installation you will be prompted:

```
Unique dashboard secret is not defined in .env file. Do you want to generate it? [Y/n]:
DASHBOARD_SECRET=h0YSFI7o0YaRjapPVOXPK7IazH1osZk0
generated and added to .env file.
Please store this secret in a safe place.
You will need it for future upgrades of the Dashboard.
```

As a message says, store the `DASHBOARD_SECRET` string somewhere safe.

Data (especially passwords) in persistent volumes in a Docker type of installation are encrypted using this `DASHBOARD_SECRET`. Losing `DASHBOARD_SECRET` when doing updates or upgrades will lead to losing your encrypted data.

During updates (from an already configured directory), you won't be prompted for this secret string.

During upgrades (from a new release directory), the install script will prompt you to provide the `DASHBOARD_SECRET`.

11.4.1. Lost DASHBOARD_SECRET

If you can find this secret string on the already configured installation directory:

```
shell> cat .env | grep DASHBOARD_SECRET
```

Alternatively, you can connect to a running docker container (See [Section 11.5, "Connecting to Docker"](#)), and issue:

```
shell> echo $DASHBOARD_SECRET
```

Kubernetes does not use a user defined (or user generated) secret, instead it uses kubernetes secret objects managed by the Kubernetes API itself. To check the secret string inside a kubernetes deployment:

```
shell> kubectl -n tungsten-dashboard get secret tungsten-dashboard-secret --template={{.data.key}} | base64 --decode
```

11.5. Connecting to Docker

Connect to a running container using a command:

```
shell> docker exec -it tungsten-dashboard-service-1 /bin/bash
```

Eventually, you can find the proper dashbaord service name using:

```
shell> docker container ls
```

Inside a container you can check persistent data using:

```
shell> ls -l persistent
shell> cat persistent/config.json
shell> cat persistent/users.json
shell> cat persistent/clusters.json
```

Appendix A. Dashboard Internals

Tungsten Dashboard uses a client/server architecture. A browser-based React client frontend connects via a websocket to a backend driven by a Jetty server. The backend application then connects to a Tungsten Cluster using a TCP session. To support these operations, the Tungsten Dashboard has three configuration files: `config.json`, `users.json` and `clusters.json`. These files are expected to be mounted to a path `/app/persistent/` when production mode is active or in the `resources/` directory if the application is in any other mode.

To explain the contents of the `resources / persistent` directory:

- `config.json` - this file houses the main application configuration such as domain, port, ssl settings, startup actions and more. All fields in the file can be overridden with environment variables.
- `clusters.json` - houses a json array that contains the details of the first connection you make to any given cluster you add to the Dashboard. It includes port, ssl settings, base64 encoded certificates and encrypted username and password for the REST API.
- `users.json` - contains username, email and password hash for all the dashboard users.
- `dynamic/tokens.edn` - A user login stores metadata for that token including the timestamp here. Using logout will purge this information and all previously issued tokens will be rejected afterwards. Please DO NOT alter this file by hand!
- `dynamic/clusters.edn` - The map in this file is the result of the REST API call to endpoint `/api/v8/manager/cluster/topology`. This information is used by the backend application to select a host for each cluster service. The strings placed under each cluster in the array are expected to all be valid fully qualified domain names that the Dashboard can connect to. Please DO NOT alter this file by hand! If you do, your changes may be lost. In case of invalid values in the file, validate the results returned by the API topology endpoint for the affected cluster.

Note

The `clusters.json` and `users.json` files contain encrypted values, and the Dashboard application expects these values to be encrypted. Inputting plain values into the encrypted fields will cause errors in the application. If you want to add clusters or users to a running Dashboard, please do so through the user interface, or through helm `values.yaml` file.

A.1. Dashboard Base URL

The base URL of the application is constructed from several configuration values:

- `domain`: The domain where the application is running [e.g. "localhost", "example.com"]
- `port`: The non-SSL port number [default 4090]
- `ssl_port`: The SSL port number [default 4091]
- `ssl`: Boolean flag indicating if SSL should be used
- `path`: Optional URL path prefix [e.g. "/dashboard"]

The final base URL is constructed as: protocol + host + port + path where protocol and port are determined by the `ssl` boolean.

A.2. Environment Variables

Tungsten Dashboard uses environment variables for some of its configuration. Environment variables are used for the Docker method of installation and development mode.

Environment variables are stored in the `.env` file.

Along with the values below, all variables available within the `values.yaml` can be included by converting them to UPPERCASE and prefixing them with `DASHBOARD_` - for example to set the keystore properties in `values.yaml` you would define `keystorePath` where the equivalent environment variable would then be `DASHBOARD_KEYSTORE_PATH`. See Section A.3, "The `values.yaml` file" for a full list.

Table A.1. Environment Variables

Option	Description
<code>DASHBOARD_ENV</code>	Environment definition [e.g. for different config files selection]
<code>DASHBOARD_SECRET</code>	32 bytes long string that will be used to encrypt cluster credentials

An example can be seen below:

```
DASHBOARD_ENV=production
```

```
DASHBOARD_BASE_URL=example.com
DASHBOARD_PORT=4090
DASHBOARD_PATH=
DASHBOARD_SSL=false
DASHBOARD_SECRET=eSI0vP+MfThNPbLbCcwf4xXrAWdf3mzK
DASHBOARD_DEFAULT_USER=admin
DASHBOARD_DEFAULT_USER_PASSWORD=admin
```

Warning

It is critical that the `DASHBOARD_SECRET` key is stored safely as it will be required for future upgrades since it serves as the encryption key. If it is lost, then there is the possibility of losing data after upgrades.

A.3. The `values.yaml` file

The `values.yaml` file allows for application customization when launching into kubernetes using helm. The supported configuration settings are divided into four areas: Environment, Start-Up, Webserver and Cluster Connection. These four are explained below:

A.3.1. Application Configuration

The values listed below will be translated into snake_case and stored in the `config.json` file.

Table A.2. Environment Settings

Option	Description
<code>configPath</code>	String, path to configuration file location. Defaults to <code>/app/persistent/</code> for production
<code>helm</code>	When set to true, warns user on cluster configuration modifications through the UI. This is used when the dashboard is deployed via Helm and cluster configurations should only be managed through Helm values.
<code>logType</code>	Type of log to be written to the console. Supported values are <code>console-edn</code>
<code>version</code>	Information attached to application logs

Table A.3. Start-Up Settings

Option	Description
<code>clustersConnectOnStart</code>	When set to true, the dashboard will connect to the clusters on start-up. (Recommended for development use only)
<code>topologyCleanOnExit</code>	When set to true, the dashboard will clean the cluster topology cache on exit by writing an empty map to the cache file. (Recommended for development use only)
<code>topologyStartupDiscover</code>	When set to true, the dashboard will discover clusters on startup. This includes a rest api call to the clusters in <code>clusters.json</code> file. (Recommended for development use only)

Table A.4. Webserver Settings

Option	Description
<code>browserPort</code>	If SSL is terminated on the dashboard, the <code>browserPort</code> should match the SSL port. If no SSL is used (not recommended), or you use an external load balancer, this value should match the appropriate port for this configuration.
<code>domain</code>	Domain that dashboard is hosted on (including subdomain) - cannot be empty ie. <code>example.com</code>
<code>keystorePassword</code>	Password of keystore file. (only relevant for ssl)
<code>keystorePath</code>	Path to the key store file that contains the TLS certificate for dashboard server. Keystore must be of type JKS (only relevant for ssl)
<code>path</code>	Directory path at which the dashboard server is expected to respond to.

Option	Description
<code>port</code>	Webserver http and websocket port
<code>ssl</code>	true/false (default false). Use ssl server
<code>sslPort</code>	webserver https and websocket secure port.
<code>tokenLifetime</code>	Time to live of authentication token in seconds.
<code>allowTokenRenewal</code>	true/false - can the existing token be used to get a new one. defaults : true

Table A.5. Cluster Connection Settings

Option	Description
<code>hostnameValidation</code>	Global default for SSL hostname validation (can be overridden per cluster)
<code>defaultRestPort</code>	Default REST port for clusters.
<code>restTimeout</code>	The amount of time Dashboard server waits for the cluster to respond to a single REST api request. Consider adjusting this higher for large clusters or clusters under heavy load to reliably make the first topology discovery and connection.
<code>defaultTcpPort</code>	Default TCP port for clusters.

A.3.2. Cluster Configuration

The values listed below will be translated into snake_case and stored in the `clusters.json` file.

Table A.6. Cluster Settings

Option	Description
<code>apiPassword</code>	API Admin Password
<code>apiUser</code>	API Admin User
<code>apiCert</code>	Base64 encoded PEM certificate (encoded including BEGIN and END markers)
<code>apiPort</code>	Rest API port number for cluster. Default: 8090
<code>cert</code>	Base64 encoded PEM certificate (encoded including BEGIN and END markers)
<code>host</code>	Initial host name to connect for a cluster discovery
<code>namespace</code>	Optional namespace label
<code>ssl</code>	Toggle SSL in TCP socket connections (true/false)
<code>tcpPort</code>	TCP port number for cluster. Default: 11999
<code>api-ssl</code>	Toggle SSL for API connections (true/false)
<code>hostnameValidation</code>	Hostname validation with https connections (set false for self signed certificates)

A.3.3. User Configuration

The values listed below will be translated into snake_case and stored in the `users.json` file. Passwords in the final deployment will be encrypted.

Table A.7. User Settings

Option	Description
<code>password</code>	Password for user
<code>role</code>	Currently only 'admin' role is supported
<code>username</code>	Username (avoid numbers, spaces and special characters)

Appendix B. Release Notes

B.1. Tungsten Dashboard 8.0.4 GA [10 Mar 2026]

Version End of Life. Not Yet Set

Release 8.0.4 contains a number of bug fixes and improvements to usability.

Improvements, new features and functionality

General usability and Performance

- Removed several unnecessarily detailed questions in kubernetes installation and replaced them with automation.
- Kubernetes installation questions are now grouped by topic, and step counter has been added.
- Kubernetes installation now allows you to install ingress-nginx service to your cluster if you do not have it, allowing you to make the Dashboard accessible in your cluster without port forwarding. Both with SSL and no-ssl options are supported for the ingress.

Issues: TD2-270

- Diagnostic tool now collects `dynamic/clusters.edn` content in Docker dashboard diagnostics.

Issues: TD2-277

- Added `host:port` syntax support and smart port inference to `getcert.pl`

Issues: TD2-279

- Added a compact layout mode that reduces cluster row height and spacing for a denser overview; toggle between compact and relaxed layout via the submenu toolbar.
- Replaced the fixed simple/advanced view toggle with a column selector dropdown in the table header, allowing users to show or hide individual columns (replicator state, latencies, sequence numbers, pipeline, etc.).
- Column visibility and layout preferences are persisted in local storage across sessions.

Issues: TD2-284

- Updated cluster status icons for topology warnings.
- The "Fetch Certs" action now displays a pending message while the operation is in progress.
- SSL status icon now shows a more specific tooltip indicating whether TCP, API, or both connections are unsecured.

Issues: TD2-287

Installation and Deployment

- The install script now lets you choose which namespace the dashboard is deployed into, defaulting to `tungsten-dashboard`, and detects conflicts when a dashboard already exists in a different namespace.

Issues: TD2-266

- Kubernetes installation now prompts user to add Tungsten clusters to the Dashboard. By providing namespace details and selecting the correct Kubernetes service a cluster can be added to the dashboard with the install script if the tungsten cluster was installed inside the same kubernetes cluster as the dashboard.

Issues: TD2-271

- Kubernetes installation script now allows user to enter certificate key and file to automatically create Kubernetes secret into dashboard namespace.
- Kubernetes installation no longer requires user to manually enter tls secret and domain details in to the helm values file. They will be automatically filled by the install script.

Issues: TD2-272

Bug Fixes

General usability and Performance

- Fixed a bug where a failed initial connection to a node with an unresolvable hostname could corrupt cluster state, causing subsequent commands to fail with a missing host target.

Issues: TD2-276

- Fixed a bug where a failed REST status poll could erase service connection options, causing subsequent commands to produce malformed URLs.

Issues: TD2-280

- Improved error handling around cluster actions. System now shows more clear errors if host data is missing or corrupted.

Issues: TD2-281

- REST URL construction now validates that a target host is present before making a call, providing a clear error message instead of a cryptic HTTP client failure.

Issues: TD2-283

Installation and Deployment

- Kubernetes installation no longer modifies `values.yaml` file after user has been prompted to validate it.

Issues: TD2-267

- Kubernetes & docker install no longer prints invalid https url at the end of the installation process when https was not enabled.

Issues: TD2-268

- Kubernetes installation's "wait for available" condition has had its timeout increased from 30 to 60 seconds to reduce timeout errors at the last step of Kubernetes installations.

Issues: TD2-269

- Kubernetes install did not store repository and registry information correctly into helm `values.yaml` file. New system stores registry and repository into separate fields in the file.

Issues: TD2-273

- Feature for fetching certificates automatically now works for hybrid SSL setups.

Improved connections handling for clusters with hybrid SSL.

Issues: TD2-282

- Fixed Kubernetes installer to correctly report deployment failures instead of showing misleading "Installation completed!" messages.
- Added automatic StorageClass detection and configuration to prevent pods from getting stuck in Pending state on clusters without a default StorageClass.

Issues: TD2-288

B.2. Tungsten Dashboard 8.0.3 GA [13 Jan 2026]

Version End of Life. Not Yet Set

Release 8.0.3 contains a number of bug fixes and improvements to usability.

Improvements, new features and functionality

General usability and Performance

- Add/edit cluster form now has a "Reset" button on each section allowing you to clear all fields in the form section.

Issues: TD2-220

- Dashboard screen Clusters sorting now mirrors sorting settings from the Clusters screen

Issues: TD2-256

- Add/edit cluster form UX improvements

Issues: TD2-259

- UI elements position and styling fixes.

Issues: TD2-195, TD2-255, TD2-261

- Cluster connections popup improvements

Issues: TD2-262

- Three new ways to fetch certificates from the cluster have been introduced, specifically to assist when certificate rotation has happened within the cluster. These are as follows:
 - via “Edit cluster” form with explicit “Fetch Certificates” button in “SSL” sub-form
 - via cluster level menu item “Fetch Certificates”
 - via warning icon in cluster row when connection error is detected

Issues: TD2-264

Bug Fixes

General usability and Performance

- Composite Active/Active clusters locking mechanism fixed.

Issues: TD2-263

- Fixed a bug with repeated installations failing when trying to generate new self-signed certificates.

Issues: TD2-265

B.3. Tungsten Dashboard 8.0.2 GA [6 Oct 2025]

Version End of Life. Not Yet Set

Release 8.0.2 contains a number of bug fixes and improvements to usability.

Behavior Changes

The following behavior changes have been made to Tungsten Dashboard which may impact the previous operational experience.

- Filtering clusters using search box is now case insensitive.

Issues: TD2-208

- Light/dark UI theme is now configurable in the Configuration screen.

Issues: TD2-216

- When adding a cluster, you can now progress through the form in steps using both “Next” button or Enter key.

Issues: TD2-231

- The install script now determines whether to use docker-compose command or the docker compose plugin automatically. docker-compose is not required anymore, when the docker compose plugin is installed.

Issues: TD2-236

Improvements, new features and functionality

General usability and Performance

- Improvements made to the performance of parallel connection initialization to clusters

Issues: TD2-180

- The Install script now has a new option `--reset`, which resets configuration the files to their default state.

Please use this option if your configuration is broken and you want to start with clean (default) configuration files.

If updating an environment installed using docker compose from insecure to secure configuration, please use this new flag and start with specifying the configuration again.

Issues: TD2-223

- The getcert script can now output the certificate encoded in base64 by specifying the new `--base64` option

Issues: TD2-224

- The tool collect-diag.pl has been included in the installation directory and serves for collecting diagnostic information packages, useful for support cases.

Issues: TD2-240

Bug Fixes

General usability and Performance

- Search box now properly reflects URL query and filters section state

Issues: TD2-192

- Unreachable clusters do not block healthy clusters anymore, when initializing the application.

Issues: TD2-228

- Standalone cluster now shows current coordinator on a service level view.

Issues: TD2-230

- General design fixes and improvements

Issues: TD2-232

- Minor UI graphical changes and fixes

Issues: TD2-234

- Namespace label is now visible in the Dashboard main screen cluster widgets, and invalid clusters. Namespace label serves also as a fast filter/link to search all clusters within specific namespace.

Issues: TD2-235

- When adding a new cluster and pressing "Test connection" at the end of the form, all connection errors are shown above the button.

Additionally, if testing the connection for a Tungsten Cluster running an older release (pre v8.0.0), the relevant error is also shown.

Issues: TD2-238

- Cluster coordinators aggregation and presentation fixes

Issues: TD2-241

- Connections per connector node and connection summaries now works properly with Tungsten Clusters version 8.0.1+

Issues: TD2-243

- Namespace label is now shown next to a cluster in Clusters sidebar menu

Issues: TD2-244

- The following issues are now fixed:

- Adjusted timeouts to cluster connections so that connections to unresponsive servers timeout faster.
- Discovering cluster topology no longer needs all clusters to respond or error out before updates show in browser.
- Cluster connection to existing clusters during first status check is now asynchronous. Unresponsive clusters no longer prevent other connections from forming.
- If a cluster is removed while http request is happening, the results of those requests are properly ignored. Preventing empty clusters from showing.

Issues: TD2-245

- Install script detects existing dashboard docker volume and offers user adequate solution to avoid corrupting existing data.

Issues: TD2-248

- Add cluster button is now visible in Dashboard widgets screen all the time.

Issues: TD2-260

Installation and Deployment

- Docker Compose installation method now offers to generate self-signed certificate for https connection. However this solution is only for development / testing and not recommended to use for production!

Issues: TD2-239

B.4. Tungsten Dashboard 8.0.1 GA [14 Jul 2025]

Version End of Life. Not Yet Set

Release 8.0.1.

Improvements, new features and functionality

Behavior Changes

- Tungsten Dashboard now supports virtual groups for clusters, to reflect Dynamic Datasource Groups.

Issues: TD2-40

- Fixed outdated packages in Dashboard frontend which might represent vulnerabilities.

Issues: TD2-79

- Show Cluster List in Dashboard sidebar, for Composite clusters, also shows services.

Issues: TD2-87

- All forms across the UI are improved for better and smoother usability.

Additionally, when adding a new cluster, certification details are automatically fetched upon selecting the SSL config tab.

Issues: TD2-117

- Dashboard frontend->backend communication has been improved.

Issues: TD2-141

- Improved the logic on how the Dashboard displays labels and colours depending on clusters and hosts states and statuses.

Issues: TD2-194

- `values.yaml` file syntax has been unified. All keys are now camelCase, originally cluster definition was kebab-case.

Issues: TD2-200

- New fast action "Add cluster" button is available in the submenu of the Clusters screen.

Issues: TD2-211

Installation and Deployment

- The `install.pl` can now be used to upgrade the Tungsten Dashboard release, additionally, The environment configuration can also be updated using the same script.

Issues: TD2-214

- The `install.pl` now supports a wider range of CPU architectures.

The architecture is automatically detected from the output of the `uname`, however should this return a value that is unexpected, the user can manually provide the appropriate image.

Issues: TD2-221

Bug Fixes

General usability and Performance

- Improved websocket responsiveness, clusters are now loaded faster on startup.
- Bad cluster connection details no longer delays broadcasting of healthy cluster data.

Issues: TD2-221

B.5. Tungsten Dashboard 8.0.0 GA [28 Apr 2025]

Version End of Life. Not Yet Set

Release 8.0.0 marks a new major release of the Dashboard. Completely re-written and re-designed to be more performant and provide the ability to work alongside Tungsten Clustering solutions launched with Tungsten Operator within Kubernetes.