Tungsten Dashboard 8.0 Manual

Continuent Ltd

Tungsten Dashboard 8.0 Manual

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 8.0.0.

Build date: 2025-05-07 (5ddb0786)

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

Table of Contents

List of Figures

List of Tables

Chapter 1. Introduction

Tungsten Dashboard version 8.0.0 (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 just docker and docker-compose, 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 prerequisities are complete, you can then install Dashboard v8 - See Chapter 3, *Installation*

Once the Dashboard is installed and running, consult Chapter 4, *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 listen 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 Compose Deployment

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

- Docker - No specific version requirements.
- docker-compose - Minimum version v2.28.1

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
```

2.2.2. docker-compose

To install docker-compose, follow the official docker-compose documentation

To confirm the installation run:

```
shell> docker-compose --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 cient - 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.

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.

For example to upload an image to a kind cluster you could use the following:

```
shell> kind load image-archive target/images/tungsten-dashboard_{{.VERSION}}-linux_{{.CURRENT_ARCH}}.tar
```

> **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

Chapter 3. Installation

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

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

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-8.0.0-42.tar.gz
shell> cd /opt/continuent/software/tungsten-dashboard-8.0.0-42
```

3.2. Installing via docker-compose

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 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-8.0.0-42
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 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'
```

3.2.2. Install Steps for docker-compose

3.2.2.1. HTTP Install (No SSL/TLS)

Below is an example of installing Tungsten Dashboard v8 using the `install.pl` script without SSL/TLS Support:

```
shell> cd /opt/continuent/software/tungsten-dashboard-8.0.0-42
shell> ./install.pl -d

Unique dashboard secret is not defined in .env file. Do you want to generate it? [y/N]: y
DASHBOARD_SECRET=LpJFoQVHYnYoxOJCjCuLkV2ZzEvuQNkN generated and added to .env file.
Please store this secret in a safe place. You will need it for future upgrades of the Dashboard.

Please enter preferred admin user name: tungsten

Please enter a password for admin user: secret

Please enter the domain for your application without schema or port.
Domain for your application [127.0.0.1]:

Configure insecure port for http connections.
```

```
Note: All connections to this port (http) will be redirected to https if it is enabled.
Please enter preferred insecure dashboard backend port [4090]:

The path is the part of the URL that comes after the domain.
For example, if your application is hosted at http://example.com/dashboard, the path is dashboard.
Please enter the path for your application [empty]:

Configure https method.
Internal delegates TLS logic to the dashboard server.
If TLS is external, dashboard server only accepts http connections in the default port, it will not validate https.

Please select preferred TLS / HTTPS method:
1) Internal (Dashboard is the SSL endpoint)
2) External (Dashboard is not the SSL endpoint)
3) None (no SSL, not recommended for Production deployments)
Enter choice [1]: 3

Insecure mode selected.
Dashboard browser will attempt to load resources over http.
This is not recommended for production environments.
Do you want to proceed? [y/N]: y

Loading Tungsten Dashboard image...
Loaded image: tungsten-dashboard:8.0.0
Starting docker-compose up...
[+] Running 3/3
 # Volume "tungsten-dashboard_persist"                          Created                                    0.0s
 # Container tungsten-dashboard-service-1                       Healthy                                    5.6s
 ! service Published ports are discarded when using host network mode                                     0.0s

Installation completed!

Opening the Dashboard on browser. If it does not open automatically, please open using URL: http://127.0.0.1:4090

To uninstall current deployment, simply run 'docker-compose down'
Remember to remove volumes either manually or by running 'docker-compose down -v'
```

3.2.2.2. HTTPS Install (SSL/TLS Supported)

Below is an example of installing Tungsten Dashboard v8 using the `install.pl` script with SSL/TLS Support:

```
shell> cd /opt/continuent/software/tungsten-dashboard-8.0.0-42
shell> ./install.pl -d
Unique dashboard secret is not defined in .env file. Do you want to generate it? [y/N]: y
DASHBOARD_SECRET=bN99HmvK2lOvZ5RNkgQJp6RGMmpVSTC6 generated and added to .env file.
Please store this secret in a safe place. You will need it for future upgrades of the Dashboard.

Please enter preferred admin user name: tungsten

Please enter a password for admin user: secret

Please enter the domain for your application without schema or port.
Domain for your application [127.0.0.1]: dashboard.example.com

Configure insecure port for http connections.
All connections to this port (http) will be redirected to https.
Please enter preferred insecure dashboard backend port [4090]:

The path is the part of the URL that comes after the domain.
For example, if your application is hosted at http://example.com/dashboard, the path is dashboard.
Please enter the path for your application [empty]:

Configure https method.
Internal delegates TLS logic to the dashboard server.
If TLS is external, dashboard server only accepts http connections in the default port, it will not validate https.

Please select preferred TLS / HTTPS method:
1) Internal
2) External
3) None (not recommended)
Enter choice [1]:

Internal TLS selected.
This requires a valid java keystore file with a certificate under alias "jetty".
Add the keystore file ie. jetty.jks to the ./cert directory.
Is the keystore file jetty.jks in the ./cert directory? [y/N]: y

Provide application SSL port: [4091]:

Provide the secret name in the docker-compose.yml file [dashboard_cert]:
Keystore file's password: tungsten

Loading Tungsten Dashboard image...
```

```
3abdd8a5e7a8: Loading layer [==================================================>]  29.72MB/29.72MB
a81b5d0b5796: Loading layer [==================================================>]  22.95MB/22.95MB
5eb803431e96: Loading layer [==================================================>]  157.6MB/157.6MB
88f8cc5af356: Loading layer [==================================================>]    158B/158B
dd4b98d123a3: Loading layer [==================================================>]  2.282kB/2.282kB
d4cd2ada82fd: Loading layer [==================================================>]    117B/117B
5f70bf18a086: Loading layer [==================================================>]     32B/32B
c175884b36b0: Loading layer [==================================================>]  47.39MB/47.39MB
73ecd4e84b34: Loading layer [==================================================>]    399B/399B
Loaded image: tungsten-dashboard:8.0.0
Starting docker-compose up...
[+] Running 3/3
 # Volume "tungsten-dashboard_persist"                          Created                                                    0.0s
 # Container tungsten-dashboard-service-1                       Healthy                                                   10.9s
 ! service Published ports are discarded when using host network mode                                                    0.0s

Installation completed!

Please open the Dashboard using URL: https://dashboard.example.com:4091

To uninstall current deployment, simply run 'docker-compose down'
Remember to remove volumes either manually or by running 'docker-compose down -v'
```

3.2.3. Uninstall

To uninstall Tungsten Dashboard, simply execute the following:

```
shell> docker-compose down
```

or, to remove persistent volumes along with the uninstallation:

```
shell> docker-compose down -v
```

Warning

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

3.3. Installing via Kubernetes using helm

Kubernetes installation via helm uses a declarative approach to configuring the Dashboard. To get started, you need to adjust the `values.yaml` or your custom yaml file to define settings before you run the installation script.

Declare the clusters:

Find the following line in the `values.yaml` file:

```
clusters: []
```

Replace the line with a configuration that matches your clusters. For example:

```
clusters:
  - host: "db1.example.com"
    namespace: "dev"
    api-user: "sample-user"
    api-password: "<clear-text-password>"
    tcp-port: 11999
    api-port: 8090
    api-ssl: true
    ssl: true
    cert: "<base64-encoded-certificate>"
    api-cert: "<base64-encoded-certificate>"
    hostname-validation: true
```

You can add as many clusters as you'd like using the "- <cluster>" syntax.

Note

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

Declare the Application Configuration:

Next, adjust the application configuration as necessary. See Section A.3, “The `values.yaml` file” for more details on each setting. Note that the configs here are in camelCase while in the final JSON they're snake_case

```
config:
  env: "production"
  version: "8.0.0"
  configPath: "/app/persistent/"
  logType: "console-edn"
  helm: true
  domain: ""
  path: ""
  port: 4090
  ssl: false
  sslPort: 4091
  browserPort: 4090
  clustersConnectOnStart: false
  topologyStartupDiscover: false
  topologyCleanOnExit: false
  defaultRestPort: 8090
  defaultTcpPort: 11999
  hostnameValidation: true
  restTimeout: 15000
```

In the configuration take note of the following 5 values: `domain`, `port`, `browserPort`, `path` and `restTimeout`.

- If you leave the `domain` and `path` empty, you'll be prompted for them in the install.
- `port` is the application's port that the webserver responds to, while `browserPort` is used in the base url for the browser.
- Normally you don't need to change the application `port` but you might need to change the `browserPort` to match your Ingress settings.
- For `restTimeout` consider increasing it for big clusters or clusters under heavy loads for a more reliable connection in the start-up phase. The indicated value is in milliseconds.

User Credentials:

Adjust the default user credentials or be prompted for a default user during installation. The `install.pl` script itself creates only one user, if you want multiple users define them here.

```
users:
  - role: admin # currently only admin role is supported.
    username: jane-doe
    password: <clear-text-password>
```

Passwords in yaml file are provided as clear text, during installation they're encrypted and final `users.json` will have hashed passwords on the server.

Configure Ingress (if required):

> **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 it's own namespace. Without a controller this resource does nothing.

To enable Dashboard-specific ingress locate the following entries and adjust accordingly:

```
ingress:
  enabled: false # When true, creates dashboard ingress resource

  # 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

  # 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
```

- Change `ingress.enabled` to true if you want to create ingress resource
- Add the `className` of your index controller, i.e. nginx
- Update `tls` section with your tls secret and the Hosts field should match `config.domain`

Being Installation

To begin the installation for kubernetes execute the following:

```
shell> cd /opt/continuent/software/tungsten-dashboard-8.0.0-42
shell> ./install.pl -m kubernetes
```

The following example shows the output from running the installation and the prompts that will require answering:

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

Please enter custom values YAML file [values.yaml]:

Please enter preferred admin user name: admin-user

Please enter a password for admin user: admin-password

Please enter the domain for your application without schema or port.
Domain for your application [127.0.0.1]: example.com

Please enter the dashboard server port [4090]:

The path is the part of the URL that comes after the domain.
For example, if your application is hosted at http://example.com/dashboard, the path is dashboard.
Please enter the path for your application [empty]:

WARNING: Ingress is currently not enabled!
Dashboard will not be accessible without port-forwarding from outside the cluster.
Do you want to proceed with the current configuration? [y/N]: y

WARNING: TLS is not configured in your ingress settings!
Running the dashboard without TLS encryption is not recommended for production use.
To enable TLS, modify values.yaml to include your TLS secret and domain configuration:

ingress:
  tls:
  - secretName: your-tls-secret
    hosts:
      - your.domain.com

Do you want to proceed with insecure configuration? [y/N]: y

With current settings your base url will be: http://example.com:4090
This is the url the browser will load the resources for the dashboard from.

Please input the container registry URL where Tungsten Dashboard image is stored (without a tag) [tungsten-dashboard]:

Please enter Tungsten Dashboard version tag [8.0.0]:

Current Kubernetes context is set to: arn:aws:eks:us-east-1:00000000000:example-cluster
Do you want to proceed with the currently selected Kubernetes context? [y/N]: y

Installing Tungsten Dashboard Helm chart...
NAME: tungsten-dashboard
LAST DEPLOYED: Mon Apr 14 16:43:52 2025
NAMESPACE: tungsten-dashboard
STATUS: deployed
REVISION: 1
Waiting for deployment to become ready...
deployment.apps/tungsten-dashboard condition met

Installation completed!

To uninstall current deployment, simply run 'helm uninstall tungsten-dashboard --namespace tungsten-dashboard'
```

> **Note**
>
> During this installation, the script will post yaml files to the cluster using the kubernetes syntax you provided via the helm installation logic. These yaml files are generated using the helm templating syntax fed by the `values.yaml` file.
>
> Helm moves the values you give into the proper yaml file while kubernetes declares what yaml syntaxes and values in those syntaxes are available.
>
> If you encounter issues during the installation start by inspecting the yaml posted to kubernetes and validating it against kubernetes documentation.

3.3.1. Uninstall

To uninstall Tungsten Dashboard, simply execute the following:

```
shell> helm uninstall tungsten-ui --namespace tungsten-ui
```

Additionally if you do not wish to keep the data from the dashboard run:

```
shell> kubectl -n tungsten-dashboard delete secret tungsten-dashboard-secret;
# check if tungsten-dashboard-vol exists.
shell> kubectl -n tungsten-dashboard get pvc;
# if it does, remove it with:
shell> kubectl -n tungsten-dashboard delete pvc tungsten-dashboard-vol;
```

Warning

Deleting or changing the secret without deleting the data will prevent users from logging in and the dashboard from making any connections, effectively locking you out of the system. If this happens to you remove the `users.json` file and restart the dashboard application. This will recreate the default user for you.

Chapter 4. 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.

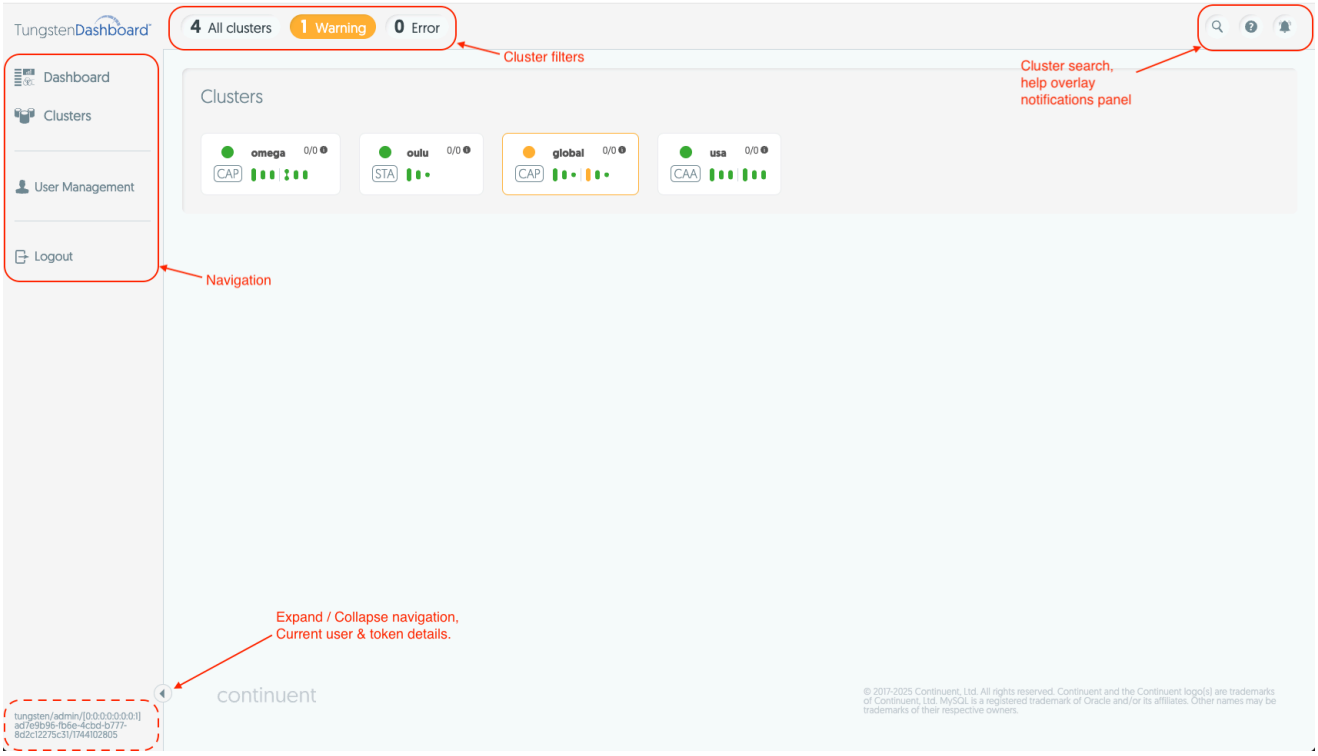
To create additional users, and to get started, see Section 4.2, “Create your own user account”

> **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.

4.1. The Front Page

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

Figure 4.1. 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.

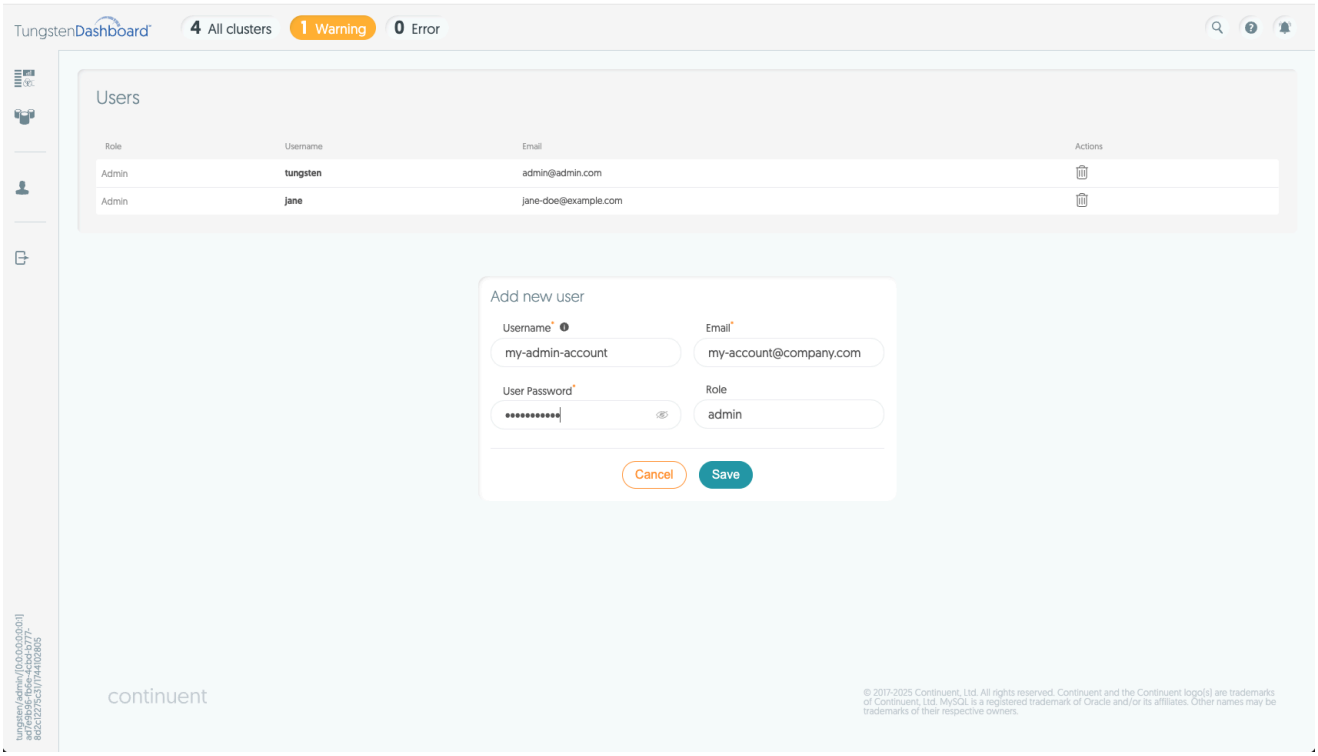
4.2. Create your own user account

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 4.2. Add User Form



> **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.

4.3. Add Clusters

Once you've logged in, click the "Add cluster" button on the fron tpage (see Figure 4.3, “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 4.4, “Add Cluster Form”)
- Finally click save to add the cluster to the Dashboard.

Figure 4.3. Cluster Front Page - Add Cluster

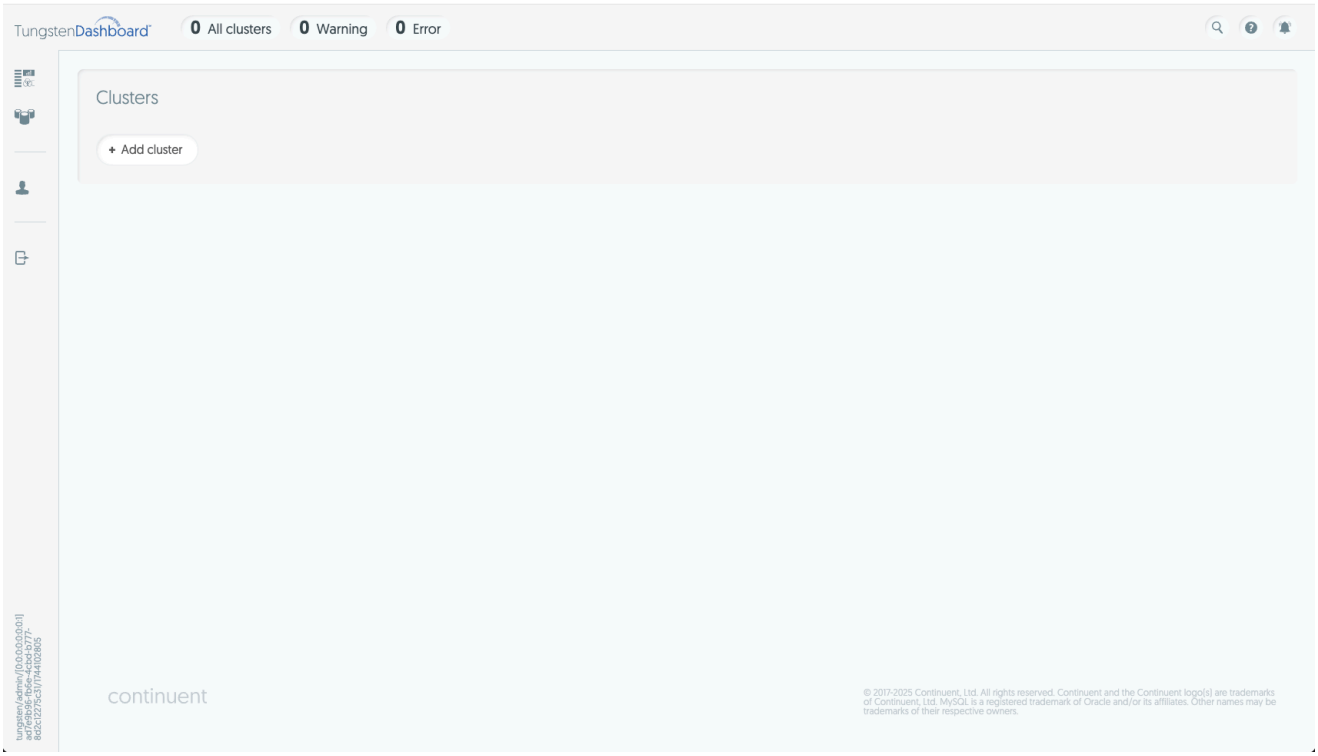
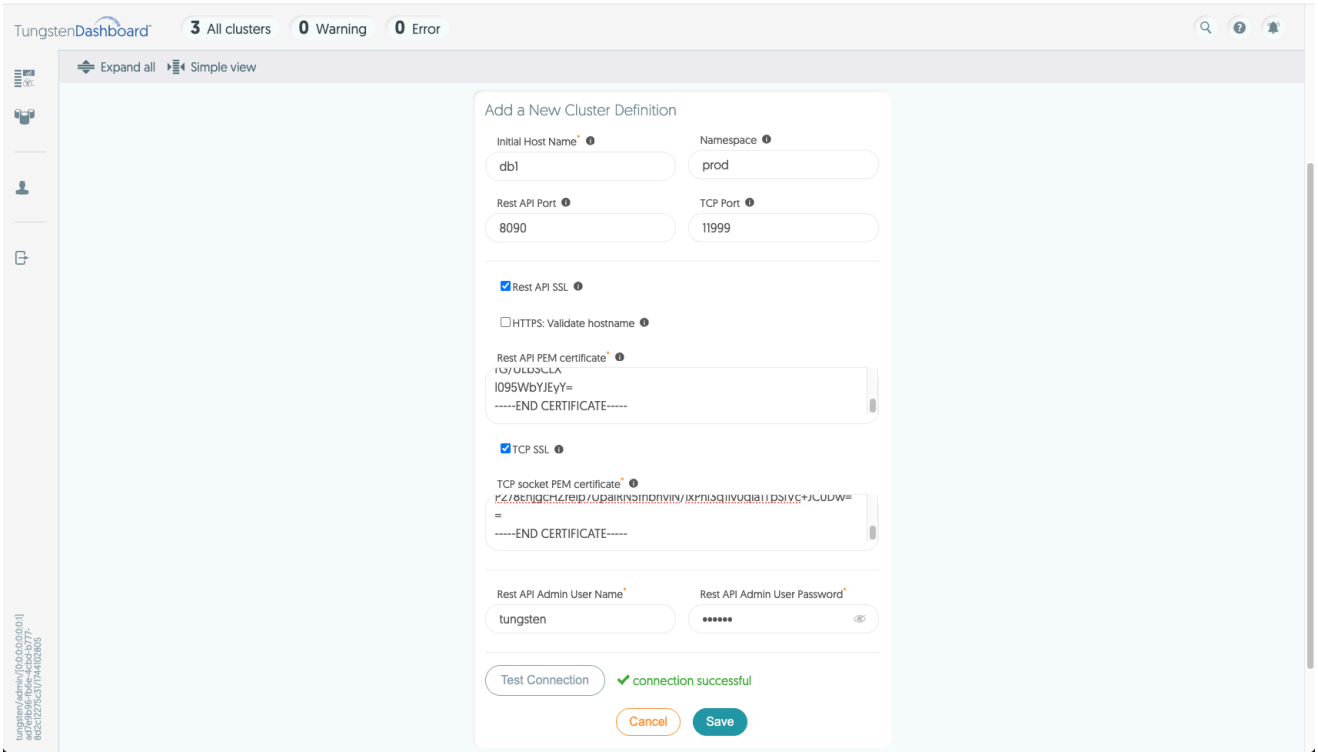


Figure 4.4. Add Cluster Form



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 it's 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.

4.4. 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 5. 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 4, *User Guide*

5.1. Front Page Interface

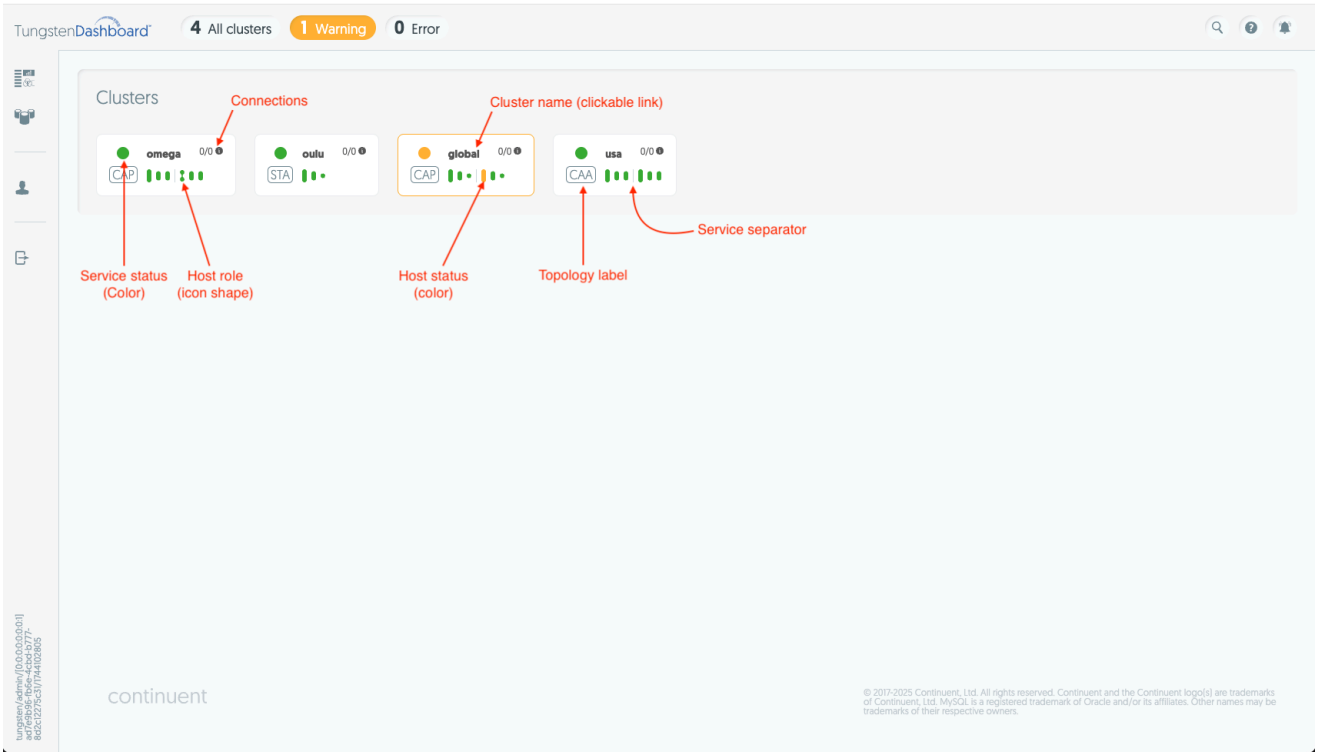
From this page you can find a quick status overview of any of the clusters you have connected the Dashboard to. Clusters are split into small "widgets" on the page. (See Figure 5.1, "Front Page Interface")

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.

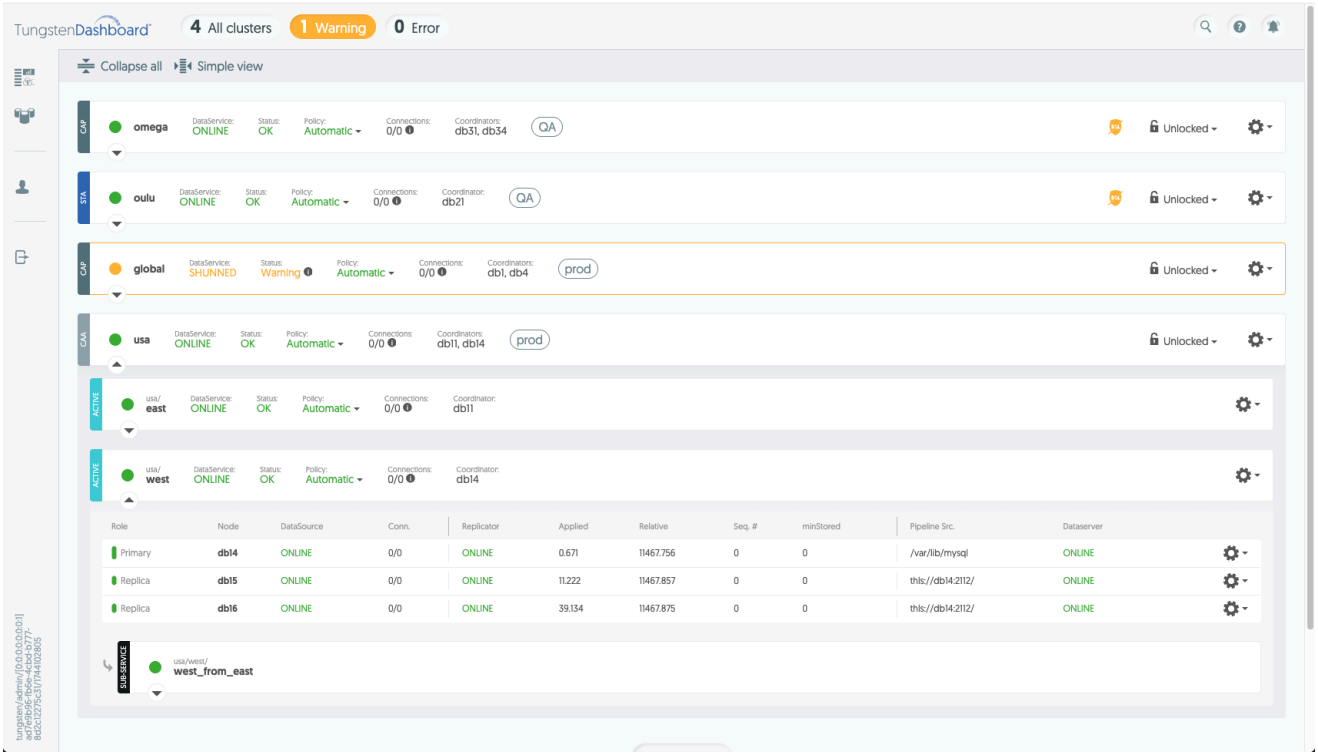
Figure 5.1. Front Page Interface



5.2. 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 5.2, “Populated Cluster Front Page”]

Figure 5.2. 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 5.3, “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 5.3. Cluster Tool Bar

Below the toolbar you can find the clusters themselves:

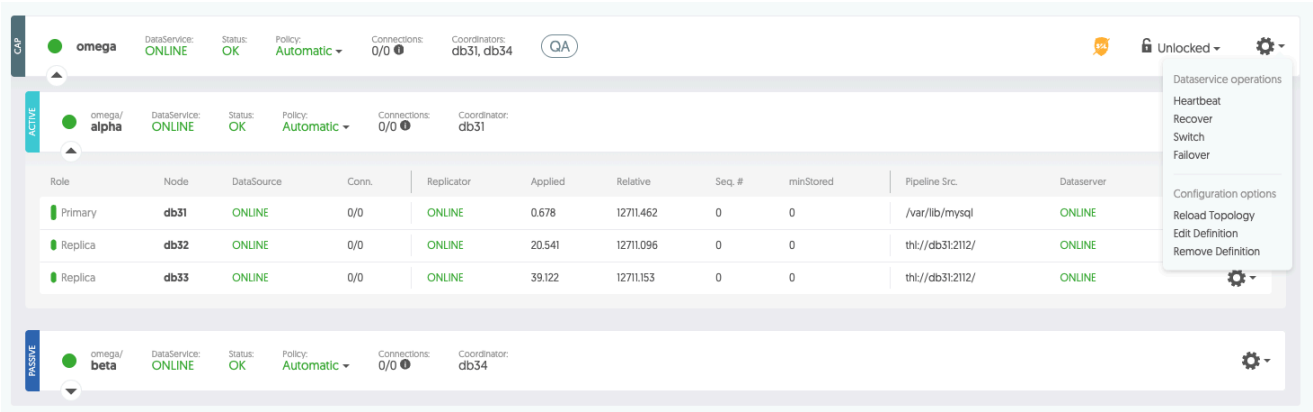
Figure 5.4. Cluster Header Detail

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 - colorized 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 5.5, "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 8, *Operations* and for additional information about individual operations, please refer to the Tungsten Clustering documentation.

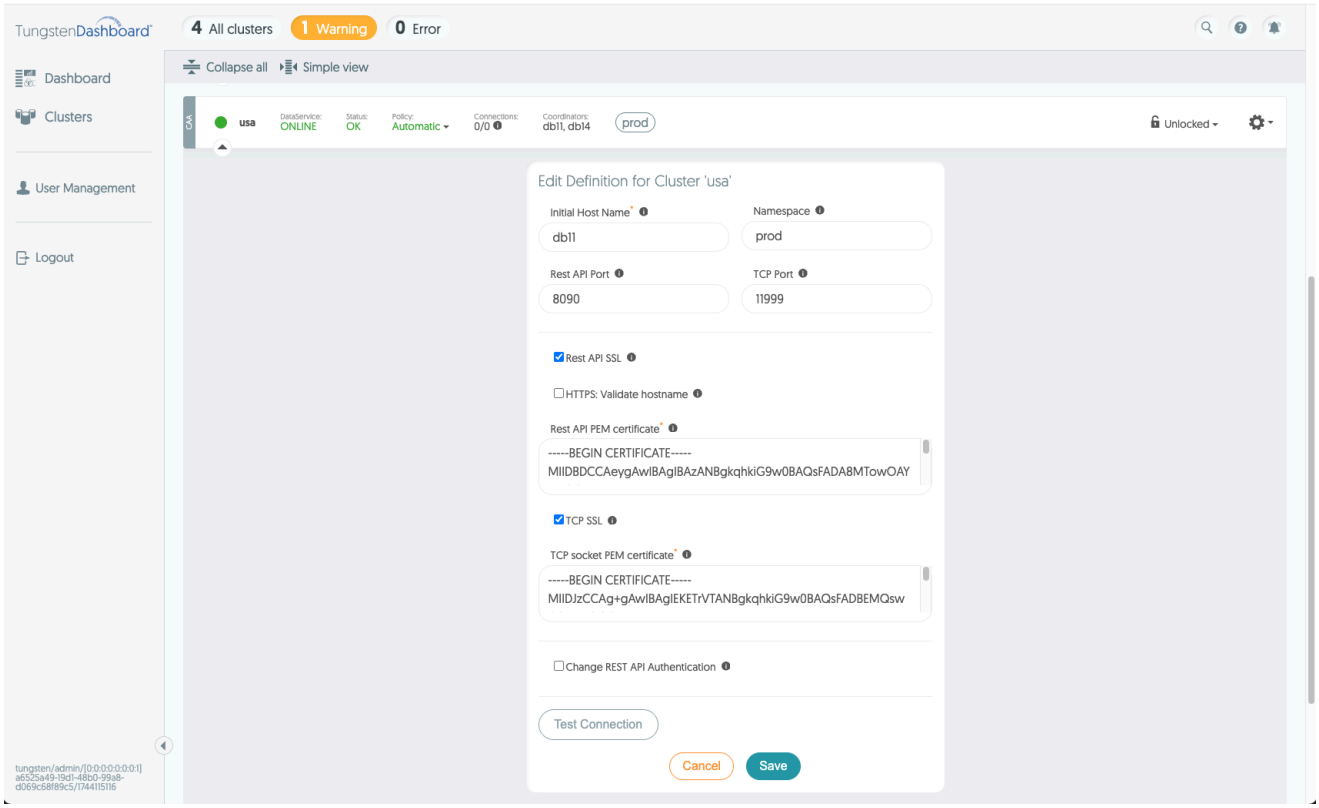
Figure 5.5. Parent Service Actions/Operations



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 5.6, "Edit Cluster Form")
- Remove cluster - Deletes the cluster from the Dashboard

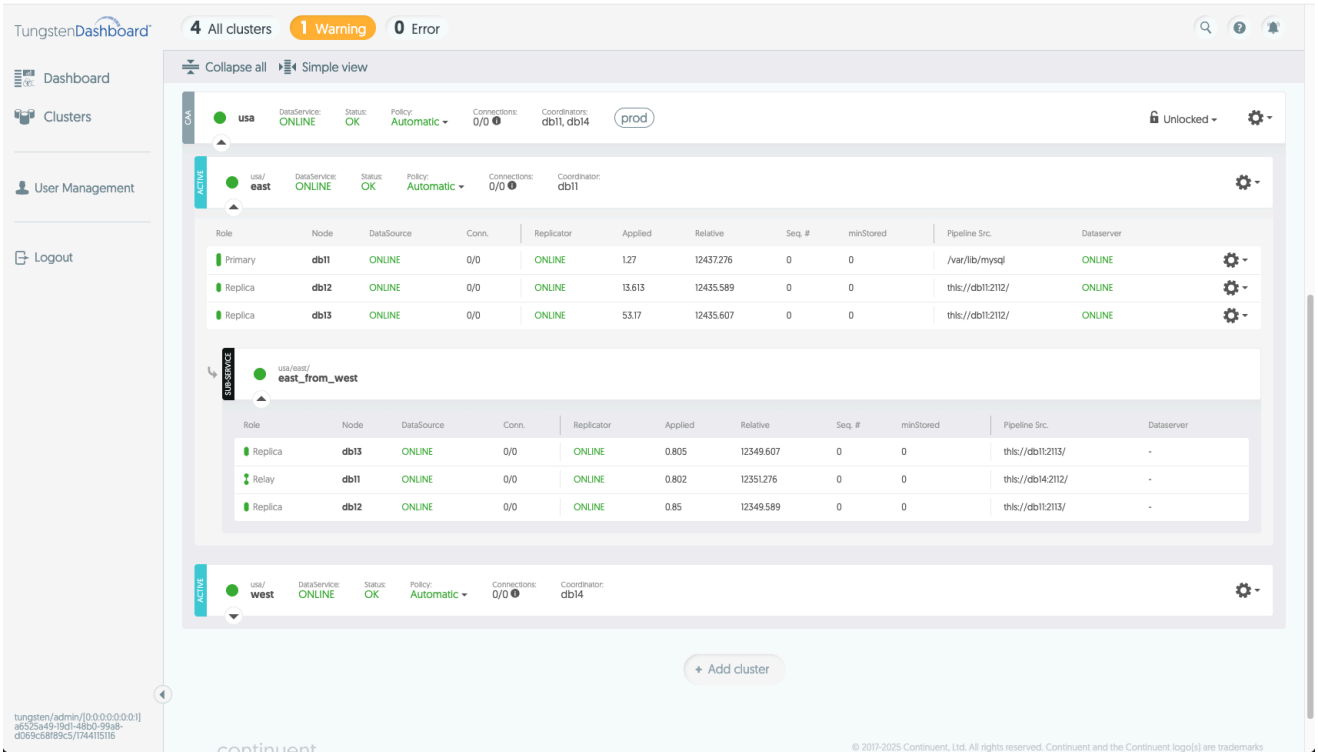
Figure 5.6. Edit Cluster Form



The cluster's host details are displayed as a table. [See Figure 5.7, “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 5.7. Cluster Host Table (CAA)



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 5.8, “Primary Host Operations” & Figure 5.9, “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 8, *Operations*

Figure 5.8. Primary Host Operations

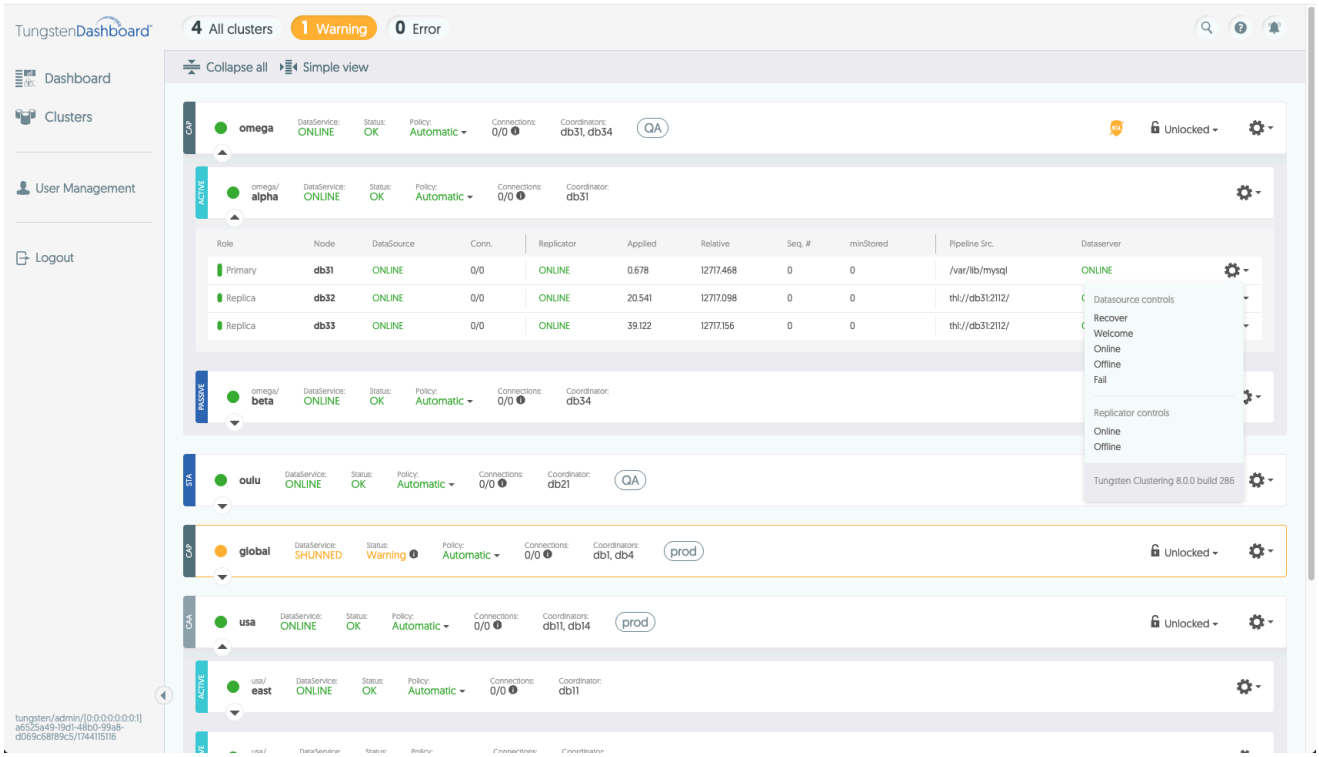
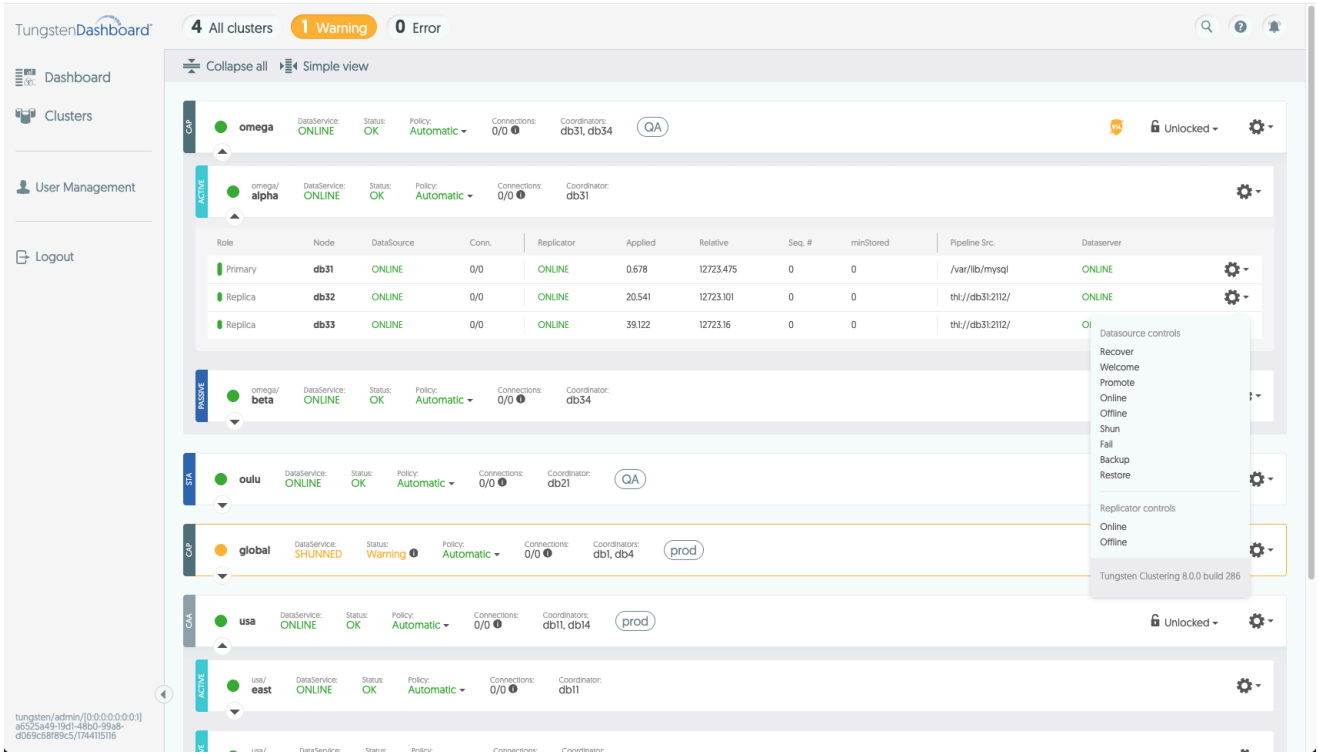


Figure 5.9. Replica Host Operations



5.3. User Management Interface

This page lists the current users stored in the Dashboard with their role, username and email. (See Figure 5.10, “User Management Page”)

From here you can add or remove users as you desire. [See Figure 5.11, “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 5.10. User Management Page

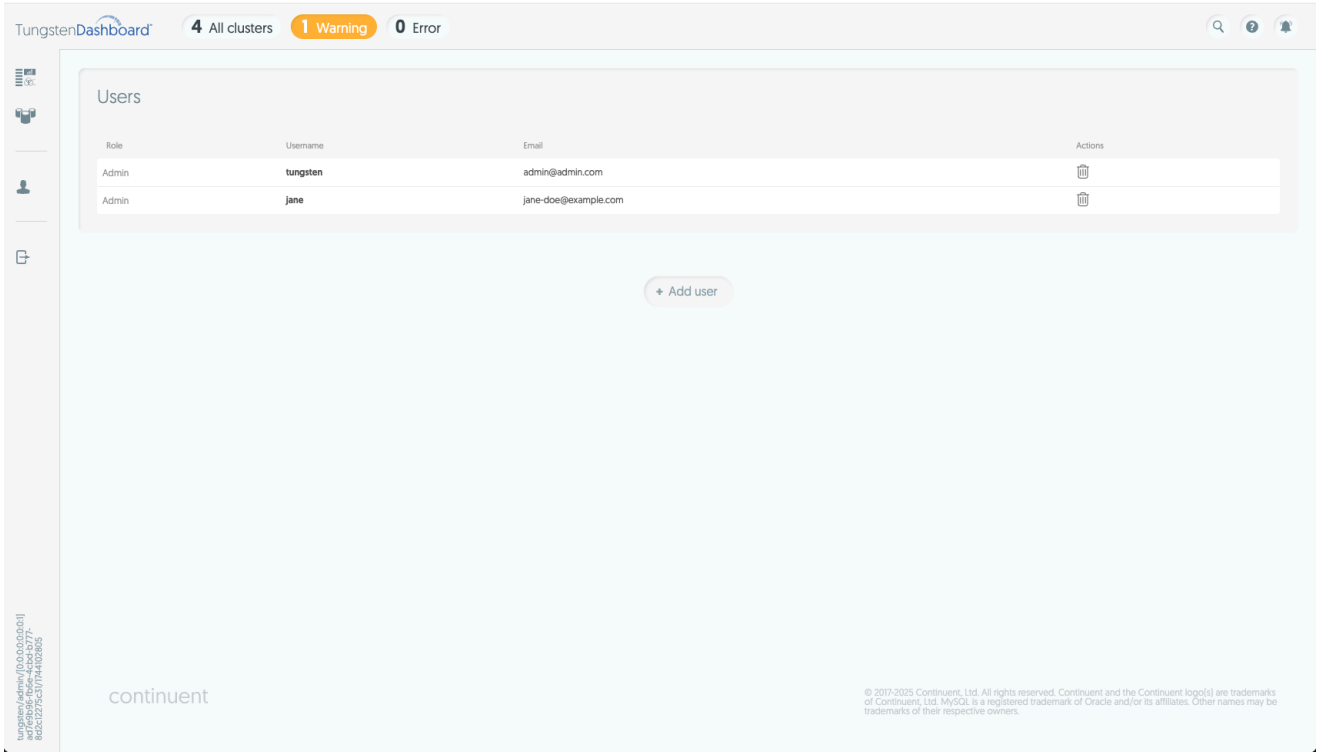
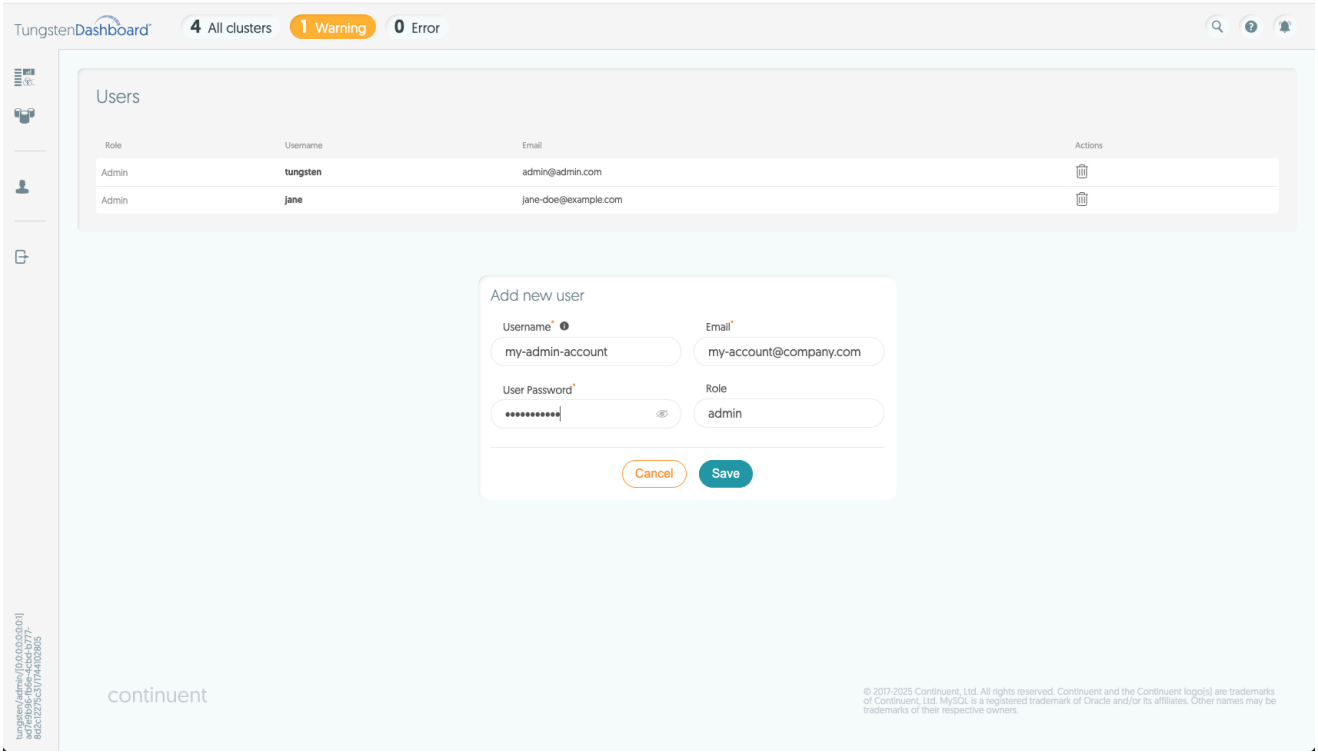


Figure 5.11. Add User Form



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 6. Command Line Tools

6.1. install.pl script

The `install.pl` script is a command-line tool for installing 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 ] [ --methos, -m ]`

Where:

Table 6.1. `install.pl` Options

Option	Description
`--docker`, `--docker-compose`, `-d`	Specify the docker-compose deployment method (The same as issuing -m docker-compose)
`--help`, `-h`	Use this option to display the help message.
`-k`, `--k8`, `--kubernetes`	Specify the kubernetes deployment method (The same as issuing -m kubernetes).
`-m`, `--methos`	Specify the deployment method (docker-compose or kubernetes).

For example, to install using Docker Compose, 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 and docker-compose for the Docker Compose 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.

6.2. 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, -a ] [ --help, -h ] [ --port, -p ] [ --quiet, -q ] [ --tcp, -t ]`

Where:

Table 6.2. `getcert.pl` Options

Option	Description
`--api`, `-a`	Use this option to fetch the Manager API certificate on the default port 8090

Option	Description
`--help`, `-h`	Use this option to display the help message.
`--port`, `-p`	Use this option to specify the port - requires either `--api` or `--tcp` argument.
`--quiet`, `-q`	Use this option to suppress verbose output and display only the certificate content.
`--tcp`, `-t`	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-----
MIIDBDCCAeygAwIBAgIBAzANBgkqhkiG9w0BAQsFADA8MTowOAYDVQQDDDFNeVNR
... 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.

6.3. 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 6.3. `letsencrypt2jetty.pl` Options

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

Chapter 7. 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 dataservices (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-compose 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 cctrl command, the APIv8 endpoint `/api/v8/manager/cluster/topology`, as well as the APIv2 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 9, *Best Practices*

7.1. Configuring SSL when Deploying with Kubernetes

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

7.1.1. 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.

7.1.2. 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 8. 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.

8.1. Service Operations

These actions can be used on standalone or composite services. They include cctrl 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 8.1, “Service Policy Dropdown”]
4. Click the "unlocked" text and choose "lock" [See Figure 8.2, “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 8.3, “Composite Service Operations” & Figure 8.4, “Service Operartions”]
7. Select the operation you want to perform

Figure 8.1. Service Policy Dropdown

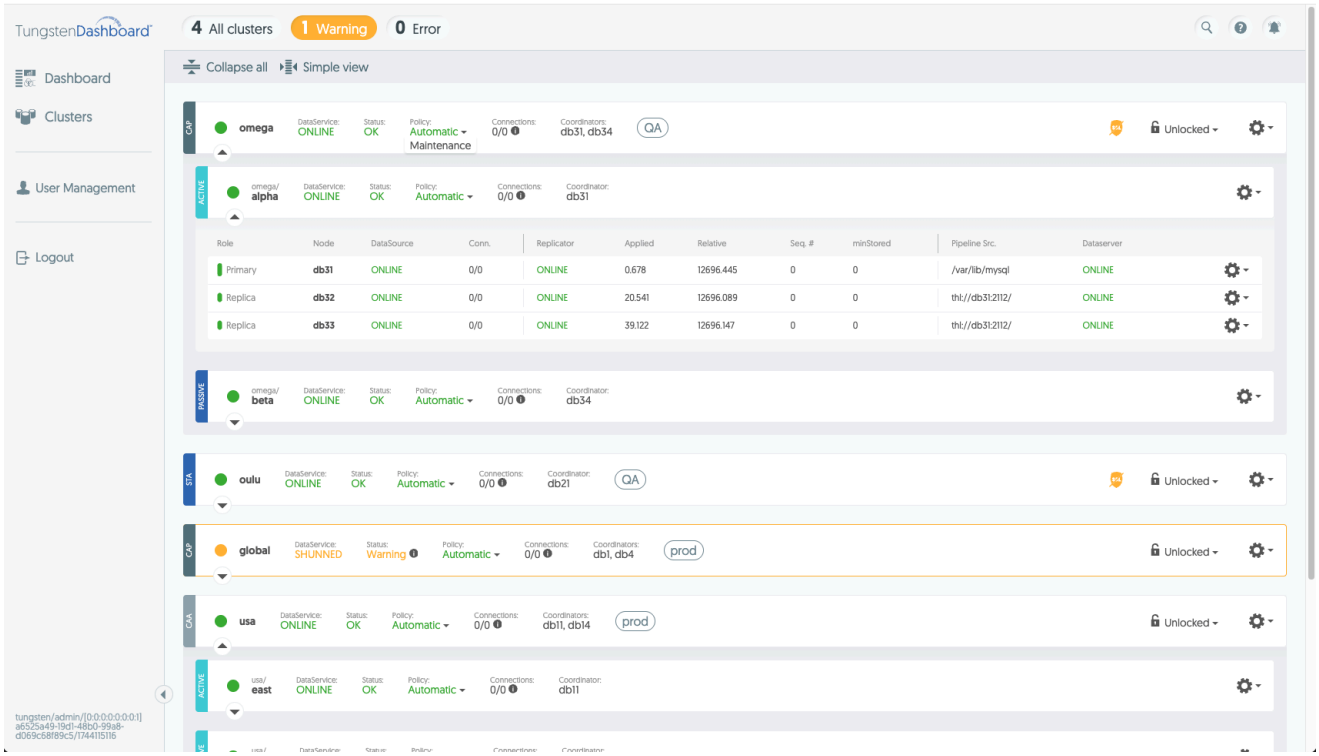


Figure 8.2. Cluster Lock Dropdown

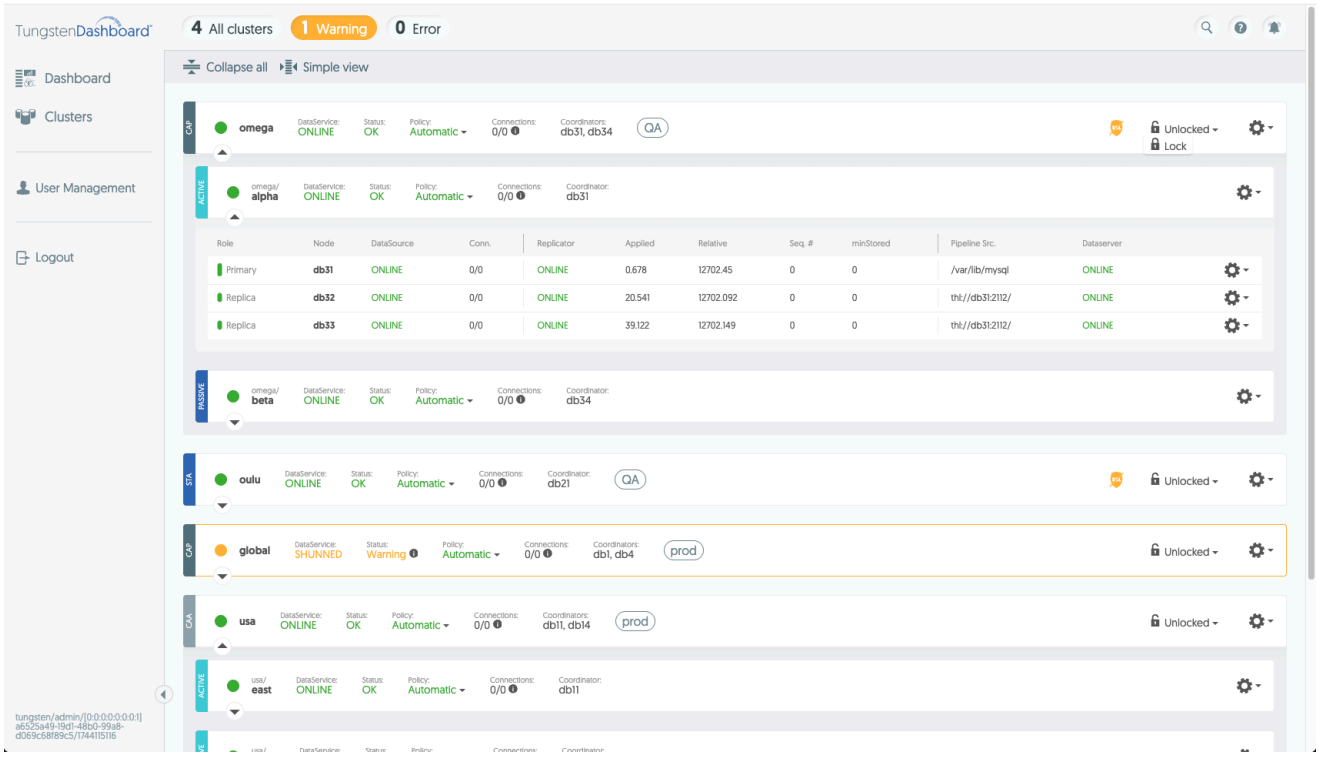


Figure 8.3. Composite Service Operations

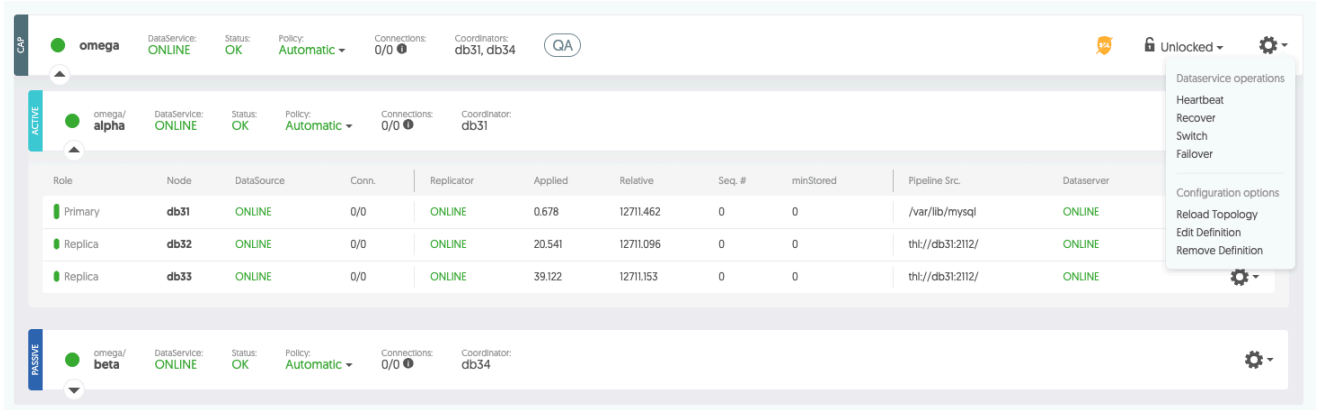
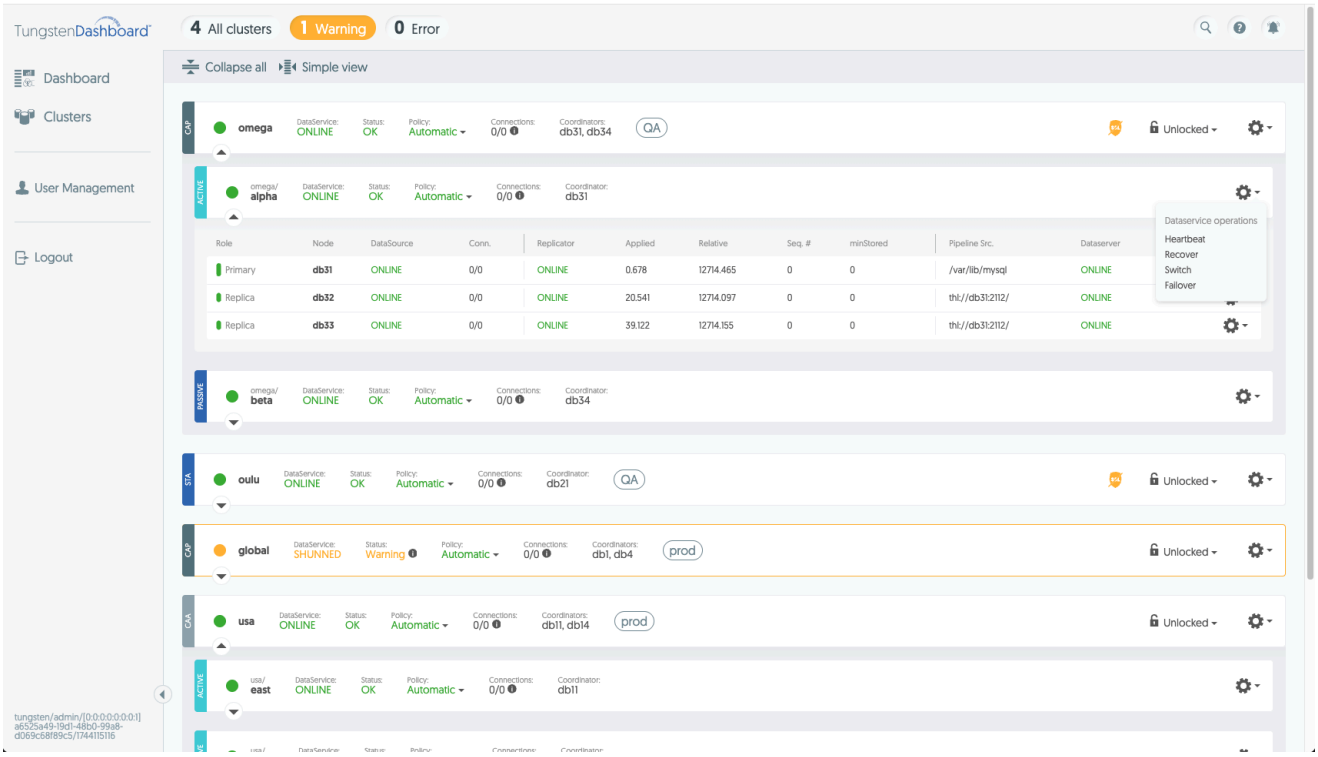


Figure 8.4. Service Operartions



8.2. 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 8.5, “Primary Host Operations” & Figure 8.6, “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 8.5. Primary Host Operations

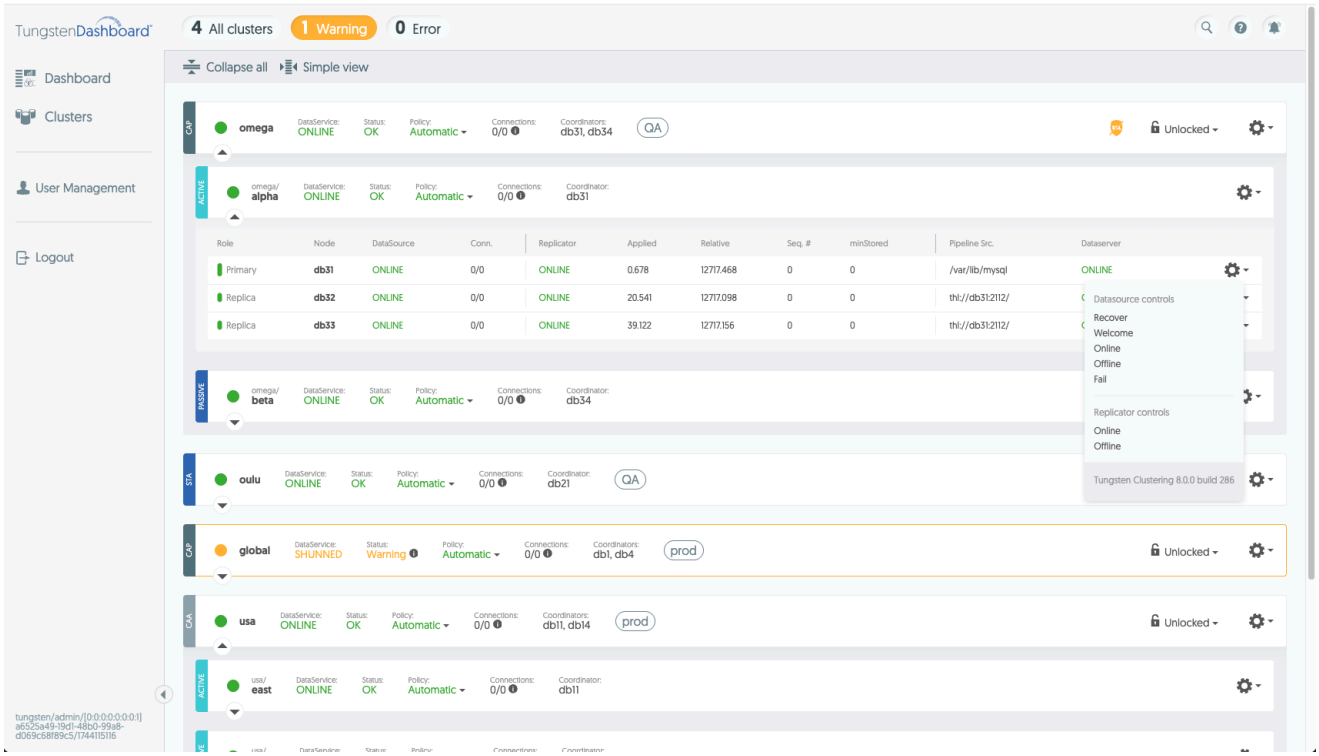
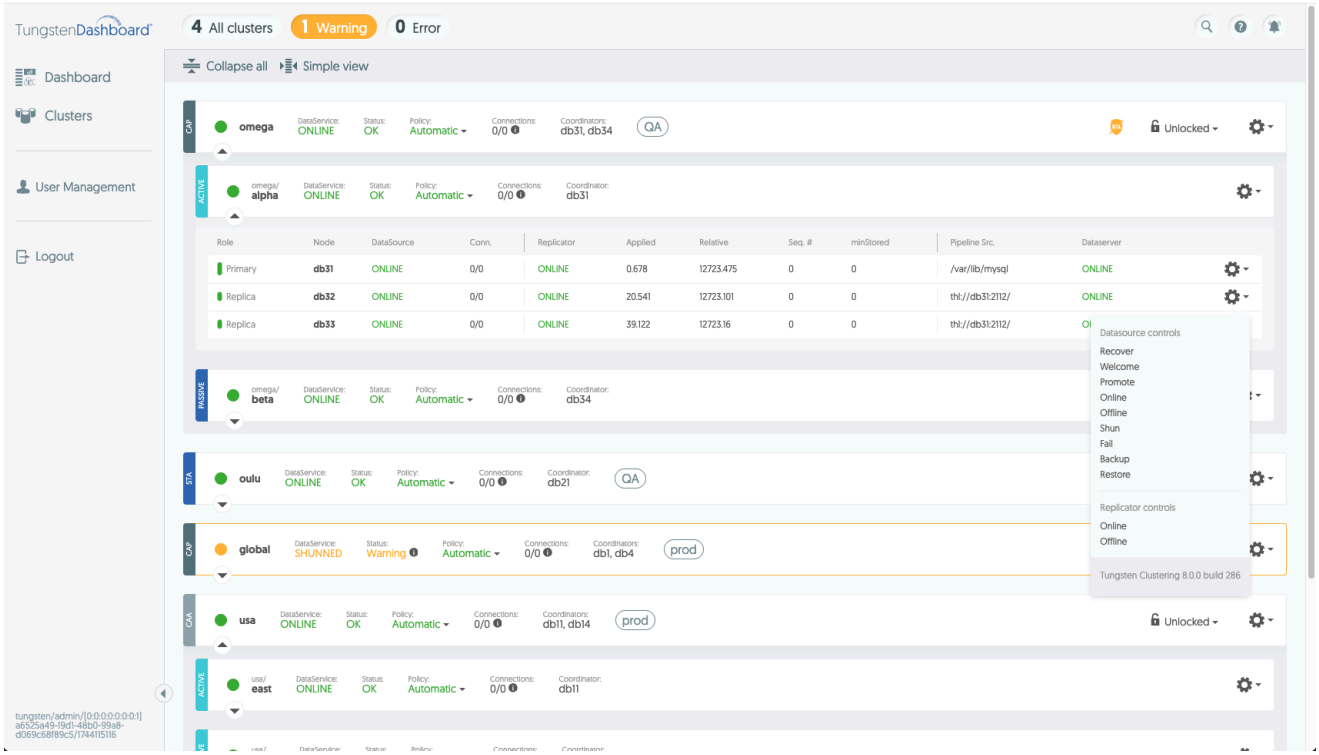


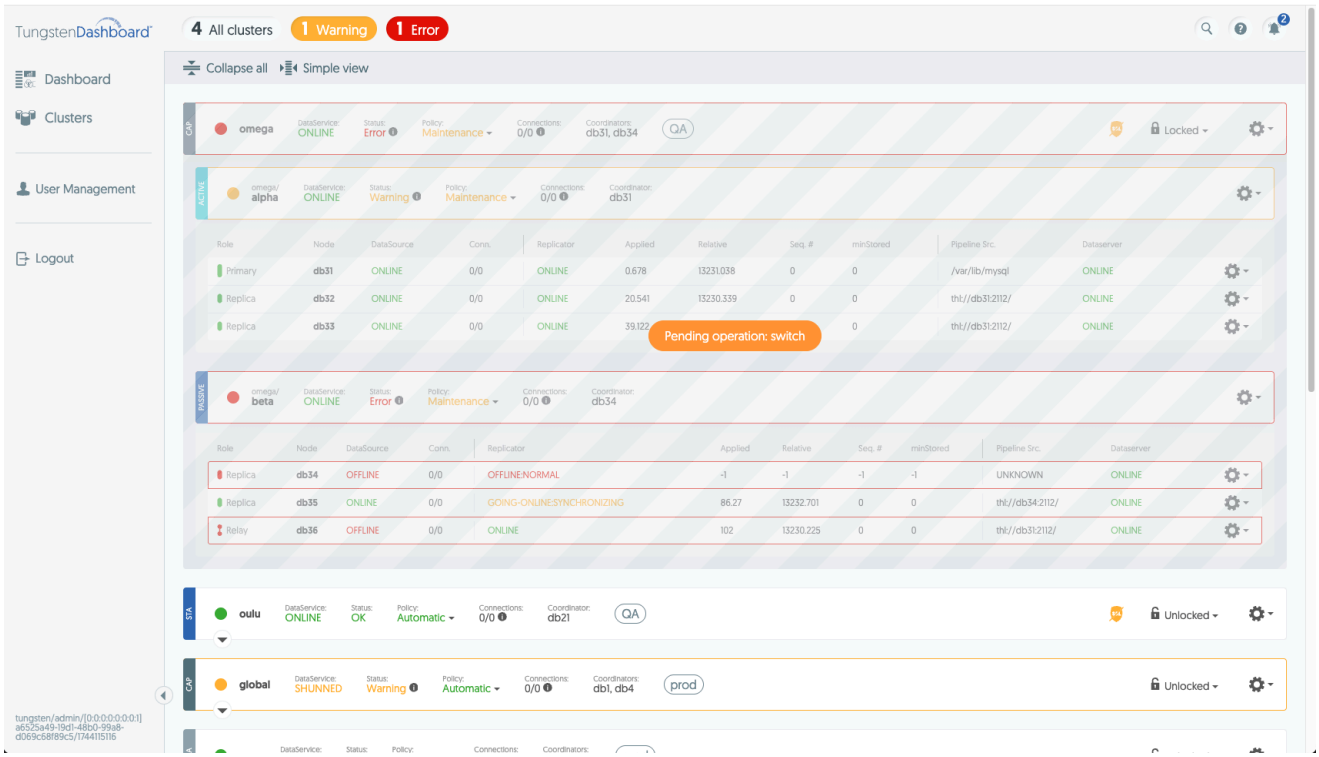
Figure 8.6. Replica Host Operations



Once an action is clicked on, the operation will lock the service and block it [See Figure 8.7, “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 8.7. Blocked Cluster



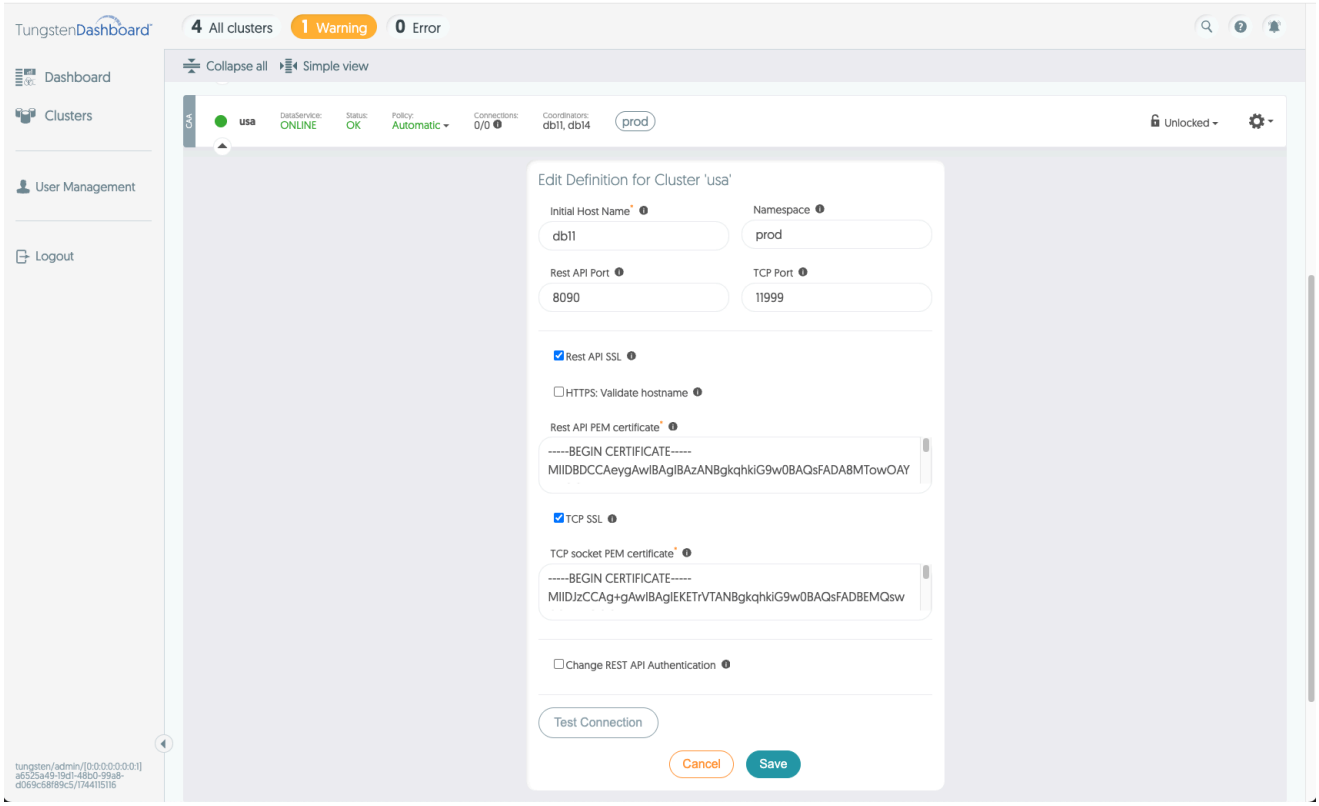
8.3. Tungsten Dashboard Exclusive operations

In addition to the control operations listed in the previous sections there are several operations unique to the Dashboard and it's 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 8.8, “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 8.3, “Composite Service Operations”] This prevents other Dashboard users from executing operations on the service without first unlocking the cluster by hand.

Figure 8.8. Cluster Edit Form



Chapter 9. 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 accesss to your encrypted data.

Chapter 10. Troubleshooting

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

10.1. 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

10.2. 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.

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 Compose 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
`DASHBOARD_ENV`	Environment definition (e.g. for different config files selection)
`DASHBOARD_SECRET`	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=e5I0vP+MFThNPBLbCcwf4xXrAWdf3mzK
DASHBOARD_DEFAULT_USER=admin
DASHBOARD_DEFAULT_USER_PASSWORD=admin
```

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
`configPath`	String, path to configuration file location. Defaults to /app/persistent/ for production
`helm`	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.
`logType`	Type of log to be written to the console. Supported values are console-edn
`version`	Information attached to application logs

Table A.3. Start-Up Settings

Option	Description
`clustersConnectOnStart`	When set to true, the dashboard will connect to the clusters on start-up. (Recommended for development use only)
`topologyCleanOnExit`	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)
`topologyStartupDiscover`	When set to true, the dashboard will discover clusters on startup. This includes a rest api call to the clusters in clusters.json file. (Recommended for development use only)

Table A.4. Webserver Settings

Option	Description
`browserPort`	If SSL is terminated on the dashbaord, the browserPort 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.
`domain`	Domain that dashboard is hosted on (including subdomain) - cannot be empty ie. example.com
`keystorePassword`	Password of keystore file. (only relevant for ssl)
`keystorePath`	Ppath to the key store file that contains the TLS certificate for dashboard server. Keystore must be of type JKS (only relevant for ssl)
`path`	Directory path at which the dashboard server is expected to respond to.
`port`	Webserver http and websocket port
`ssl`	true/false (default false). Use ssl server
`sslPort`	webserver https and websocket secure port.
`tokenLifetime`	Time to live of authentication token in seconds.

Option	Description
allowTokenRenewal	true/false - can the existing token be used to get a new one. defaults : true

Table A.5. Cluster Connection Settings

Option	Description
hostnameValidation	Global default for SSL hostname validation (can be overridden per cluster)
defaultRrestPort	Default REST port for clusters.
restTimeout	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.
defaultTcpPort	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
api-password	API Admin Password
api-user	API Admin User
api-cert	Base64 encoded PEM certificate (encoded including BEGIN and END markers)
api-port	Rest API port number for cluster. Default: 8090
api-ssl	Toggle SSL for API connections (true/false)
cert	Base64 encoded PEM certificate (encoded including BEGIN and END markers)
host	Initial host name to connect for a cluster discovery
namespace	Optional namespace label
ssl	Toggle SSL in TCP socket connections (true/false)
tcp-port	TCP port number for cluster. Default: 11999
hostname-validation	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
password	Password for user
role	Currently only 'admin' role is supported
username	Username (avoid numbers, spaces and special characters)

Appendix B. Release Notes

B.1. 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.