Manuel d'installation sous Kubernetes

Serveur Gargantua

Table des matières

Prérequis et préparation

Prérequis matériel

Un cluster Linux capable d'héberger un environnent Kubernetes fonctionnel.

Optionnellement, un serveur NAS ou autre service de stockage accessible dans le cluster Kubernetes, avec suffisamment d'espace disponible pour héberger le volume de données estimé :

- 4Go ou plus pour la base de données PostgreSQL
- 1Go ou plus pour le répertoire 'home'
- pour les zones de stockage des fichiers, selon le volume estimé

Ce chiffrage doit être adapté selon l'utilisation prévue pour le serveur Gargantua.

Prérequis logiciels

Un environnement Kubernetes version 1.28 ou plus récent.

Le gestionnaire de paquets helm pour Kubernetes, version 3.

Fichier de configuration

Pour obtenir le fichier de valeurs du chart de l'application Gargantua :

```
helm show values oci://harbor.node0.siatel.com/charts/gargantua
```

Le fichier par défaut au moment de la redaction de ce document est :

```
# Default values for gargantua.
# This is a YAML-formatted file.
# Declare variables to be passed into your templates.

plugins: [ "admininstaller", "browserplugins", "emailplugins", "officeplugins",
"cvtoivt" ]

# Database parameters
database:
  # The JDBC URL of the database; e.g:
"jdbc:postgresql://someservice.somenamespace.svc.cluster.local:5432/somedbname"
  # The database MUST have a user named 'siatel' configured with ownership-equivalent
access to the configured database
  # If left empty then a database with matching requirements will be deployed according
to supported and specified engine
  jdbcUrl: "" #"jdbc:postgresql://pgs1.default.svc.cluster.local:5432/dev8"

  # The password for the 'siatel' user; 'secretName' is favored over 'clearText'.
  password:
    secretName: "" # MUST have the 'siatel-password' field; will be looked up in the
namespace where the release is installed
    clearText: "siatel123" # highly discouraged in production.

  # Parameters for including and initializing a database with the deployment of
Gargantua
  type: "postgresql" # postgresql [| postgresql-ha | sqlserver | oracle] default is
"postgresql"

  # The included PostgreSQL deployment configuration.
  postgresql:
    image: postgres:17.0
    persistence:
      # The PostgreSQL data volume: [storageClass/size | existingClaim | hostPath]
      hostPath: ""
      existingClaim: ""
      storageClass: ""
      size: "8Gi"
    # Pod resources
    resources:
      requests:
        cpu: 100m
        memory: 128Mi
      limits:
        cpu: 2
        memory: 2Gi

##############################################
# Gargantua deployment parameters start here #
##############################################

# The deployment container image
```

```
image:
  repository: harbor.node0.siatel.com/releases/gargantua
  # This sets the pull policy for images.
  pullPolicy: IfNotPresent
  # Overrides the image tag whose default is the chart appVersion.
  tag: ""

# Persistent data stores.
persistence:

  # The 'home' datastore (i.e. configuration files)
  home:
    # The 'home' data volume: [storageClass/size | existingClaim | hostPath]
    hostPath: ""
    existingClaim: ""
    storageClass: ""
    size: "4Gi"

  # The 'storage' datastore (i.e. documents' contents)
  storage:
    # The 'storage' data volume: [hostPath | existingClaim | storageClass/size]
    hostPath: ""
    existingClaim: ""
    storageClass: ""
    size: "64Gi"

license:
  # The volume containing the 'server.lic' file [secretName | existingClaim | hostPath]
  secretName: "gargantua-license"
  existingClaim: ""
  hostPath: "" # "/mnt/f/rancher/g8-cloud-license"

# Additional volumes on the output Deployment definition.
volumes: [ ]
# - name: foo
#   secret:
#     secretName: mysecret
#     optional: false

# Additional volumeMounts on the output Deployment definition.
volumeMounts: [ ]
# - name: foo
#   mountPath: "/etc/foo"
#   readOnly: true

# Pod resources
resources:
  requests:
    cpu: 1
    memory: 2Gi
  limits:
    cpu: 8
    memory: 8Gi

# This will set the replicaset count more information can be found here:
https://kubernetes.io/docs/concepts/workloads/controllers/replicaset/
replicaCount: 1

# This is to override the chart name.
nameOverride: ""
fullnameOverride: ""

#This section builds out the service account more information can be found here:
https://kubernetes.io/docs/concepts/security/service-accounts/
```

```
serviceAccount:
  # Specifies whether a service account should be created
  create: false
  # Automatically mount a ServiceAccount's API credentials?
  automount: true
  # Annotations to add to the service account
  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: ""

# This is for setting Kubernetes Annotations to a Pod.
# For more information checkout: https://kubernetes.io/docs/concepts/overview/working-with-objects/annotations/
podAnnotations: { }
# This is for setting Kubernetes Labels to a Pod.
# For more information checkout: https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/
podLabels: { }

podSecurityContext: { }
# fsGroup: 2000

securityContext: { }
  # capabilities:
  #   drop:
  #   - ALL
  # readOnlyRootFilesystem: true
  # runAsNonRoot: true
# runAsUser: 1000

# This is for setting up a service more information can be found here: https://kubernetes.io/docs/concepts/services-networking/service/
service:
  # This sets the service type more information can be found here: https://kubernetes.io/docs/concepts/services-networking/service/#publishing-services-service-types
  type: ClusterIP
  # This sets the ports more information can be found here: https://kubernetes.io/docs/concepts/services-networking/service/#field-spec-ports
  port: 8080

# This block is for setting up the ingress for more information can be found here: https://kubernetes.io/docs/concepts/services-networking/ingress/
ingress:
  enabled: false
  className: ""
  annotations: { }
  # cert-manager.io/cluster-issuer: letsencrypt-staging
  # kubernetes.io/ingress.class: nginx
  # kubernetes.io/tls-acme: "true"
  # nginx.ingress.kubernetes.io/proxy-body-size: '1G'
  # nginx.ingress.kubernetes.io/configuration-snippet: >
  #   sub_filter 'http://{{.Release.Name}}.{{.Values.cloudSuffix}}:80' 'https://{{.Release.Name}}.{{.Values.cloudSuffix}}:443';
  #   sub_filter 'http://{{.Release.Name}}.{{.Values.cloudSuffix}}' 'https://{{.Release.Name}}.{{.Values.cloudSuffix}}';
  #   sub_filter_once off;
  hosts:
    - host: my-app.mydomain
      paths:
        - path: /
          pathType: ImplementationSpecific
  tls: [ ]
```

```
  #  - secretName: my-app-ingress-tls
  #    hosts:
  #      - my-app.mydomain

# This is to setup the liveness and readiness probes more information can be found here:
https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-
startup-probes/
#livenessProbe:
#  httpGet:
#    path: /
#    port: http
#readinessProbe:
#  httpGet:
#    path: /
#    port: http

#This section is for setting up autoscaling more information can be found here:
https://kubernetes.io/docs/concepts/workloads/autoscaling/
autoscaling:
  enabled: false
  minReplicas: 1
  maxReplicas: 1
  targetCPUUtilizationPercentage: 80
  # targetMemoryUtilizationPercentage: 80

nodeSelector: { }

tolerations: [ ]

affinity: { }
```

Personnaliser le fichier de configuration

Les paramétrages les plus fréquents à vérifier ou changer sont :

- **image**

```
image:
  repository: harbor.node0.siatel.com/releases/gargantua
```

- **ingress**

```
ingress:
  enabled: true
```

- **database**

```
# Database parameters
database:
  jdbcUrl: "" # example : "jdbc:postgresql://pgs1.default.svc.cluster.local:5432/
siatel"

  # The password for the 'siatel' user; 'secretName' is favored over 'clearText'.
  password:
    secretName: "" # MUST have the 'siatel-password' field; will be looked up in
the namespace where the release is installed
    clearText: "siatel123" # highly discouraged in production.

  # The included PostgreSQL deployment configuration.
  postgresql:
    persistence:
      existingClaim: "" # when configured using claims
      storageClass: ""
      size: "8Gi"
    # Pod resources
    resources:
      requests:
        cpu: 100m
        memory: 128Mi
      limits:
        cpu: 2       # set according to projected usage
        memory: 2Gi # set according to projected usage
```

- **persistence**

```
# Persistent data stores.
persistence:
  # The 'home' datastore (i.e. configuration files)
  home:
    existingClaim: ""  # when configured using claims
    storageClass: ""
    size: "4Gi"        # set according to projected usage

  # The 'storage' datastore (i.e. documents' contents)
  storage:
    existingClaim: "" # when configured using claims
    storageClass: ""
    size: "64Gi"      # set according to projected usage
```

Remplacer par les tailles requises par votre installation

Modules Gargantua

La clé `plugins` doit contenir tous les modules serveur Gargantua qui doivent être disponibles. Consultez votre licence pour récupérer la liste complète des modules serveur à installer.

```
plugins: [ "admininstaller", "browserplugins", "emailplugins", "officeplugins",
"cvtoivt" ]
```

Chaque module doit être inclus dans la liste avec son nom simple, pas le nom complet du fichier .jar

Les modules serveur Gargantua doivent être accompagnés par une licence valide, sinon ils seront désactivés.

Volumes nécessaires

Une installation Gargantua utilise au moins 3 volumes; une configuration avec plusieurs volumes est certainement possible et sera décrite dans la suite.

Les trois volumes requis par une installation Gargantua sont :

base de données

le volume pour la base de données PostgreSql est séparé des autres; sa taille doit être suffisante pour héberger les fichiers de la base; en général, 4 Go doivent être suffisants pour une base à usage général, environ 1 million documents, sans workflows complexes.

ce volume se retrouve dans le chemin `/var/lib/postgresql/data` dans le conteneur PostgreSQL

répertoire home

le répertoire home comprend les fichiers nécessaires pour le bon fonctionnement du serveur; parmi ceux-ci on a la configuration, les logs et la base texte intégral; les modules serveur sauvegardent aussi leur données dans cet espace, donc en fonction de la configuration il faut prévoir 1 Go ou plus.

ce volume se retrouve dans le chemin `/srv/siatel/home` dans le conteneur Gargantua

répertoire stockage

le répertoire de stockage est destiné au stockage des fichiers; ce volume inclut toujours la zone de stockage par défaut et potentiellement les autres zones de stockage

ce volume se retrouve dans le chemin `/srv/siatel/storage` dans le conteneur Gargantua

Eléments optionnels

Volumes optionnels

Certains répertoires de l'installation Gargantua peuvent être situés dans des volumes dédies.

répertoire logs applicatifs

chemin dans le conteneur : `/srv/siatel/home/logs`

répertoire de la base texte intégral

chemin dans le conteneur : `/srv/siatel/home/solr`

répertoire temporaire

chemin dans le conteneur : `/srv/siatel/home/temp`

répertoires supplémentaires de stockage de fichiers

chemin dans le conteneur : `/srv/siatel/storage/...` (`default` est un nom réservé)

répertoire des polices

si nécessaire, à part les polices standard, il faut fournir les polices personnalisées.

chemin dans le conteneur : `/srv/siatel/home/convert/msttcorefonts`

Installer Gargantua

Après avoir modifié le fichier valeurs, pour installer les composants nécessaires il faut exécuter **helm**.

```
helm install <app-name> oci://harbor.node0.siatel.com/charts/gargantua -n siatel --
values /srv/helm/gargantua.yaml
```

Résultat de l'installation

Apres avoir installé avec succès à l'aide de helm, dans le cluster on doit retrouver à minima dans le namespace choisi :

- 2 StatefulSet
  - `<app-name>-database`
  - `<app-name>-gargantua`
- Les trois PersistentVolumeClaim requis
  - `<app-name>-database-pvc`
  - `<app-name>-home-pvc`
  - `<app-name>-storage-pvc`
- 2 Services de type ClusterIP
  - `<app-name>-database`
  - `<app-name>-gargantua`
- 1 Ingress : `<app-name>-ingress`

Selon la config choisie, il faut vérifier la création de tous les autres objets configurés.

Structure des répertoires

```
|-- convert
|   |-- fonts
|   |-- msttcorefonts
|   |-- template
|   |   `-- siatel
|-- home
|   |-- conf
|   |-- convert -> /srv/siatel/convert
|   |-- logs
|   |-- plugin
|   |-- solr
|   |-- storage -> /srv/siatel/storage
|-- logs
|-- plugin
|-- solr
|-- storage
```

```
    |-- default
```

/convert

dossier racine des convertisseurs de documents

/convert/fonts, /convert/msttcorefonts

dossiers des polices personnalisées; pour un meilleur rendu des documents convertis (en imagettes ou en PDF) les polices utilisées doivent être disponibles dans ces dossiers

/home

dossier racine des l'applicatif Gargantua

/home/conf

dossier des fichiers de configuration, y compris la licence

/home/logs

dossier des fichiers logs applicatifs; en fonction du niveau de détail et période de rétention, l'espace disponible doit être suffisant

/home/plugins

dossier racine des modules d'extension Gargantua; dans ce dossiers on retrouve les fichiers des modules eux-mêmes (fichiers .jar) ainsi que les fichiers de configuration, s'ils existent, et les données spécifiques à chaque plugin, si c'est le cas.

/home/solr

dossier racine des bases texte intégral

/home/storage

dossier racine des zones de stockage pour les bases documentaires; pour une meilleure organisation des données, il est conseillé de monter ici les dossiers racine pour tous les espaces de stockage; chaque espace de stockage sera un dossier dans ce dossier racine, puis ce chemin est configuré dans Gargantua.

A noter qu'il n'y a pas d'obligation pour que la structure des espaces de stockage soit localisée dans ce dossier

/home/storage/default

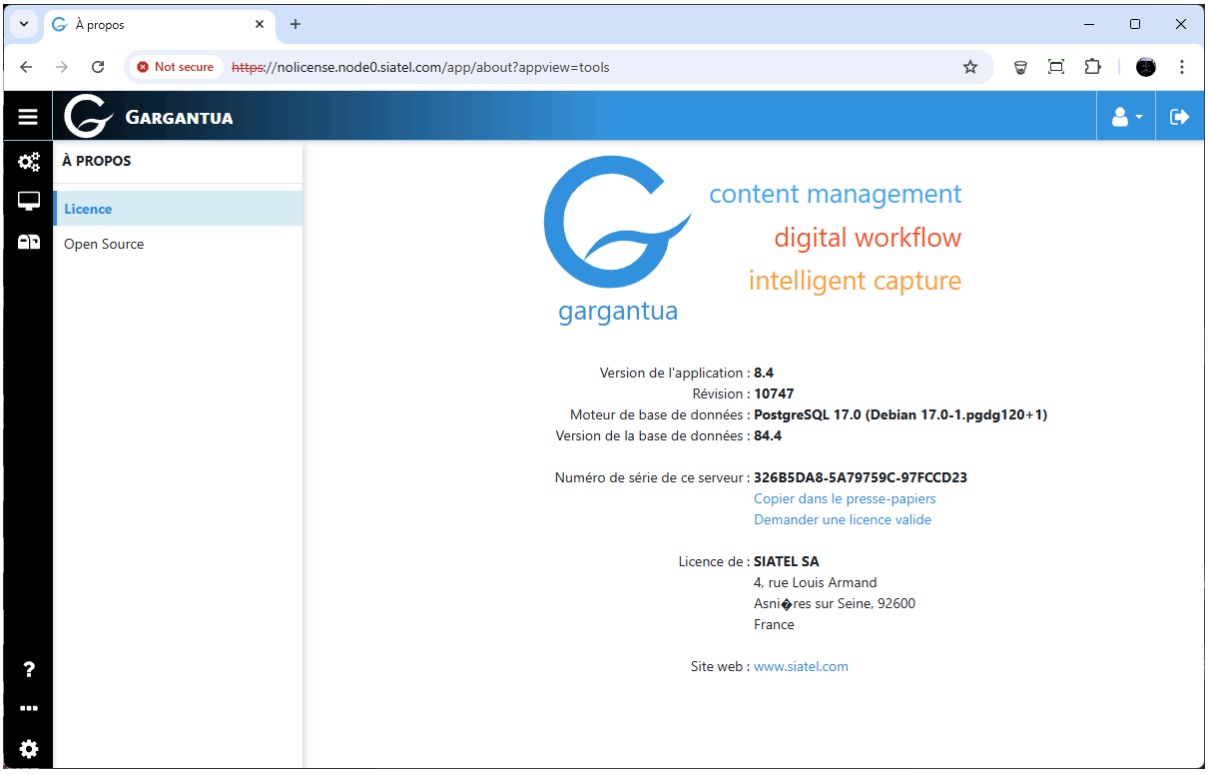
dossier racine de la zone de stockage par défaut; la zone stockage par défaut est la zone qui accueille les fichiers qui ne doivent pas être déposes dans une autre zone de stockage.

De point de vue volumes, chaque dossier inventorié ci-dessus peut-être un volume indépendant; cependant, il n'est pas vraiment nécessaire d'avoir autant de volumes que de dossiers; seulement les zones de stockage peuvent utiliser des volumes différents.

Après l'installation

Lors de l'installation sans fichier licence, le serveur Gargantua démarre en mode dégradé; la connexion n'est possible qu'avec le compte Admin, et aucun autre objet ne peut être crée.

Après la connexion en mode web, ouvrez la page 'A propos' pour récupérer le numéro de série du serveur :



Communiquez le numéro de série du serveur à SIATEL pour générer votre licence.

Après avoir reçu la licence, il faut la rendre disponible dans le StatefulSet `<app-name>-gargantua` dans le moint point `/license`, à travers un moyen de votre choix (par exemple un PersistentVolumeClaim, ConfigMap ou Secret).