



MANUAL DE DESPLIEGUE CON STATEFULSET K8S

Release Notes	2
Compatibilidad de versiones	2
Upgrade del despliegue a 4.s5	3
Paso 1 Ajustar configmaps de configuración	3
Paso 2 Recuperar personalizaciones	4
Paso 3 Desplegar	4
Despliegue del producto	5
Archivos de configuración de los microservicios	5
Opción 1: Archivos de configuración en local	5
Opción 2: Conectar Edusa a un repositorio externo	5
Despliegue de los microservicios	6
Secreto para credenciales	10
Secretos de los microservicios	10
PrivateKey Edusa	10
Configuración de los microservicios	10
Microservicios Core de Anjana	10
Plugins de Anjana	11
MinIO, Solr/Zookeeper ,Postgresql	11
Despliegue Solr/Zookeeper	11
Despliegue LDAP	12
Despliegue Minio	12
Monitorización de Anjana	13
CONFIGURACIÓN PREVIA:	14
COMANDOS ÚTILES K8S	19

Versión	Fecha de publicación	Responsable	Aprobador	Resumen de cambios
1.0	30/11/2023	Dpto DS	Responsable DS	Creación del documento

Release Notes

- Se ha implementado el despliegue de los nuevos plugins a través de statefulset.
- Se ha cambiado la configuración para que se adecúe a la versión 23.1

Compatibilidad de versiones

La presente versión del operador puede ser usada para versiones de Anjana Data:

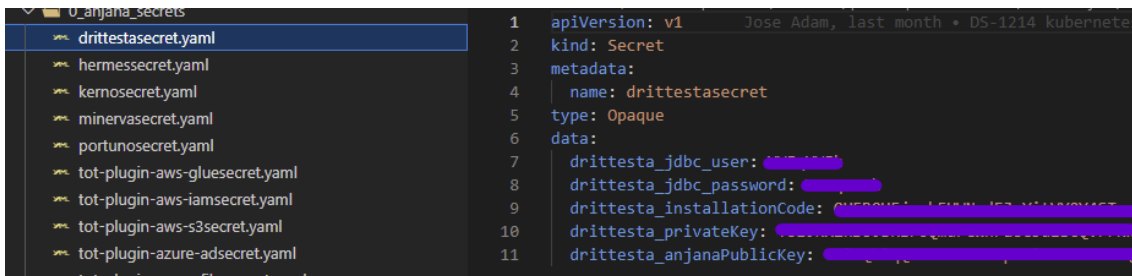
- 4.4
- 23.1

NOTA: En la presente versión del kit se incluyen preconfiguradas las últimas versiones bugfix de cada elemento tratado en el momento de la publicación, pudiendo ser publicadas versiones independientes de dichos elementos en fechas posteriores a la publicación del presente kit. Recuerde revisar y ajustar las versiones de los elementos a desplegar a la última versión de bugfix disponible.

Upgrade del despliegue a 4.s5

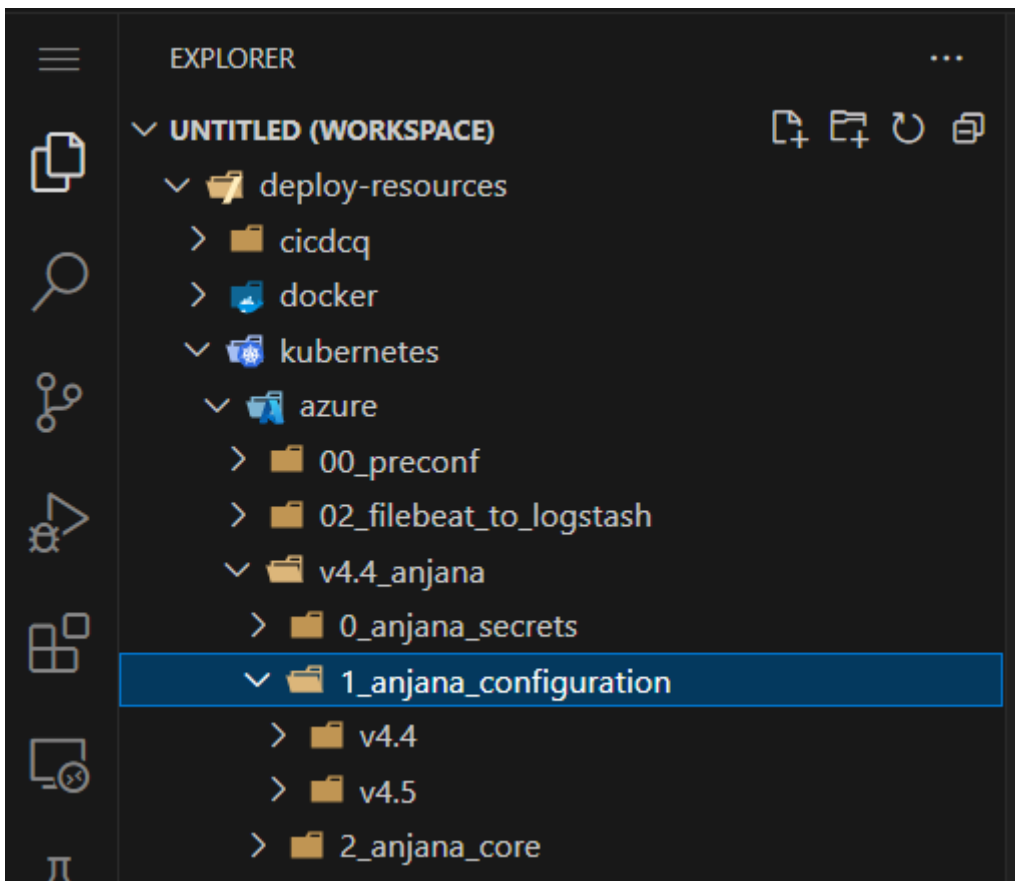
Paso 1 Ajustar configmaps de configuración

Debido al método de despliegue de los archivos de configuración mediante configmaps, el primer paso es configurar y crear los secretos con el contenido sensible de los archivos de configuración, como las credenciales.



```
1  apiVersion: v1
2  kind: Secret
3  metadata:
4    name: dritttestasecret
5  type: Opaque
6  data:
7    dritttesta_jdbc_user: [REDACTED]
8    dritttesta_jdbc_password: [REDACTED]
9    dritttesta_installationCode: [REDACTED]
10   dritttesta_privateKey: [REDACTED]
11   dritttesta_anjanaPublicKey: [REDACTED]
```

Dependiendo de la versión de Anjana, tenemos que entrar a la carpeta 4.4 o 4.5 .



```

UNTITLED (WORKSPACE)  deploy-resources > kubernetes > azure > v4.4_anjana > 1_anjana_configuration > v4.5 > configdrittestayaml > apiVersion
1  apiVersion: v1
2  kind: ConfigMap
3  metadata:
4    name: drittesta-config
5    namespace: anjana-system
6    labels:
7      app: drittesta
8  data:
9    application-default.yaml: |
10     logging:
11     pattern:
12     console: "%clr(ts="%d{yyyy-MM-dd HH:mm:ss.SSS}")%{faint} ms=DRITTESTA %clr(level=${LOG_LEVEL_PATTERN:
13     server:
14     port: 8095
15
16     spring:
17     profile: default
18     datasource:
19     url: jdbc:postgresql://rdbservice:5432/anjana?currentSchema=zeus
20     username: ${drittesta_jdbc_user}
21     password: ${drittesta_jdbc_password}
22     jpa:
23     properties:
24     hibernate:
25     default_schema: zeus
  
```

Paso 2 Recuperar personalizaciones

Los cambios o personalizaciones que se hicieran en el anterior despliegue deben ajustarse en el nuevo kit para no perderlo. Por ejemplo si se editó algún servicio o de los statefulset alguna image, puerto, namespaces, etc.

Paso 3 Desplegar

Al desplegar se tomará el lanzamiento como actualizaciones de los recursos existentes.

```

root@anjana17:/mnt/c/Users/eduardo/Documents/work/a_devops/deploy-resources/kubernetes/azure# kubectl apply -f v4.4_anjana/0_anjana_configuration
configmap/drittesta-config configured
configmap/edusa-config unchanged
configmap/hecate-config unchanged
configmap/hermes-config configured
configmap/kerno-config configured
configmap/minerva-config configured
configmap/portuno-config configured
configmap/tot-config unchanged
configmap/tot-p-aws-glu-config created
configmap/tot-p-aws-iam-config configured
configmap/tot-p-aws-s3-config configured
configmap/tot-p-az-ad-config configured
configmap/tot-p-az-file-config configured
configmap/tot-p-az-storg-config configured
configmap/tot-p-gcp-bq-config configured
configmap/tot-p-gcp-iam-config configured
configmap/tot-p-gcp-storg-config configured
configmap/tot-p-hdfs-config configured
configmap/tot-p-hive-config configured
configmap/tot-p-jdbc-config configured
configmap/tot-p-jdbc-deno-config configured
configmap/tot-p-jdbc-orac-config configured
configmap/tot-p-jdbc-reds-config configured
configmap/tot-p-jdbc-sqls-config configured
configmap/tot-p-ldap-config configured
configmap/tot-p-powerbi-config configured
configmap/viator-config configured
configmap/zeus-config configured

root@anjana17:/mnt/c/Users/eduardo/Documents/work/a_devops/deploy-resources/kubernetes/azure# kubectl apply -f v4.4_anjana/1_anjana_core
configmap/zeus-config configured
service/drittetaserver unchanged
statefulset.apps/drittesta configured
service/edusaserver unchanged
statefulset.apps/edusa configured
service/hecatesserver unchanged
statefulset.apps/hecate configured
service/heimdalsserver unchanged
statefulset.apps/heimdaldal unchanged
service/hermessserver unchanged
statefulset.apps/hermes configured
service/horusserver unchanged
statefulset.apps/horus configured
service/kernoserver unchanged
statefulset.apps/kerno configured
service/minervaserver unchanged
statefulset.apps/minerva configured
service/portunoserver unchanged
statefulset.apps/portuno configured
service/totserver unchanged
statefulset.apps/tot configured
service/viatorsserver unchanged
statefulset.apps/viator configured
service/websservice unchanged
configmap/httpd-config unchanged
statefulset.apps/web configured
service/webportunoservice unchanged
configmap/webportuno-config unchanged
statefulset.apps/webportuno configured
service/zeusserver unchanged
statefulset.apps/zeus configured
  
```

Despliegue del producto

Archivos de configuración de los microservicios

Debemos de configurar cada archivo acorde a la documentación (Anjana Data 23.1 - DS - Guía de configuración técnica), estos archivos de configuración serán guardados en un repositorio de git, o directamente en local, en este documento, explicaremos ambos.

Opción 1: Archivos de configuración en local

En el kit descargado, veremos que hay una carpeta llamada `1_anjana_configuration` , en la cual hay dos carpetas , una por versión de anjana soportada con este kit, donde estarán los archivos de configuración de los microservicios de anjana. Si queremos la instalación con este tipo de configuración, deberemos de lanzar esta carpeta antes de los statefulsets.

A su vez, se ha creado una carpeta llamado `0_anjana_secrets`, donde, en formato base64, tenemos todas las contraseñas y datos sensibles.

Opción 2: Conectar Edusa a un repositorio externo

En esta opción, deberemos ir a la configuración de Edusa, que se encuentra en la carpeta `1_anjana_configuration`, y cambiar el texto de `application.yaml` por uno como el siguiente.

```
configedusa.yaml M X
└─ deploy-resources > kubernetes > azure > v4.4_anjana > 1_anjana_configuration > v4.5 > configedusa.yaml > {} data >
  You, 36 seconds ago | 2 authors (eduardomasegosa and others)
  1 apiVersion: v1
  2 kind: ConfigMap
  3 metadata:
  4   name: edusa-config
  5   namespace: anjana-system
  6   labels:
  7     app: edusa
  8 data:
  9   application.yaml: |
 10     server:
 11       port: 8888
 12     spring:
 13       profiles: default
 14       application:
 15         name: <config_branch>
 16       cloud:
 17         config:
 18           server:
 19             git:
 20               uri: <url_repo_git> # git@bitbucket.org:repo_company/repo.git
 21               default-label: develop
 22               skipSslValidation: true
 23               timeout: 10
 24               clone-on-start: true
 25               force-pull: true
 26               searchPaths: '{application}'
 27               ignoreLocalSshSettings: true
 28               privateKey: "${PRIVATEKEY-CONFIGSERVER}"
```

La privateKey está lanzado como un secreto, que más adelante configuraremos.

Despliegue de los microservicios

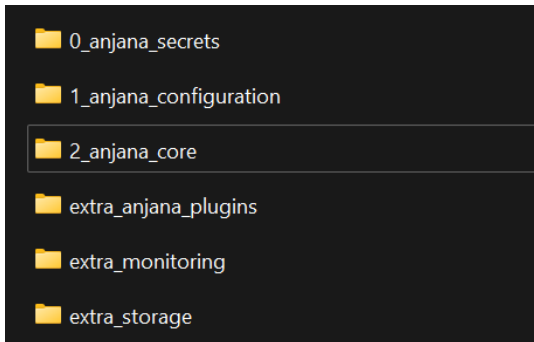
Cada archivo contiene un statefulset y un servicio que expone este servicio con una ip privada para simplificar el enrutado y las conectividades desde programas externos. Se puede cambiar este mecanismo, pero recuerda usar cualquier alternativa para exponer el producto web y la api a los usuarios y programas externos

Hay varias dependencias a considerar.

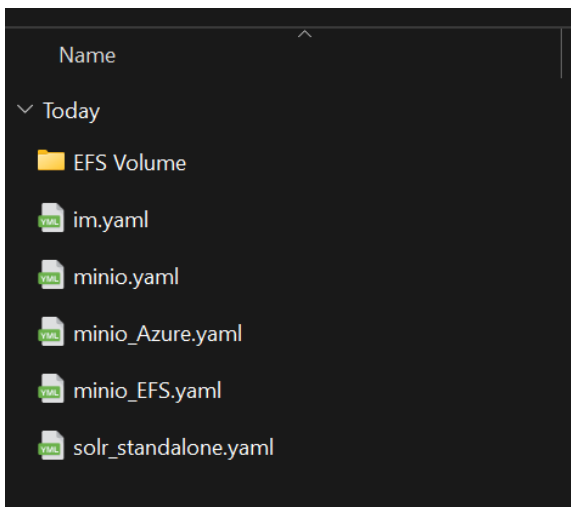
- Solr, Zk, postgresql y los programas externos deben ser lanzados antes de que se lance anjana, ya que si no, no podrán iniciarse
- Solr depende de Zookeeper
- Se necesita que tanto los buckets de s3 como los schemas de bbdd estén anteriormente creados, en caso contrario, fallará
- Antes de ser configurados los microservicios tenemos que lanzar los secretos.
- Todos los microservicios deben ser configurados antes de lanzarse.
- Todos los microservicios dependen de Edusa, ya que es quien le otorga los archivos de configuración.

Tenemos un tgz en Nexus con todos los yaml de los microservicios de anjana llamado kit-k8s-<version>. Con este kit, podemos descargar e instalar anjana con Statefulsets en Kubernetes

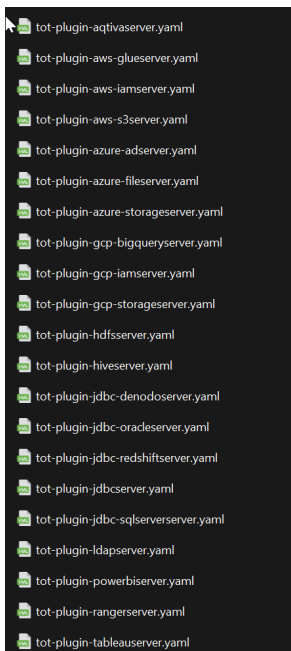
Si descomprimos el archivo, se verá así



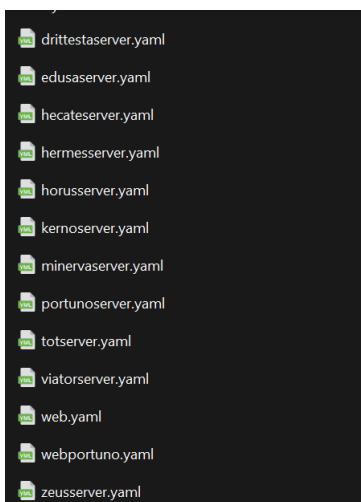
En la carpeta, de extra_storage, tendremos todo lo necesario para lanzar una instancia de solr y minio.



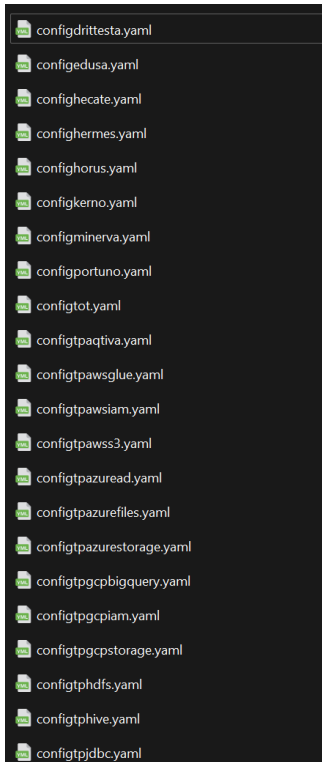
En la carpeta extra_anjana_plugins, encontraremos los Statefulset necesarios para la instalación de todos los plugins de Anjana.



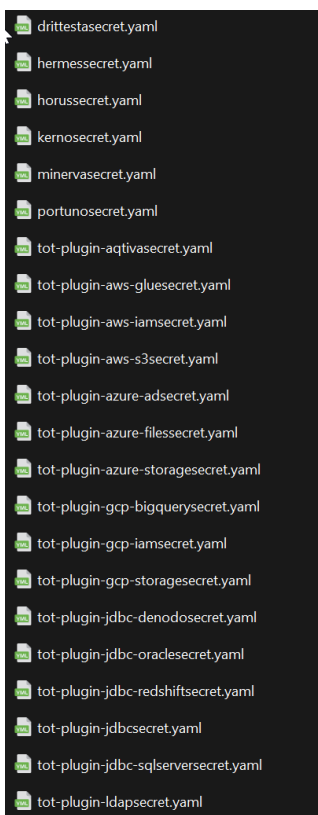
En la carpeta 2_anjana_core encontraremos todos los archivos para realizar un lanzamiento de Anjana.



En la carpeta 1_anjana_configuration encontraremos dos carpetas, diferenciando las versiones soportadas por este kit. Dentro de ellos, encontraremos todos los archivos de configuración de anjana, tanto de core, como plugins.



y por último en la carpeta 0_anjana_secrets encontraremos los secrets que tenemos que editar y lanzar para poner los secrets que después vamos a usar para editar la configuración.



Secreto para credenciales

De forma previa a la configuración de los microservicios y sus secretos, será necesario establecer el secreto que contiene las credenciales de acceso al repositorio de imágenes.

```
kubectl create ns anjana-system
kubectl --namespace anjana-system create secret docker-registry anjanadr
--docker-server=dr-releases.anjanadata.org --docker-username=<user> --docker-password=<pass>
--docker-email=<admin@email>
```

Secretos de los microservicios

En la configuración de los microservicios, usamos variables de entorno. Para que esas variables de entorno se creen en el pod, tenemos que lanzar antes los secretos. Para ello lanzaremos el siguiente comando.

```
# Para lanzar todos los archivos
kubectl apply -f 0_anjana_secrets -n anjana-system

# Para lanzar un archivo
kubectl apply -f 0_anjana_secrets/<nombre_servicio>secret.yaml -n anjana-system
```

PrivateKey Edusa

En la configuración de Edusa, podemos elegir entre un repositorio local o un repositorio Git. Para git, tenemos que pasarle la llave privada de git, que para no ponerlo en plano, lanzamos de la siguiente manera.

```
kubectl create secret generic privatekey-configserver \
-n anjana-system \
--from-literal=PRIVATEKEY-CONFIGSERVER='-----BEGIN RSA PRIVATE KEY-----
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX==
-----END RSA PRIVATE KEY-----'
```

Configuración de los microservicios

Una vez modificados para adecuarlos a nuestras necesidades, podemos lanzar todos los configmaps de los microservicios de la siguiente manera. Es importante hacerlo antes de lanzar los microservicios, ya que si no, estos fallarán.

```
# Para lanzar todos los archivos
kubectl apply -f 1_anjana_configuration/v4.X -n anjana-system

# Para lanzar un archivo
kubectl apply -f 1_anjana_configuration/n/v4.X/config<nombre_servicio>.yaml -n anjana-system
```

Microservicios Core de Anjana

Una vez lanzados los archivos de configuración, debemos de lanzar todos los archivos de dentro de la carpeta 2_anjana_core para levantar un anjana correctamente. A su vez, es obligatorio lanzar los yaml uno a uno para evitar los errores de liquibase.

```
# Orden de lanzamiento
edusa,hecate,portuno,zeus,drittista,kerno,minerva,viator,hermes,tot,web,webportuno
# Para lanzar un archivo
kubectl apply -f 2_anjana_core/<nombre_servicio>server.yaml -n anjana-system
```

Plugins de Anjana

De necesitar el lanzamiento de algunos de los plugins de Anjana, deberemos de, una vez lanzado su archivo de configuración, lanzar el yaml del plugin correspondiente

```
# Para lanzar todos los archivos
kubectl apply -f extra_anjana_plugins -n anjana-system
# Para lanzar un archivo
kubectl apply -f extra_anjana_plugins/<nombre_plugin>.yaml -n anjana-system
```

MinIO, Solr/Zookeeper ,Postgresql

Despliegue Solr/Zookeeper

Para desplegar estos servicios, debemos de ir donde tengamos el kit descargado, a la carpeta extra_storage, y lanzar el yaml correspondiente de Solr junto con Zk. Por ejemplo:

```
kubectl apply -f extra_storage/solr_standalone.yaml -n anjana-system
```

Se han añadido varias opciones utilizadas pero no mantenidas por Anjana que pueden ayudar a persistir los datos de Solr, como el uso de discos de Azure, volúmenes EBS de AWS o los volúmenes incluidos en los nodos de EKS/máquina en la que esté desplegado el cluster (especificados como Baremetal en el sts de Solr).

Para su uso es necesario descomentar la sección y rellenar los datos correspondientes, y aprovisionar los recursos que se quieran utilizar.

```
volumes:
  - name: config-volume
    configMap:
      name: solr-config

  ## ONLY FOR BAREMETAL
  # - hostPath:
  #   path: /opt/data/solrdata
  #   type: DirectoryOrCreate
  #   name: solrvc
  # - hostPath:
  #   path: /opt/solrconfig
  #   type: DirectoryOrCreate
  #   name: solrvc2

  # ONLY FOR AWS EBS
  - awsElasticBlockStore:
      volumeID: "volume-ID"
      fsType: ext4
      name: solrvc
  - awsElasticBlockStore:
      volumeID: "volume-ID"
      fsType: ext4
      name: solrvc2

  ## ONLY FOR AZURE
  # volumeClaimTemplates:
  #   - metadata:
  #       name: solrvc
  #       namespace: anjana-system
  #     spec:
  #       accessModes:
  #         - ReadWriteOnce
  #       storageClassName: azuredisk-retain
  #       resources:
  #         requests:
  #           storage: 4Gi
  #   - metadata:
  #       name: solrvc2
  #       namespace: anjana-system
  #     spec:
  #       accessModes:
  #         - ReadWriteOnce
  #       storageClassName: azuredisk-retain
  #       resources:
  #         requests:
  #           storage: 4Gi
```

IMPORTANTE: Si el nodo del cluster donde se encuentra Solr cae en una zona de disponibilidad distinta a la de los volúmenes, éstos no se podrán adjuntar. Habrá que crear un snapshot de ellos y restaurar en la zona correspondiente, o buscar sistemas de almacenamiento alternativos.

Despliegue LDAP

Para desplegar estos servicios, debemos de ir donde tengamos el kit descargado, a la carpeta `extra_storage`, y lanzar el yaml correspondiente de IM. Por ejemplo:

```
kubectl apply -f extra_storage/im.yaml -n anjana-system
```

Despliegue Minio

Para Minio, tenemos varios modos de lanzamiento.

Tenemos el `minio.yaml`, el cual nos proporciona un minio con un volumen de AWS EBS, el cual tendremos que configurar previamente, y añadir el `VolumeID` al archivo. También debemos de añadir el `accesskey` y el `secret_key` de minio en las siguientes variables

```
- name: 'MINIO_ROOT_USER'
  value: '<access_key>'
- name: 'MINIO_ROOT_PASSWORD'
  value: '<secret_key>'
ports:
- containerPort: 9000
  name: s3service
- containerPort: 9001
  name: minioui
volumeMounts:
- name: data
  mountPath: /data
volumes:
- awsElasticBlockStore:
  volumeID: "<volume_id>"
  fsType: ext4
  name: data
```

También tenemos otro yaml de lanzamiento de minio, llamado `minio_EFS`, el cual está configurado para aceptar un volumen EFS de Amazon. Para ello nos vamos a la carpeta llamada `EFS Volume`, y en el archivo llamado `pv.yaml`, debemos de añadir el `volume_id` del volumen EFS que hemos creado anteriormente en AWS, al igual que especificar el storage que tiene ese volumen.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: efs-pv
  namespace: anjana-system
spec:
  capacity:
    storage: 5Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  storageClassName: efs-sc
  csi:
    driver: efs.csi.aws.com
    volumeHandle: <volume_id>
    volumeAttributes:
      encryptInTransit: "true"
```

Cambiaremos también las variables de `minio_root_user` y `minio_root_password` en el archivo `minio_EFS` para adecuarlas a nuestro entorno.

```
env:
- name: 'MINIO_ROOT_USER'
  value: '<access_key>'
- name: 'MINIO_ROOT_PASSWORD'
  value: '<secret_key>'
```

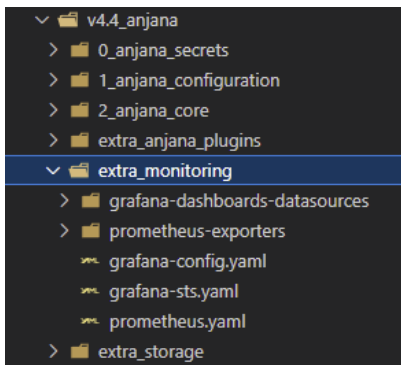
Para que anjana funcione correctamente, deben estar anteriormente creados y rellenos los buckets en minio. Hay que crearlos todos en privado, menos el bucket de CDN, que debe de ser público

Monitorización de Anjana

Hemos añadido la posibilidad de desplegar una plataforma de monitorización, que se compone de:

- Grafana, donde podemos crear cuadros de mando y configuramos los Data sources como fuente de datos
- Prometheus el cual provee las métricas a los cuadros de mandos
- Exporters de Solr/Zookeeper, Apache, PostgreSQL y Node, los cuales aportan métricas extras a Prometheus

Todos los archivos se encuentran disponibles en la siguiente carpeta del kit de Statefulset.



IMPORTANTE: La plataforma de monitorización está orientada a monitorizar el estado del aplicativo por lo que **NO** se ha provisto de persistencias. Al borrar los pods de la plataforma de monitorización o redespugarla **se perderá el histórico de todas las métricas**.

En el kit de Statefulset también están incluidos los exporters. Apache-exporter es el único que se despliega automáticamente con en el pod del front. El resto tienen que ser desplegados de manera manual.

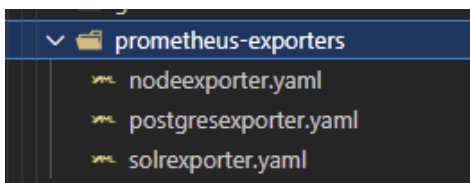
Para que la plataforma de monitorización recoja los datos necesarios, los siguientes exporters tienen que estar presentes:

- node-exporter
- postgres-exporter
- solr-exporter
- apache-exporter (ya predesplegado con el front)

CONFIGURACIÓN PREVIA:

Debemos de configurar los siguientes archivos para decirle a Prometheus y los exporters de qué máquinas se van a coger las métricas.

Para los exporters:



- **Node Exporter:** este exporter viene preconfigurado para levantar un pod por cada nodo del cluster de kubernetes. El exporter publicará un endpoint de métricas automáticamente con el puerto 32000, el cual puede ser configurado aquí:

```

> 2_anjana_core 36
  > extra_anjana_plugins 37
    > extra_monitoring 38
      > grafana-dashboards-datasources 39
        > prometheus-exporters 40
          nodeexporter.yaml 41
          postgresexporter.yaml 42
          solreexporter.yaml 43
          grafana-config.yaml 44
          grafana-sts.yaml 45
          prometheus.yaml 46
          extra_storage 47
        spec:
          imagePullSecrets:
            - name: anjanadr
          containers:
            - image: bitnami/node-exporter:latest
              imagePullPolicy: Always
              name: nodeexporter
              command:
                - /opt/bitnami/node-exporter/bin/node_exporter
                - --collector.textfile
                - --collector.textfile.directory=/var/lib/node_exporter
                - --web.listen-address=0.0.0.0:32000
                - --web.telemetry-path=/metrics
              ports:

```

- **Postgres Exporter:** para este exporter será necesario sustituir el valor “rdbservice” por el dominio/IP de la máquina/servicio donde se encuentre alojada la instancia de PostgreSQL a la que se quiere hacer referencia, así como el usuario y contraseña que se utilizará para la conexión.

```

> 1_anjana_configuration 160
> 2_anjana_core 161
  > extra_anjana_plugins 162
    > extra_monitoring 163
      > grafana-dashboards-datasources 164
        > prometheus-exporters 165
          nodeexporter.yaml 166
          postgresexporter.yaml 167
          solreexporter.yaml 168
          grafana-config.yaml 169
          grafana-sts.yaml 170
          prometheus.yaml 171
          extra_storage 172
        spec:
          imagePullSecrets:
            - name: anjanadr
          containers:
            - image: bitnami/postgres-exporter:latest
              imagePullPolicy: Always
              name: pgsqlexporter
              env:
                - name: DATA_SOURCE_NAME
                  value: postgres://anjana:anjana@rdbservice:5432/postgres?sslmode=disable
                - name: PG_EXPORTER_EXTEND_QUERY_PATH
                  value: /opt/postgres-queries.yml
              command:
                - /opt/bitnami/postgres-exporter/bin/postgres_exporter
                - --web.listen-address=:9187
              ports:
                - containerPort: 9187
                  name: pgsqlexporter
                  protocol: TCP

```

- **Solr Exporter:** este exporter viene preconfigurado para apuntar a un pod de Solr en kubernetes. Para apuntar a una instancia aparte habrá que sustituir la cadena “solr-0.indexservice.anjana-system.svc.cluster.local” por el DNS/IP correspondiente. También habrá que ajustar el modo en el que se encuentre desplegado Solr, por defecto se ha dejado seleccionado el modo solrcloud, siendo posible comentar esa línea y descomentar las otras dos para apuntar a un pod/instancia de Solr standalone, ajustando igualmente la cadena de conexión.

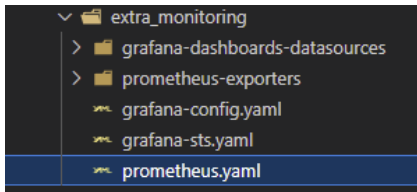
```

> 1_anjana_configuration 38
> 2_anjana_core 39
  > extra_anjana_plugins 40
    > extra_monitoring 41
      > grafana-dashboards-datasources 42
        > prometheus-exporters 43
          nodeexporter.yaml 44
          postgresexporter.yaml 45
          solreexporter.yaml 46
          grafana-config.yaml 47
          grafana-sts.yaml 48
          prometheus.yaml 49
          extra_storage 50
        spec:
          imagePullSecrets:
            - name: anjanadr
          containers:
            - image: releasesdr.anjanadata.org:11000/solreexporter:8.8.1
              imagePullPolicy: Always
              name: solreexporter
              command:
                - /opt/solr-8.8.1/contrib/prometheus-exporter/bin/solr-exporter
                - -p=9854
                - -z=solr-0.indexservice.anjana-system.svc.cluster.local:2181/solrcloud ## For SolrCloud Solr only
                - -z=solr-0.indexservice.anjana-system.svc.cluster.local:2181 ## For Standalone Solr only
                - -b=http://solr-0.indexservice.anjana-system.svc.cluster.local:8983/solr ## For Standalone Solr only
                - -f=/opt/solr-exporter-config.xml
                - -n=1
              ports:

```


Para Prometheus:

Vamos al archivo de configuración 0-prometheus.yaml.



El archivo viene preconfigurado para apuntar a los exporters configurados previamente, el único valor a reemplazar será el “s3service” por el DNS/IP de la instancia de MinIO.

Cabe destacar que cuando se elige la opción de Solr standalone, las métricas de zookeeper no están disponibles, por lo que su correspondiente dashboard no mostrará ninguna información. En caso de ser necesario, se pueden ajustar los valores para apuntar a exporters ubicados fuera de este kit.

```
- job_name: 'solr'
  static_configs:
    - targets:
      - solrexporter-0.solrexporterservice.anjana-system.svc.cluster.local:9854
      - nodeexporter-0.nodeexporterservice.anjana-system.svc.cluster.local:32000
    labels:
      env: 'solr'

- job_name: 'apache'
  static_configs:
    - targets:
      - web-0.webservice.anjana-system.svc.cluster.local:9117
    labels:
      env: 'apache'

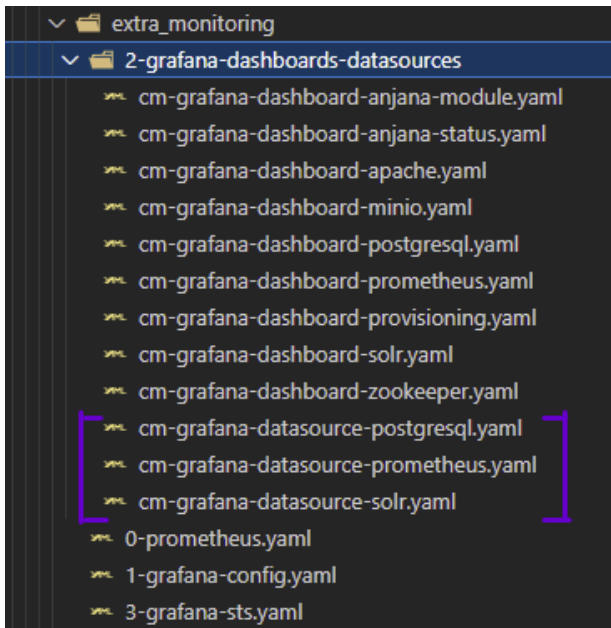
- job_name: 'zookeeper'
  static_configs:
    - targets:
      - solr-0.indexservice.anjana-system.svc.cluster.local:7000
    labels:
      __metrics_path__: /metrics
      env: 'zookeeper'

- job_name: 'minio'
  static_configs:
    - targets:
      - s3service:9000
    labels:
      __metrics_path__: /minio/v2/metrics/cluster
      env: 'minio'

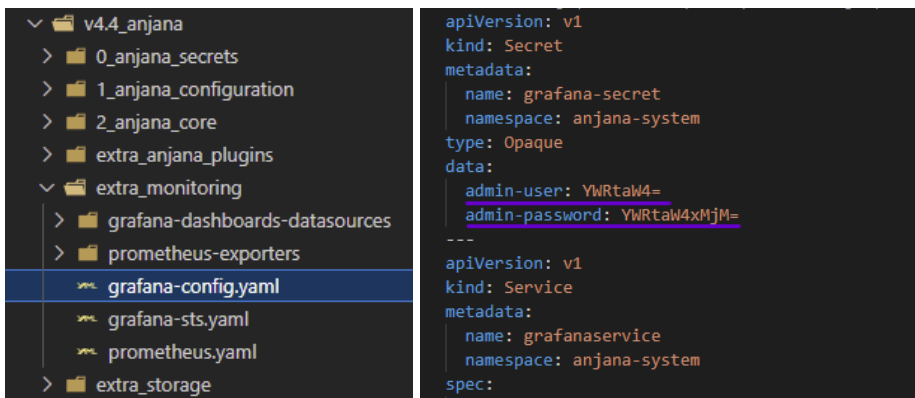
- job_name: 'postgres'
  static_configs:
    - targets:
      - pgsqlexporter-0.pgsqlexporterservice.anjana-system.svc.cluster.local:9187
      - nodeexporter-0.nodeexporterservice.anjana-system.svc.cluster.local:32000
    labels:
      env: 'postgres'
```

Se puede establecer como fuente de datos más de un nodo del mismo tipo, es decir, más de un apache o más de un Solr si fuese necesario. De esta manera, si tenemos el exporter de apache o el que corresponda en cada una de las máquinas, Prometheus cogerá las métricas de ambas máquinas.

También habrá que buscar y sustituir los valores “rdbservice” e “indexservice” por el DNS/IP de las instancias de PostgreSQL y Solr respectivamente en los datasources que se configurarán en Grafana. Los ficheros se encuentran en:



Las credenciales de acceso a Grafana tienen que ser configuradas en el archivo siguiente, en la parte correspondiente al secret. Serán rellenados en base64:

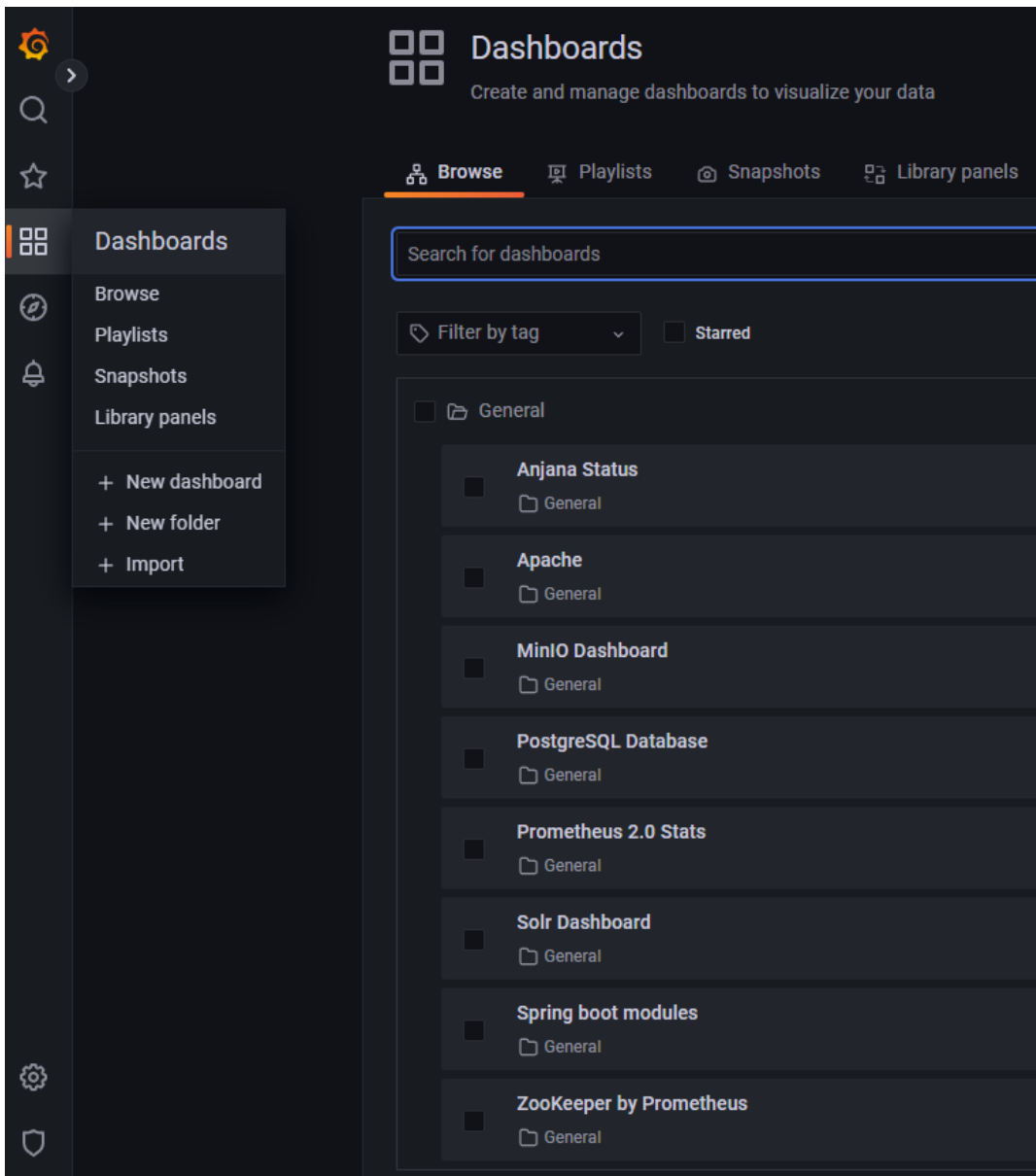


Una vez configurado todo para lanzar la plataforma de monitorización al completo sería con el siguiente comando:

```
kubectl apply -f extra_monitoring/ -R
```

Una vez lanzado y editado las configuraciones pertinentes, ya podremos entrar a Grafana y ver los dashboards. Para poder entrar habrá que establecer un port forwarding con el siguiente comando, y tras eso estará disponible en localhost:3000. Por defecto las credenciales son usuario “admin” y contraseña “admin123”.

```
kubectl port-forward svc/grafanaservice 3000:3000
```



The screenshot shows the 'Dashboards' section of the Anjana interface. The main heading is 'Dashboards' with the subtitle 'Create and manage dashboards to visualize your data'. Below this, there are navigation tabs for 'Browse', 'Playlists', 'Snapshots', and 'Library panels'. A search bar is present with the placeholder text 'Search for dashboards'. There are also filter options: 'Filter by tag' (with a dropdown arrow) and a 'Starred' checkbox. A list of dashboards is displayed under a 'General' folder, each with a checkbox and a folder icon:

- Anjana Status (General)
- Apache (General)
- MinIO Dashboard (General)
- PostgreSQL Database (General)
- Prometheus 2.0 Stats (General)
- Solr Dashboard (General)
- Spring boot modules (General)
- ZooKeeper by Prometheus (General)

The left sidebar contains navigation icons for settings, search, star, and a dashboard grid icon. A dropdown menu is open for the dashboard grid icon, showing options: 'Dashboards', 'Browse', 'Playlists', 'Snapshots', 'Library panels', '+ New dashboard', '+ New folder', and '+ Import'.

COMANDOS ÚTILES K8S

Las credenciales para conectar al cluster están en :

```
/root/.kube/config
```

Listar todos los recursos (-o wide muestra más detalle):

```
kubectl get namespace
kubectl get pod -n <namespace_anjanadata> -o wide
kubectl get service -n <namespace_anjanadata> -o wide
kubectl get statefulset -n <namespace_anjanadata> -o wide
kubectl get configmaps -n <namespace_anjanadata>
kubectl get all -n <namespace_anjanadata>
```

Acciones sobre un recurso en concreto añadiendo el nombre al comando anterior:

```
kubectl get pod edusa-0 -n <namespace_anjanadata> -o wide
kubectl delete pod edusa-0 -n <namespace_anjanadata>
kubectl edit statefulset edusa -n <namespace_anjanadata>
```

Ver el log de un microservicio

```
kubectl logs edusa-0 -n <namespace_anjanadata>
kubectl logs edusa-0 -n <namespace_anjanadata> -f (Con -f muestra el log en vivo "stream")
kubectl logs edusa-0 -n <namespace_anjanadata> --previous (Con --previous muestra el log del anterior pod y se puede ver el error por el que se borró)
```

Ver detalle de un recurso

```
kubectl describe pod edusa-0 -n <namespace_anjanadata>
```

Ejecutar comandos en un pod o contra un pod

```
kubectl exec edusa-0 -n <namespace_anjanadata> -- cat /opt/data/...
kubectl exec -it edusa-0 -n <namespace_anjanadata> -- sh
kubectl cp <namespace_anjanadata>/edusa-0:/opt/data/configrepo/zeus/application-default.yaml
application-default.yaml
```

Si un pod no llega a arrancar y queremos hacerle troubleshooting podemos forzar en el comando de arranque del pod un bucle para poder lanzar el comando "exec -it" y entrar en modo interactivo.

```
kubectl edit statefulset edusa -n <namespace_anjanadata>
  command: [ "/bin/sh", "-c", "--" ]
  args: [ "while true; do sleep 30; done;" ]
```

```
- name: anjanadr
containers:
- name: edusa
  image: releasesdr.anjanadata.org:11000/edusa:4.4.0
  command: [ "/bin/sh", "-c", "--" ]
  args: [ "while true; do sleep 30; done;" ]
  imagePullPolicy: Always
  ports:
```

```
kubectl exec -it edusa-0 -n <namespace_anjanadata> -- sh
```

Hacer un escalado con el número de pod de cada statefulset. Por ejemplo, antes de un upgrade, cuando tengamos que parar el backend, podemos utilizar estos comandos:

```
kubectl scale statefulset edusa --replicas=0
kubectl scale sts --replicas=0 -l layer=anjanabackend,env=pre -n <namespace_anjanadata> (con
-l puedes filtrar por los labels)
```

tip: para ver las labels

```
kubectl get sts -n <namespace_anjanadata> --show-labels
```