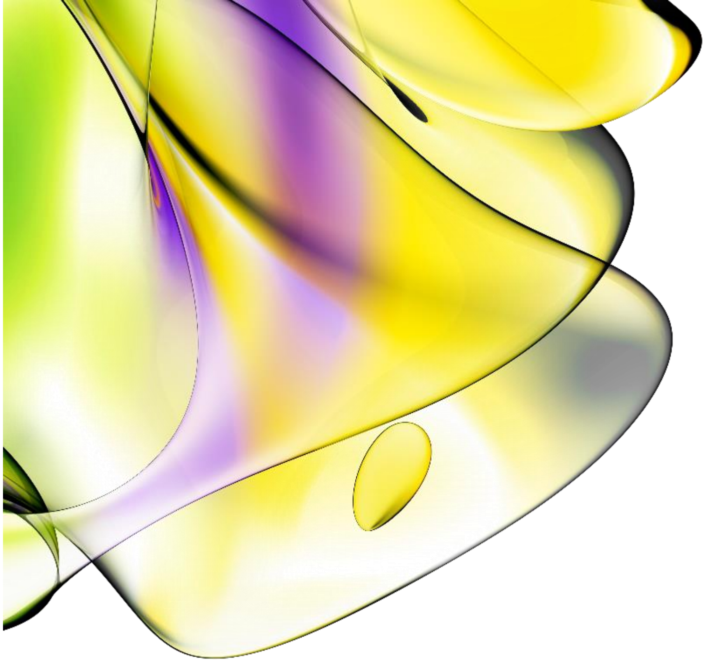


Revvity Signals Inventa™ 3.5.1

Installation Guide (Bare Metal Kubernetes)

Powered by Spotfire®

Last Modified: April 12, 2024

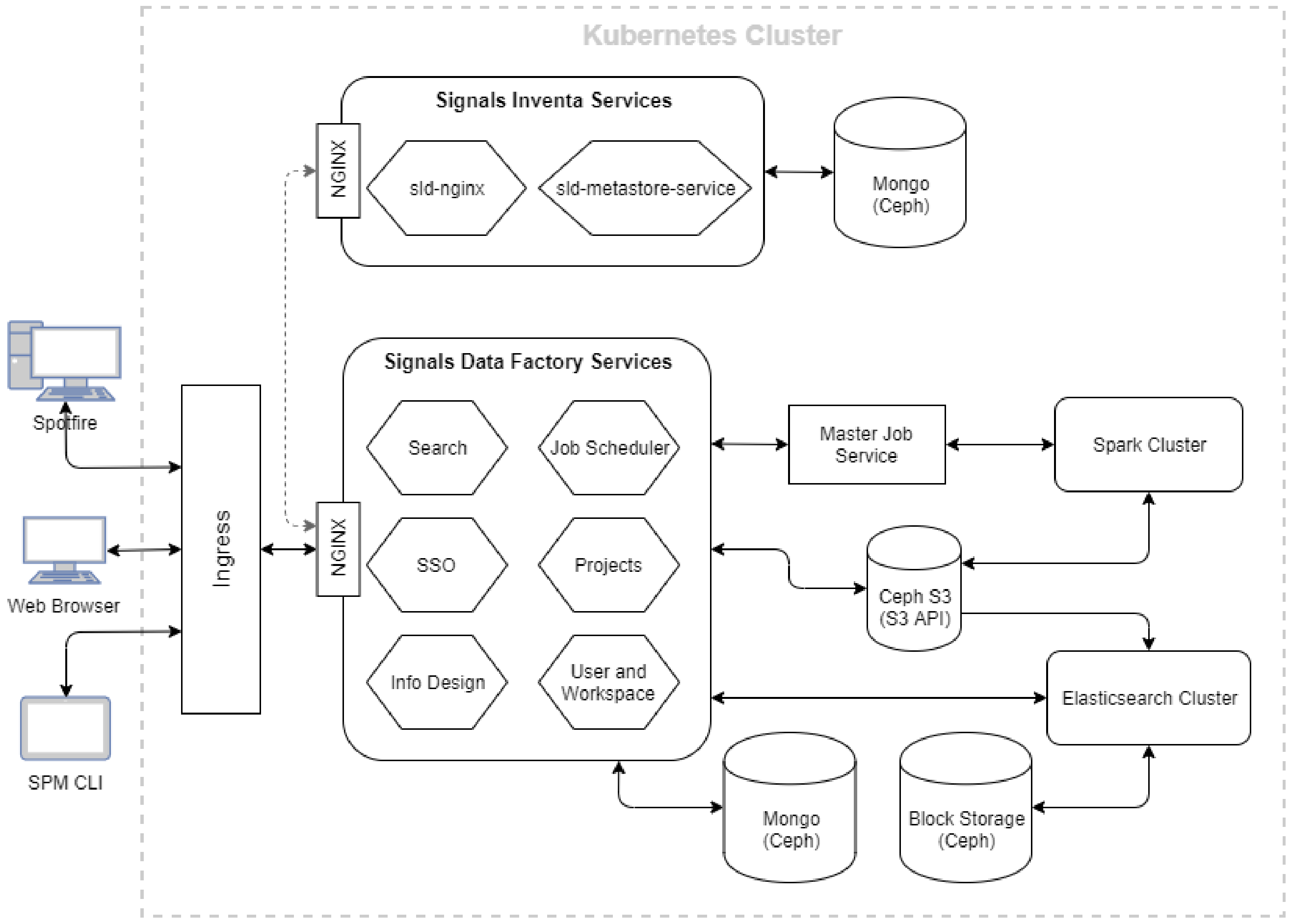
Table of Contents

1 Introduction

Signals Inventa^{TM} contains the following components:

- The Signals Data Factory^{TM} with which users are able to configure the research projects, upload and transform the scientific data and store the transformed data in a global index such that the data could be queried and downloaded.
- A reversed proxy service which delegates the front-end API requests to the proxied back-end services.
- A Metastore service with which users can manage application metadata and publication datasets. It contains the Metastore application and the underlying MongoDB NoSQL database.
- A data publishing tool from which users can define the mappings between measurement types and Spotfire table columns and publish the data from the selected Spotfire table to a centralized location.
- A Global Search tool with which users can specify either compound batches criteria or assay results criteria or both to query against the Signals Data Factory to see the summary of data and download both compound batches and assay results from Signals Data Factory.
- A Publish Measurements app, a Global Search app and a SAR Analysis app which integrated Signals Apps framework to provide easy-to-use data publishing, data searching, downloading and analyzing capabilities.
- An Assay Results Review app, allowing users to compare the historical data downloaded from Signals Data Factory with the current assay data
- A Target Engagement Profile app that incorporates both in vitro inhibition data and in vivo pharmacokinetic and efficacy data such that compounds can be efficiently compared for clinical nomination

The following high-level deployment diagram demonstrates the components of Signals Inventa.



In the above diagram:

- sld-metastore-service microservice will be used to maintain the metadata needed by Signals Inventa. The data will be stored in the backend MongoDB NoSQL database.
- sld-nginx microservice acts as a gateway to the internal Signals Inventa APIs.
- Signals Data Factory service components provides RESTful APIs for managing projects, designing metadata, searching for or downloading compounds and assays, and scheduling backend jobs for data transformation and indexing.
- Master Job Service is a back-end service component which provides the capability of scheduling and managing data import and publish jobs. It will communicate with the Spark and ElasticSearch cluster for data processing.
- Client components, such as data publishing tool and global search GUI, will communicate with Signals Inventa services to achieve the data publishing, querying and downloading features.

2 Prerequisites

This Installation Guide explains the steps required to deploy and set up Signals Inventa, based on the assumption that the following prerequisites are satisfied:

1. **Spotfire and required components are installed correctly**
    - Spotfire Analyst and Web Player has been installed and configured.
    - Revvity Lead Discovery Premium has been correctly deployed and configured in Spotfire. Please refer to *"Revvity Lead Discovery Premium v3.5 Installation Guide.pdf"* for details.
2. **1 or 3+ machines are available to be used as nodes for the cluster**
    - The number of cluster nodes must be an odd number
    - Minimum 8 CPUs, 64GB RAM with 1 TB storage for each machine
        - It is required that all cluster machine storage is using Solid State Hard Drives (SSDs). The recommendation is that these SSDs should provide at least 3000 IOPS (Input/Output Operations per second) and 250MB/s throughput for the better performance on data importing, publishing and searching.
        - You should use an unformatted attached disk as the storage device and reserve 850GB at minimal on it for storing the Signals Inventa metadata and the imported and indexed assay data
        - You should reserve 150GB at minimal on your storage device for storing docker images and runtime data. By default, this data is stored in `/var/lib/docker`, but the location can change depending on your OS distribution. If you don't have the sufficient disk space being mounted to the root (/), you need to change the docker daemon directory to the location that is on a mounted storage which has sufficient disk space. For the detailed steps, please check the Docker official documentation: https://docs.docker.com/config/daemon/#docker-daemon-directory
        - A small amount of runtime data is stored at /var/lib/rook and this is currently not configurable. Please ensure that at least 1GB of space is available in this filesystem location.
    - Operating System: Either Ubuntu 20.04 LTS or Ubuntu 22.04 LTS. For security considerations on Ubuntu 22.04 LTS, please refer to the **Important Note for Installing Inventa on Ubuntu 22.04 LTS** section below
    - Networking prerequisites:
        - Following ports should be allowed for inbound traffic for the nodes to communicate with each other: 22, 80, 443, 2376, 2379, 2380, 6443, 8472/udp, 9099, 10250, 10254, 10256, 30000-32767, 30000-32767/udp.
        - Following ports should be allowed for outbound traffic for the nodes to communicate with each other: 443, 2379, 2380, 6443, 8472/udp, 9099, 10250, 10254, 10256.
        - It is recommended using machines that have 25Gbps network bandwidth, and that lower networking bandwidth may reduce performance.
    - (**Important**) Time updating should be configured correctly by default. Most Linux operating systems include a mechanism to auto-synchronize the time from a centralized service like time.gov, this capability should be enabled and configured properly by default on all the nodes in the cluster. Specifically, these nodes should be able to communicate with the time synchronization services via Network Time Protocol on specific ports, most commonly it is the

UDP port 13 or 123, depending on the time service. If you have a restrictive environment, it's worth checking that some form of time update service is enabled and working on the machines, otherwise it will cause some unpredictable issues to your deployment. Followings are the online resources that describe the way of setting up time synchronization on different systems:

- Ubuntu: Use timedatectl to control the system time and date
- Ubuntu: How to Set up Time Synchronization With NTP

3. **A client machine for managing the cluster installation is available**

- It must be either an **Ubuntu 20.04 LTS** or an **Ubuntu 22.04 LTS** machine.
- The installation process itself only requires a marginal amount of storage, however, if you want to perform the data backup/restore tasks described in Appendix E: Data Backup and Restore in the future on this client machine, it needs to have enough storage to be able to download a compressed copy of all the data in the system. It is helpful to have several hundred gigabytes reserved, depending on your data size. However, if you configure the backups to be downloaded to another location, you don't need the space here.
- It is a good practice to put this client machine in a different subnet than the 3 node machines that will be used for the Kubernetes cluster.
- It must have **jq**, the lightweight and flexible command-line JSON processor, installed. For installation instructions, please refer to: https://stedolan.github.io/jq/download/
- It must have **kubectl** installed. Supported **kubectl** versions are 1.20, 1.21 and 1.22. For installation instructions, please refer to: https://kubernetes.io/docs/tasks/tools/install-kubectl/. To make sure that the kubectl is installed correctly, run the following command:

```
$ kubectl version --client
```

  This should give you the Client Version information.

- It must have **helm 3** installed. Supported **helm** versions are from 3.4.1 to 3.11.1. For installation instructions, please refer to: https://helm.sh/docs/intro/install/. To make sure that helm is installed correctly, run the following:

```
$ helm version
```

  This should give you the version information of the helm that you have installed.

- It must have **Rancher RKE** version **1.5.3** installed; no other Rancher RKE version is supported. For installation instructions, please refer to: https://rancher.com/docs/rke/latest/en/installation/. To make sure that Rancher RKE is installed correctly, run the following:

```
$ rke --version
```

  This should give you the version number of Rancher RKE.

- It must have **unzip** installed. Use the following command to install unzip:
  For Ubuntu:

```
$ sudo apt install -y unzip
```

To make sure that unzip is installed correctly, run the following:

```
$ unzip -v
```

This should give you the about information of unzip

- It must have **Python 3** installed. **Note that Python 2 is NOT supported.** The preferred version of Python 3 is 3.6 or above. Please refer to https://www.python.org/downloads to get the installer. After the installation completes, run the following command:

```
$ python3 --version
```

This should show the version of the Python 3 that you've installed

- It must have **pip3** installed. Use the following command to install pip3:

For Ubuntu:

```
$ sudo apt install -y python3-pip
```

Execute the following command to check whether pip3 has been installed correctly.
This should show the version of pip3:

```
$ pip3 -V
```

- It must have **SELinux bindings for Python3** installed if your client machine is running with SELinux enabled. To install **SELinux bindings for Python3**:

For Ubuntu:

```
$ sudo apt install -y python3-selinux
```

- Use the following command to install **ansible**:

```
$ sudo python3 -m pip install ansible-core==2.13.9 ansible==6.1.0 jmespath==1.0.1
pyyaml==6.0 kubernetes==12
```

Supported toolset versions:

- **ansible**: 5.8, 6.0 and 6.1 (With the core versions 2.12.10 or 2.13.9)
- **jmespath**: 1.0.0 and 1.0.1
- **pyyaml**: 6.0
- **kubernetes** (python library): 12

- Now you can execute ansible command under the subshell:

```
$ ansible --version
```

This will give the output similar to the following:

```
ansible [core 2.13.9]
  config file = None
  configured module search path = ['/home/ubuntu/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/local/lib/python3.8/dist-packages/ansible
  ansible collection location = /home/ubuntu/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/local/bin/ansible
  python version = 3.8.10 (default, Mar 13 2023, 10:26:41) [GCC 9.4.0]
  jinja version = 3.1.2
  libyaml = True
```

Important Note for Installing Inventa on Ubuntu 22.04 LTS

Ubuntu 22.04 LTS shipped with a newer version of `sshd`, which disabled the support of ssh-rsa HostkeyAlgorithm by default. In order to install Signals Inventa on Ubuntu 22.04 LTS, some custom configurations must be applied to this built-in `sshd` service before you can continue with the installation. Before making the changes, please consider your organization's internal security policies compliance and query your IT/Security department for the details when necessary.

To enable the ssh-rsa as a supported HostkeyAlgorithm, on each of the node machine, do the followings:

1. Query the supported host key types by executing the following command:

   ```
   $ ssh -Q sig | tr '\n' ',' | sed 's/,$/\n/'
   ```

   This will give the results similar to the following:

   ```
   ssh-ed25519,sk-ssh-ed25519@openssh.com,ssh-rsa,rsa-sha2-256,rsa-sha2-512,ssh-dss,ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,ecdsa-sha2-nistp521,sk-ecdsa-sha2-nistp256@openssh.com
   ```

2. Open sshd configuration file with a text editor, for example:

   ```
   $ sudo vim /etc/ssh/sshd_config
   ```

3. Add the following line to the sshd_config file:

   ```
   HostkeyAlgorithms <supported_key_types>
   ```

   Where <supported_key_types> is the value you got from step 1 above

4. Add or uncomment the setting `PubkeyAuthentication` in the configuration file, set its value to `yes`. For example:

   ```
   PubkeyAuthentication yes
   ```

5. Add the following line to the configuration file to enable the ssh-rsa as a supported key type for PubkeyAuthentication:

   ```
   PubkeyAcceptedKeyTypes+=ssh-rsa
   ```

6. Save the changes, exit the text editor and execute the following command to apply the changes:

   ```
   $ sudo service sshd restart
   ```

3 Installing Services

This section explains the installation procedures for installing Signals Inventa services.

3.1 Setting up Node Machines ready for Kubernetes Installation

Before installing Kubernetes, node machines need to be provisioned so that they can be added to the cluster. Each node must be setup and configured correctly.

Note: Each of these node machines should be setup for authentication via an SSH key. This SSH key will be used for setting up the Kubernetes cluster in later steps.

Note: lvm2 package must be installed on each of the node machine.

Run the following commands on each of the cluster nodes to configure and provision them:

```
sudo apt-get update &&
sudo apt-get install -y \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg-agent \
    software-properties-common \
    lvm2
```

Note that after executing the above commands, you should install the docker on each of the node machines. The supported docker versions are 17.03, 17.06, 17.09, 18.06, 18.09, 19.03 and 20.10. For more information about the docker installation, please refer to the following documentation:

https://docs.docker.com/engine/install/

3.2 Installing Kubernetes Cluster and Signals Data Factory Services

Follow the steps below to setup the Kubernetes cluster and install the Signals Data Factory services:

1. Logon to the **client machine** which already has the prerequisites installed. Then copy the Signals Data Factory installation package (**sdf-standalone-cluster-deployment-ansible-sld.zip**) to this machine.
2. Unzip the installation package:

```
$ unzip sdf-standalone-cluster-deployment-ansible-sld.zip -d ~
```

3. Switch to the sdf-standalone-cluster-deployment-ansible folder:

```
$ cd ~/sdf-standalone-cluster-deployment-ansible
```

4. Make a copy of `cluster-configuration.yaml`:

```
$ cp cluster-configuration-ceph.yaml cluster-configuration.yaml
```

5. Edit the `cluster-configuration.yaml` file with a text editor, perform the following edits:

a. `ingress_host`: (*Required*). The host name that you wish to use for the Signals Data Factory deployment. Note that the IP address is not supported. For example: **sdf.example.com**

   Note: If your Kubernetes nodes are deployed on Amazon Web Services, you may create a Record Set in your Route 53 configuration on Amazon Web Services portal and point it to the IP addresses of all the nodes of your cluster. The `ingressHost` here is the name of that Record Set

b. `ansible_private_key_file`: (*Required*). The path to the SSH key file which is used for connecting the node machines. This key file should be put onto the **client machine** and set the path to this key file in this `ansible_private_key_file` configuration

c. `ansible_user`: (*Required*). The name of the SSH user to log into the cluster node machine that you have just provisioned for the Kubernetes cluster installation

d. `cluster_name`: (*Required*). Give a name to your cluster

e. `dataStorageDevice`: The device id of the drive you wish to use on every node machine for data storage. For example, you can use the lsblk command on the node machine to get a list of the storage devices, then choose the one that you wish to use it as the data storage device and put its id as the value of this setting

f. `enforceSecureCookie`: (*Optional*). Set this flag to `true` to make sure that Signals Data Factory Single-Sign-On (SSO) could work properly in integrating with Spotfire**. Important Note**: If your Signals Data Factory is going to be deployed under HTTP rather than HTTPS, please set this flag to `false`, otherwise you will not be able to login to Signals Data Factory

g. `enableSciStream`: (*Optional*). Set this flag to `true` if you want to use the SciStream features

h. `showPipelines`: (*Optional*). Set this flag to `true` if you want to show pipelines under the projects

i. `enableMeasurementRename`: (Optional). Set this flag to `true` if you want to enable the measurement type renaming feature. **Important Note**: Renaming a measurement type could cause unpredictable results to your system, so it should be avoided. Please don't use this feature unless you understand the consequences

j. `hierarchy`: (Optional). If you want to change the nomenclature of the hierarchical entities, you can configure it here. Please make sure that you only change the "Compound" and "Batch" and leave others untouched:

```
hierarchy: "Group,Set,Row,Column,Project,Compound,Batch,Sample"
```

k. `hosts`: The primary/external IP of the node machine that you have just provisioned for the Kubernetes cluster installation. You can get the IP of the node machine by logging on to the node machine and execute **ifconfig** command. If you are using AWS EC2 instances as the node machines, it is the **IPv4 Public IP** of your node machine. Note that you should only specify the IP of one of your node machines here, to specify the IP of other node machines, you should duplicate all the settings under the **hosts** section for all of the node machines and specify the IP for each of the nodes separately

Note that you must duplicate the node settings under the **hosts** section to reflect the number of node machines you have for the cluster setup. Following is an example of the **hosts** section of the `cluster-configuration.yaml` file which sets up a three-node cluster:

```
k8s_nodes:
  hosts:
    "192.168.100.101":
      internal_ip: "10.1.21.122"
    "192.168.100.102":
      internal_ip: "10.1.20.83"
    "192.168.100.103":
      internal_ip: "10.1.16.215"
```

You can just specify one entry under the `hosts` section if you only have one node in your cluster setup.

6. Execute the following command under the same directory as **cluster-configuration.yaml** to install Kubernetes cluster, the Ceph storage, the core infrastructure services and Signals Data Factory client services:

```
$ ./scripts/deploy-sdf-storage-k8s.sh
```

You will finally see the play recap message similar to the following, with all “failed” equal to zero, indicating that everything is installed successfully:

```
[06:35:41] system | -- Play recap --
127.0.0.1                : ok=68   changed=21   unreachable=0    failed=0
192.168.100.101          : ok=20   changed=5    unreachable=0    failed=0
192.168.100.102          : ok=20   changed=5    unreachable=0    failed=0
192.168.100.103          : ok=20   changed=5    unreachable=0    failed=0
```

You will also see the deployment information similar to below:

```
{
  - msg: SDF Client URL: <your_sdf_url>
         Ceph dashboard URL:<your_sdf_url>:32300
         Ceph dashboard password: &L]Q;o+2B!BR;:!^6"a`
         Number of Spark Clusters: 1
         Livy memory: 2Gi
         Livy memory limit: 10Gi
         Spark worker memory: 12Gi
         Spark worker memory limit: 15Gi
         Spark driver memory: sparkDriverMemory: 6Gi
         Elastic Search memory: 23Gi
         Elastic Search heap memory: 11Gi
}
```

Make a note of the Ceph dashboard password, it can be useful for troubleshooting purposes

Note: If you encounter the error similar to the following one while executing the `deploy-sdf-storage-k8s.sh` script:

```
UNREACHABLE!: Failed to create temporary directory.In some cases, you may have been able to
authenticate and did not have permissions on the target directory. Consider changing the remote
tmp path in ansible.cfg to a path rooted in "/tmp", for more error information use -vvv. Failed
command was: ( umask 77 && mkdir -p "` echo /home/ubuntu/.ansible/tmp `"&& mkdir "` echo
/home/ubuntu/.ansible/tmp/ansible-tmp-1629092103.1590614-25539-55236818610831 `" && echo ansible-
tmp-1629092103.1590614-25539-55236818610831="` echo /home/ubuntu/.ansible/tmp/ansible-tmp-
1629092103.1590614-25539-55236818610831 `" ), exited with result 1
```

Please follow the steps below and then try executing the deploy script again:

a) Under the `sdf-standalone-cluster-deployment-ansible` directory, open the `ansible.cfg` file with a text editor

b) Add `remote_tmp = /tmp/.ansible-${USER}/tmp` to the end of this configuration file

c) Save the changes and exit the editor

7. Once the setup has finished successfully, a new file named **kube_config_cluster.yml** will have been generated under the current directory.

   Execute the following command to copy the file to the Kubernetes configuration folder such that `kubectl` and `helm` tools could communicate with the cluster:

```
$ mkdir -p ~/.kube && cp kube_config_cluster.yml ~/.kube/config
```

   Note: A successful setup will also generate the `cluster.yml` and `cluster.rkestate` files under the current directory, please make a copy of these files as they will be used when you are going to upgrade Signals Inventa in future.

8. Verify that the Kubernetes cluster has been installed successfully:

```
$ kubectl get nodes
```

   This will list all the nodes being added to the Kubernetes cluster, for example:

```
NAME               STATUS   ROLES                       AGE   VERSION
192.168.100.101    Ready    controlplane,etcd,worker    5m    v1.19.2
192.168.100.102    Ready    controlplane,etcd,worker    5m    v1.19.2
192.168.100.103    Ready    controlplane,etcd,worker    5m    v1.19.2
```

9. You can check if all pods are in “Running” or “Completed” state by executing the following command and find the pods whose name starts with `sdf`.

```
$ kubectl get po
```

   You will get the results similar to the following:

```
NAME                                                        READY   STATUS    RESTARTS   AGE
sdf-ci-master-job-apis-6764f44f4-7ckj9                      1/1     Running   0          8m16s
sdf-ci-master-job-mongo-67b5cc4b7f-cfnlb                    1/1     Running   0          8m16s
sdf-ci-master-job-scheduler-6fdbbd95b4-qw2sc                1/1     Running   0          8m16s
sdf-core-infrastructure-elasticsearch-master-0              1/1     Running   0          7m56s
sdf-core-infrastructure-elasticsearch-master-1              1/1     Running   0          7m56s
sdf-core-infrastructure-elasticsearch-master-2              1/1     Running   0          7m55s
sdf-cs-signals-data-transformation-service-8ffc6669f-jcbkz  1/1     Running   0          5m56s
sdf-cs-signals-data-transformation-service-8ffc6669f-lqssv  1/1     Running   0          5m56s
sdf-cs-signals-job-scheduler-898647669-r4n4t                1/1     Running   0          5m56s
```

```
sdf-cs-signals-mongo-6587566576-64g2v                       1/1     Running     0          5m56s
sdf-cs-signals-nginx-74f995578d-2vjw8                       1/1     Running     0          5m56s
sdf-cs-signals-nginx-74f995578d-kv7mx                       1/1     Running     0          5m56s
sdf-cs-signals-primary-api-64dcfcfbff-v9m6h                 1/1     Running     0          5m56s
sdf-cs-signals-primary-api-64dcfcfbff-zgtqg                 1/1     Running     0          5m56s
sdf-cs-signals-seeder2-8tdbr                                0/1     Completed   2          5m56s
sdf-cs-signals-sso-9ff8fb675-6vfrv                          1/1     Running     1          5m56s
sdf-cs-signals-ui-68d66fdc64-4m4bf                          1/1     Running     0          5m56s
sdf-spark-1-sdf-spark-standalone-livy-6769c5ff4b-76wth      1/1     Running     0          6m51s
sdf-spark-1-sdf-spark-standalone-master-56c5f974d8-9tqwb    1/1     Running     0          6m51s
sdf-spark-1-sdf-spark-standalone-worker-7ff89dd5ff-5625s    1/1     Running     0          6m51s
sdf-spark-1-sdf-spark-standalone-worker-7ff89dd5ff-hj4r6    1/1     Running     0          6m51s
sdf-spark-1-sdf-spark-standalone-worker-7ff89dd5ff-r8qzx    1/1     Running     0          6m51s
```

10. Now navigate to the URL configured by `ingress_host` setting in your web browser. You should see the Signals Data Factory login page, this implies that you've deployed Signals Data Factory successfully on the Kubernetes cluster.
11. Login Signals Data Factory with the username "`super`" and password "`password`". You can then manage user accounts from the **Signals** menu. For more information about account management and permission settings, please refer to **Revvity Signals Inventa User Guide.**
12. After logging into the Signals Data Factory, you need to create a default project used by Signals Inventa. Follow the instructions in the **Revvity Signals Inventa User Guide** to create a project named Signals Inventa, which will be used in the later steps when setting up Signals Inventa services.

3.3 Integrating SAML-compliant Identity Providers (Optional)

Signals Data Factory can support SAML-compliant identity providers (SAML IdPs), so that you can integrate the SAML identity provider of your corporate with Signals Data Factory and allow end users to authenticate by using their credentials from the corporate SAML identity provider.

Note: You will need to prepare another host name to be used by the SAML-compliant identity provider integration, end users should use this new host name to access the login page provided by the configured identity provider.

To enable the SAML-compliant identity provider integration, please follow the steps below.

1. On the **client machine**, from **sdf-standalone-cluster-deployment-ansible** folder, open `cluster-configuration.yaml` file with a text editor, make the following changes:

   `enableSAMLAuthentication`: Set to true

   `samlIngressHost`: The host name that you've prepared for the SAML-compliant identity provider integration. For example: **sdf-saml.example.com.** Note that the IP address is not supported

   `samlTLSCertificateName`: (optional) If you want to enable SSL on the SAML-compliant integration, you can use this setting to specify the name of the TLS certificate that you've imported into Kubernetes. For more information about enabling SSL, please refer to: Appendix A: Enable SSL on Service Deployment.
2. Execute the following command to update your Signals Data Factory client services deployment:

```
$ ./scripts/deploy-client-services.sh
```

3. On the **client machine**, extract the Signals Inventa deployment package with the following command:

```
$ mkdir -p ~/sia-cluster-deployment && tar zxvf sia-cluster-deployment.tar.gz -C ~/sia-
cluster-deployment
```

4. From **~/sia-cluster-deployment** folder, open `config-saml.json` file with a text editor, make the following changes:

    - `idpDescription:` Specify a descriptive text for the SAML IdP
    - `idpKey`: The BASE64-encoded value of the SAML IdP certificate in PEM format
    - `loginUrl`: The URL of the login page provided by SAML IdP
    - `assertUrl`: The SAML assertion post back URL. For example, if the `samlIngressHost` is set to "**sdf-saml.example.com**" and the SSL is also enabled on the service deployment, the assertUrl should be set to https://**sdf-saml.example.com**/sso-auth/saml-consume
    - `postLogoutUrl`: The logout URL provided by SAML IdP. For example, in an Auth0 SAML provider, the postLogoutUrl is similar to: **https://my-tenant.auth0.com/v2/logout?client_id=kZisifji232ascohhorvKL**

    Note: For some scenarios, you may need the following settings to be configured on your SAML tenant. For example, in the Azure SSO integration scenario, you should set the `idpType` to be equal to "`saml`" and "`samlRequestRequired`" to be "`true`". These settings are not provided by default in the `config-saml.json file`, however, you can modify the `config-saml.json` file to add them manually so that the SAML-compliant IdP integration can meet your deployment requirements.

    - `SamlRequestSpID`
    - `IdpType`
    - `SamlRequestForceLogin`
    - `SamlRequestRequired`

5. Save the changes, on the same **client machine**, execute the following command to configure the spm tool:

    ```
    $ echo "default http://<ingress_host> super password" > ~/.signals.cfg
    ```

    Where `<ingress_host>` is the host name you configured to the `ingress_host` setting in section 3.2 step 5 above. For example, **sdf.example.com.** Note that if you enabled the SSL on the service deployment, you will need to use **https** instead of **http**. Also note that there is no tailing slash at the end of the URL, for example, the URL http://sdf.example.com/ is an invalid URL

6. On the same **client machine**, switch to the directory where the spm tool exists by using:

    ```
    $ cd ~/sdf-standalone-cluster-deployment-ansible/tools/linux-amd
    ```

7. Use the following command to create a new workspace for the new host name you've set for `samlIngressHost`:

    ```
    ./spm -e "create-workspace-host -d 'Revvity Signals' -c <sia-cluster-deployment-
    path>/config-saml.json -workspace-name sdf <samlIngressHost>"
    ```

    Where:

    - `<sia-cluster-deployment-path>` is the **full path** where Signals Inventa deployment package has been extracted to in step 3 above

- `<samlIngressHost>` is the host name you configured for the `samlIngressHost` setting in section 3.3 step 1 above. For example, **sdf-saml.example.com**

8. You will need to configure or customize the claims mapping in your SAML IdP, such that claims information from the SAML IdP could be correctly routed and identified by Signals Data Factory. For example, you may prefer to use an email address from SAML IdP as the username in Signals Data Factory, you can have the following claims mapping set up if you use Auth0 as your SAML IdP:

```
{
  "mappings": {
    "email": "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier"
  }
}
```

   Note that different identity providers will have different ways of setting up the claims mapping, please refer to the user guide of your SAML IdP for the detailed steps.

9. Signals Data Factory will not automatically synchronize the user accounts from your SAML IdP, so you will need to do that manually to create the user accounts in Signals Data Factory, the username of these user accounts should match the ones in your SAML IdP. Switch to the directory where the spm tool exists and execute the following command to create the user account under the new host name:

```
$ ./spm -e "create-user -roles superadmin,admin -password <user_pass> -host
<samlIngressHost> <user_name>"
```

   Note:

   - The `-roles` parameter specifies the roles that the newly created user account will belong to. The command above will create a user account with both `SUPER_ADMIN` and `ADMIN` roles. The different between `SUPER_ADMIN` and `ADMIN` is that, `SUPER_ADMIN` will have the privilege of managing user accounts. You can create a regular user (non-admin user) by omitting the `-roles` parameter
   - `<user_pass>`: Specify the password for the newly created user account. In the SAML IdP integration scenario, this password will not be used for authentication, the authentication workflow will be handled by the SAML IdP
   - `<samlIngressHost>` is the host name you configured for the `samlIngressHost` setting in section 3.6 step 1 above. For example, **sdf-saml.example.com**
   - `<user_name>` is the name of the user account to be created

To test the deployment, access the host you set for the `samlIngressHost` setting in a web browser, you should see your SAML IdP login page, login with the configured user account, you should be navigated to the dashboard of Signals Data Factory.

3.4 Installing Signals Inventa Services

3.4.1 Import Base Information Design

Before Signals Data Factory is going to be used by Signals Inventa, some initial data needs to be created in the system. For example, the Assay Endpoint Results measurement type is a special measurement type that must exist in order for Signals Inventa to be functioning correctly. Such initial data is called Base Information Design in Signals Data Factory.

To import Base Information Design:

1. On the **client machine**, extract the Signals Inventa Command Line Utilities (siautils) tool package with the following command:

   ```
   $ mkdir -p ~/siautils && unzip siautils-linux-x64-<version>.zip -d ~/siautils && cd
   ~/siautils
   ```

   Where the `<version>` is the version number of the command line utilities tool.

   Note: You can also use this command line utilities tool under Windows operating system. The package name for Windows is `siautils-win-x64-<version>.zip`. For more information about the system requirements of the Signals Inventa Command Line Utilities tool, please refer to **Revvity Signals Inventa Command Line Utilities User Guide**

2. (Linux only) Make sure that the file siautils is executable under the current user account:

   ```
   $ chmod +x siautils
   ```

3. Execute the following command to generate the `base-info-design.csv` file that needs to be loaded into Signals Data Factory:

   ```
   $ ./siautils create-info-design --url http://<ingress_host> -u super
   ```

   Where `<ingress_host>` is the host name you configured to the `ingress_host` setting in section 3.2 step 5 above. For example, **sdf.example.com.** Note that if you enabled the SSL on the service deployment, you will need to use **https** instead of **http**. Also note that there is no tailing slash at the end of the URL, for example, the URL http://sdf.example.com/ is an invalid URL.

   After entering the password, the command will generate the base-info-design.csv and the SPM script under the `info-design-scripts` folder. The output of the command execution would be something similar to the following:

   ```
   Configured hierarchical entities: [Compound, Batch]

   Writing base-info-design.csv ...

   Generating SPM script ...

   Done.
   ```

   Note: For more information about the options of the create-info-design command, please refer to **Revvity Signals Inventa Command Line Utilities User Guide**

   Note: The Out-of-the-Box (OotB) datasets provided in the Signals Inventa installation package already contains the measurement types and maps definitions used for Signals VitroVivo 4P Dose-Response IC50 Assay workflow as well as all datasets for Inventa OotB Dashboard example. If you are going to load the Out-of-the-Box datasets that are provided in the Signals Inventa installation package, please go directly to step 4 below. You can also use the `create-info-design` command with the `-b` parameter to generate all those files. For example:

   ```
   $ ./siautils create-info-design --url http://<ingress_host> -u super -b
   ```

   For more information about the steps of loading OotB datasets that is provided in the Inventa installation package, please refer to Importing Out-of-the-Box Datasets (Optional) chapter.

4. On the same **client machine**, create a symbolic link to the Signals Data Factory SPM command line tool under the `/usr/local/bin` folder by using the following command:

```
$ sudo ln -s ~/sdf-standalone-cluster-deployment-ansible/tools/linux-amd/spm
/usr/local/bin/spm
```

5. On the same **client machine**, execute the following command to configure the spm tool if you didn't do the same thing before:

```
$ echo "default http://<ingress_host> super password" > ~/.signals.cfg
```

   Where `<ingress_host>` is the host name you configured to the `ingress_host` setting in section 3.2 step 5 above. For example, **sdf.example.com.** Note that if you enabled the SSL on the service deployment, you will need to use **https** instead of **http**. Also note that there is no tailing slash at the end of the URL, for example, the URL http://sdf.example.com/ is an invalid URL

6. On the same **client machine**, switch to the `info-design-scripts` directory by using:

```
$ cd ~/siautils/info-design-scripts
```

7. Execute the following command to import the base information design and the OotB measurement types and maps if you've chosen to import them as well:

```
$ spm default.spm
```

   After the command has executed successfully, you should see output on the terminal similar to the following, with no errors:

```
Importing Information Design File

Attributes Report

        Attributes (created)

                Batch  :  Batch Project ID

                Column  :  Assay Date

                Column  :  Endpoint Name

                Column  :  Endpoint Qualifier

                Column  :  Endpoint Type

                Column  :  Endpoint Unit

                Column  :  Endpoint Value (Numeric)

                Column  :  Endpoint Value (Text)

                Column  :  Is Numeric

                Column  :  Parent Measurement Row ID

                Column  :  Parent Measurement Type

                Column  :  s_h1

                Column  :  s_h2

                Column  :  s_h3

                Compound  :  Chemical Structure


Measurement Report

        Measurement 1:

                created  :  Assay Endpoint Results

                Measurement attributes:
                        linked  :  Assay Date
                        linked  :  Endpoint Name
                        linked  :  Endpoint Qualifier
                        linked  :  Endpoint Type
                        linked  :  Endpoint Unit
                        linked  :  Endpoint Value (Numeric)
                        linked  :  Endpoint Value (Text)
                        linked  :  Is Numeric
                        linked  :  Parent Measurement Row ID
                        linked  :  Parent Measurement Type
                        linked  :  s_h1
                        linked  :  s_h2
                        linked  :  s_h3

Parse report
{
    "missingGroups": []
}
base-info-design.csv uploaded successfully
```

Important Note: The measurement types and attributes imported from base-info-design.csv are important to make sure that the system can function correctly, so, please do not change any portion of these imported measurement types and attributes (e.g. names, data types, relationships, etc.) under any circumstances.

3.4.2 Deploying Signals Inventa Services

There are three microservices in Signals Inventa that are essential for the entire system to work properly. This section introduces the details to deploy and configure these four microservices. Follow the steps below to deploy the services:

1. From the **client machine**, switch to the helm directory (where Signals Inventa helm chart exists by) using the following command :

   ```
   $ cd ~/sia-cluster-deployment/helm
   ```

2. Copy `values-template.yaml` to `values.yaml` by using the following command

   ```
   $ cp values-template.yaml values.yaml
   ```

3. Go to parent folder and execute the following command:

   ```
   $ cd .. && ./deploy-inventa.sh ../sdf-standalone-cluster-deployment-ansible sia
   ```

   The script file 'deploy-inventa.sh' requires two arguments to execute:

   - the SDF installer directory which contains the 'cluster-configuration.yaml' file
   - the target HELM deployment name for the services

   The default values will be used as the above command example if no arguments are specified.

4. Execute the following command to see if all pods are in the "Running" state:

   ```
   $ kubectl get po -l "app=sia"
   ```

This will give the output similar to the following:

```
NAME                                      READY    STATUS     RESTARTS    AGE
sia-metastore-service-7b4cc6d9f6-flb7q    1/1      Running    0            5m
sia-mongo-7487ff57c-wfjlf                 1/1      Running    0            5m
sia-nginx-687cc79bdd-xsc67                1/1      Running    0            5m
```

3.4.3 Enabling Signals Inventa Integration

In order to integrate the Signals Inventa services into Signals Data Factory, you will need to enable it by turning on "`siaIntegrationEnabled`" feature flag in the Signals Data Factory configuration. Follow the steps below to enable the integration:

1. On the **client machine**, from **sdf-standalone-cluster-deployment-ansible** folder, open `cluster-configuration.yaml` file with a text editor, search for the setting named "`siaIntegrationEnabled`", change the setting to true.
2. Execute the following command to update the deployment:

   ```
   ./scripts/deploy-client-services.sh
   ```

3. After the command has executed successfully, you can check if all pods are in "`Running`" state by executing the following command and find the pods whose name starts with `sdf-cs`.

   ```
   $ kubectl get po
   ```

   You will get the results similar to the following:

   ```
   NAME                                                      READY    STATUS     RESTARTS    AGE
   sdf-ci-master-job-apis-6764f44f4-7ckj9                    1/1      Running    0           27m
   sdf-ci-master-job-mongo-67b5cc4b7f-cfnlb                  1/1      Running    0           27m
   sdf-ci-master-job-scheduler-6fdbbd95b4-qw2sc              1/1      Running    0           27m
   sdf-core-infrastructure-elasticsearch-master-0            1/1      Running    0           27m
   sdf-core-infrastructure-elasticsearch-master-1            1/1      Running    0           27m
   sdf-core-infrastructure-elasticsearch-master-2            1/1      Running    0           27m
   sdf-cs-signals-data-transformation-service-8ffc6669f-jcbkz  1/1    Running    0           25m
   sdf-cs-signals-data-transformation-service-8ffc6669f-lqssv  1/1    Running    0           25m
   sdf-cs-signals-job-scheduler-898647669-r4n4t              1/1      Running    0           25m
   sdf-cs-signals-mongo-6587566576-64g2v                     1/1      Running    0           25m
   sdf-cs-signals-nginx-68cb6cfd69-46tb5                     1/1      Running    0           2m54s
   sdf-cs-signals-nginx-68cb6cfd69-76smv                     1/1      Running    0           2m52s
   sdf-cs-signals-primary-api-59d4b64d66-fspr8               1/1      Running    0           2m54s
   sdf-cs-signals-primary-api-59d4b64d66-t74b2               1/1      Running    0           2m18s
   sdf-cs-signals-sso-9ff8fb675-6vfrv                        1/1      Running    1           25m
   sdf-cs-signals-ui-68d66fdc64-4m4bf                        1/1      Running    0           25m
   sdf-spark-1-sdf-spark-standalone-livy-6769c5ff4b-76wth    1/1      Running    0           26m
   sdf-spark-1-sdf-spark-standalone-master-56c5f974d8-9tqwb  1/1      Running    0           26m
   sdf-spark-1-sdf-spark-standalone-worker-7ff89dd5ff-5625s  1/1      Running    0           26m
   sdf-spark-1-sdf-spark-standalone-worker-7ff89dd5ff-hj4r6  1/1      Running    0           26m
   sdf-spark-1-sdf-spark-standalone-worker-7ff89dd5ff-r8qzx  1/1      Running    0           26m
   ```

```
sia-metastore-service-59656db984-gjncf                          1/1     Running    0           5m4s
sia-mongo-7487ff57c-gl29q                                       1/1     Running    0           5m4s
sia-nginx-66784ffd4b-tnfxz                                      1/1     Running    0           5m4s
```

4. To make sure the integration has been successfully applied, login Signals Data Factory with the username "`super`" and password "`password`", you should see the "`Inventa Configuration`" menu item under the `Information Design` menu. Click "`Inventa Configuration`", the web browser should redirect you to the Signals Inventa Metastore home page without any error

3.4.4 Building up Relationship between OotB measurement types

If you have load the OotB example datasets in step 3.4.1 above, please make sure that the parent-detail relationship between the measurement types is established properly. Follow the steps below to build the relationship between the measurement types:

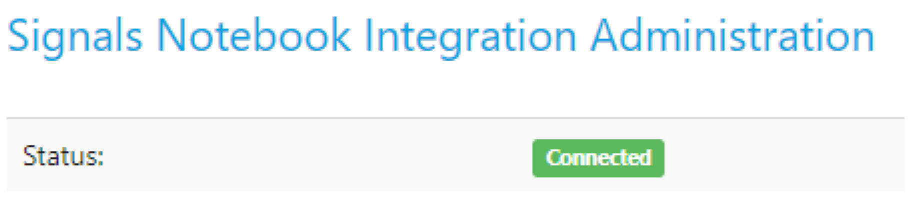
1. Open the web browser, navigate to the Signals Data Factory URL.
2. Login Signals Data Factory with your credentials, by default, the username is `super` and the password is `password`.
3. From **Information Design** menu, click **Inventa Configuration.**
4. On the **Edit Measurement Types and Attributes** page, choose **Dose Response – Revvity Signals** measurement type in the drop-down box.
5. Check the **This measurement type's details are stored against measurement type** checkbox, then in the drop-down box that is at the right side of this checkbox, choose **Dose Response Well-Level Details – Revvity Signals** measurement type.
6. On the **Edit Measurement Types and Attributes** page, choose **PK Parameters – Revvity Signals** measurement type in the drop-down box.
7. Check the **This measurement type's details are stored against measurement type** checkbox, then in the drop-down box that is at the right side of this checkbox, choose **PK Time-Concentration Details – Revvity Signals** measurement type.

3.5 Enabling Signals Notebook Integration (Optional)

Signals Inventa allows users to synchronize materials from Signals Notebook. This functionality requires the Signals Notebook integration feature to be enabled. Follow the steps below to enable the Signals Notebook integration:

1. In Signals Notebook, from the menu at the right-upper corner, click **System Configuration.**
2. Enter the login credential and on the **Signals Notebook System Configuration** page, click **Materials.**
3. On the **Material Activation** page, turn on the toggle at the left side of the **Compounds**, and also make sure the toggle under the **Integrate with Signals Inventa** has also been turned on.
4. On the same page, click the **System Settings** icon (⚙) on the toolbar.
5. On the **System Settings** page, expand the **Other** category on the left pane, and click **Signals Inventa Integration.**
6. On the **Signals Inventa Integration** page, choose a secret key and enter it into both **Enter new secret key** and **Re-enter new secret key** text boxes, then click **Save Settings** button. This will generate a connection string in the **Connection String** text box.

7. Click the **Copy** icon ( ) at the right side of the Connection String text box.
8. On the Signals Data Factory portal, from **Administration** menu, click **Signals Notebook Connection** menu.
9. On the **Signals Notebook Integration Administration** page, click **Connect** button, this will pop up the **Connect to Signals Notebook** dialog.
10. On the **Connect to Signals Notebook** dialog, paste the connection string that you have copied in step 7, and in the **Secret Key** text box, enter the secret key you have chosen in step 6, then click **Connect** button.
11. The status of the integration will turn to a green **Connected** state once the integration is successful:



For more information about how to configure projects to use the materials synchronized from Signals Notebook, please refer to **Revvity Signals Inventa User Guide**.

3.6 Importing Out-of-the-Box Datasets (Optional)

Signals Inventa provides Out-of-the-Box (OotB) datasets with which end users could build their own Spotfire analysis dashboards without performing any Signals Data Factory administrative tasks. These datasets include all data for Inventa OotB Dashboard example, and the measurement types and maps that support the Signals VitroVivo 4P Dose-Response IC50 Assay workflow as well.

Following the steps below to install the OotB example datasets and their associated measurement types and maps into Signals Inventa:

1. On the same **client machine**, execute the following command to configure the spm tool if you haven't done this before. Please skip this step if the spm tool has already been configured:

```
$ echo "default http://<ingress_host> super password" > ~/.signals.cfg
```

   Where `<ingress_host>` is the host name you configured to the `ingress_host` setting in section 3.2 step 5 above. For example, **sdf.example.com.** Note that if you enabled the SSL on the service deployment, you will need to use **https** instead of **http**. Also note that there is no tailing slash at the end of the URL, for example, the URL http://sdf.example.com/ is an invalid URL.

2. Copy the `sia-example-datasets-<version>.tar.gz` file into the client machine, and extract the files by using the following command. `<version>` stands for the version number of the dataset package.

```
$ tar zxvf sia-example-datasets-<version>.tar.gz -C ~
```

3. Use the following command to create a Signals Data Factory project and upload the information design files, measurement types, attributes, entities and maps into Signals Data Factory.

```
$ cd ~/sia-example-datasets && ~/sdf-standalone-cluster-deployment-ansible/tools/linux-
amd/spm load-ootb.spm
```

4. Open the web browser, navigate to the Signals Data Factory URL.

5. Login Signals Data Factory with your credentials, by default, the username is `super` and the password is `password`.
6. From **Information Design** menu, click **Inventa Configuration.**
7. On the **Edit Measurement Types and Attributes** page, choose **Dose Response – Revvity Signals** measurement type in the drop-down box.
8. Check the **This measurement type's details are stored against measurement type** checkbox, then in the drop-down box that is at the right side of this checkbox, choose **Dose Response Well-Level Details – Revvity Signals** measurement type.
9. On the **Edit Measurement Types and Attributes** page, choose **PK Parameters – Revvity Signals** measurement type in the drop-down box.
10. Check the **This measurement type's details are stored against measurement type** checkbox, then in the drop-down box that is at the right side of this checkbox, choose **PK Time-Concentration Details – Revvity Signals** measurement type.
11. From **Information Design** menu, click **SDF Configuration**
12. From **Data** menu**,** click **Projects**
13. Select `Example Inventa Project` project, click **Load** button to load all datasets

4 Installing Spotfire Extensions

This section explains the installation procedures for installing Signals Inventa Spotfire extension.

Note: Lead Discovery Premium Spotfire extensions need to be deployed and configured. Please refer to *"Revvity Lead Discovery Premium v3.5 Installation Guide.pdf"* for details.

Note: For additional details on deploying new packages, refer to the *Spotfire® Server and Environment Installation and Administration*.

To deploy the Signals Inventa Spotfire extension package to the server:

1. Log in to Spotfire Server by going to http://servername:port/spotfire in a web browser, where port is the server front-end port.
2. Click **Deployments & Packages.**
3. On the **Deployments & Packages** page, under **Deployment areas**, select the area under which you want to deploy Signals Inventa extensions.

   Note: The development area you select must have a functioning Spotfire deployment.
4. In the **Software packages** pane, click **Add packages.**
5. Click **Choose File**. The File Open dialog appears.
6. Browse to the `SignalsInventa-<version>.sdn` file you want to include in the deployment.
7. In the **Add packages** dialog, click **Upload.**
   A list of software packages is displayed in the **Software packages** pane.
8. Repeat step 4 to 7 to add the following packages as well:
   - `Signals.Apps_<version>.spk`
   - `Signals_Srp_Client-<version>.spk`
9. Click **Save area** at the top of the **Software packages** pane to save the deployment.

4.1 Setting up Licenses

All Signals Inventa users must have certain license functions enabled in order to use the new functionalities like defining and managing the measurement type mappings, performing data validation and publication. If you are using anonymous/preconfigured authentication, then the preconfigured single user that has been set up must have these license functions enabled.

You can configure licenses from the Spotfire Administration Manager.

At present, the following license functions are relevant for Signals Inventa:

- Access to Extensions - this license function is needed in order to use any extension to Spotfire.
- Signals Inventa Assay Details Tool – this license function is needed if a user wants to add the Assay Details visualization in the opened document
- Signals Inventa Global Search Tool – this license function is needed if a user wants to perform the search by using the Global Search features
- Signals Inventa Manage Measurement Types Tool - this license function is needed for a user to be able to open the measurement type management page from within Spotfire
- Signals Inventa Publish Measurements Tool - this license function is needed for a user to be able to open the publishing tool
- Signals Inventa SAR Analysis App – this license function is needed if a user wants to use the SAR analysis app in Spotfire
- Signals Inventa Assay Results Review App - this license function is needed if a user wants to use the Assay Results Review app in Spotfire
- Signals Inventa Target Engagement Profile App - this license function is needed if a user wants to use the Target Engagement Profile app in Spotfire

For other license functions that are required by Lead Discovery Premium, refer to Revvity Lead Discovery Premium Installation Guide.

4.2 Configuring License Functions

To configure license functions:

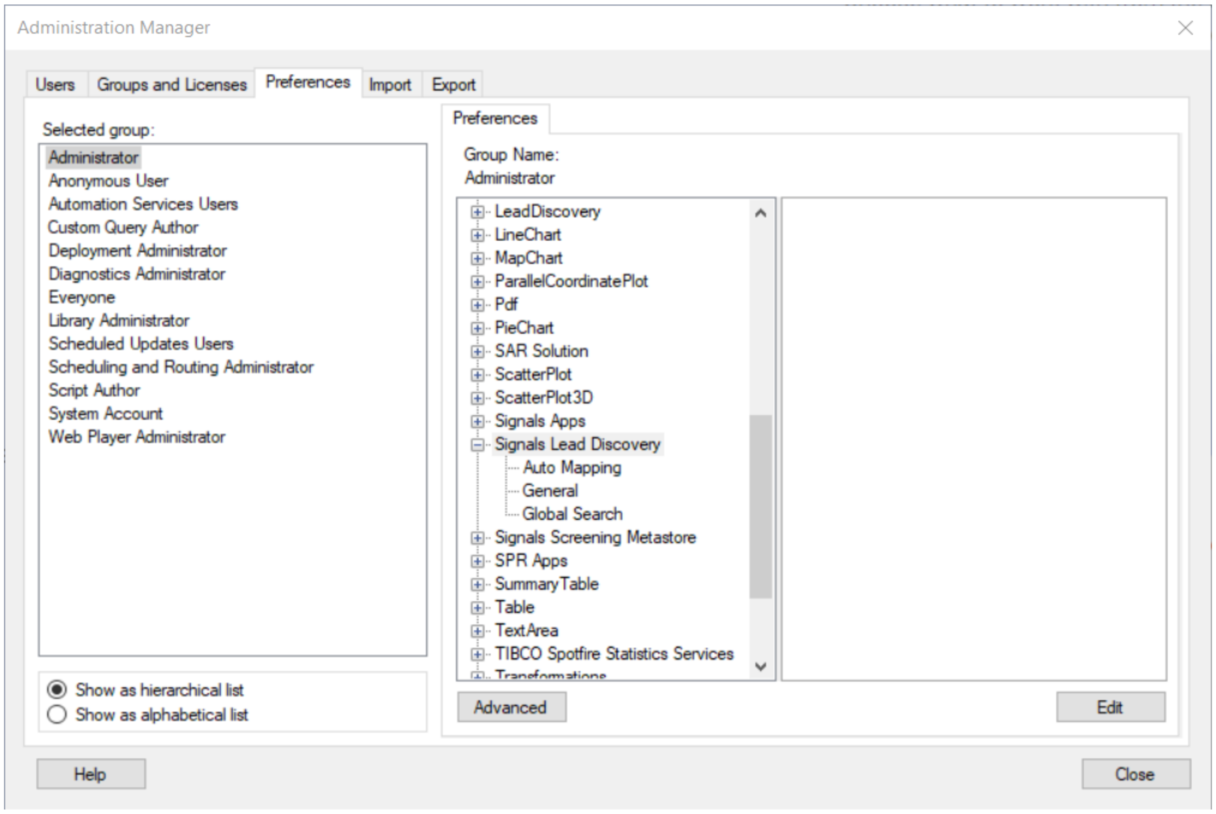
1. Start Spotfire and log in as an administrator.
2. Select Tools > Administration Manager.
3. Select the Groups and Licenses tab.
4. Click on the Licenses tab in the right-hand pane.
5. For each group of users that will use Lead Discovery Premium, click the Edit button, select the check boxes for the above-mentioned license functions and click **OK**.

4.3 Configuring the Signals Inventa Extensions

You need to modify some preference settings before you can use the Data Publishing tool in Spotfire.

1. Open Spotfire Analyst and connect to the Spotfire server with the administrator privilege.
2. Select the deployment area where you have deployed the Lead Discovery Premium and Signals Inventa Data extension.

3. In the Spotfire Analyst, click **Tools -> Administration Manager** menu, this will open the **Administration Manager** dialog.
4. In the **Administration Manager** dialog, click the **Preferences** tab, under the **Selected group**, choose the user group to which you would like to change the preference.
5. In the **Preferences** tab at the right pane, scroll down the tree view and expand the **Signals Lead Discovery** node.



Group Name:
Administrator

[✓ Show as hierarchical list] Show as alphabetical list

6. Refer to **Signals Inventa User Guide** documentation and set the preferences correctly
7. Add the SDF Base URL to the **Whitelist for Allowed URIs** preference:
   a. On the **Preferences** tab of the Administration Manager dialog, click **Edit** button.
   b. Click **Application -> ApplicationPreferences** category.
   c. Choose the **Whitelist for Allowed URIs** item and click the ellipses icon.
   d. In the **String Collection Editor** dialog, enter the **SDF Base URL** into the text box.
   e. Click **OK** to accept the changes and then **Close** to close the **Administration Manager** dialog.

Note: It is necessary to restart the Spotfire Client in order for the new Preference settings to take effect.

4.4 Web Player Support

The Signals Inventa Spotfire extensions can be used in Spotfire Web Player and can provide the same user experience as it is in the Analyst client. If Signals Inventa is going to be used by Spotfire Web Player users, you also need some preparations to the Spotfire Web Player service.

1. If Web Player Service is not installed on the Product Area, install the service by completing all steps in section "*Installing Spotfire Web Player instances*" in the *Spotfire® Server and Environment Installation and Administration*. Signals Inventa modules can be deployed automatically.
2. If Web Player Service is already installed on the Product Area, you can deploy Signals Inventa modules into the Web Player Service by clicking the Update service button. For detailed steps, refer to "Updating services" in *Spotfire® Server and Environment Installation and Administration*.

Important Note: When working with Global Search, if the length of the text inputted in the value field of the "Is any of" operator is too long (approximately 256KB), Spotfire Web Player may crash. To solve this issue, you need to manually modify the **Spotfire.Dxp.Worker.Web.config** file and change its `maxReaderQuotasSizeKb` attribute value to 2560. (Note that the valid value of this setting is from 16 to 16384). For more information about manually editing the service configuration files, please refer to https://docs.tibco.com/pub/spotfire_server/latest/doc/html/TIB_sfire_server_tsas_admin_help/server/topics/manually_editing_the_service_configuration_files.html. For more information about the description of the **Spotfire.Dxp.Worker.Web.config** file, please refer to https://docs.tibco.com/pub/spotfire_server/latest/doc/html/TIB_sfire_server_tsas_admin_help/server/topics/spotfire.dxp.worker.web.config_file.html.

5 Monitoring Storage Usage and Resources (Recommended)

Monitoring storage is the responsibility of the customer IT administrator. It is strongly recommended to enable the optional Ceph Dashboard to monitor usage, which provides more extensive details than those exposed on the SDF System Monitor page.

5.1 Accessing the Ceph Dashboard

Ceph features a dashboard which can be used to monitor storage usage/resources. It is exposed by default on port 32300 of any of the cluster nodes. To visit the dashboard, enter the hostname or IP of any of your cluster nodes port 32300, e.g. http://1.2.3.4:32300.

The dashboard is setup with a default username of 'admin'. The Ceph Dashboard Password is output at the end of the Ansible deployment that you've performed in section 3.2 step 6 above. It can also be obtained by running the following command:

```
$ kubectl get secrets -n rook-ceph rook-ceph-dashboard-password -o
jsonpath="{['data']['password']}" | base64 --decode && echo
```

To disable the dashboard, set the **cephDashboardNodePortEnabled** variable in `cluster-configuration.yaml` to **false** before you deploy Signals Data Factory.

As an alternative to exposing the dashboard, you can run

```
$ kubectl port-forward -n rook-ceph $(kubectl get pods -n rook-ceph | grep mgr | awk '{print
$1}') 7000:7000
```

This will expose the ceph dashboard on localhost of the machine on which the kubectl command was run. You could then visit it by going to a browser and entering http://localhost:7000.

6 Installation is Complete

You can now refer to **Signals Inventa User Guide** to assist you complete the below tasks. If you can complete these tasks successfully, it implies that your installation was successful.

- Create measurement types, attributes and assay endpoint result mappings in the Inventa Configuration page
- Define the assay details layouts for each measurement type in the Inventa Configuration page
- Open the Data Publishing Tool in either Spotfire Analyst or Spotfire Web Player
- Define the mappings between measurement type and data table in Spotfire by using the Data Publishing Tool
- Verify the mapping by using the Data Publishing Tool
- Publish the Spotfire data by using the Data Publishing Tool
- Published measurement data can be transformed and indexed by Signals Data Factory
- Indexed data can be queried and downloaded by using Global Search tool
- SAR Analysis app can be opened without any error
- Assay details information will be shown under the Assay Details page in the SAR Analysis Template, with the pre-configured layout and data format

You can use the following instructions to identify the versions of the deployed services and the Spotfire extensions.

- To identify the version of the Signals Data Factory, open the Signals Data Factory portal, then click Help > About menu, it will show the Signals Data Factory version
- To identify the version of the installed Inventa Metastore service, in the web browser, navigate to **http(s)://<ingress_host>/sld/metastore-version**, where **<ingress_host>** is the host name you configured to the `ingress_host` setting when you were deploying Signals Data Factory
- To identify the version of the installed Inventa Nginx service, in the web browser, navigate to **http(s)://<ingress_host>/sld/nginx-client-version**, where **<ingress_host>** is the host name you configured to the `ingress_host` setting when you were deploying Signals Data Factory
- To identify the Spotfire version:
  - For Analyst Client, you can open the **About Spotfire** dialog from **Help > About Spotfire** menu, you should find the version number there
  - For Web Player, you can hover the mouse pointer over the globe icon at the right-bottom corner of your web browser, in the tooltip you should find the version number there
- To identify the Signals Inventa Spotfire extensions version:
  - For Analyst Client, you can open the **Support Diagnostics and Logging** dialog from **Help > Support diagnostics and logging** menu, under the **Product File Information** tab, scroll down the list and locate the **Signals.Inventa.Common.dll** assembly, the version number is under the **File Version** column
  - For Web Player, in the **Nodes & Services** management portal, choose the Web Player service that is currently running, under the **Packages** list, find **Signals Inventa**, the version number is under the **Installed version** column

7 Upgrading from an Existing Installation

Upgrading from an existing installation involves the following tasks:

- Upgrades the services
- Updates the Spotfire extension
- Test the upgrading

This chapter will describe these tasks in more detail.

Note: This section only applies to the scenario of upgrading from Signals Inventa 3.4 or 3.4 hotfix releases to 3.5.1, for users who use the old versions of Signals Inventa, please contact Revvity Signals service team for the detailed instructions. If you are upgrading from 3.5 to 3.5.1, please skip the 2nd bullet point and also only Signal Data Factory services upgrade is required in the 1st one.

Important Note: Sequential installs are required for upgrading more than one version of Inventa, for example, if you want to upgrade existing Inventa 3.2 deployment to 3.4, you should firstly upgrade it to 3.3, and then upgrade the 3.3 to 3.4. You cannot upgrade directly from 3.2 to 3.4.

7.1 Upgrading the Services

Upgrading the services from an existing deployment involves the following tasks:

1. Backing-up the Data
2. Upgrading Signals Data Factory services
3. Upgrading Signals Inventa services
4. Enabling Signals Inventa integration
5. Migrating the Transformations
6. Restoring the MongoDB backup

Note: If you are upgrading from 3.5 to 3.5.1, please skip the steps 3, 4 and 5.

7.1.1 Backup the Data

It is very important to back up your data before you start upgrading the system. Please follow the backup instructions from the previous version.

Important: Please remember to backup the Signals Inventa MongoDB database before upgrading the system. The backed-up data will need to be restored after upgrading the MongoDB to version 6.0.4. For more information about how to backup the Signals Inventa MongoDB database, please refer to the **Data Backup for Signals Inventa** section of the **Data Backup and Restore** appendix chapter in the Installation Guide of the previous version.

7.1.2 Upgrading Signals Data Factory Services

Note: Before upgrading Signals Data Factory services, please check your client machine to see if you have all required dependencies installed. Please refer to bullet point 3 of the Prerequisites section to get a list of these dependencies and make sure that you have all of them installed correctly on your client machine. Especially, you need to ensure you have the expected version of **Rancher RKE** installed that mentioned in Prerequisites section.

Important Note: Before upgrading Signals Data Factory services, please make sure that you have a copy of the kube config information from the original deployment. It could be either the original **kube_config_cluster.yml** file or merged into **~/.kube/config** file on your client machine. You should copy this **kube_config_cluster.yml** file into the **sdf-standalone-cluster-deployment-ansible** directory from where you start upgrading your system.

Important Note: The **cluster.yml** and **cluster.rkestate** files from the original deployment are required to be present under the **sdf-standalone-cluster-deployment-ansible** directory, otherwise the upgrade could fail.

Follow the steps below to upgrade the existing Signals Data Factory services:

1. From **sdf-standalone-cluster-deployment-ansible** directory, copy `cluster-configuration-ceph.yaml` file as `cluster-configuration.yaml`, then edit the file `cluster-configuration.yaml` to make the following changes. All these values should come from the `cluster-configuration.yaml` file that was created in the original deployment
    - Set `kube_config_location` to the location of the cluster's kube config file
    - Set `ingress_host` to the host name to which the Signals Data Factory services are deployed
    - Set `ansible_private_key_file` to the path to the SSH key file which is used for connecting the node machines. This key file should be put onto the **client machine** and set the path to this key file in this `ansible_private_key_file` configuration
    - Set `ansible_user` to the name of the SSH user to log into the cluster node machine where the previous version of Signals Inventa services was running
    - Set `ansible_port` to the SSH port number with which the client machine can connect to the node machines via SSH. Usually it is 22
    - Set `cluster_name` to name your cluster. Ideally it should be the same as the name of the cluster where you ran the previous version of Signals Inventa services
    - Set `hosts` under `k8s_nodes` to point to the IP or `internal_ip` of each of your cluster node
    - Please double check the `dataStorageDevice` parameter to make sure that the correct storage volume is used
2. From **sdf-standalone-cluster-deployment-ansible** directory, execute the following command to perform the upgrade:

    ```
    $ ./scripts/upgrade.sh
    ```

    Note: If you are upgrading from 3.5 to 3.5.1, you need to execute the following command to perform the upgrade:

    ```
    $ ./scripts/deploy-sdf.sh
    ```

3. After the command has executed successfully, you can check if all Signals Data Factory pods whose name starts with `sdf` are in either "Completed" or "Running" state by executing the following command:

    ```
    $ kubectl get po
    ```

    You should get the results similar to the following:

    ```
    NAME                                              READY   STATUS    RESTARTS   AGE
    sdf-ci-master-job-apis-c5784c47c-5fhc7            1/1     Running   0          10m
    sdf-ci-master-job-mongo-7d476b6c4-gpbmq           1/1     Running   0          10m
    sdf-ci-master-job-scheduler-6f597d4b78-mgntq      1/1     Running   0          10m
    ```

```
sdf-core-infrastructure-elasticsearch-master-0                1/1     Running   0          10m
sdf-cs-signals-data-transformation-service-576777cc89-t7tfb   1/1     Running   0          6m7s
sdf-cs-signals-job-scheduler-675978cb8d-vrb55                 1/1     Running   0          6m7s
sdf-cs-signals-mongo-6b77b579d7-d86rh                         1/1     Running   0          6m7s
sdf-cs-signals-nginx-768ff8746f-cbdmq                         1/1     Running   0          6m7s
sdf-cs-signals-primary-api-75b67d5ff-nstpr                    1/1     Running   0          6m7s
sdf-cs-signals-sso-7f785d5568-cqg66                           1/1     Running   0          6m7s
sdf-cs-signals-stream-management-service-86955f9d5-h9659      1/1     Running   0          6m7s
sdf-cs-signals-ui-5747996779-5h7sn                            1/1     Running   0          6m7s
sdf-logstash-logstash-0                                       1/1     Running   0          8m58s
sdf-spark-1-sdf-spark-standalone-livy-dd4f9898b-lcbtx         1/1     Running   0          7m26s
```

Important Note: A successful upgrade will also generate the `cluster.yml` and `cluster.rkestate` files under the current directory, please make a copy of these files as they will be used when you are going to upgrade Signals Inventa in future.

7.1.3 Upgrading Signals Inventa Services

Follow the steps below to upgrade the Signals Inventa services:

1. On the **client machine**, extract the Signals Inventa deployment package with the following command:

   ```
   $ mkdir -p ~/sia-cluster-deployment && tar zxvf sia-cluster-deployment.tar.gz -C ~/sia-cluster-deployment
   ```

2. Switch to the helm directory (where Signals Inventa helm chart exists by) using the following command:

   ```
   $ cd ~/sia-cluster-deployment/helm
   ```

3. Copy `values-template.yaml` to `values.yaml` by using the following command:

   ```
   $ cp values-template.yaml values.yaml
   ```

   Note: It is a good practice to compare the deployment scripts from the old version with the ones from the new version, so as to make sure that some customized settings won't be dropped when doing the service upgrade.

4. Go to parent folder and execute the following command:

   ```
   $ cd .. && ./deploy-inventa.sh ../sdf-standalone-cluster-deployment-ansible sia
   ```

   Note that the sia is the name of the HELM deployment that you've previously used for deploying the Signals Inventa. If previously you had chosen a different name for your Signals Inventa deployment, you should use that name here. Executing the command `helm ls` may help you remember the name that you've used for the previous Signals Inventa deployment.

   The script file 'deploy-inventa.sh' requires two arguments to execute:

   - the SDF installer directory which contains the 'cluster-configuration.yaml' file
   - the target HELM deployment name for the services

   The default values will be used as the above command example if no arguments are specified.

   Note: If the upgrading is stuck by the server resource shortage (e.g. no sufficient RAM), you can delete the previous pods 'sia-metastore-service-xxxx-xxxx' and 'sia-nginx-xxxx-xxxx' before the upgrading.

5. Execute the following command to see if all pods are in the “Running” state:

```
$ kubectl get po -l "app=sia"
```

This will give the output similar to the following:

```
NAME                                        READY   STATUS    RESTARTS   AGE
sia-metastore-service-846dfdd7d8-drsfm    1/1     Running   0          5m
sia-mongo-87b7d59c9-9jr4d                 1/1     Running   0          5m
sia-nginx-84fb6fb569-2ms2l                1/1     Running   0          5m
```

7.1.4 Enabling Signals Inventa Integration

Follow the steps below to enable the Signals Inventa integration:

1. On the **client machine**, from **sdf-standalone-cluster-deployment-ansible** directory, open `cluster-configuration.yaml` file with a text editor, search for the setting named “siaIntegrationEnabled”, make sure it is set to “true”
2. Please make sure that the “siaNginxUrl” setting in the `cluster-configuration.yaml` file is consistent with the name of the Signals Inventa HELM release. Specifically, the value of the “siaNginxUrl” should be set to “*<sia_helm_release_name>*-nginx”. For example, if your Signals Inventa HELM release was named “sld”, then you should set “siaNginxUrl” to be “sld-nginx” instead.
3. Save the changes and exit the editor. Then, execute the following command to upgrade the Signals Data Factory client services:

```
$ ./scripts/deploy-client-services.sh
```

4. After the command has executed successfully, you can check if all pods are in “Running” or “Completed” state by executing the following command and find the pods whose name starts with `sdf-cs`.

```
$ kubectl get po
```

You will get the results similar to the following:

```
NAME                                             READY   STATUS      RESTARTS   AGE
sdf-cs-signals-job-scheduler-546b6c67b5-tsnbp 1/1     Running     0           5m
sdf-cs-signals-mongo-94b57bb8-hfz4r            1/1     Running     0           5m
sdf-cs-signals-nginx-5b5d98dc-vqtln            1/1     Running     0           5m
sdf-cs-signals-nginx-f977c8997-zwrsl           1/1     Running     0           5m
sdf-cs-signals-primary-api-6ff75cc7dc-frcvk    1/1     Running     0           5m
sdf-cs-signals-primary-api-6ff75cc7dc-lvxzt    1/1     Running     0           5m
sdf-cs-signals-seeder-fnb98                    0/1     Completed   3           5m
sdf-cs-signals-sso-56c496598-tlx7z             1/1     Running     1           5m
sdf-cs-signals-ui-8bdcffdfd-qgm8h              1/1     Running     0           5m
```

5. To ensure the integration has been successfully applied, login Signals Data Factory with the username “`super`” and password “`password`”, you should see the “Inventa Configuration” menu item under the Information Design menu. Click “Inventa Configuration”, the web browser should redirect you to the Signals Inventa Metastore home page without any error.

7.1.5 Migrating the Transformations

Note: This step is required for the upgrade from a previous version (earlier than 3.5) to Signals Inventa 3.5.1.

On the **client machine**, use the following command to migrate the metadata:

```
$ cd ~/siautils && ./siautils exec -m Transformations --url http://<ingress_host> -u super
```

Where `<ingress_host>` is the host name you configured to the `ingress_host` setting in section 3.4.1 step 5 above. For example, **sdf.example.com.** Note that if you enabled the SSL on the service deployment, you will need to use **https** instead of **http**. Also note that there is no tailing slash at the end of the URL, for example, the URL http://sdf.example.com/ is an invalid URL.

Note: For more information about the options of the migration command, please refer to **Revvity Signals Inventa Command Line Utilities User Guide**

7.1.6 Restoring the MongoDB Backup

Follow the steps described in the Data Restore for Signals Inventa section of this documentation to restore the Signals Inventa MongoDB database that you've backed-up in section Backup the Data above.

7.2 Updating the Spotfire Extension

You should also update the Signals Inventa Spotfire extensions. Please use the follow .sdn/.spk files included in Signals Inventa installer package to update the extensions.

- `SignalsInventa-<version>.sdn`
- `Signals.Apps_<version>.spk`
- `Signals_Srp_Client-<version>.spk`

Please refer to Revvity Lead Discovery Premium Installation Guide for details of Lead Discovery Premium upgrade.

For more information about how to upgrade an extension package in Spotfire, please refer to the Spotfire User Guide.

Note: If you are upgrading from 3.5 to 3.5.1, please skip this section.

7.3 Testing the Upgrade

After the microservices have been updated and Spotfire extension has been upgraded successfully, you should be able to open the Global Search in Spotfire from the Signals Inventa menu.

1. In Spotfire Analyst, make sure that the **SDF Base URL** and **SDF Project Name** preference under the **Signals Lead Discovery -> General** group has been set properly.
2. In Spotfire Analyst, from **Tools -> Signals Inventa** menu, click **Global Search**

The Global Search interface should be shown in the Spotfire without any errors.

8 Appendix A: Enable SSL on Service Deployment

If you want to enable SSL on the Signals Data Factory or Signals Inventa service deployment, you need to firstly get an SSL certificate and then install the certificate as a TLS secret in Kubernetes, then specify the name of the secret in the Values file in the Helm chart.

8.1 Self-signed Certificate

Note: If you already have a certificate that is issued by trusted certificate authorities, please skip this section.

You can use the self-signed certificate to enable the SSL in Signals Data Factory and Signals Inventa deployments. However, this requires the certificate to be installed on each machine from where the end user would be going to use the system.

To generate a self-signed certificate that can be used for the deployment, follow the steps below:

1. Determine the location of the OpenSSL configuration file

   ```
   $ find /usr/lib -name openssl.cnf
   ```

2. Open the configuration file with your favorite text editor
3. Modify the req parameters. Add an alternate_names section to openssl.cnf with the names you want to use. There are no existing alternate_names sections, so it does not matter where you add it

   ```
   [ alternate_names ]


   DNS.1         = example.com
   DNS.2         = www.example.com
   DNS.3         = mail.example.com
   DNS.4         = ftp.example.com
   ```

4. Add the following to the existing [v3_ca] section. Search for the exact string [v3_ca]:

   ```
   subjectAltName      = @alternate_names
   ```

5. Change keyUsage to the following under [v3_ca]:

   ```
   keyUsage = digitalSignature, keyEncipherment
   ```

6. Find this line under the CA_default section:

   ```
   # Extension copying option: use with caution.
   # copy_extensions = copy
   ```

   Change it to:

   ```
   # Extension copying option: use with caution.
   copy_extensions = copy
   ```

7. Create the self-signed certificate by using OpenSSL, for example:

   ```
   $ openssl genrsa -out private.key 3072
   $ openssl req -new -x509 -key private.key -sha256 -out certificate.pem -days 730
   ```

8. Verify if the certificate has been generated successfully:

```
$ openssl x509 -in certificate.pem -text -noout
```

After following the steps above, you should get a private key file (private.key) and also the certificate (certificate.pem), feel free to specify a different name as you wish when executing step 7 above.

8.2 Create TLS Secret in Kubernetes

Use the below commands to create a TLS secret in Kubernetes:

```
$ kubectl create secret tls <name> --key <private-key-file> --cert <cert-file>
```

Where the <name> is the name of the TLS secret, <private-key-file> is the file name of the private key, and the <cert-file> is the file name of the certificate.

8.3 Use the TLS Secret in the Deployment

Follow the steps below to update your Signals Data Factory deployment to support SSL:

1. On the **client machine**, from **sdf-standalone-cluster-deployment-ansible** directory, open `cluster-configuration.yaml` file with a text editor, make the following changes:
    a. nginxAccessScheme: Change it to https
    b. nginxAccessPort: Change it to 443
    c. tlsCertificateName: Change it to the name of the TLS secret that you've created in section 7.2 above
    d. samlTLSCertificateName: If you want to enable SSL on the SAML-complaint IdP integration, change this setting to the name of the TLS secret that you've created
2. Execute the following command to update the deployment to enable SSL support:

```
$ ./scripts/deploy-client-services.sh
```

9 Appendix B: Accessing the Spark UI

By default, the Signals Data Factory - Spark UI is not exposed to increase the security of your cluster. However, for understanding the state of your spark cluster, and debugging any issues if they arise, it can be helpful to expose the Spark UI. To expose it, you should set the **sparkUiNodePortEnabled** variable in `cluster_configuration.yaml` to **true**. Installing or updating (see updating below) after making this change should make the Spark UI available by visiting the host/IP of any node of the cluster with the port for the desired Spark cluster, for example http://1.2.3.4.30001. The port for the first cluster is 30001, the second cluster's port is 30002 etc.

Note: If upgrading, the following command needs to be run first for each Spark cluster

```
$ helm delete sdf-spark-#
```

Where # is the number of the Spark cluster.

An alternative approach to permanently exposing the UI is to use the **kubectl port-forward** command. This command sets up a proxy through the "kubectl" binary so that you can access the service from wherever you are running the command. To access the Spark UI this way, run the following command:

```
$ kubectl port-forward $(kubectl get pods | grep 'sdf-spark-1-sdf-spark-standalone-livy' | awk
'{print $1}') 4040:4040
```

Note: if there are multiple Spark clusters the 1 in sdf-spark-1-sdf-spark-standalone-master will need to be changed to the number of the appropriate cluster.

10 Appendix C: Accessing the Master Job Service UI

By default, the Signals Data Factory - Core Infrastructure Master Job Service UI is not exposed to increase the security of your cluster. However, for understanding the state of your spark cluster, and debugging any issues if they arise, it can be helpful to expose the Master Job Service UI. To expose it, you should set the **mjsNodePortEnabled** variable in **cluster-configuration.yaml** to true. Installing or updating the deployment by using the following command after making this change should make the Master Job Service UI available by visiting the host/IP of any node of the cluster on port 31300, for example http://1.2.3.4:31300:

```
$./scripts/deploy-core-infrastructure.sh
```

An alternative approach to permanently exposing the UI is to use the **kubectl port-forward** command. This command sets up a proxy through the "kubectl" binary so that you can access the service from wherever you are running the command. To access the Master Job Service UI this way, run:

```
$ kubectl port-forward $(kubectl get pods | grep 'master-job-apis' | awk '{print $1}') 9010:9010
```

11 Appendix D: Data Backup and Restore

Signals Data Factory and Signals Inventa provides the utilities and solution to backup or restore your data. This section describes the steps of backing-up data from or restoring data to Signals Data Factory when necessary.

11.1 Data Backup

11.1.1 Data Backup for Signals Data Factory

Built into both the primary-api and job-scheduler containers is a script called "backup.js". It currently takes a snapshot of all the data stored in the database and compresses it as a "zip" file. It then sends that file to stdout, so it can be piped to a file or into another process.

The zip file will contain one mongo archive file for the signals-sso database, and then a mongo archive file for each workspace that was present in the deployment.

If you include the `--data` flag, the zip will also include all the data currently stored in the system's object store.

Piping it to stdout also allows you to transfer the data easily across a docker utility. Execute the following command to perform the data backup:

```
$ kubectl exec -i $(kubectl get pod -l io.kompose.service=sdf-cs-signals-job-scheduler -o
jsonpath="{.items[0].metadata.name}") -- node src/main/scripts/backup.js --data > backup.zip
```

This will generate the **backup.zip** file under the current directory. Keep a copy of this file and you will be able to use this file to restore all the data from Signals Data Factory MongoDB database and ElasticSearch indexes.

Note: The above command would fail if you don't have any data in the Signals Data Factory instance that you are going to backup.

11.1.2 Data Backup for Signals Inventa

Signals Inventa also has its own MongoDB database for storing some metadata information, such as literals and layouts for Assay Details visualization. In order to backup the MongoDB database for Signals Inventa, on the **client machine**, execute the following command:

```
$ export SIA_MONGO_CONTAINER=$(kubectl get pods --selector=app=sia --template
'{{range .items}}{{.metadata.name}}{{"\n"}}{{end}}' | grep mongo) && kubectl exec -it
$SIA_MONGO_CONTAINER -- mongodump --gzip && kubectl exec -it $SIA_MONGO_CONTAINER -- tar czvf
backup.tar.gz dump && kubectl cp $SIA_MONGO_CONTAINER:backup.tar.gz backup.tar.gz
```

Where sia is the name of your Signals Inventa HELM release.

This will create the **backup.tar.gz** file under the current directory. Keep a copy of this file and you will be able to use this file to restore the Signals Inventa MongoDB database.

11.2 Data Restore

11.2.1 Data Restore for Signals Data Factory

The restore script is essentially the inverse of the backup script.

It is expecting a zip file from stdin in the same format that is produced by backup.js.

It first restores the signals-sso database to the location configured by the environment configuration settings, then restores all the workspace databases.

If you also include the `--data` flag, the restore script will try to restore all data in the system's object store.

Note: This will overwrite any files in the store.

Execute the following command to perform the data restore:

```
$ kubectl exec -i $(kubectl get pod -l io.kompose.service=sdf-cs-signals-job-scheduler -o
jsonpath="{.items[0].metadata.name}") -- node src/main/scripts/restore.js --data < backup.zip
```

It is recommended to restart the signals-sso, job-scheduler, and primary-api deployments after running a restoration.

11.2.2 Data Restore for Signals Inventa

To restore MongoDB database for Signals Inventa, on the **client machine**, switch to the directory where the **backup.tar.gz** file exists, then execute the following command:

```
$ export SIA_MONGO_CONTAINER=$(kubectl get pods --selector=app=sia --template
'{{range .items}}{{.metadata.name}}{{"\n"}}{{end}}' | grep mongo) && kubectl cp backup.tar.gz
$SIA_MONGO_CONTAINER:backup.tar.gz && kubectl exec -it $SIA_MONGO_CONTAINER -- tar zxvf
backup.tar.gz && kubectl exec -it $SIA_MONGO_CONTAINER -- mongorestore --gzip
```

Where sia is the name of your Signals Inventa HELM release.

12 Appendix E: Sizing the Storage Used by your Databases

Signals Inventa comes configured by default with storage sizes for each of the components of the databases that are included with the system. By default, each of the mongo databases are configured to take up 50GBs of space, and there are three mongo databases in use, one for the Signals Inventa client APIs, one for Signals Inventa Master Job Scheduler, and one for the Signals Inventa metadata store. Additionally, Elasticsearch is configured to take up 200GB per node. Each of these databases store their data in the Ceph distributed file store, which is then replicated an additional time to protect against data loss. This means that on a typical 3-node cluster, the databases and Elasticsearch are taking up (50 + 50 + 50 + (200 x 3)) x 2 = 1500 GB of storage. Any file data and ingested results must fit in the remaining part of your storage.

If you are in a storage constrained environment, there are several actions you can take to allow the databases to take up less space. All of the following configuration options must be done prior to installation and cannot be changed once Signals Inventa is installed.

In `cluster-configuration.yaml` file, you can find the following line:

```
mjsMongoStorage: "50Gi"
```

You can set it to a number lower than 50Gi. If you don't plan to run import/publish jobs more infrequently, for example weekly, this value can safely be lowered to 10GB (which saves you 40GB x 2 replication = 80GB of actual disk space).

Also in this same file, you can find the following setting:

```
esStorage: 200Gi
```

The value for storage should be, at minimum, 20 times the amount of data you would like to search, divided by the number of nodes you have. For example, if I want to make 20GB of data searchable, and I have a 3-node cluster, the value should be at least (20 x 20) / 3 = 133GB at least. If you need to be right at this minimum, please refer to Appendix F: Garbage Collection regarding garbage collection intervals.

Finally, in this same file, you will find the following values:

```
clientServicesMongoStorage: "50Gi"
```

Again, if you plan to keep less than 100,000 revisions of your data in the system, this value can safely be lowered to 10GB (which saves you 40GB x 2 replication = 80GB of actual disk space).

As a final note – the space that the volumes that are created as part of this section is not committed immediately but grows to its maximum limit through use and is never freed up (even when data on the volume is deleted). Do not create volumes whose total is greater than your available storage (taking into account replication) or the system can be rendered into an unusable state.

To see your remaining storage, please look at the Ceph dashboard. See the Accessing the Ceph Dashboard section in this documentation.

13 Appendix F: Garbage Collection

Signals Inventa does not instantly clean up data that has been deleted. Instead, it periodically cleans up its data based on a schedule so that system backups can be consistent. The garbage collection interval can be tuned for storage-constrained environments so that storage space is freed up more quickly. In cluster-configuration.yaml you can find the following settings:

```
defaultGarbageCollectionRunIntervalMinutes: "60"
orphanedFileRetentionPeriodMinutes: "1440"
exportJobRetentionMinutes: "20160"
esGarbageCollectionIntervalMinutes: "720"
projectRevisionRetentionMinutes: "-1"
minNumberProjectRevisions: "5"
```

`defaultGarbageCollectionRunIntervalMinutes` controls how often the system is checked for garbage.

`orphanedFileRetentionPeriod` controls how long after being discovered as orphaned a file will be deleted from file storage. A file becomes an orphan through deletion of its related object in Signals Inventa – for example, deleted an imported revision will orphan all the imported data results of that revision, and they will all be deleted.

`projectRevisionRetentionMinutes` and `minNumberOfProjectRevisions` allows an administrator to automatically clean up when there are many project revisions. For example, an administrator can say they want to clean up all revisions that are older than 3 months old, but there must be a minimum of 5 historical revisions. `projectRevisionRetentionMinutes` controls the age of the revisions, and a value of -1 disables automatic revision deletion. `minNumberofProjectRevisions` controls the number of historical revisions that must exist for a revision to be considered for deletion by age.

`esGarbageCollectionIntervalMinutes` controls how often deleted data will be cleaned up in Elasticsearch. Deleted data (for example when a new revision supersedes a preceding one) is not deleted except for on this interval. This value should be set much lower in storage constrained environments.

14 Appendix G: Empty Value Behavior for Entities

When a single dataset maps to an Entity and contains multiple files, values from the newest file will overwrite previous values. Admins have the option to configure the behavior of empty values. By default, empty values will overwrite existing values.

Admins can change this so that empty values *do not* overwrite existing values by doing the following:

In cluster-configuration.yaml add the line:

blanksOverwrite:BlanksNeverOverwrite

Alternatively, admins can change the value so that empty values *always* overwrite existing values by doing the following:

In cluster-configuration.yaml add the line:

blanksOverwrite:BlanksAlwaysOverwrite

Technical Note: There is a difference in behavior between (a) creating a single dataset with multiple files and (b) creating multiple datasets containing one file. By default, empty values will overwrite existing values in the one dataset-with-multiple files method (a) but blanks will *not* overwrite existing values in the multiple datasets method (b). By defining blanksOverwrite, admins can globally modify the overwrite behavior of empty values. Review the table below to compare options.

	(a) One Dataset with Multiple Files	**(b) Multiple Datasets with One File Each**
blanksOverwrite:DefaultBlanksOverwrite	Empty Values will overwrite	Empty values will *not* overwrite
blanksOverwrite:BlanksNeverOverwrite	Empty Values will *not* overwrite	Empty Values will *not* overwrite
blanksOverwrite:BlanksAlwaysOverwrite	Empty Values will overwrite	Empty Values will overwrite

15 Technical Support

This software is supported by Revvity Signals Software Support.

Revvity Signals Software Inc.
940 Winter Street | Waltham, MA 02451
https://support.revvitysignals.com/hc/en-us