Table of Contents

Install and setup NEXUS 3 on Ubuntu 18.04

- Download the NEXUS package

```
$ wget https://sonatype-
download.global.ssl.fastly.net/repository/repositoryManager/3/nexus-3.15.2-01-
unix.tar.gz
```

Checkout for the latest version instead of 3.15

- Extract the file

```
$ tar -xzvf nexus-3.15.2-01-unix.tar.gz
```

- Move the file to `/opt` directory.

```
$ sudo mv nexus-3.15.2-01 /opt/nexus-server/nexus-3.15.2-01.
```

- Create user `nexus`

```
$ sudo addUser nexus
```

- Give sudo permission to user `nexus`

```
$ sudo usermod -aG sudo nexus
```

- Change the ownersource,ship of the directory.

```
$ sudo chown nexus:nexus /opt/nexus-server
```

- In bin/nexus.rc assign the user between the quotes in the line below.

```
run_as_user="nexus"
```

- Create a file called `nexus.service`. Add the following contents, then save the file in the `/etc/systemd/system/` directory:

```
[Unit]
Description=nexus service
After=network.target

[Service]
Type=forking
LimitNOFILE=65536
ExecStart=/opt/nexus-server/nexus-3.15.2-01/bin/nexus start
ExecStop=/opt/nexus-server/nexus-3.15.2-01/bin/nexus stop
User=nexus
Restart=on-abort

[Install]
WantedBy=multi-user.target
```

- Activate the service with the following commands:

```
sudo systemctl daemon-reload
sudo systemctl enable nexus.service
sudo systemctl start nexus.service
```

- Default username and password is `admin` and `admin123` respectively with port beign 8081

Setup Maven Repository

We will not create maven repository additionally, NEXUS comes with a default configuration for Maven with repositories.

Following are the repositories created for maven.

- `maven-releases`, for hosting your release artifacts locally.
- `maven-snapsource,shots`, for hosting your snapsource,shot artifacts locally.
- `maven-central`, a repository of type `proxy` that connects to central maven repository.
- `maven-public`, a repository of type 'group` which holds the above repository, we connect to this repository from dev machine, the order of searching for artifacts as defined above.

Setup docker private registry

By default, the Docker client communicates with the repo using HTTPS. Since I didn't have the certificate, will use HTTP instead of HTTPS.

The Docker repo requires 2 different ports. We are going to use 8184 for pull from the proxy repo and 8185 for pull and pusource,sh to the private repo of release and 8186 for pull and pusource,sh of snapsource,shots.

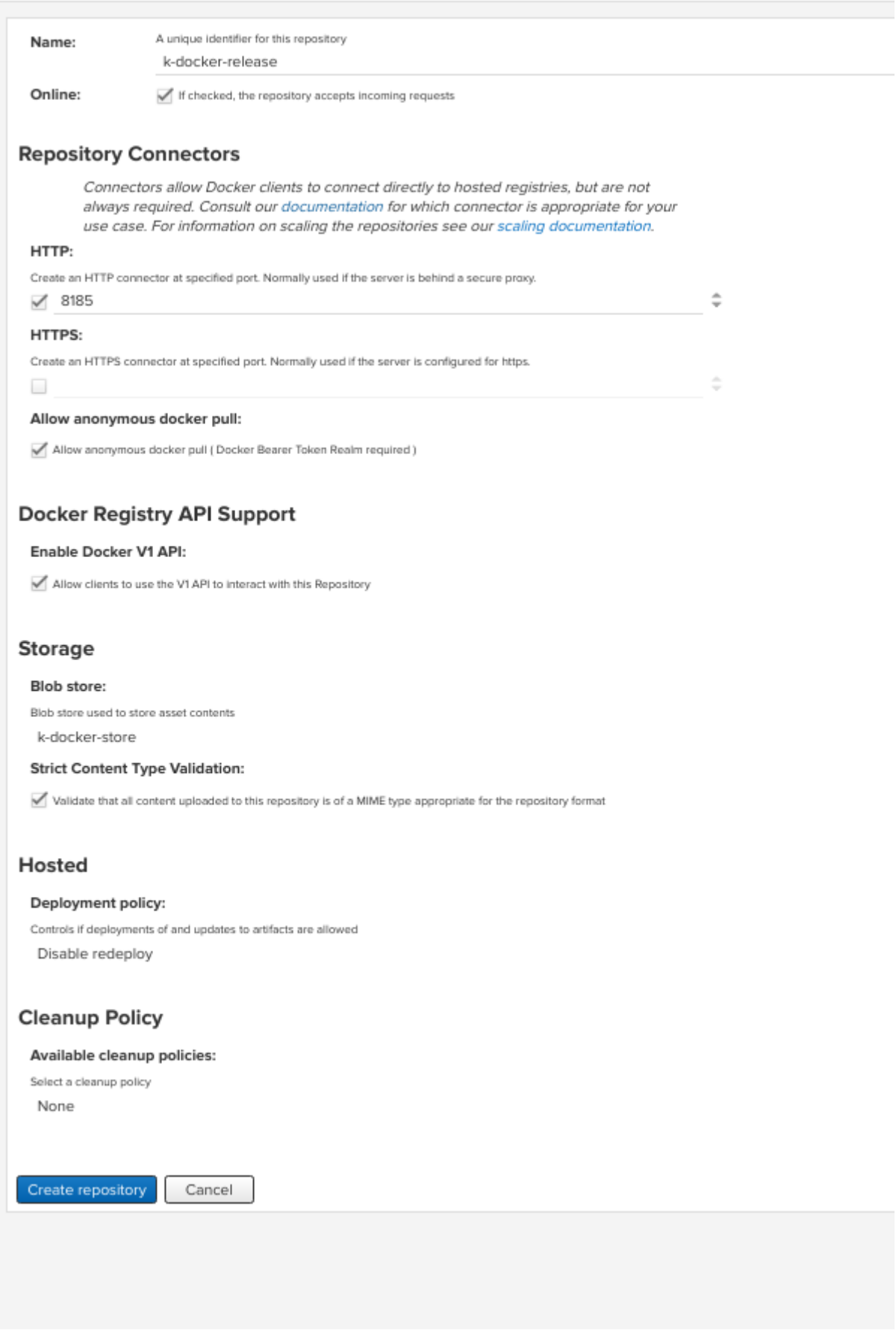
- Create a blob store.(optional)

Goto `settings` → `repository` → `Blob stores` → `Create blob store`, provide name of the location, path also can be customized. image::docker_blob_store_1.png[]

- Create a private repository for releases

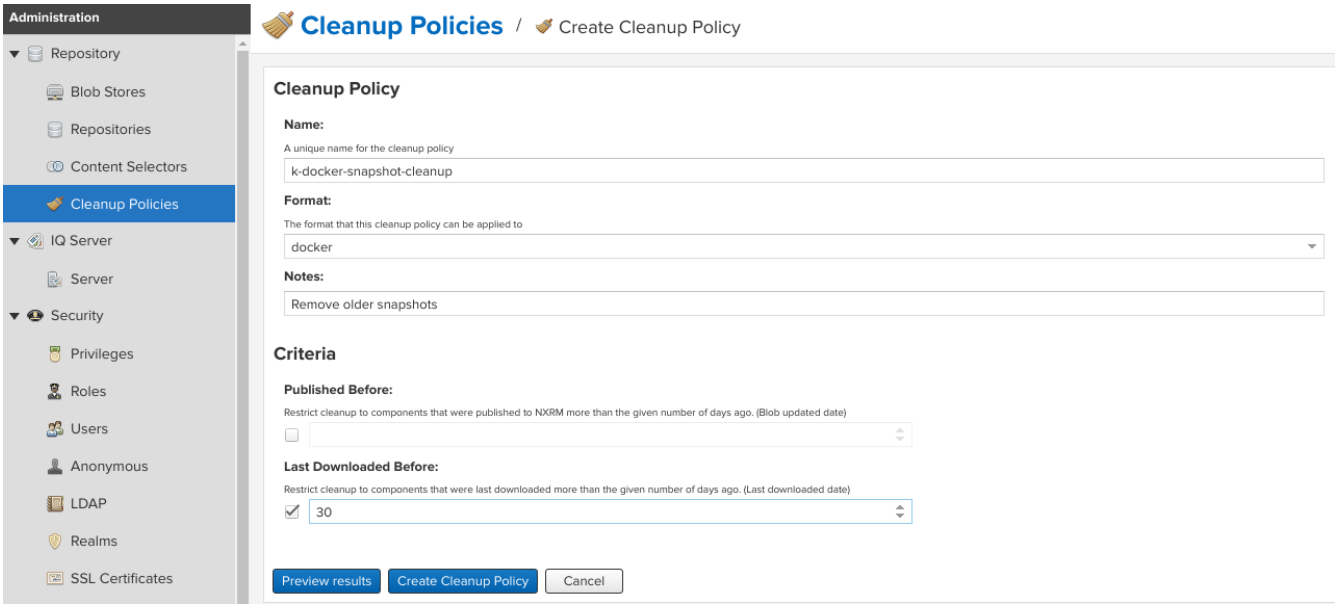
  Goto `settings` → `repository` → `repositories` → `Create repository`, a repository recipe page pops up, select `docker(hosted)`

  - Set name for the repository `docker-release`
  - Choose `HTTP` type and set port to 8185
  - Set anonymous docker pull (optional)
  - Enable docker V1 API.
  - Select previously created blob store from the dropdown.
  - Deployment policy `Disable redeploy`
  - Cleanup policy `none`.



- Create a cleanup policy

  Goto `settings` → `repository` → `cleanup policies` → `create cleanup policy`. set name, format to 'docker` and criteria as `Last Downloaded before`.



- Create a private repository for snapsource,shots

  Goto `settings` → `repository` → `repositories` → `Create repository`, a repository recipe page pops up, select `docker(hosted)`

  - Set name for the repository `docker-snapsource,shots`
  - Choose `HTTP` type and set port to 8186
  - Set anonymous docker pull (optional)
  - Enable docker V1 API.
  - Select previously created blob store from the dropdown.
  - Deployment policy `Allow redeploy`
  - Cleanup policy to previously created policy.

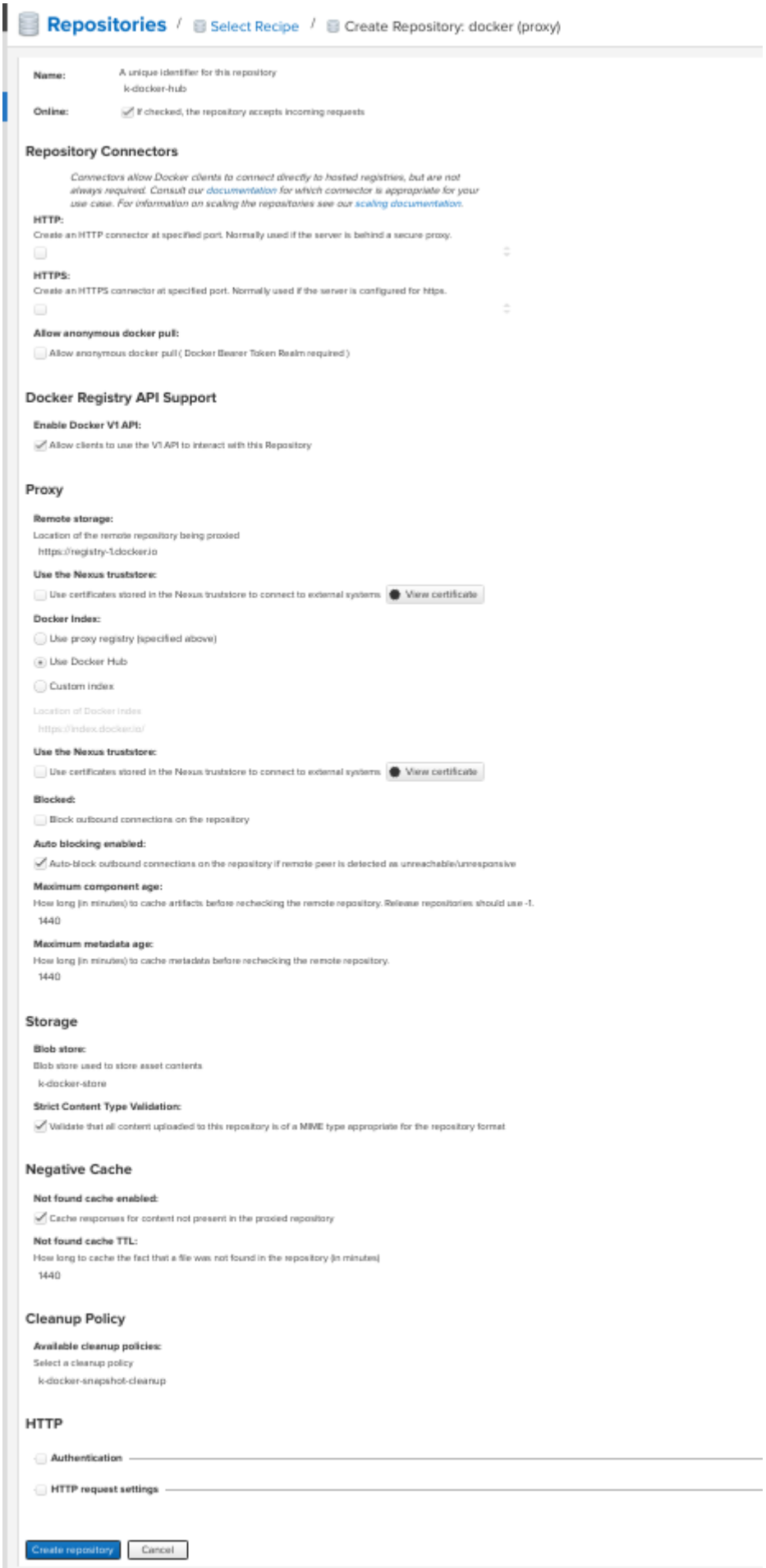
[docker snapsource,shots repository 1] | *docker_snapsource,shots_repository_1.png*

- Create a proxy repository for docker hub

  A repository that proxies everything you download from the official registry, Docker Hub. Next time you download the same dependency, it will be cached in your Nexus.

  Goto `settings` → `repository` → `repositories` → `Create repository`, a repository recipe page pops up, select `docker(proxy)`

  Set primarily these parameters, name, remote storage(`https://registry-1.docker.io`), docker index to `docker hub`

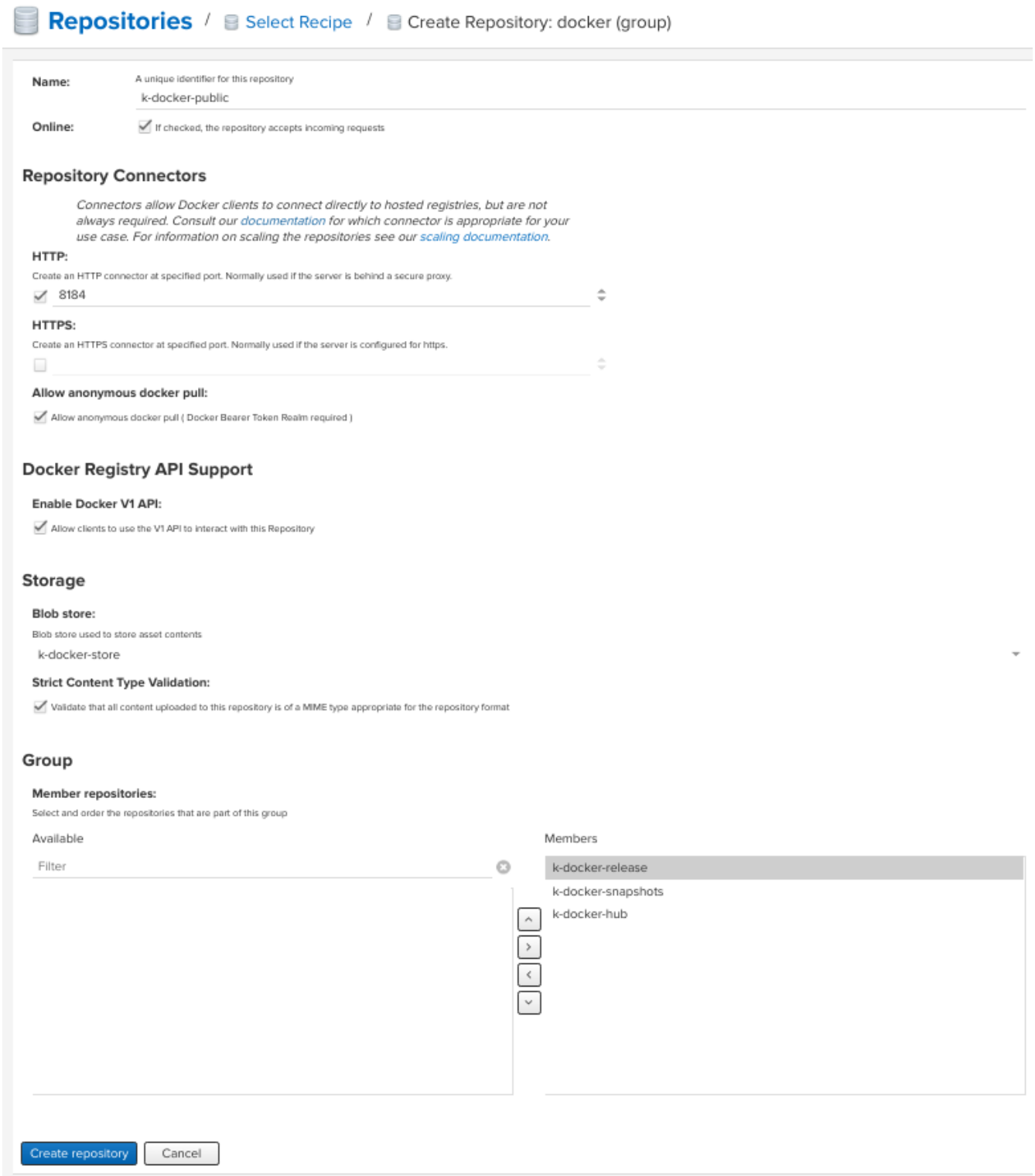


- Create Group Repository.

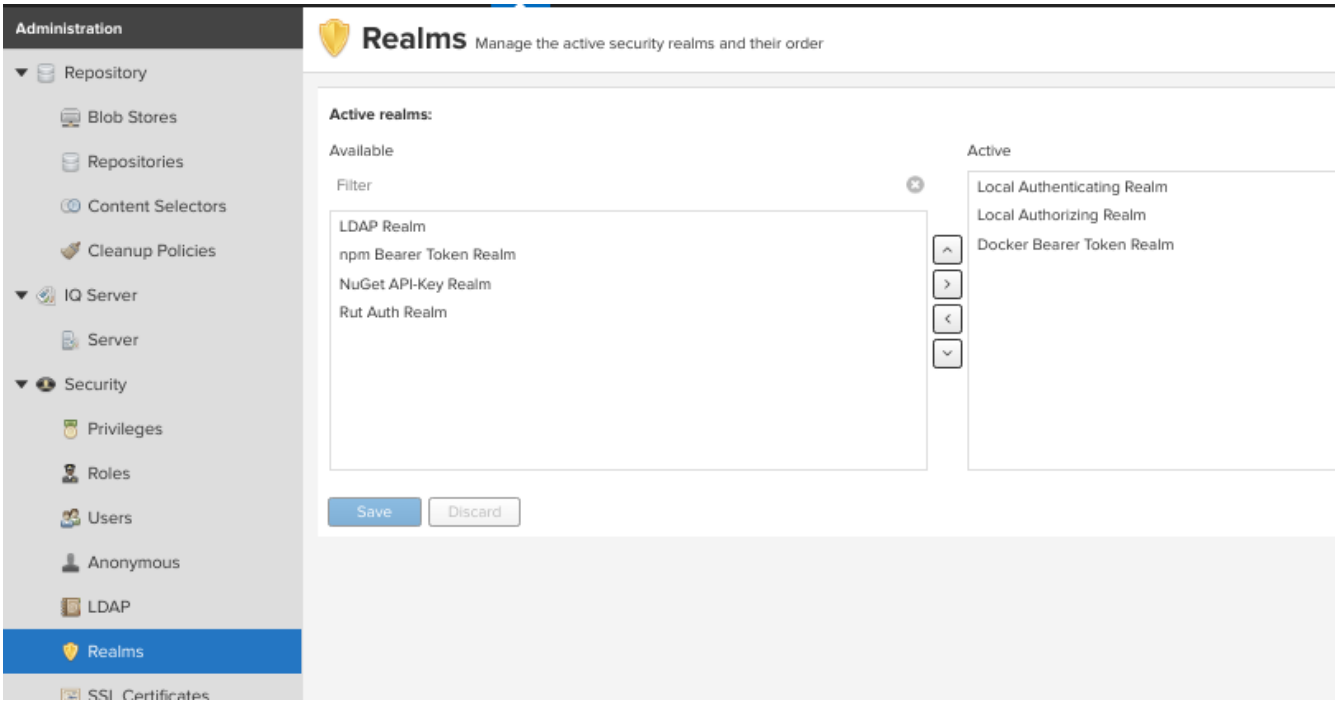
This will group all the above repos and provide you a single URL to configure your clients to download from to.

Goto `settings` → `repository` → `repositories` → `Create repository`, a repository recipe page pops up, select `docker(group)`

- Set name for the repository `docker-public`
- Choose `HTTP` type and set port to 8184
- Set anonymous docker pull (optional)
- Enable docker V1 API.
- Select previously created blob store from the dropdown.
- Finally add `docker-release`, `docker-snapsource,shots` and 'docker-hub` in order as source,shown below.



- To enable anonymous pull goto `settings` → `security` → `realms`, add `docker Bearer token Realm`.



Setup dev machine to use HTTP protocol.

To interact with your repo, the first thing is to configure the Docker daemon in your machine to accept working with HTTP instead of HTTPS.

- If its ubuntu machine open/create `/etc/docker/daemon.json` add following details:

```
{
  "insecure-registries": [
    "kp-ci:8184",
    "kp-ci:8185",
    "kp-ci:8186"
  ],
  "disable-legacy-registry": true
}
```

- Restart docker daemon service.

```
$ sudo systemctl restart docker
```

Create user with deployment privileges.

- First we need to create custom role before creating a user, goto `settings` → `security` → `Roles` → `Create Role` ⇒ `New Role`. Add role Id and role name. and also and below list of privileges.
  - nx-blobstores-all
  - nx-component-upload(most probably this source,should alone with view source,should suffice, though I did not test)
  - nx-repository-admin---*

    - nx-repository-view---*
- Next goto `settings` → `security` → `users` → `create user`, a new popup comes prompting for user details fill all the details and set `status` to `Active` and also add the role created at previous step by moving the role from left hand side box to right hand side box.

Install Helm Repository.

Build from source code.

- Clone the project.

```
git clone https://github.com/sonatype-nexus-community/nexus-repository-helm.git
```

- change directory

```
cd nexus-repository-helm
```

- Build the source.

```
$ mvn clean package -DskipTests
```

If you want stable release checkout by last released version tag.

Enable Helm repository

- Copy the bundle from target to Copy the bundle into `<nexus_dir>/system/org/sonatype/nexus/plugins/nexus-repository-helm/0.0.7/nexus-repository-helm-0.0.7.jar`

```
 sudo mkdir -p /opt/nexus-server/nexus-3.15.2-
01/system/org/sonatype/nexus/plugins/nexus-repository-helm/0.0.7/
 sudo target/nexus-repository-helm-0.0.7.jar \
     /opt/nexus-server/nexus-3.15.2-01/system/org/sonatype/nexus/plugins/nexus-
repository-helm/0.0.7/
 sudo chown nexus:nexus -R /opt/nexus-server/nexus-3.15.2-
01/system/org/sonatype/nexus/plugins/nexus-repository-helm/0.0.7/
```

- Update OSGi feature by updating `features.xml` at `<nexus_home>/system/org/sonatype/nexus/assemblies/nexus-core-feature/3.15.2-01/nexus-core-feature-3.15.2-01-features.xml`
    - Add entire highlighted line under `nexus-core-feature` section.

```
<feature name="nexus-core-feature"
description="org.sonatype.nexus.assemblies:nexus-core-feature"
version="3.15.2.01">
        <details>org.sonatype.nexus.assemblies:nexus-core-feature</details>
        <feature version="3.15.2.01" prerequisite="false" dependency="false"
>nexus-audit-plugin</feature>
        <feature version="3.15.2.01" prerequisite="false" dependency="false"
>nexus-blobstore-tasks</feature>
        <feature version="3.15.2.01" prerequisite="false" dependency="false"
>nexus-ssl-plugin</feature>
        <feature version="3.15.2.01" prerequisite="false" dependency="false"
>nexus-coreui-plugin</feature>
        <feature version="3.15.2.01" prerequisite="false" dependency="false"
>nexus-repository-httpbridge</feature>
        <feature version="3.15.2.01" prerequisite="false" dependency="false"
>nexus-repository-maven</feature>
        <feature version="3.15.2.01" prerequisite="false" dependency="false"
>nexus-repository-npm</feature>
        <feature version="3.15.2.01" prerequisite="false" dependency="false"
>nexus-repository-pypi</feature>
        <feature version="3.15.2.01" prerequisite="false" dependency="false"
>nexus-repository-raw</feature>
        <strong><feature version="0.0.7" prerequisite="false" dependency=
"false">nexus-repository-helm</feature></strong>
        <feature version="3.15.2.01" prerequisite="false" dependency="false"
>nexus-restore-maven</feature>
        <feature version="3.15.2.01" prerequisite="false" dependency="false"
>nexus-blobstore-s3</feature>
        <feature version="3.15.2.01" prerequisite="false" dependency="false"
>nexus-restore-npm</feature>
        <feature version="3.15.2.01" prerequisite="false" dependency="false"
>nexus-restore-pypi</feature>
        <feature version="3.15.2.01" prerequisite="false" dependency="false"
>nexus-restore-raw</feature>
        <feature version="3.15.2.01" prerequisite="false" dependency="false"
>nexus-script-plugin</feature>
        <feature version="3.15.2.01" prerequisite="false" dependency="false"
>nexus-task-log-cleanup</feature>
        <feature prerequisite="true" dependency="false">wrap</feature>
</feature>
```

- Add below xml section either below `nexus-core-feature` or just above `</features`

```
<feature name="nexus-repository-helm"
description="org.sonatype.nexus.plugins:nexus-repository-helm" version="0.0.7">
    <details>org.sonatype.nexus.plugins:nexus-repository-helm</details>
    <bundle>mvn:org.sonatype.nexus.plugins/nexus-repository-helm/0.0.7</bundle>
</feature>
```

- Once `features.xml` file is updated restart the nexus service.

```
$ sudo systemctl restart nexus.service
```

- Verify nexus service start is successful.

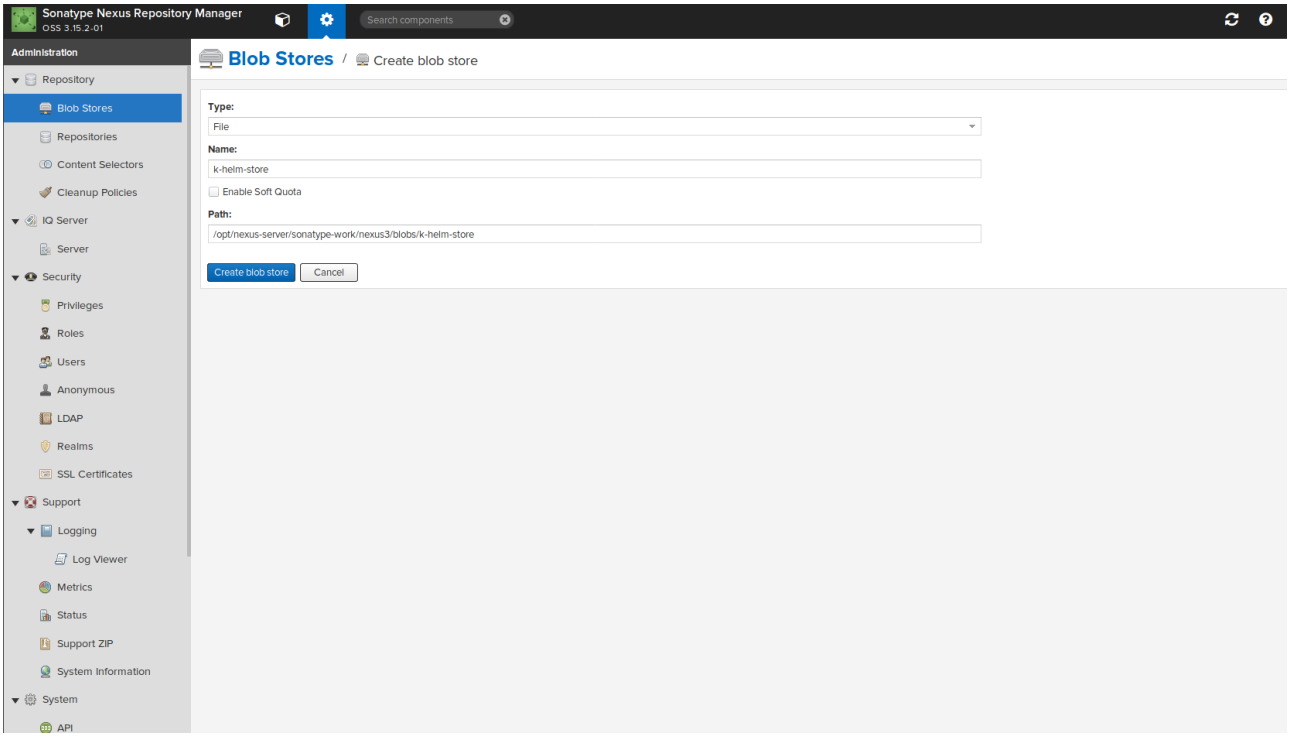
```
$ sudo systemctl status nexus.service
```

Create a hosted Helm repository

Current implementation supports hosted and proxy repositories, and does not support group, We will be creating only hosted helm repository, creating proxy should be straight forward.

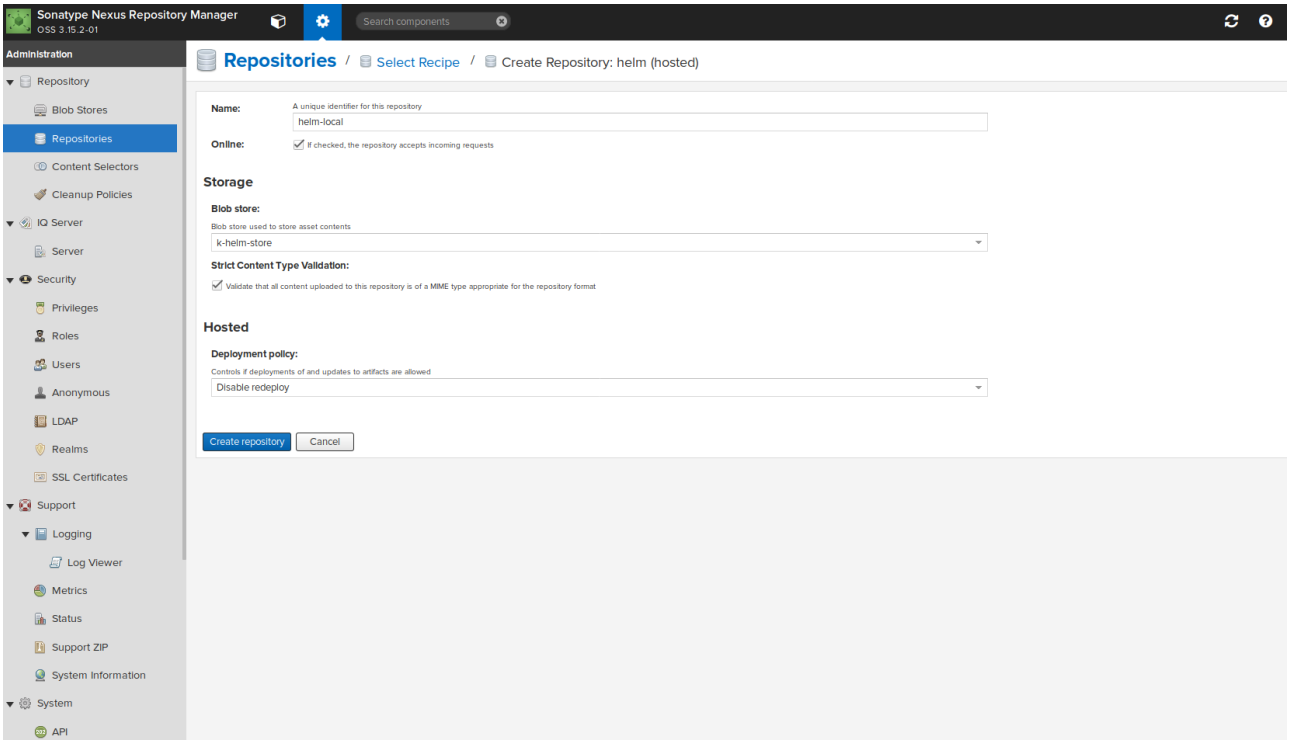
- Create blob to store helm packages.

  Go to `settings` → `repository` → `Blob stores` → `Create blob`, fill the blob name, if required change the path keep the blob store directory/path as is.



- Create helm local repository.

  Go to `settings` → `repository` → `Repositories` → `Create Repository`, and select `helm(hosted)` from list of repositories. Set the desired name and the also set the blob name that was created in previous step.


Sonatype Nexus Repository Manager
OSS 3.15.2-01
Search components
Administration
Repository
Blob Stores
Repositories
Content Selectors
Cleanup Policies
IQ Server
Server
Security
Privileges
Roles
Users
Anonymous
LDAP
Realms
SSL Certificates
Support
Logging
Log Viewer
Metrics
Status
Support ZIP
System Information
System
API
Repositories / Select Recipe / Create Repository: helm (hosted)
Name:
A unique identifier for this repository
helm-local
Online:
If checked, the repository accepts incoming requests
Storage
Blob store:
Blob store used to store asset contents
k-helm-store
Strict Content Type Validation:
Validate that all content uploaded to this repository is of a MIME type appropriate for the repository format
Hosted
Deployment policy:
Controls if deployments of and updates to artifacts are allowed
Disable redeploy
Create repository
Cancel