From monolith to microservices

Simple path for your application

Tutorial

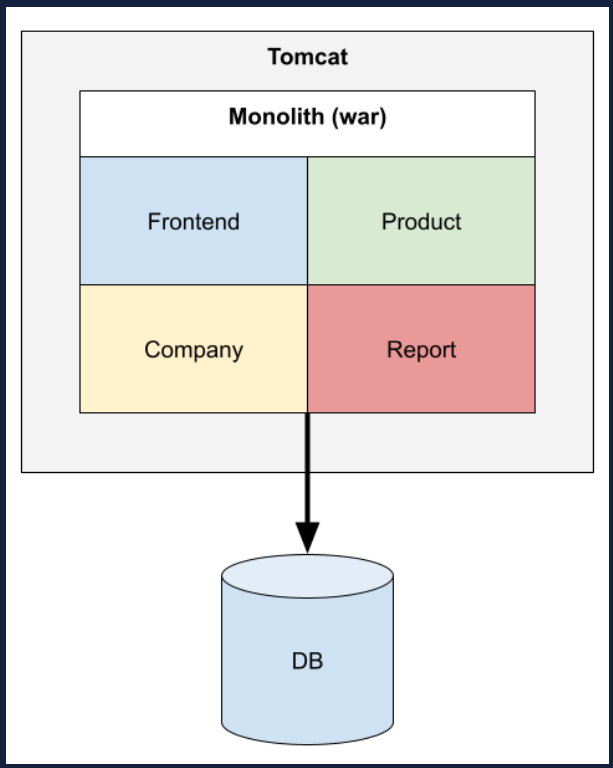
May 2022

Overview

In this tutorial, we'll see how you can migrate from monolith application to microservices with Onteon.

Example application

Example application is a simple Tomcat application, that allows user to manage products and reports.



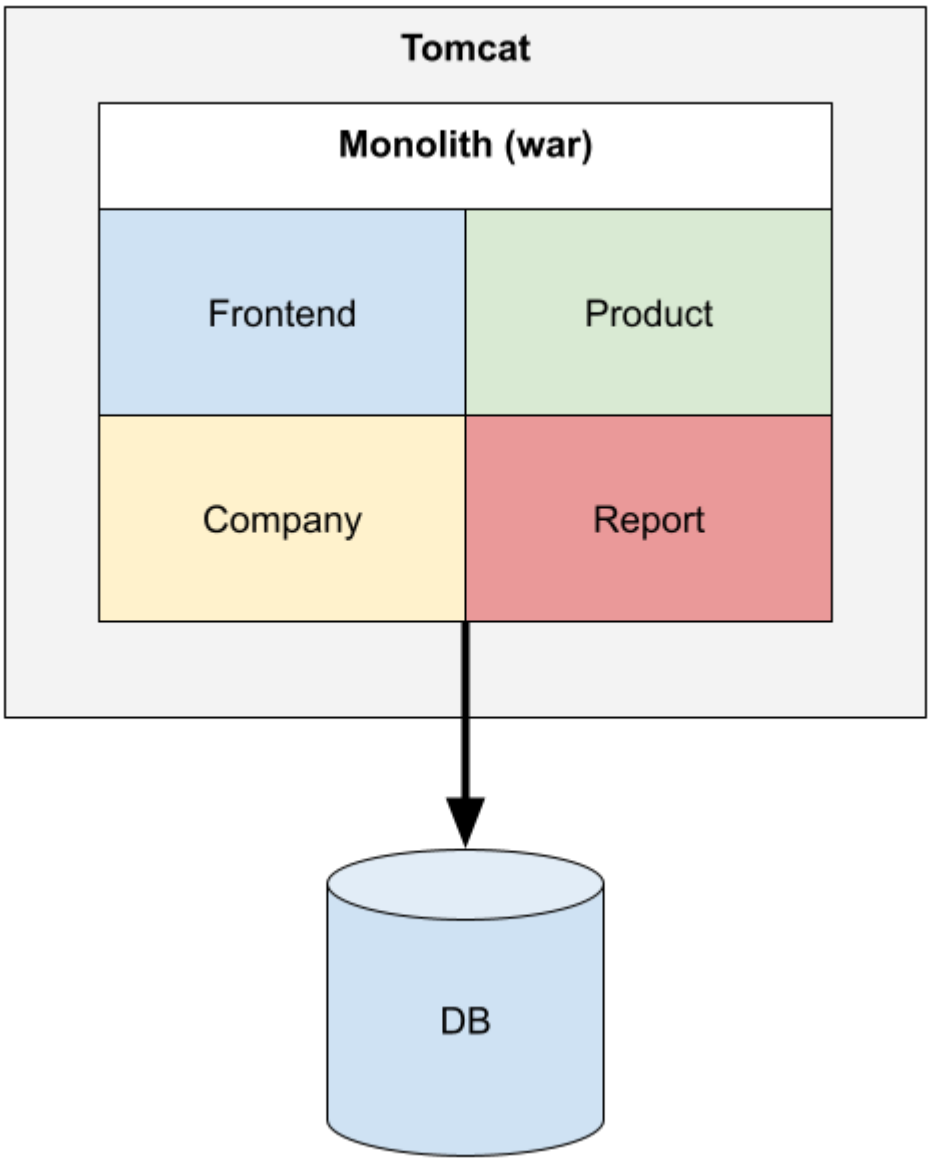
Requirements

Before reading the tutorial, make sure that you have Java with Maven and Docker with Docker Compose installed on your machine. Application uses PostgreSQL database.

Table of Contents

Step 0 - Running the example application without Onteon

In this step we will download and start application in Tomcat, without Onteon.



Files

First, you need to download and unzip application files. Click here to download files.

Database

Before starting the application, you need to start the database. To do so, you can use Docker. Simply run `docker run -e POSTGRES_PASSWORD=password -p 5432:5432 postgres`.

Tomcat

Now you need to download and untar Tomcat. To do so, go to Tomcat download page, download package and untar it.

```
monolith$ wget https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.62/bin/apache-tomcat-9.0.62.tar.gz
monolith$ tar -xvf apache-tomcat-9.0.62.tar.gz
```

Now, clean the tomcat's `webapps` directory.

```
monolith$ rm -r apache-tomcat-9.0.62/webapps/*
```

Application

To prepare application, go to `monolith` directory and run `mvn clean package`.

```
$ cd monolith
monolith$ mvn clean package
```

After this, copy war file from target directory to webapps.

```
monolith$ cp target/monolith.war apache-tomcat-9.0.62/webapps
```

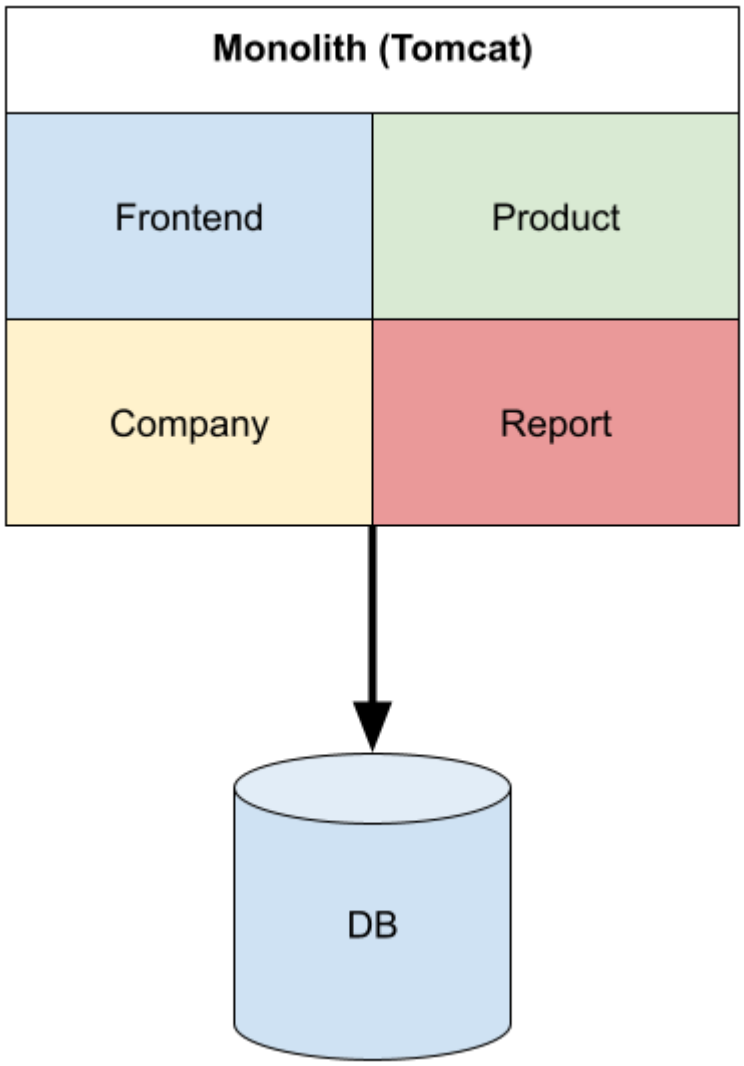
Start application

To start application, run Tomcat by executing `./apache-tomcat-9.0.62/bin/catalina.sh run` and go to http://localhost:8080/monolith.

If it works, press **CTRL + C** to stop the application, stop the Docker container and remove the `apache-tomcat-9.0.62` directory.

Step 1 - Running and scaling monolith application with Onteon

In this step we will run and scale application with Onteon.



Files

If you don't have files from previous step, you can:

- download **initial** files from here
- download **complete** files from here

Onteon

Let's start the Onteon with Docker Compose. Create `docker-compose.yml` file in the root directory and paste the content.

```
version: "3.8"
services:
  onteoncc-master:
    image: onteon/control-center:1.2.0
    ports:
      - "8050:8050"
      - "9096:9096"
      - "27017:27017"
      - "27018:27018"
      - "27019:27019"
    environment:
      ONTEON_MONITORING_ENABLED: "false"
    command: [ "./start-master.sh" ]
  onteon-node-manager-1:
    image: onteon/node-manager:1.2.0
    privileged: true
    tty: true
    depends_on:
      - "onteoncc-master"
    ports:
      - "8020:8020"
  onteon-node-manager-2:
    image: onteon/node-manager:1.2.0
    privileged: true
    tty: true
    depends_on:
      - "onteoncc-master"
    ports:
      - "8021:8021"
  db:
    image: postgres
    environment:
      POSTGRES_PASSWORD: password
    ports:
      - "5432:5432"
  onteoncli:
    profiles: [ "cli-only" ]
    image: onteon/onteoncli-with-demo-apps:1.2.0
    stdin_open: true
    tty: true
    volumes:
      - ".:/opt/onteon"
    depends_on:
      - "onteoncc-master"
```

Now, start Onteon with `docker-compose up` command.

OnteonCLI

To communicate with Onteon Cluster, you can use the OnteonCLI tool. You don't have to install this on your machine. Simply execute `docker-compose run --rm onteoncli`, and here you will be able to run OnteonCLI commands.

```
$ docker-compose run --rm onteoncli
Creating tutorial-migrating-from-monolith-to-microservices-step0_onteoncli_run ... done
root@4e71436c1dc2:/opt/onteon# onteoncli --version
1.2.0 (72a185028983f0349bf2d934fc43af80334eda2a)
```

Now, you need to log in to cluster. Simply execute `onteoncli login --cluster-url http://onteoncc-master:8050` and accept code in your browser (copy the link displayed in the console and paste it in browser).

Database

Now, we need to change database url, since we will use database described in docker-compose.yml file.

Go to `monolith/src/main/resources/application.properties` and change the `spring.datasource.url` property to `jdbc:postgresql://db:5432/postgres`. Full file should look like this:

```
spring.datasource.url=jdbc:postgresql://db:5432/postgres
spring.datasource.username=postgres
spring.datasource.password=password
spring.jpa.hibernate.ddl-auto=update
spring.jpa.database-platform=org.hibernate.dialect.PostgreSQL94Dialect
```

URLs

Onteon uses two specific url patterns to access applications. First, chooses the application `_by_name`, second - `_by_name_and_version`. We need to change the fronted urls to backend. Go to `monolith/src/main/resources/templates/index.html` and change the value of `baseUrl` variable to `/_by_name/monolith`.

```
const baseUrl = '/_by_name/monolith';
const companyApiBaseUrl = `${baseUrl}/api/v1/companies`;
const productApiBaseUrl = `${baseUrl}/api/v1/products`;
const reportApiBaseUrl = `${baseUrl}/api/v1/reports`;
```

Tomcat configuration

Since we want to start many instances of our application, we need to somehow change the ports for every instance. It is possible with Onteon and `ont_port` placeholder.

Let's create `monolith/tomcat/conf` directory, and then create a `server.xml` file in it. It will be the Tomcat configuration file. Onteon will check the file and replace all placeholders in it.

Here is the content of `monolith/tomcat/conf/server.xml` file.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  Licensed to the Apache Software Foundation (ASF) under one or more
  contributor license agreements.  See the NOTICE file distributed with
  this work for additional information regarding copyright ownership.
  The ASF licenses this file to You under the Apache License, Version 2.0
  (the "License"); you may not use this file except in compliance with
  the License.  You may obtain a copy of the License at

      http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License.
-->
<!-- Note:  A "Server" is not itself a "Container", so you may not
     define subcomponents such as "Valves" at this level.
     Documentation at /docs/config/server.html
 -->
<Server port="${ont_port_1}" shutdown="SHUTDOWN">
    <Listener className="org.apache.catalina.startup.VersionLoggerListener"/>
    <!-- Security listener. Documentation at /docs/config/listeners.html
    <Listener className="org.apache.catalina.security.SecurityListener" />
    -->
    <!-- APR library loader. Documentation at /docs/apr.html -->
    <Listener className="org.apache.catalina.core.AprLifecycleListener" SSLEngine="on"/>
    <!-- Prevent memory leaks due to use of particular java/javax APIs-->
    <Listener className="org.apache.catalina.core.JreMemoryLeakPreventionListener"/>
```

```
    <Listener className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener" />
    <Listener className="org.apache.catalina.core.ThreadLocalLeakPreventionListener" />

    <!-- Global JNDI resources
         Documentation at /docs/jndi-resources-howto.html
    -->
    <GlobalNamingResources>
        <!-- Editable user database that can also be used by
             UserDatabaseRealm to authenticate users
        -->
        <Resource name="UserDatabase" auth="Container"
                  type="org.apache.catalina.UserDatabase"
                  description="User database that can be updated and saved"
                  factory="org.apache.catalina.users.MemoryUserDatabaseFactory"
                  pathname="conf/tomcat-users.xml" />
    </GlobalNamingResources>

    <!-- A "Service" is a collection of one or more "Connectors" that share
         a single "Container" Note:  A "Service" is not itself a "Container",
         so you may not define subcomponents such as "Valves" at this level.
         Documentation at /docs/config/service.html
     -->
    <Service name="Catalina">

        <!--The connectors can use a shared executor, you can define one or more named thread pools-->
        <!--
        <Executor name="tomcatThreadPool" namePrefix="catalina-exec-"
            maxThreads="150" minSpareThreads="4" />
        -->


        <!-- A "Connector" represents an endpoint by which requests are received
             and responses are returned. Documentation at :
             Java HTTP Connector: /docs/config/http.html
             Java AJP  Connector: /docs/config/ajp.html
             APR (HTTP/AJP) Connector: /docs/apr.html
             Define a non-SSL/TLS HTTP/1.1 Connector on port 8080
        -->
        <Connector port="${ont_port_2}" protocol="HTTP/1.1"
                   connectionTimeout="20000"
                   redirectPort="${ont_port_3}" />
        <!-- A "Connector" using the shared thread pool-->
        <!--
        <Connector executor="tomcatThreadPool"
                   port="8080" protocol="HTTP/1.1"
                   connectionTimeout="20000"
                   redirectPort="8443" />
        -->
        <!-- Define an SSL/TLS HTTP/1.1 Connector on port 8443
             This connector uses the NIO implementation. The default
             SSLImplementation will depend on the presence of the APR/native
             library and the useOpenSSL attribute of the AprLifecycleListener.
             Either JSSE or OpenSSL style configuration may be used regardless of
             the SSLImplementation selected. JSSE style configuration is used below.
        -->
        <!--
        <Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
                   maxThreads="150" SSLEnabled="true">
            <SSLHostConfig>
                <Certificate certificateKeystoreFile="conf/localhost-rsa.jks"
                             type="RSA" />
            </SSLHostConfig>
        </Connector>
        -->
        <!-- Define an SSL/TLS HTTP/1.1 Connector on port 8443 with HTTP/2
             This connector uses the APR/native implementation which always uses
             OpenSSL for TLS.
             Either JSSE or OpenSSL style configuration may be used. OpenSSL style
             configuration is used below.
        -->
        <!--
        <Connector port="8443" protocol="org.apache.coyote.http11.Http11AprProtocol"
                   maxThreads="150" SSLEnabled="true" >
            <UpgradeProtocol className="org.apache.coyote.http2.Http2Protocol" />
            <SSLHostConfig>
                <Certificate certificateKeyFile="conf/localhost-rsa-key.pem"
                             certificateFile="conf/localhost-rsa-cert.pem"
                             certificateChainFile="conf/localhost-rsa-chain.pem"
                             type="RSA" />
            </SSLHostConfig>
        </Connector>
        -->

        <!-- Define an AJP 1.3 Connector on port 8009 -->
        <!--
        <Connector protocol="AJP/1.3"
                   address="::1"
                   port="8009"
                   redirectPort="8443" />
        -->

        <!-- An Engine represents the entry point (within Catalina) that processes
             every request.  The Engine implementation for Tomcat stand alone
             analyzes the HTTP headers included with the request, and passes them
             on to the appropriate Host (virtual host).
```

```
             Documentation at /docs/config/engine.html -->

        <!-- You should set jvmRoute to support load-balancing via AJP ie :
        <Engine name="Catalina" defaultHost="localhost" jvmRoute="jvm1">
        -->
        <Engine name="Catalina" defaultHost="localhost">

            <!--For clustering, please take a look at documentation at:
                /docs/cluster-howto.html  (simple how to)
                /docs/config/cluster.html (reference documentation) -->
            <!--
            <Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster"/>
            -->

            <!-- Use the LockOutRealm to prevent attempts to guess user passwords
                 via a brute-force attack -->
            <Realm className="org.apache.catalina.realm.LockOutRealm">
                <!-- This Realm uses the UserDatabase configured in the global JNDI
                     resources under the key "UserDatabase".  Any edits
                     that are performed against this UserDatabase are immediately
                     available for use by the Realm.  -->
                <Realm className="org.apache.catalina.realm.UserDatabaseRealm"
                       resourceName="UserDatabase"/>
            </Realm>

            <Host name="localhost" appBase="webapps"
                  unpackWARs="true" autoDeploy="true">

                <!-- SingleSignOn valve, share authentication between web applications
                     Documentation at: /docs/config/valve.html -->
                <!--
                <Valve className="org.apache.catalina.authenticator.SingleSignOn" />
                -->

                <!-- Access log processes all example.
                     Documentation at: /docs/config/valve.html
                     Note: The pattern used is equivalent to using pattern="common" -->
                <Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"
                       prefix="localhost_access_log" suffix=".txt"
                       pattern="%h %l %u %t &amp;quot;%r&amp;quot; %s %b"/>

            </Host>
        </Engine>
    </Service>
</Server>
```

As you can see, we used different port placeholders: ont_port_1, ont_port_2, ont_port_3. Onteon automatically detects the placeholders and chooses as many ports as you want for your application instance. In this case, Onteon will choose three different ports.

Application structure

In Onteon applications are stored in tar.gz files. Every tar file has a bin and a conf directory. A bin directory contains all application files. Conf contains conf.yml file, which is a configuration file that describes application.

Bin

Let's create a `monolith/app/bin` directory. Here you can copy the Tomcat files. Full structure should look like this:

```
monolith/app
└── bin
    ├── bin
    ├── BUILDING.txt
    ├── conf
    ├── CONTRIBUTING.md
    ├── lib
    ├── LICENSE
    ├── logs
    ├── NOTICE
    ├── README.md
    ├── RELEASE-NOTES
    ├── RUNNING.txt
    ├── temp
    ├── webapps
    └── work
```

Now, clean the `webapps` directory.

```
$ rm -rf monolith/app/bin/webapps/*
```

Also, replace `monolith/tomcat/conf/server.xml` with `monolith/app/bin/conf/server.xml`.

Go to `monolith` directory and build application war file.

```
$ cd monolith
monolith$ mvn clean package
```

Now, copy the war file to `webapps`, and change it name to ROOT.war.

```
monolith$ cp target/monolith.war app/bin/webapps/ROOT.war
```

Conf

Let's create an application configuration file. First, create `monolith/app/conf/conf.yml` file.

Now let's write the content of `conf.yml` file.

```yaml
app:
  name: 'monolith'
  version: '1.0.0'
  appType: 'standard'
  procType: 'native'
  processProvider:
    name: 'GenericOsProcessProviderImpl'
    version: '1.0.0'
    executable:
      start:
        command: '${ont_app_path}/bin/bin/catalina.sh run'
        successLine: 'Server startup'
      stop:
        command: '${ont_app_path}/bin/bin/catalina.sh stop'
      terminate:
        command: '${ont_app_path}/bin/bin/catalina.sh stop -force'
  placeHolder:
    name: 'PlaceHolderManagerImpl'
    version: '1.0.0'
    filesToReplace:
      - '${ont_app_path}/bin/conf/server.xml'
    variables:
  serviceRepository:
    healthCheckUrl: 'http://localhost:${ont_port_2}'
    entities:
      - entity:
          priority: 1
          port: ${ont_port_2}
          protocol:
            type: 'HTTP'
            version: '1.1'
          isExternal: true
          isInternal: true
```

The first part of configuration contains basic information about app, such as name, version and process type.

```
app:
  name: 'monolith'
  version: '1.0.0'
  appType: 'standard'
  procType: 'native'
```

Then, we specify commands that start, stops and terminates application.

```
  processProvider:
    name: 'GenericOsProcessProviderImpl'
    version: '1.0.0'
    executable:
      start:
        command: '${ont_app_path}/bin/bin/catalina.sh run'
        successLine: 'Server startup'
      stop:
        command: '${ont_app_path}/bin/bin/catalina.sh stop'
      terminate:
        command: '${ont_app_path}/bin/bin/catalina.sh stop -force'
```

Then, we specify the files which will have the placeholders replaced. Also, you can define your custom placeholders here in `variables` section.

```
  placeHolder:
    name: 'PlaceHolderManagerImpl'
    version: '1.0.0'
    filesToReplace:
      - '${ont_app_path}/bin/conf/server.xml'
    variables:
```

The last part defines the health check url, and the information about availability on load balancers (we will talk about later). For now, application will be available on external and internal load balancers.

```
  serviceRepository:
    healthCheckUrl: 'http://localhost:${ont_port_2}'
    entities:
      - entity:
          priority: 1
          port: ${ont_port_2}
          protocol:
            type: 'HTTP'
            version: '1.1'
          isExternal: true
          isInternal: true
```

Application Package

Now, we are ready to create application package. Simply tar both bin and directory package.

```
$ cd monolith/app
monolith/app$ tar -zcvf monolith.tar.gz *
```

Full tar structure should look like this:

```
monolith.tar.gz
├── bin
│   ├── bin
│   ├── BUILDING.txt
│   ├── conf
│   ├── CONTRIBUTING.md
│   ├── lib
│   ├── LICENSE
│   ├── logs
│   ├── NOTICE
│   ├── README.md
│   ├── RELEASE-NOTES
│   ├── RUNNING.txt
│   ├── temp
│   ├── webapps
│   └── work
└── conf
    └── conf.yml
```

Upload application

To upload application, run `onteoncli application-registry upload monolith.tar.gz` (you can do this from docker container with preinstalled OnteonCLI). You can see that application was created by executing `onteoncli application list`.

```
monolith/app$ onteoncli application-registry upload monolith.tar.gz
monolith/app$ onteoncli application list
```

Distribution

Now let's create a distribution that will automatically create and start specified number of instances.

Let's create a `distribution.yml` file that describe distribution.

```
application: monolith:1.0.0
numberOfInstances: 2
scripts:
  checkIfNodeCanAcceptNewApplicationInstance: defaultAvailableNodeOnlyCINCANAIV1
  selectNodeForNewApplicationInstance: defaultApplicationInstancesCountOnlySNFNAIV1
  selectApplicationInstanceToRemove: defaultApplicationInstancesCountOnlySAITRV1
```

As you can see, the file contains information about application that will be distributed, desired number of instances, and scripts that decides how to distribute application. We will use the default scripts.

Let's create a distribution and see how distribution works.

```
monolith/app$ onteoncli distribution create-from-file distribution.yml
id:                                          624ac75217e51441a673b2ad
createdAT:                                   2022-04-04T10:24:18.257Z
updatedAt:                                   2022-04-04T10:24:18.257Z
applicationId:                               624ac68017e51441a673b22e
numberOfInstances:                           2
checkIfNodeCanAcceptNewApplicationInstanceScriptId: 624abbfa17e51441a673abd3
selectNodeForNewApplicationInstanceScriptId:        624abbfa17e51441a673abcf
selectApplicationInstanceToRemoveScriptId:          624abbfa17e51441a673abd1

monolith/app$ watch onteoncli application-instance list

id                       createdAt               applicationName applicationVersion applicationType applicationId            nodeId                   status
133a1ea2f703a66a68341f22 2022-04-04T10:24:34.599Z monolith        1.0.0              standard        624ac68017e51441a673b22e 0e1edbf6d26653d5376ece20 running
e3111ea6b17a9afd6796b9e7 2022-04-04T10:24:23.317Z monolith        1.0.0              standard        624ac68017e51441a673b22e 3c76d9bea9c92f15a673d6e8 running
9b75555ef6f1dfe23abbbf61 2022-04-04T09:36:08.683Z nginx-edge      1.1.3              embedded        0a1769f83abc3224fe6cb477 3c76d9bea9c92f15a673d6e8 running
441227fb169c58fcb8786e6e 2022-04-04T09:36:08.681Z nginx-edge      1.1.3              embedded        0a1769f83abc3224fe6cb477 0e1edbf6d26653d5376ece20 running
7eb19ec4ab6ac31d0a2a2680 2022-04-04T09:36:02.488Z nginx-inner     1.1.3              embedded        157958ad0cdf14266501ad70 0e1edbf6d26653d5376ece20 running
6751d5bd71f171ffd6e1e855 2022-04-04T09:36:02.486Z nginx-inner     1.1.3              embedded        157958ad0cdf14266501ad70 3c76d9bea9c92f15a673d6e8 running
```

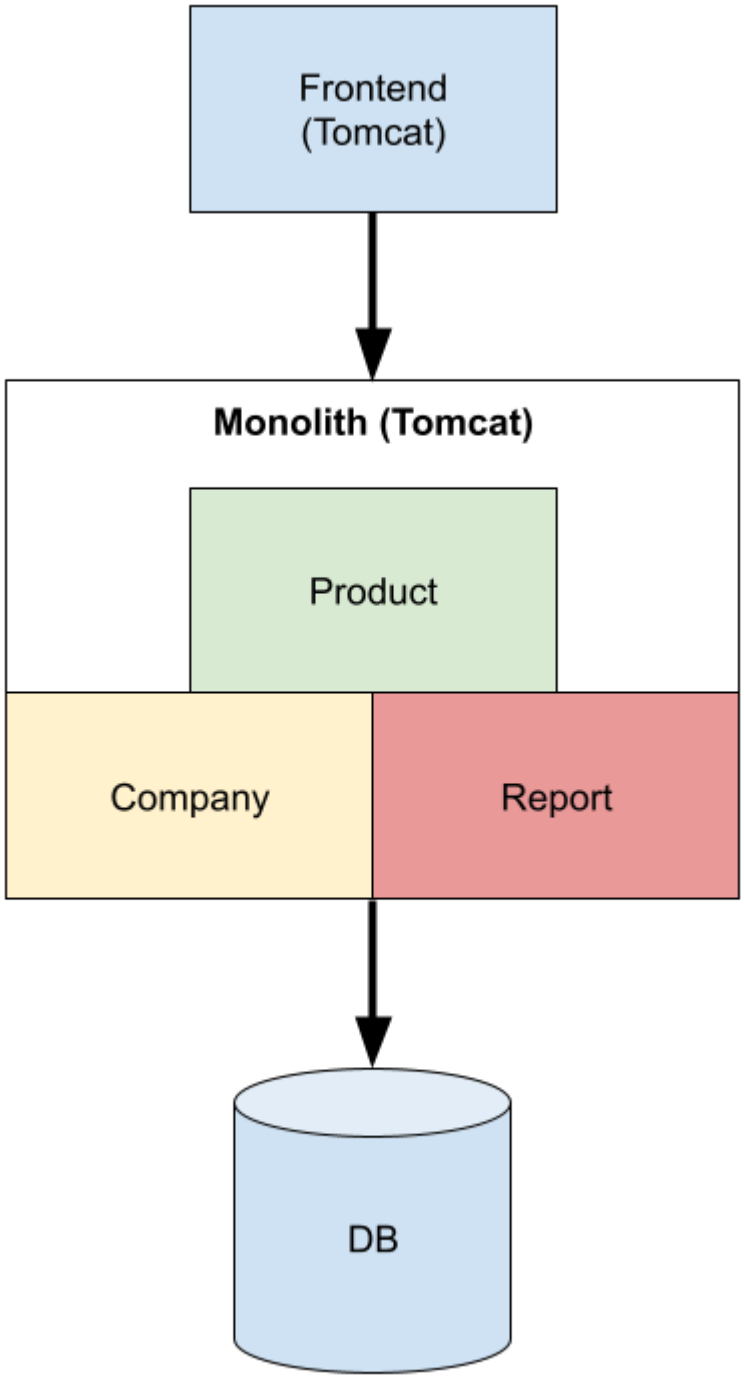
As you can see, distribution automatically created two instances of monolith application.

Access application

To access the application, go to http://localhost:8020/_by_name/monolith.

Step 2 - Extracting frontend from monolith and upgrading application without disruptions

In this step we will separate frontend and backend. Then, we will perform zero downtime upgrade of monolith application.



Files

If you don't have files from previous step, you can:

- download **initial** files from here
- download **complete** files from here

Frontend

This time, we will add a script that will build application package for us.

Create a `frontend/scripts` directory and put here a build.sh script.

```
#!/usr/bin/env bash

set -e

SOURCE="${BASH_SOURCE[0]}"
while [ -h "$SOURCE" ]; do
  DIR="$(cd -P "$( dirname "$SOURCE" )" >/dev/null 2>&1 && pwd)"
  SOURCE="$(readlink "$SOURCE")"
  [[ $SOURCE != /* ]] && SOURCE="$DIR/$SOURCE"
done
SCRIPT_DIR="$(cd -P "$(dirname "$SOURCE")" >/dev/null 2>&1 && pwd)"
ROOT_DIR="${SCRIPT_DIR}/.."
TARGET_DIR="${ROOT_DIR}/target"
DOWNLOAD_DIR="${ROOT_DIR}/download"
TOMCAT_SOURCE="${DOWNLOAD_DIR}/tomcat.tar.gz"
TOMCAT_DOWNLOAD_URL="http://archive.apache.org/dist/tomcat/tomcat-9/v9.0.60/bin/apache-tomcat-9.0.60.tar.gz"
TOMCAT_DIR_NAME="apache-tomcat-9.0.60"
APP_BIN_DIR="${TARGET_DIR}/app/bin"
APP_CONF_DIR="${TARGET_DIR}/app/conf"


mkdir -p "${DOWNLOAD_DIR}"
if [ ! -f "${TOMCAT_SOURCE}" ]; then
  wget -O "${TOMCAT_SOURCE}" "${TOMCAT_DOWNLOAD_URL}"
fi

rm -rf "${TARGET_DIR}"
mkdir -p "${APP_BIN_DIR}"
mkdir -p "${APP_CONF_DIR}"
tar -xvf "${TOMCAT_SOURCE}" -C "${TARGET_DIR}"
cp -r "${TARGET_DIR}/${TOMCAT_DIR_NAME}/"* "${APP_BIN_DIR}"
rm -rf "${APP_BIN_DIR}/webapps/"*
cp "${ROOT_DIR}/onteon/conf.yml" "${APP_CONF_DIR}"
cp -r "${ROOT_DIR}/tomcat/"* "${APP_BIN_DIR}"
cd "${TARGET_DIR}/app" && tar -zcvf "${TARGET_DIR}/frontend.tar.gz" *
```

Then, set the proper rights of file, by executing the `chmod 750 frontend/scripts/build.sh` command.

Now, create a `frontend/onteon` directory and put there a conf.yml file, which will be a configuration file for frontend application. It will be similar to the configuration from the monolith. The only difference will be the name of the application.

```
app:
  name: 'frontend'
  version: '1.0.0'
  appType: 'standard'
  procType: 'native'
  processProvider:
    name: 'GenericOsProcessProviderImpl'
    version: '1.0.0'
    executable:
      start:
        command: '${ont_app_path}/bin/bin/catalina.sh run'
        successLine: 'Server startup'
      stop:
        command: '${ont_app_path}/bin/bin/catalina.sh stop'
      terminate:
        command: '${ont_app_path}/bin/bin/catalina.sh stop -force'
  placeHolder:
```

```
    name: 'PlaceHolderManagerImpl'
    version: '1.0.0'
    filesToReplace:
      - '${ont_app_path}/bin/conf/server.xml'
    variables:
  serviceRepository:
    healthCheckUrl: 'http://localhost:${ont_port_2}'
    entities:
      - entity:
          priority: 1
          port: ${ont_port_2}
          protocol:
            type: 'HTTP'
            version: '2.0'
          isExternal: true
          isInternal: true
```

Now, let's create a `frontend/tomcat/conf/server.xml` file, with placeholders.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  Licensed to the Apache Software Foundation (ASF) under one or more
  contributor license agreements.  See the NOTICE file distributed with
  this work for additional information regarding copyright ownership.
  The ASF licenses this file to You under the Apache License, Version 2.0
  (the "License"); you may not use this file except in compliance with
  the License.  You may obtain a copy of the License at

      http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License.
-->
<!-- Note:  A "Server" is not itself a "Container", so you may not
     define subcomponents such as "Valves" at this level.
     Documentation at /docs/config/server.html
 -->
<Server port="${ont_port_1}" shutdown="SHUTDOWN">
    <Listener className="org.apache.catalina.startup.VersionLoggerListener"/>
    <!-- Security listener. Documentation at /docs/config/listeners.html
    <Listener className="org.apache.catalina.security.SecurityListener" />
    -->
    <!-- APR library loader. Documentation at /docs/apr.html -->
    <Listener className="org.apache.catalina.core.AprLifecycleListener" SSLEngine="on"/>
    <!-- Prevent memory leaks due to use of particular java/javax APIs-->
    <Listener className="org.apache.catalina.core.JreMemoryLeakPreventionListener"/>
    <Listener className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener"/>
    <Listener className="org.apache.catalina.core.ThreadLocalLeakPreventionListener"/>

    <!-- Global JNDI resources
         Documentation at /docs/jndi-resources-howto.html
    -->
    <GlobalNamingResources>
        <!-- Editable user database that can also be used by
             UserDatabaseRealm to authenticate users
        -->
        <Resource name="UserDatabase" auth="Container"
                  type="org.apache.catalina.UserDatabase"
                  description="User database that can be updated and saved"
                  factory="org.apache.catalina.users.MemoryUserDatabaseFactory"
                  pathname="conf/tomcat-users.xml"/>
    </GlobalNamingResources>

    <!-- A "Service" is a collection of one or more "Connectors" that share
         a single "Container" Note:  A "Service" is not itself a "Container",
         so you may not define subcomponents such as "Valves" at this level.
         Documentation at /docs/config/service.html
     -->
    <Service name="Catalina">

        <!--The connectors can use a shared executor, you can define one or more named thread pools-->
        <!--
        <Executor name="tomcatThreadPool" namePrefix="catalina-exec-"
            maxThreads="150" minSpareThreads="4"/>
        -->


        <!-- A "Connector" represents an endpoint by which requests are received
             and responses are returned. Documentation at :
             Java HTTP Connector: /docs/config/http.html
             Java AJP  Connector: /docs/config/ajp.html
             APR (HTTP/AJP) Connector: /docs/apr.html
             Define a non-SSL/TLS HTTP/1.1 Connector on port 8080
        -->
        <Connector port="${ont_port_2}" protocol="HTTP/1.1"
                   connectionTimeout="20000"
                   redirectPort="${ont_port_3}"/>
        <!-- A "Connector" using the shared thread pool-->
        <!--
```

```
        <Connector executor="tomcatThreadPool"
                   port="8080" protocol="HTTP/1.1"
                   connectionTimeout="20000"
                   redirectPort="8443" />
        -->
        <!-- Define an SSL/TLS HTTP/1.1 Connector on port 8443
             This connector uses the NIO implementation. The default
             SSLImplementation will depend on the presence of the APR/native
             library and the useOpenSSL attribute of the AprLifecycleListener.
             Either JSSE or OpenSSL style configuration may be used regardless of
             the SSLImplementation selected. JSSE style configuration is used below.
        -->
        <!--
        <Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
                   maxThreads="150" SSLEnabled="true">
            <SSLHostConfig>
                <Certificate certificateKeystoreFile="conf/localhost-rsa.jks"
                             type="RSA" />
            </SSLHostConfig>
        </Connector>
        -->
        <!-- Define an SSL/TLS HTTP/1.1 Connector on port 8443 with HTTP/2
             This connector uses the APR/native implementation which always uses
             OpenSSL for TLS.
             Either JSSE or OpenSSL style configuration may be used. OpenSSL style
             configuration is used below.
        -->
        <!--
        <Connector port="8443" protocol="org.apache.coyote.http11.Http11AprProtocol"
                   maxThreads="150" SSLEnabled="true" >
            <UpgradeProtocol className="org.apache.coyote.http2.Http2Protocol" />
            <SSLHostConfig>
                <Certificate certificateKeyFile="conf/localhost-rsa-key.pem"
                             certificateFile="conf/localhost-rsa-cert.pem"
                             certificateChainFile="conf/localhost-rsa-chain.pem"
                             type="RSA" />
            </SSLHostConfig>
        </Connector>
        -->

        <!-- Define an AJP 1.3 Connector on port 8009 -->
        <!--
        <Connector protocol="AJP/1.3"
                   address="::1"
                   port="8009"
                   redirectPort="8443" />
        -->

        <!-- An Engine represents the entry point (within Catalina) that processes
             every request.  The Engine implementation for Tomcat stand alone
             analyzes the HTTP headers included with the request, and passes them
             on to the appropriate Host (virtual host).
             Documentation at /docs/config/engine.html -->

        <!-- You should set jvmRoute to support load-balancing via AJP ie :
        <Engine name="Catalina" defaultHost="localhost" jvmRoute="jvm1">
        -->
        <Engine name="Catalina" defaultHost="localhost">

            <!--For clustering, please take a look at documentation at:
                /docs/cluster-howto.html  (simple how to)
                /docs/config/cluster.html (reference documentation) -->
            <!--
            <Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster"/>
            -->

            <!-- Use the LockOutRealm to prevent attempts to guess user passwords
                 via a brute-force attack -->
            <Realm className="org.apache.catalina.realm.LockOutRealm">
                <!-- This Realm uses the UserDatabase configured in the global JNDI
                     resources under the key "UserDatabase".  Any edits
                     that are performed against this UserDatabase are immediately
                     available for use by the Realm.  -->
                <Realm className="org.apache.catalina.realm.UserDatabaseRealm"
                       resourceName="UserDatabase"/>
            </Realm>

            <Host name="localhost" appBase="webapps"
                  unpackWARs="true" autoDeploy="true">

                <!-- SingleSignOn valve, share authentication between web applications
                     Documentation at: /docs/config/valve.html -->
                <!--
                <Valve className="org.apache.catalina.authenticator.SingleSignOn" />
                -->

                <!-- Access log processes all example.
                     Documentation at: /docs/config/valve.html
                     Note: The pattern used is equivalent to using pattern="common" -->
                <Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"
                       prefix="localhost_access_log" suffix=".txt"
                       pattern="%h %l %u %t &quot;%r&quot; %s %b" />

            </Host>
        </Engine>
```

```
    </Service>
</Server>
```

Then, create a `frontend/tomcat/webapps/ROOT/index.html` which is a html for our frontend.

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet"
          integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3" crossorigin="anonymous">

    <title>Products</title>
</head>
<body>

<div class="container">
    <div class="row companies">
        <div class="col">
            <h1>Companies</h1>
            <div class="list">
                <div class="row">
                    <div class="col-1">
                        <label class="form-label">Id</label>
                    </div>
                    <div class="col-9">
                        <label class="form-label">Company name</label>
                    </div>
                </div>
            </div>
            <form class="new-company-form mt-4">
                <h2>Create new company</h2>
                <div class="mb-3">
                    <label for="company-name" class="form-label">Company name</label>
                    <input type="text" class="form-control" id="company-name">
                </div>
                <button type="submit" class="btn btn-primary">Create</button>
            </form>
        </div>
    </div>

    <div class="row products mt-5">
        <div class="col">
            <h1>Products</h1>
            <div class="list">
                <div class="row">
                    <div class="col-1">
                        <label class="form-label">Id</label>
                    </div>
                    <div class="col-5">
                        <label class="form-label">Product name</label>
                    </div>
                    <div class="col-2">
                        <label class="form-label">Amount</label>
                    </div>
                    <div class="col-2">
                        <label class="form-label">Company id</label>
                    </div>
                </div>
            </div>
            <form class="new-product-form mt-4">
                <h2>Create new product</h2>
                <div class="mb-3">
                    <label for="product-name" class="form-label">Product name</label>
                    <input type="text" class="form-control" id="product-name">
                </div>
                <div class="mb-3">
                    <label for="product-amount" class="form-label">Product amount</label>
                    <input type="number" min="0" value="10" class="form-control" id="product-amount">
                </div>
                <div class="mb-3">
                    <label for="product-company" class="form-label">Product company</label>
                    <select id="product-company" class="form-control">
                    </select>
                </div>
                <button type="submit" class="btn btn-primary">Create</button>
            </form>
        </div>
    </div>

    <div class="row reports mt-5">
        <div class="col">
            <h1>Reports</h1>
            <div class="list">
                <div class="row">
                    <div class="col-1">
                        <label class="form-label">Id</label>
                    </div>
                    <div class="col-5">
                        <label class="form-label">Report name</label>
```

```
                    </div>
                    <div class="col-3">
                        <label class="form-label">Status</label>
                    </div>
                </div>
            </div>
            <form class="new-report-form mt-4">
                <h2>Create new report</h2>
                <div class="mb-3">
                    <label for="report-name" class="form-label">Report name</label>
                    <input type="text" class="form-control" id="report-name">
                </div>
                <button type="submit" class="btn btn-primary">Create</button>
            </form>
        </div>
    </div>
</div>


<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"
        integrity="sha384-ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p"
        crossorigin="anonymous"></script>
<script
        src="https://code.jquery.com/jquery-3.6.0.slim.min.js"
        integrity="sha256-u7e5khyithlIdTpu22PHhENmPcRdFiHRjhAuHcs05RI="
        crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>

<script>
    const baseUrl = '/_by_name/monolith';
    const companyApiBaseUrl = `${baseUrl}/api/v1/companies`;
    const productApiBaseUrl = `${baseUrl}/api/v1/products`;
    const reportApiBaseUrl = `${baseUrl}/api/v1/reports`;

    function initCompaniesForm() {
        $(".new-company-form").submit(e => {
            e.preventDefault(e);
            addCompany($("#company-name").val());
        })
    }

    function initCompaniesList(companies) {
        companies.forEach(company => {
            const containerId = `company-${company.id}-container`;
            const elementHtml = `
                <div class="row mt-3">
                    <div id="${containerId}" class="col">
                        <div class="row">
                            <div class="col-1 d-flex align-items-center">
                                <div>${company.id}</div>
                            </div>
                            <div class="col-9 d-flex align-items-center">
                                <input type="text" value="${company.name}" class="form-control name">
                            </div>
                            <div class="col-2 d-flex align-items-center">
                                <button type="submit" class="btn btn-primary update mx-1">Update</button>
                                <button type="submit" class="btn btn-danger delete">Delete</button>
                            </div>
                        </div>
                    </div>
                </div>
            `;
            $(".companies .list").append(elementHtml)
            $(`#${containerId} .update`).on(
                    "click",
                    () => updateCompany(company.id, $(`#${containerId} .name`).val())
            );
            $(`#${containerId} .delete`).on("click", () => deleteCompany(company.id));
        })
    }

    function addCompany(name) {
        axios.post(companyApiBaseUrl, {name: name})
                .then(() => {
                    document.location.reload(true)
                })
                .catch(error => {
                    console.log(error);
                });
    }

    function updateCompany(id, name) {
        axios.put(`${companyApiBaseUrl}/${id}`, {name: name})
                .then(() => {
                    document.location.reload(true)
                })
                .catch(error => {
                    console.log(error);
                });
    }

    function deleteCompany(companyId) {
        axios.delete(`${companyApiBaseUrl}/${companyId}`)
                .then(() => {
                    document.location.reload(true)
```

```
            })
            .catch(error => {
                console.log(error);
            });
}

function initNewProductCompanyList(companies) {
    companies.forEach(company => {
        $("#product-company").append(`<option value="${company.id}">${company.name}</option>`)
    })
}

function initProductsForm() {
    $(".new-product-form").submit(e => {
        e.preventDefault(e);
        addProduct(
                $("#product-name").val(),
                $("#product-amount").val(),
                $("#product-company").val()
        );
    })
}

function addProduct(name, amount, companyId) {
    axios.post(productApiBaseUrl, {name: name, amount: amount, companyId: companyId})
            .then(() => {
                document.location.reload(true)
            })
            .catch(error => {
                console.log(error);
            });
}

function initProductsList(products, companies) {
    products.forEach(product => {
        const containerId = `product-${product.id}-container`;
        const elementHtml = `
            <div class="row mt-3">
                <div id="${containerId}" class="col">
                    <div class="row">
                        <div class="col-1 d-flex align-items-center">
                            <div>${product.id}</div>
                        </div>
                        <div class="col-5 d-flex align-items-center">
                            <input type="text" value="${product.name}" class="form-control name">
                        </div>
                        <div class="col-2 d-flex align-items-center">
                            <input type="number" min="0" value="${product.amount}" class="form-control amount">
                        </div>
                        <div class="col-2 d-flex align-items-center">
                            <select class="form-control company">
                                ${
                companies
                        .map(company =>
                                product.companyId === company.id
                                        ? `<option selected="selected" value="${company.id}">${company.name}</option>`
                                        : `<option value="${company.id}">${company.name}</option>`
                        )
                        .join("")
        }
                            </select>
                        </div>
                        <div class="col-2 d-flex align-items-center">
                            <button type="submit" class="btn btn-primary update mx-1">Update</button>
                            <button type="submit" class="btn btn-danger delete">Delete</button>
                        </div>
                    </div>
                </div>
            </div>
        `;
        $(".products .list").append(elementHtml)
        $(`#${containerId} .update`).on(
                "click",
                () => updateProduct(
                        product.id,
                        $(`#${containerId} .name`).val(),
                        $(`#${containerId} .amount`).val(),
                        $(`#${containerId} .company`).val()
                )
        );
        $(`#${containerId} .delete`).on("click", () => deleteProduct(product.id));
    })
}

function updateProduct(id, name, amount, companyId) {
    axios.put(`${productApiBaseUrl}/${id}`, {name: name, amount: amount, companyId: companyId})
            .then(() => {
                document.location.reload(true)
            })
            .catch(error => {
                console.log(error);
            });
}

function deleteProduct(productId) {
```

```
    axios.delete(`${productApiBaseUrl}/${productId}`)
            .then(() => {
                document.location.reload(true)
            })
            .catch(error => {
                console.log(error);
            });
}

function initReportsForm() {
    $(".new-report-form").submit(e => {
        e.preventDefault(e);
        addReport(
                $("#report-name").val()
        );
    })
}

function addReport(name) {
    axios.post(reportApiBaseUrl, {name: name})
            .then(() => {
                document.location.reload(true)
            })
            .catch(error => {
                console.log(error);
            });
}

function initReportsList(reports) {
    reports.forEach(report => {
        const containerId = `report-${report.id}-container`;
        const elementHtml = `
            <div class="row mt-3">
                <div id="${containerId}" class="col">
                    <div class="row">
                        <div class="col-1 d-flex align-items-center">
                            <div>${report.id}</div>
                        </div>
                        <div class="col-5 d-flex align-items-center">
                            <input type="text" value="${report.name}" class="form-control name">
                        </div>
                        <div class="col-3 d-flex align-items-center">
                            <div>${report.status}</div>
                        </div>
                        <div class="col-3 d-flex align-items-center">
                            <button type="submit" class="btn btn-secondary download mx-1">Download</button>
                            <button type="submit" class="btn btn-primary update mx-1">Update</button>
                            <button type="submit" class="btn btn-danger delete">Delete</button>
                        </div>
                    </div>
                </div>
            </div>
        `;
        $(".reports .list").append(elementHtml)
        $(`#${containerId} .download`).on("click", () => window.location.href = `${reportApiBaseUrl}/${report.id}`);
        $(`#${containerId} .update`).on(
                "click",
                () => updateReport(report.id, $(`#${containerId} .name`).val())
        );
        $(`#${containerId} .delete`).on("click", () => deleteReport(report.id));
    })
}

function updateReport(id, name) {
    axios.put(`${reportApiBaseUrl}/${id}`, {name: name})
            .then(() => {
                document.location.reload(true)
            })
            .catch(error => {
                console.log(error);
            });
}

function deleteReport(id) {
    axios.delete(`${reportApiBaseUrl}/${id}`)
            .then(() => {
                document.location.reload(true)
            })
            .catch(error => {
                console.log(error);
            });
}

$(document).ready(function () {
    initCompaniesForm();

    axios.get(companyApiBaseUrl)
            .then(response => {
                const companies = response.data.sort((a, b) => a.id - b.id);
                initCompaniesList(companies);
                initNewProductCompanyList(companies);
                initProductsForm();
                axios.get(productApiBaseUrl)
                        .then(response => {
                            const products = response.data.sort((a, b) => a.id - b.id);
```

```
                            initProductsList(products, companies);
                        })
                        .catch(error => {
                            console.log(error);
                        })
                })
                .catch(error => {
                    console.log(error);
                })

        initReportsForm();
        axios.get(reportApiBaseUrl)
                .then(response => {
                    const reports = response.data.sort((a, b) => a.id - b.id);
                    initReportsList(reports);
                })
                .catch(error => {
                    console.log(error);
                })
    });
</script>

</body>
</html>
```

Let's create a `frontend/onteon/distribution.yml` file for frontend.

```
application: frontend:1.0.0
numberOfInstances: 2
scripts:
  checkIfNodeCanAcceptNewApplicationInstance: defaultAvailableNodeOnlyCINCANAIV1
  selectNodeForNewApplicationInstance: defaultApplicationInstancesCountOnlySNFNAIV1
  selectApplicationInstanceToRemove: defaultApplicationInstancesCountOnlySAITRV1
```

Finally, let's execute `frontend/scripts/build.sh` script, upload built package and create distribution.

```
$ frontend/scripts/build.sh
$ onteoncli application-registry upload frontend/target/frontend.tar.gz
Starting upload session...
Sending part no 0...
Sending part no 1...
Finishing upload session...
uploaded: true

$ onteoncli distribution create-from-file frontend/onteon/distribution.yml
id:                                          624ae0428fa5e24f38c400ed
createdAT:                                   2022-04-04T12:10:42.037Z
updatedAt:                                   2022-04-04T12:10:42.037Z
applicationId:                               624ae0268fa5e24f38c400ce
numberOfInstances:                           2
checkIfNodeCanAcceptNewApplicationInstanceScriptId: 624ada7b8fa5e24f38c3fb35
selectNodeForNewApplicationInstanceScriptId:        624ada7b8fa5e24f38c3fb31
selectApplicationInstanceToRemoveScriptId:          624ada7b8fa5e24f38c3fb33

$ watch onteoncli application-instance list

id                       createdAt               applicationName applicationVersion applicationType applicationId            nodeId                   status
66ad936502d57141d6b8bc38 2022-04-04T12:11:02.837Z frontend        1.0.0              standard        624ae0268fa5e24f38c400ce af7d3125ac011c589f71ff89 running
a699f0807c3ab9efa55df209 2022-04-04T12:10:52.855Z frontend        1.0.0              standard        624ae0268fa5e24f38c400ce ad9c9125652890d8002ed8d3 running
554c361ead539ada06482108 2022-04-04T11:47:26.53Z  monolith        1.0.0              standard        624adaae8fa5e24f38c3fb5f af7d3125ac011c589f71ff89 running
c51066e7821383a483501b26 2022-04-04T11:47:14.231Z monolith        1.0.0              standard        624adaae8fa5e24f38c3fb5f ad9c9125652890d8002ed8d3 running
a582a6c4851e1116774ab8f9 2022-04-04T11:46:19.338Z nginx-edge      1.1.3              embedded        0a1769f83abc3224fe6cb477 ad9c9125652890d8002ed8d3 running
b904b4d94e010d485f2552dd 2022-04-04T11:46:19.335Z nginx-edge      1.1.3              embedded        0a1769f83abc3224fe6cb477 af7d3125ac011c589f71ff89 running
67590baa82e4e5d9c44aed9f 2022-04-04T11:46:13.155Z nginx-inner     1.1.3              embedded        157958ad0cdf14266501ad70 ad9c9125652890d8002ed8d3 running
967cff6e752fa2b8ca86573f 2022-04-04T11:46:13.152Z nginx-inner     1.1.3              embedded        157958ad0cdf14266501ad70 af7d3125ac011c589f71ff89 running
```

Go to http://localhost:8020/_by_name/frontend/ to see if it works.

Monolith

Now we need to remove frontend from monolith.

Removed useless files and dependencies

First, let's remove the:

- monolith/src/main/resources/templates/index.html file
- monolith/src/main/java/com/example/monolith/frontend package

Then go to pom.xml, and remove some Thymeleaf dependency:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.6.5</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>

    <groupId>com.example</groupId>
    <artifactId>monolith</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>monolith</name>
    <description>monolith</description>
    <packaging>war</packaging>

    <properties>
        <java.version>11</java.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-validation</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-tomcat</artifactId>
            <scope>provided</scope>
        </dependency>

        <dependency>
            <groupId>com.github.librepdf</groupId>
            <artifactId>openpdf</artifactId>
            <version>1.3.27</version>
        </dependency>

        <dependency>
            <groupId>org.postgresql</groupId>
            <artifactId>postgresql</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <finalName>${artifactId}</finalName>

        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <configuration>
                    <excludes>
                        <exclude>
                            <groupId>org.projectlombok</groupId>
```

```
                        <artifactId>lombok</artifactId>
                    </exclude>
                </excludes>
            </configuration>
        </plugin>
    </plugins>
</build>
</project>
```

IsAliveController

Since monolith don't serve a frontend, we need to add some endpoint that will determine if application is alive.

Let's create `monolith/src/main/java/com/example/monolith/isalive/controller/IsAliveController.java` controller with one endpoint, that will return nothing.

```
package com.example.monolith.isalive.controller;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

/**
 * @author Patryk Borchowiec
 */
@RestController
public class IsAliveController {
    @GetMapping("/is-alive")
    public void isAlive() {

    }
}
```

Modify configuration

Now, let's change the version and health check url of monolith application in `conf.yml`. Also make sure that conf.yml has correct path: `monolith/onteon/conf.yml`.

```
app:
  name: 'monolith'
  version: '1.1.0'
  appType: 'standard'
  procType: 'native'
  processProvider:
    name: 'GenericOsProcessProviderImpl'
    version: '1.0.0'
    executable:
      start:
        command: '${ont_app_path}/bin/bin/catalina.sh run'
        successLine: 'Server startup'
      stop:
        command: '${ont_app_path}/bin/bin/catalina.sh stop'
      terminate:
        command: '${ont_app_path}/bin/bin/catalina.sh stop -force'
  placeHolder:
    name: 'PlaceHolderManagerImpl'
    version: '1.0.0'
    filesToReplace:
      - '${ont_app_path}/bin/conf/server.xml'
    variables:
  serviceRepository:
    healthCheckUrl: 'http://localhost:${ont_port_2}/is-alive'
    entities:
      - entity:
          priority: 1
          port: ${ont_port_2}
          protocol:
            type: 'HTTP'
            version: '1.1'
          isExternal: true
          isInternal: true
```

Application Package

Here we will also use a script for building application package. Let's modify the `monolith/scripts/build.sh` script.

```
#!/usr/bin/env bash

set -e

SOURCE="${BASH_SOURCE[0]}"
while [ -h "$SOURCE" ]; do
  DIR="$(cd -P "$( dirname "$SOURCE" )" >/dev/null 2>&1 && pwd)"
  SOURCE="$(readlink "$SOURCE")"
  [[ $SOURCE != /* ]] && SOURCE="$DIR/$SOURCE"
done
SCRIPT_DIR="$(cd -P "$(dirname "$SOURCE")" >/dev/null 2>&1 && pwd)"
ROOT_DIR="${SCRIPT_DIR}/.."
TARGET_DIR="${ROOT_DIR}/target"
DOWNLOAD_DIR="${ROOT_DIR}/download"
TOMCAT_SOURCE="${DOWNLOAD_DIR}/tomcat.tar.gz"
TOMCAT_DOWNLOAD_URL="http://archive.apache.org/dist/tomcat/tomcat-9/v9.0.60/bin/apache-tomcat-9.0.60.tar.gz"
TOMCAT_DIR_NAME="apache-tomcat-9.0.60"
APP_BIN_DIR="${TARGET_DIR}/app/bin"
APP_CONF_DIR="${TARGET_DIR}/app/conf"


mkdir -p "${DOWNLOAD_DIR}"
if [ ! -f "${TOMCAT_SOURCE}" ]; then
  wget -O "${TOMCAT_SOURCE}" "${TOMCAT_DOWNLOAD_URL}"
fi

cd "${ROOT_DIR}" && mvn clean package
tar -xvf "${TOMCAT_SOURCE}" -C "${TARGET_DIR}"
mkdir -p "${APP_BIN_DIR}"
mkdir -p "${APP_CONF_DIR}"
cp -r "${TARGET_DIR}/${TOMCAT_DIR_NAME}/"* "${APP_BIN_DIR}"
rm -rf "${APP_BIN_DIR}/webapps/"*
cp -r "${TARGET_DIR}/monolith.war" "${APP_BIN_DIR}/webapps/ROOT.war"
cp "${ROOT_DIR}/onteon/conf.yml" "${APP_CONF_DIR}"
cp -r "${ROOT_DIR}/tomcat/"* "${APP_BIN_DIR}"
cd "${TARGET_DIR}/app" && tar -zcvf "${TARGET_DIR}/monolith.tar.gz" *
```

Let's build and upload application package.

```
$ ./monolith/scripts/build.sh
$ onteoncli application-registry upload monolith/target/monolith.tar.gz
$ onteoncli application list
id                      createdAt               updatedAt               name     version type     processType
624ae4b48fa5e24f38c4071e 2022-04-04T12:29:40.566Z 2022-04-04T12:29:40.566Z monolith 1.1.0   standard native
624ae0268fa5e24f38c400ce 2022-04-04T12:10:14.809Z 2022-04-04T12:10:14.809Z frontend 1.0.0   standard native
624adaae8fa5e24f38c3fb5f 2022-04-04T11:46:54.782Z 2022-04-04T11:46:54.782Z monolith 1.0.0   standard native
```

Our application is uploaded.

Upgrade

Now you can upgrade application. First, list all applications and choose the id of application that you want to replace, and id of application that will replace the old one. Then, you need to execute `onteoncli application-change create <old-app-id> <new-app-id>` And then, you can check how it automatically replaces old application with new one.

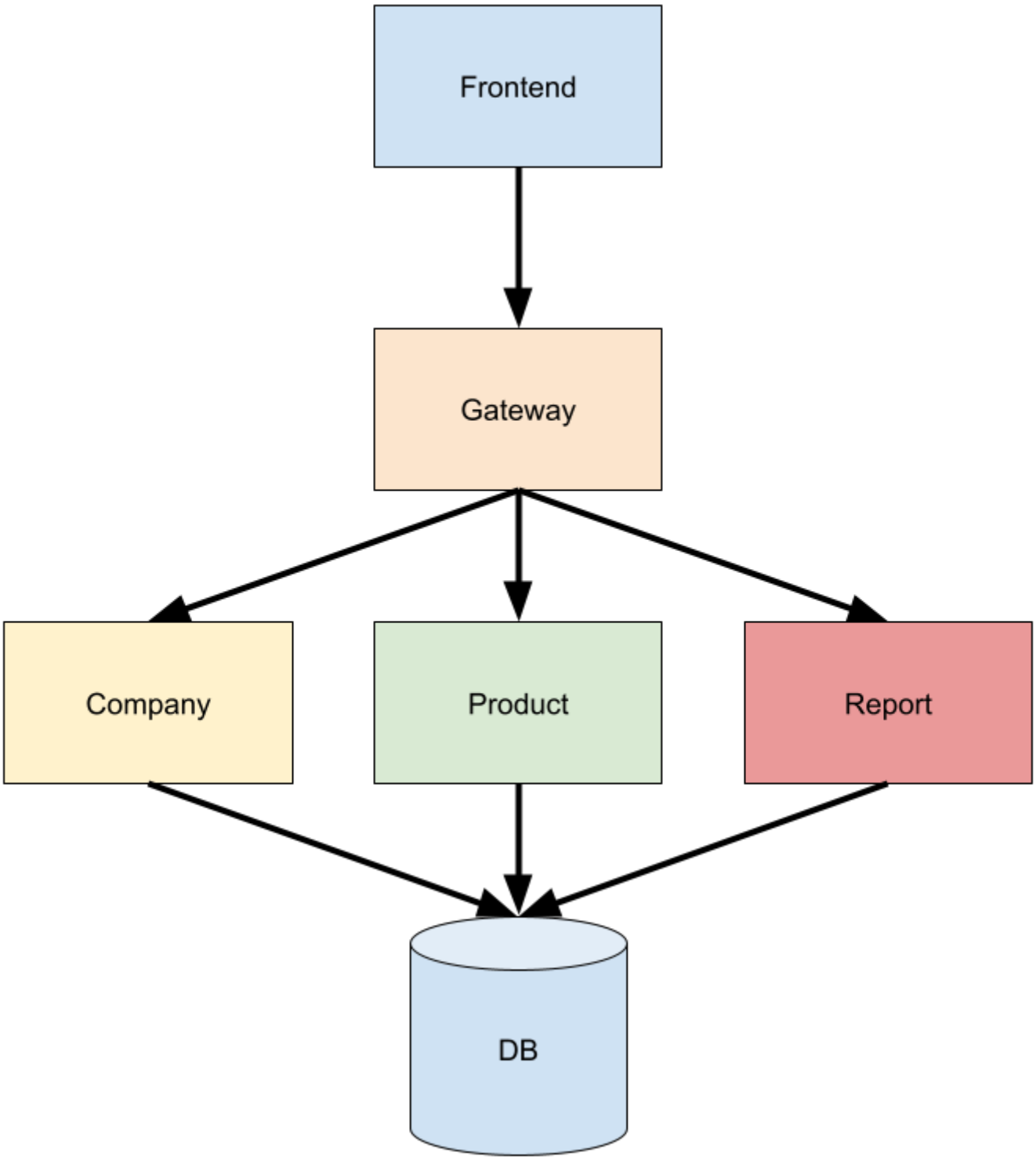
```
monolith/app$ onteoncli application list
id                      createdAt               updatedAt               name     version type     processType
624ae4b48fa5e24f38c4071e 2022-04-04T12:29:40.566Z 2022-04-04T12:29:40.566Z monolith 1.1.0   standard native
624ae0268fa5e24f38c400ce 2022-04-04T12:10:14.809Z 2022-04-04T12:10:14.809Z frontend 1.0.0   standard native
624adaae8fa5e24f38c3fb5f 2022-04-04T11:46:54.782Z 2022-04-04T11:46:54.782Z monolith 1.0.0   standard native

monolith/app$ onteoncli application-change create 624adaae8fa5e24f38c3fb5f 624ae4b48fa5e24f38c4071e
id:               624ae5578fa5e24f38c40803
createdAt:        2022-04-04T12:32:23.755Z
updatedAt:        2022-04-04T12:32:23.755Z
oldApplicationId: 624adaae8fa5e24f38c3fb5f
newApplicationId: 624ae4b48fa5e24f38c4071e

monolith/app$ onteoncli application-instance list
```

```
id                       createdAt               applicationName applicationVersion applicationType applicationId            nodeId                   status
7fb6e61dc85560468fc181dd 2022-04-04T12:33:31.665Z monolith        1.1.0              standard        624ae4b48fa5e24f38c4071e af7d3125ac011c589f71ff89 running
dccc6c79c1ce495ab85163f5 2022-04-04T12:32:33.337Z monolith        1.1.0              standard        624ae4b48fa5e24f38c4071e ad9c9125652890d8002ed8d3 running
66ad936502d57141d6b8bc38 2022-04-04T12:11:02.837Z frontend        1.0.0              standard        624ae0268fa5e24f38c400ce af7d3125ac011c589f71ff89 running
a699f0807c3ab9efa55df209 2022-04-04T12:10:52.855Z frontend        1.0.0              standard        624ae0268fa5e24f38c400ce ad9c9125652890d8002ed8d3 running
a582a6c4851e1116774ab8f9 2022-04-04T11:46:19.338Z nginx-edge      1.1.3              embedded        0a1769f83abc3224fe6cb477 ad9c9125652890d8002ed8d3 running
b904b4d94e010d485f2552dd 2022-04-04T11:46:19.335Z nginx-edge      1.1.3              embedded        0a1769f83abc3224fe6cb477 af7d3125ac011c589f71ff89 running
67590baa82e4e5d9c44aed9f 2022-04-04T11:46:13.155Z nginx-inner     1.1.3              embedded        157958ad0cdf14266501ad70 ad9c9125652890d8002ed8d3 running
967cff6e752fa2b8ca86573f 2022-04-04T11:46:13.152Z nginx-inner     1.1.3              embedded        157958ad0cdf14266501ad70 af7d3125ac011c589f71ff89 running
```

Finally, go to http://localhost:8020/_by_name/frontend/ and see if it works.

Step 3 - Dividing monolith into microservices

In this step we will split monolith into microservices. So far, microservices will not communicate with each other. They will use database to gather all data.



Files

If you don't have files from previous step, you can:

- download **initial** files from here
- download **complete** files from here

Company

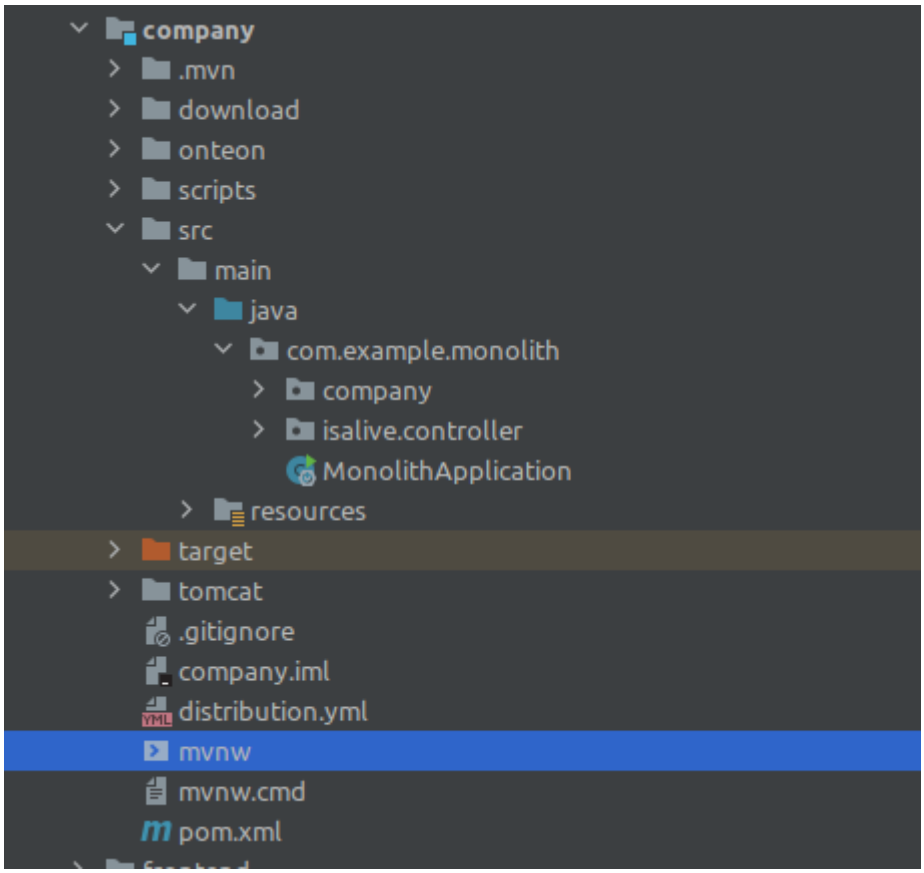
First, copy the `monolith` directory and rename it to `company`.

Then go to pom.xml, change names and remove openpdf dependency (it is needed only by report microservice).

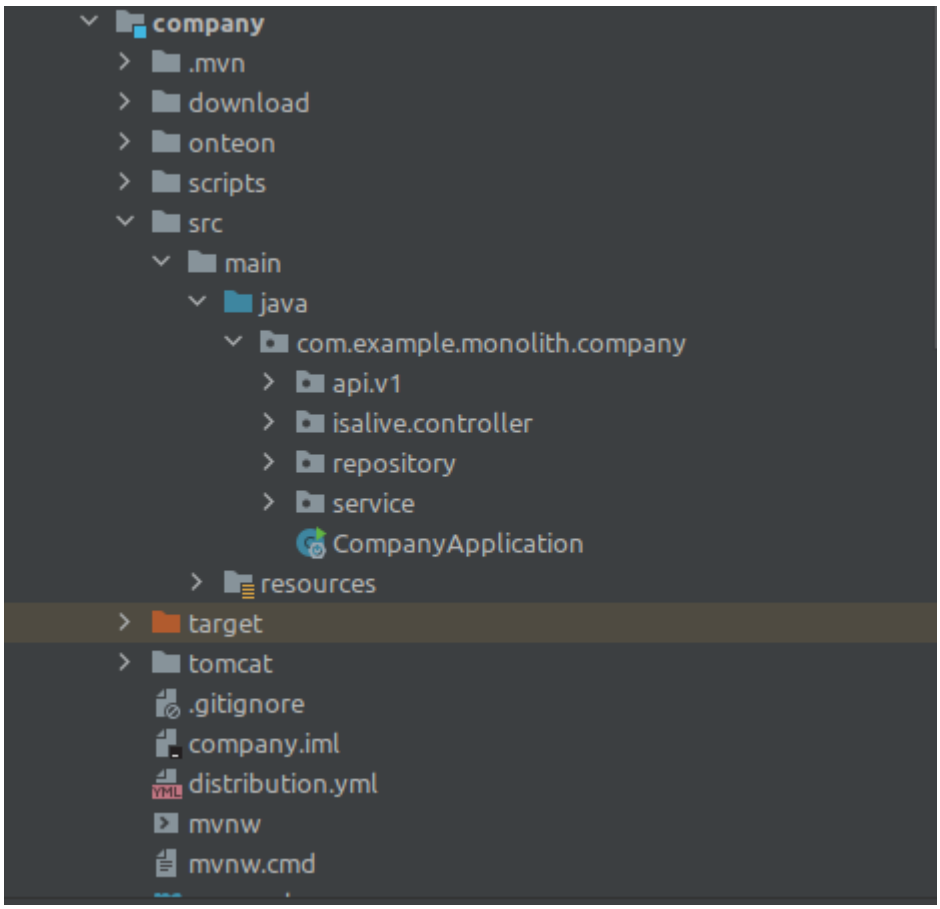
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.6.5</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>

    <groupId>com.example</groupId>
    <artifactId>company</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>company</name>
    <description>company</description>
    <packaging>war</packaging>

    <properties>
        <java.version>11</java.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-validation</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-tomcat</artifactId>
            <scope>provided</scope>
        </dependency>

        <dependency>
            <groupId>org.postgresql</groupId>
            <artifactId>postgresql</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <finalName>${artifactId}</finalName>

        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
```

```
                <configuration>
                    <excludes>
                        <exclude>
                            <groupId>org.projectlombok</groupId>
                            <artifactId>lombok</artifactId>
                        </exclude>
                    </excludes>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

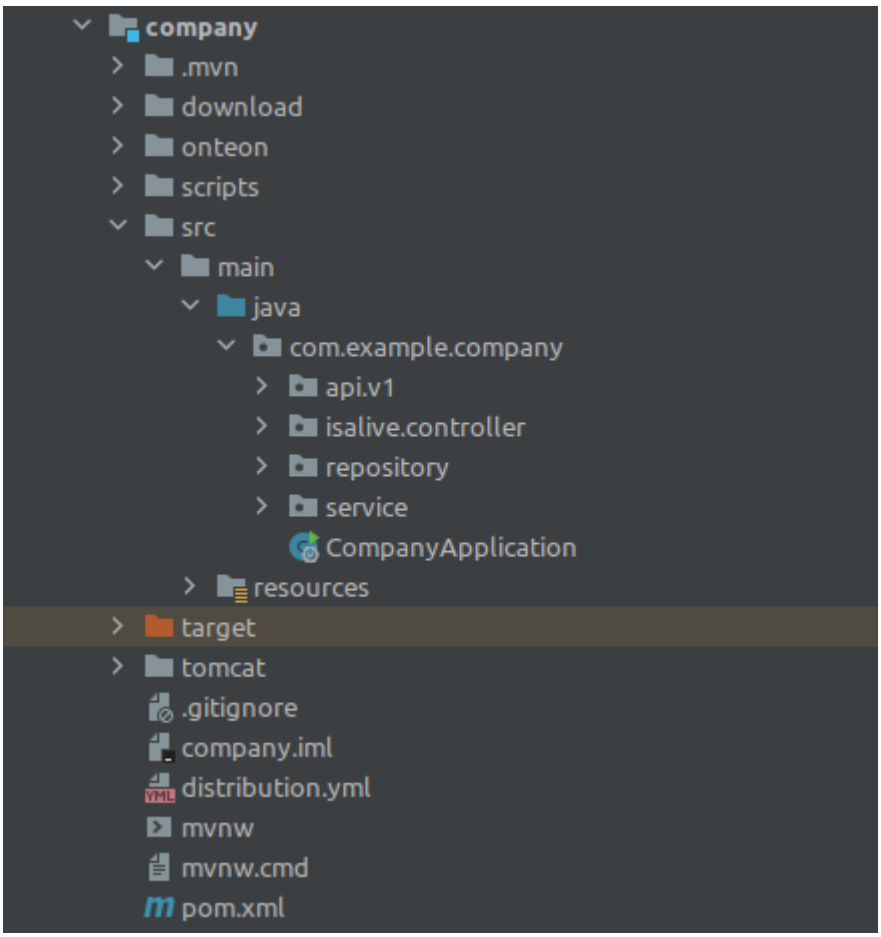
Next step is to remove the `product` and `report` packages. The packages should look like this.



Now, rename `MonolithApplication` to `CompanyApplication` and, move `CompanyApplication` and `isalive` package to `company` package.



Finally, rename `com.example.monolith.company` package to `com.example.company`.



Onteon Configuration

Let's modify `company/onteon/conf.yml` file.

```
app:
  name: 'company'
  version: '1.0.0'
  appType: 'standard'
  procType: 'native'
  processProvider:
    name: 'GenericOsProcessProviderImpl'
    version: '1.0.0'
    executable:
      start:
        command: '${ont_app_path}/bin/bin/catalina.sh run'
        successLine: 'Server startup'
      stop:
        command: '${ont_app_path}/bin/bin/catalina.sh stop'
      terminate:
        command: '${ont_app_path}/bin/bin/catalina.sh stop -force'
  placeHolder:
    name: 'PlaceHolderManagerImpl'
    version: '1.0.0'
    filesToReplace:
      - '${ont_app_path}/bin/conf/server.xml'
    variables:
  serviceRepository:
    healthCheckUrl: 'http://localhost:${ont_port_2}/is-alive'
    entities:
      - entity:
          priority: 1
          port: ${ont_port_2}
          protocol:
            type: 'HTTP'
            version: '1.1'
          isExternal: false
          isInternal: true
```

Let's see the serviceRepository section. As you can see the `isExternal` property is set to false. It means that, application instance will not be accessible in the external port. It will be available only for internal communication.

Distribution

Create `company/onteon/distribution.yml`.

```
application: company:1.0.0
numberOfInstances: 2
scripts:
  checkIfNodeCanAcceptNewApplicationInstance: defaultAvailableNodeOnlyCINCANAIV1
  selectNodeForNewApplicationInstance: defaultApplicationInstancesCountOnlySNFNAIV1
  selectApplicationInstanceToRemove: defaultApplicationInstancesCountOnlySAITRV1
```

Building with script

Let's modify a `company/scripts/build.sh` script.

```
#!/usr/bin/env bash

set -e

SOURCE="${BASH_SOURCE[0]}"
while [ -h "$SOURCE" ]; do
  DIR="$(cd -P "$( dirname "$SOURCE" )" >/dev/null 2>&1 && pwd)"
  SOURCE="$(readlink "$SOURCE")"
  [[ $SOURCE != /* ]] && SOURCE="$DIR/$SOURCE"
done
SCRIPT_DIR="$(cd -P "$(dirname "$SOURCE")" >/dev/null 2>&1 && pwd)"
ROOT_DIR="${SCRIPT_DIR}/.."
TARGET_DIR="${ROOT_DIR}/target"
DOWNLOAD_DIR="${ROOT_DIR}/download"
TOMCAT_SOURCE="${DOWNLOAD_DIR}/tomcat.tar.gz"
TOMCAT_DOWNLOAD_URL="http://archive.apache.org/dist/tomcat/tomcat-9/v9.0.60/bin/apache-tomcat-9.0.60.tar.gz"
TOMCAT_DIR_NAME="apache-tomcat-9.0.60"
APP_BIN_DIR="${TARGET_DIR}/app/bin"
APP_CONF_DIR="${TARGET_DIR}/app/conf"


mkdir -p "${DOWNLOAD_DIR}"
if [ ! -f "${TOMCAT_SOURCE}" ]; then
  wget -O "${TOMCAT_SOURCE}" "${TOMCAT_DOWNLOAD_URL}"
fi

cd "${ROOT_DIR}" && mvn clean package
tar -xvf "${TOMCAT_SOURCE}" -C "${TARGET_DIR}"
mkdir -p "${APP_BIN_DIR}"
mkdir -p "${APP_CONF_DIR}"
cp -r "${TARGET_DIR}/${TOMCAT_DIR_NAME}/"* "${APP_BIN_DIR}"
rm -rf "${APP_BIN_DIR}/webapps/"*
cp -r "${TARGET_DIR}/company.war" "${APP_BIN_DIR}/webapps/ROOT.war"
cp "${ROOT_DIR}/onteon/conf.yml" "${APP_CONF_DIR}"
cp -r "${ROOT_DIR}/tomcat/"* "${APP_BIN_DIR}"
cd "${TARGET_DIR}/app" && tar -zcvf "${TARGET_DIR}/company.tar.gz" *
```

and now you can run script. The package can be found in target directory.

```
$ ./company/scripts/build.sh
$ ls company/target/company.tar.gz
```

Start application

Let's upload application and create distribution.

```
$ onteoncli application-registry upload company/target/company.tar.gz
Starting upload session...
Sending part no 0...
Sending part no 1...
Sending part no 2...
Sending part no 3...
```

```
Sending part no 4...
Sending part no 5...
Sending part no 6...
Sending part no 7...
Sending part no 8...
Finishing upload session...
uploaded: true

$ onteoncli distribution create-from-file company/onteon/distribution.yml
id:                                          624af1868fa5e24f38c4192b
createdAT:                                   2022-04-04T13:24:22.004Z
updatedAt:                                   2022-04-04T13:24:22.004Z
applicationId:                               624af1418fa5e24f38c418c8
numberOfInstances:                           2
checkIfNodeCanAcceptNewApplicationInstanceScriptId: 624ada7b8fa5e24f38c3fb35
selectNodeForNewApplicationInstanceScriptId:        624ada7b8fa5e24f38c3fb31
selectApplicationInstanceToRemoveScriptId:          624ada7b8fa5e24f38c3fb33


$ watch onteoncli application-instance list
id                       createdAt               applicationName applicationVersion applicationType applicationId            nodeId                   status
c3533175b60794a9cba93fbe 2022-04-04T13:24:45.401Z company         1.0.0              standard        624af1418fa5e24f38c418c8 af7d3125ac011c589f71ff89 running
2755d475d46ec64a5019752d 2022-04-04T13:24:33.334Z company         1.0.0              standard        624af1418fa5e24f38c418c8 ad9c9125652890d8002ed8d3 running
7fb6e61dc85560468fc181dd 2022-04-04T12:33:31.665Z monolith        1.1.0              standard        624ae4b48fa5e24f38c4071e af7d3125ac011c589f71ff89 running
dccc6c79c1ce495ab85163f5 2022-04-04T12:32:33.337Z monolith        1.1.0              standard        624ae4b48fa5e24f38c4071e ad9c9125652890d8002ed8d3 running
66ad936502d57141d6b8bc38 2022-04-04T12:11:02.837Z frontend        1.0.0              standard        624ae0268fa5e24f38c400ce af7d3125ac011c589f71ff89 running
a699f0807c3ab9efa55df209 2022-04-04T12:10:52.855Z frontend        1.0.0              standard        624ae0268fa5e24f38c400ce ad9c9125652890d8002ed8d3 running
a582a6c4851e1116774ab8f9 2022-04-04T11:46:19.338Z nginx-edge      1.1.3              embedded        0a1769f83abc3224fe6cb477 ad9c9125652890d8002ed8d3 running
b904b4d94e010d485f2552dd 2022-04-04T11:46:19.335Z nginx-edge      1.1.3              embedded        0a1769f83abc3224fe6cb477 af7d3125ac011c589f71ff89 running
67590baa82e4e5d9c44aed9f 2022-04-04T11:46:13.155Z nginx-inner     1.1.3              embedded        157958ad0cdf14266501ad70 ad9c9125652890d8002ed8d3 running
967cff6e752fa2b8ca86573f 2022-04-04T11:46:13.152Z nginx-inner     1.1.3              embedded        157958ad0cdf14266501ad70 af7d3125ac011c589f71ff89 running
```

Product

First, copy the `monolith` directory and rename it to `product`.

Then go to pom.xml, change names and remove openpdf dependency (it is needed only by report microservice).

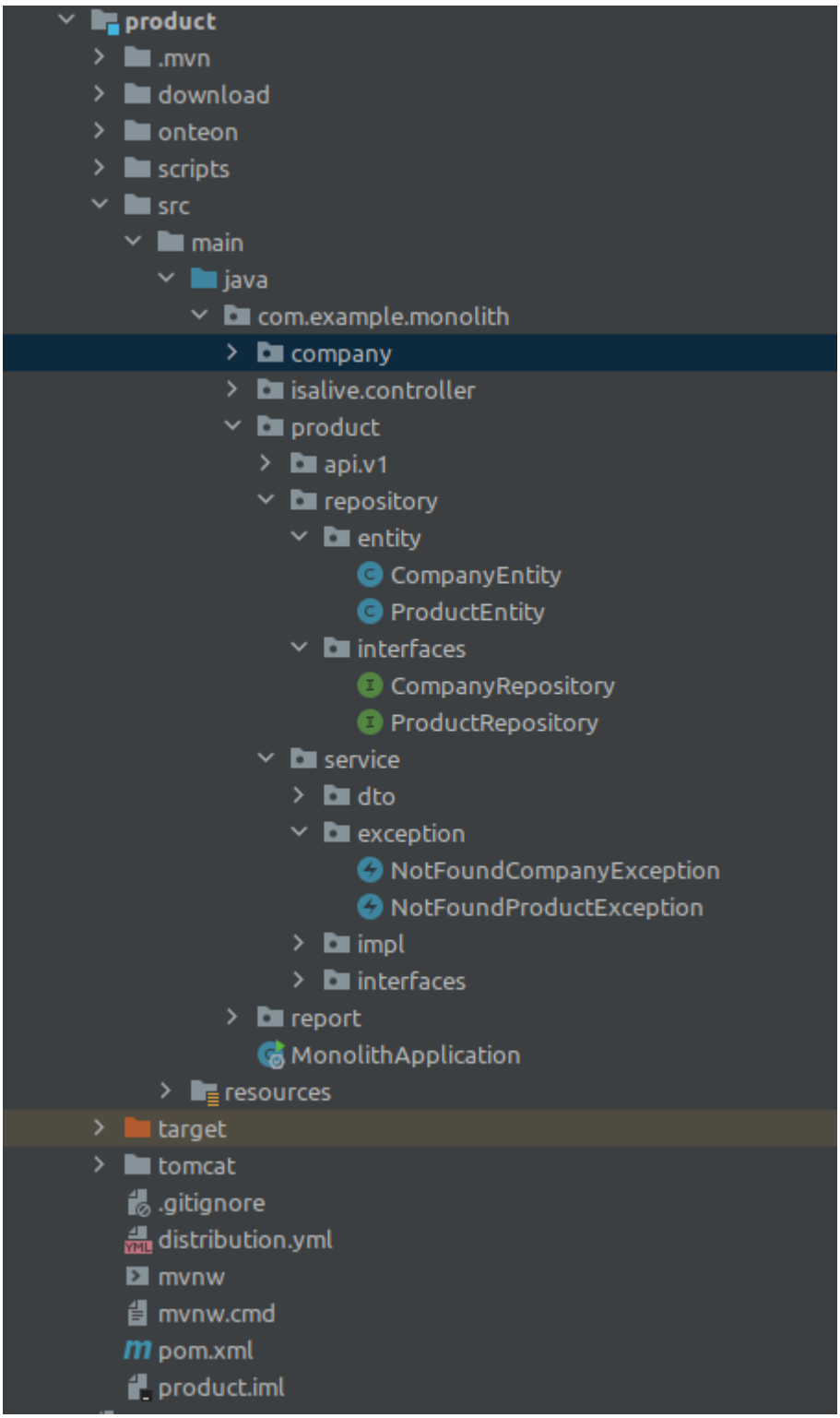
```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.6.5</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>

    <groupId>com.example</groupId>
    <artifactId>product</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>product</name>
    <description>product</description>
    <packaging>war</packaging>

    <properties>
        <java.version>11</java.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-validation</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-tomcat</artifactId>
            <scope>provided</scope>
        </dependency>

        <dependency>
            <groupId>org.postgresql</groupId>
            <artifactId>postgresql</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
```

```
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <finalName>${artifactId}</finalName>

        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <configuration>
                    <excludes>
                        <exclude>
                            <groupId>org.projectlombok</groupId>
                            <artifactId>lombok</artifactId>
                        </exclude>
                    </excludes>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

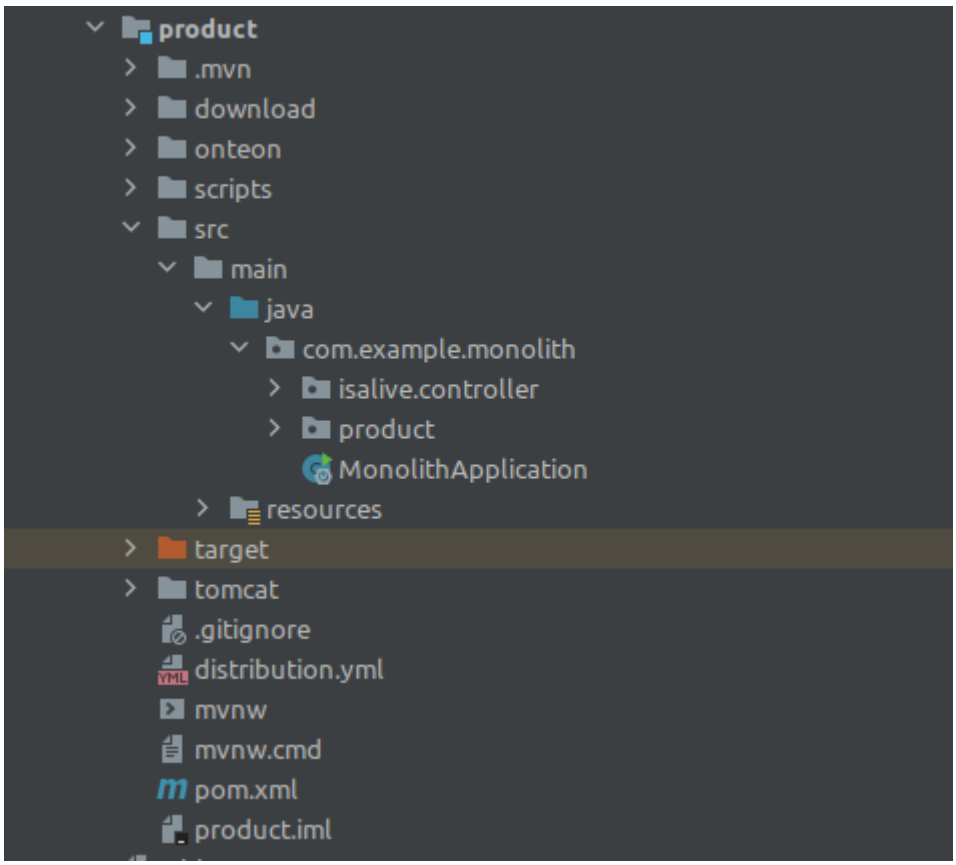
Now, let's open the `ProductServiceImpl`, and check if there are imports from other packages than `product`.

```
package com.example.monolith.product.service.impl;

import com.example.monolith.company.repository.entity.CompanyEntity;
import com.example.monolith.company.repository.interfaces.CompanyRepository;
import com.example.monolith.company.service.exception.NotFoundCompanyException;
import com.example.monolith.product.repository.entity.ProductEntity;
import com.example.monolith.product.repository.interfaces.ProductRepository;
import com.example.monolith.product.service.dto.ProductDTO;
import com.example.monolith.product.service.dto.converter.ProductConverter;
import com.example.monolith.product.service.exception.NotFoundProductException;
import com.example.monolith.product.service.interfaces.ProductService;
import lombok.NonNull;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.stream.Collectors;

/**
 * @author Patryk Borchowiec
 */
@Service
public class ProductServiceImpl implements ProductService {
    private final ProductRepository productRepository;
    private final CompanyRepository companyRepository;
    private final ProductConverter productConverter;

    public ProductServiceImpl(
            final ProductRepository productRepository,
            final CompanyRepository companyRepository,
            final ProductConverter productConverter
    ) {
        this.productRepository = productRepository;
        this.companyRepository = companyRepository;
        this.productConverter = productConverter;
    }
```

As you can see, ProductServiceImpl uses CompanyEntity, CompanyRepository, NotFoundCompanyException from `company` package. We need to copy these classes to product package. Product package should look like this:

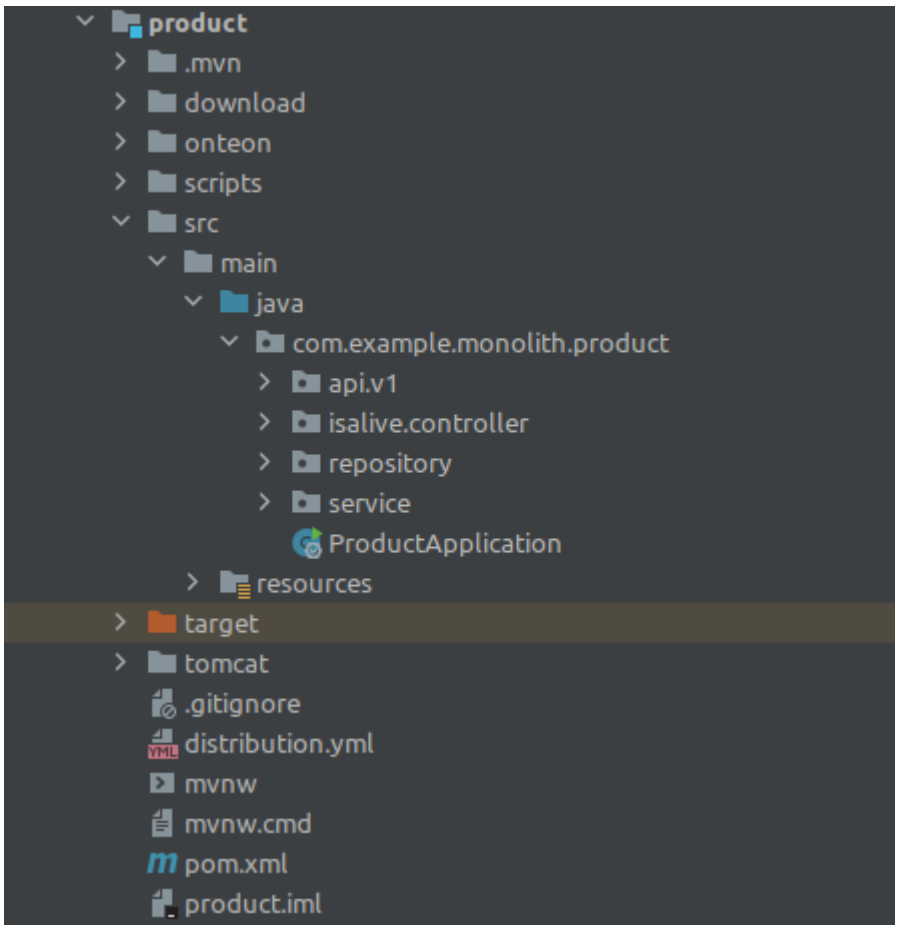


Next step is to remove the `company` and `report` packages. The packages should look like this:

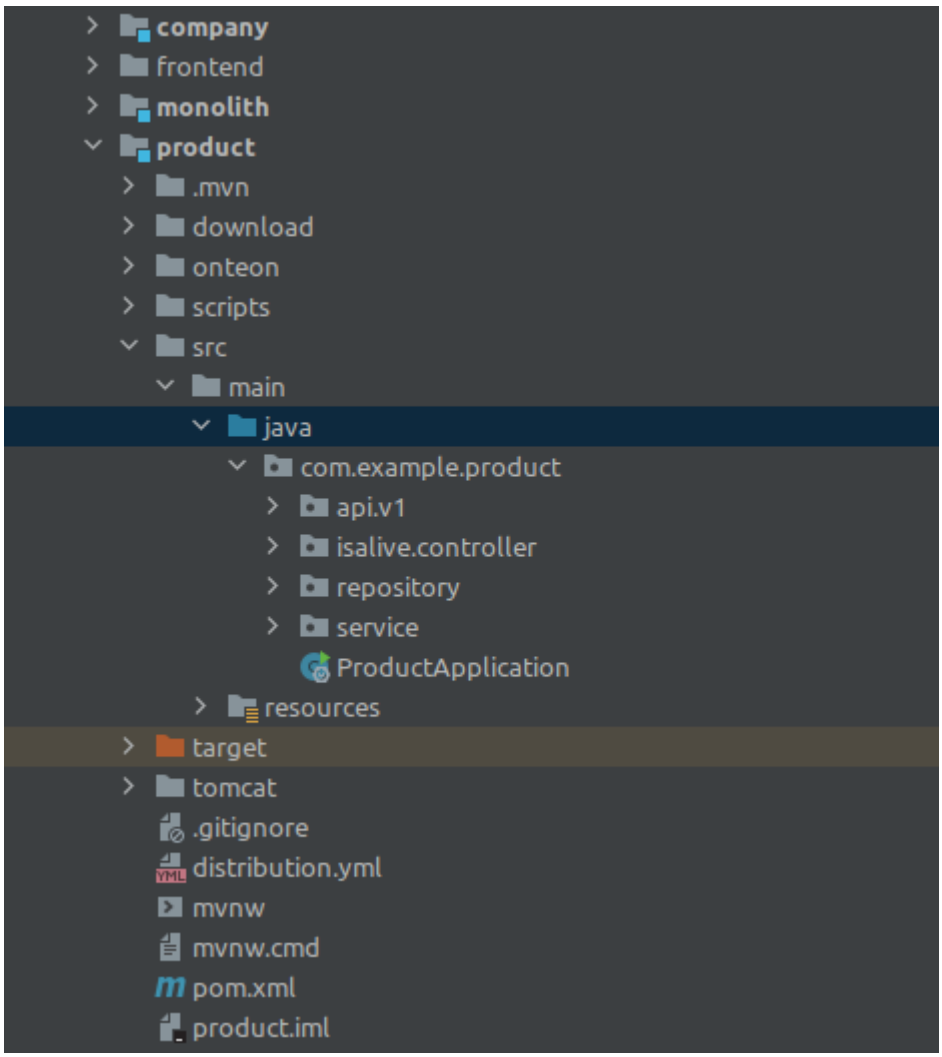


Here you can run `mvn clean package` to make sure that product project has everything.

Now, rename `MonolithApplication` to `ProductApplication` and, move `ProductApplication` and `isalive` package to `product` package.



Finally, rename `com.example.monolith.product` package to `com.example.product`.



Onteon Configuration

Let's create `product/onteon/conf.yml` file.

```
app:
  name: 'product'
  version: '1.0.0'
  appType: 'standard'
  procType: 'native'
  processProvider:
    name: 'GenericOsProcessProviderImpl'
    version: '1.0.0'
    executable:
      start:
        command: '${ont_app_path}/bin/bin/catalina.sh run'
        successLine: 'Server startup'
      stop:
        command: '${ont_app_path}/bin/bin/catalina.sh stop'
      terminate:
        command: '${ont_app_path}/bin/bin/catalina.sh stop -force'
  placeHolder:
    name: 'PlaceHolderManagerImpl'
    version: '1.0.0'
    filesToReplace:
      - '${ont_app_path}/bin/conf/server.xml'
    variables:
  serviceRepository:
    healthCheckUrl: 'http://localhost:${ont_port_2}/is-alive'
    entities:
      - entity:
          priority: 1
          port: ${ont_port_2}
          protocol:
            type: 'HTTP'
```

```
      version: '1.1'
    isExternal: false
    isInternal: true
```

The configuration is pretty similar to the previous one. This application will also be available only in internal network.

Distribution

Create `product/onteon/distribution.yml`.

```
application: product:1.0.0
numberOfInstances: 2
scripts:
  checkIfNodeCanAcceptNewApplicationInstance: defaultAvailableNodeOnlyCINCANAIV1
  selectNodeForNewApplicationInstance: defaultApplicationInstancesCountOnlySNFNAIV1
  selectApplicationInstanceToRemove: defaultApplicationInstancesCountOnlySAITRV1
```

Building with script

Let's modify a `product/scripts/build.sh` script.

```
#!/usr/bin/env bash

set -e

SOURCE="${BASH_SOURCE[0]}"
while [ -h "$SOURCE" ]; do
  DIR="$(cd -P "$( dirname "$SOURCE" )" >/dev/null 2>&1 && pwd)"
  SOURCE="$(readlink "$SOURCE")"
  [[ $SOURCE != /* ]] && SOURCE="$DIR/$SOURCE"
done
SCRIPT_DIR="$(cd -P "$(dirname "$SOURCE")" >/dev/null 2>&1 && pwd)"
ROOT_DIR="${SCRIPT_DIR}/.."
TARGET_DIR="${ROOT_DIR}/target"
DOWNLOAD_DIR="${ROOT_DIR}/download"
TOMCAT_SOURCE="${DOWNLOAD_DIR}/tomcat.tar.gz"
TOMCAT_DOWNLOAD_URL="http://archive.apache.org/dist/tomcat/tomcat-9/v9.0.60/bin/apache-tomcat-9.0.60.tar.gz"
TOMCAT_DIR_NAME="apache-tomcat-9.0.60"
APP_BIN_DIR="${TARGET_DIR}/app/bin"
APP_CONF_DIR="${TARGET_DIR}/app/conf"


mkdir -p "${DOWNLOAD_DIR}"
if [ ! -f "${TOMCAT_SOURCE}" ]; then
  wget -O "${TOMCAT_SOURCE}" "${TOMCAT_DOWNLOAD_URL}"
fi

cd "${ROOT_DIR}" && mvn clean package
tar -xvf "${TOMCAT_SOURCE}" -C "${TARGET_DIR}"
mkdir -p "${APP_BIN_DIR}"
mkdir -p "${APP_CONF_DIR}"
cp -r "${TARGET_DIR}/${TOMCAT_DIR_NAME}/"* "${APP_BIN_DIR}"
rm -rf "${APP_BIN_DIR}/webapps/"*
cp -r "${TARGET_DIR}/product.war" "${APP_BIN_DIR}/webapps/ROOT.war"
cp "${ROOT_DIR}/onteon/conf.yml" "${APP_CONF_DIR}"
cp -r "${ROOT_DIR}/tomcat/"* "${APP_BIN_DIR}"
cd "${TARGET_DIR}/app" && tar -zcvf "${TARGET_DIR}/product.tar.gz" *
```

and now you can run script. The package can be found in target directory.

```
$ ./product/scripts/build.sh
$ ls product/target/product.tar.gz
```

Start application

Let's upload application and create distribution.

```
$ onteoncli application-registry upload product/target/product.tar.gz
Starting upload session...
Sending part no 0...
Sending part no 1...
Sending part no 2...
Sending part no 3...
Sending part no 4...
Sending part no 5...
Sending part no 6...
Sending part no 7...
Sending part no 8...
Finishing upload session...
uploaded: true

$ onteoncli distribution create-from-file product/onteon/distribution.yml
id:                                          624bf12ddbe2ff0e819bc85d
createdAT:                                   2022-04-05T07:35:09.069Z
updatedAt:                                   2022-04-05T07:35:09.069Z
applicationId:                               624bf11bdbe2ff0e819bc849
numberOfInstances:                           2
checkIfNodeCanAcceptNewApplicationInstanceScriptId: 624ada7b8fa5e24f38c3fb35
selectNodeForNewApplicationInstanceScriptId:        624ada7b8fa5e24f38c3fb31
selectApplicationInstanceToRemoveScriptId:          624ada7b8fa5e24f38c3fb33


$ watch onteoncli application-instance list --per-page 20
id                       createdAt                applicationName applicationVersion applicationType applicationId            nodeId                   status
00df60104dfb3cd4bda8b709 2022-04-05T07:35:25.63Z  product         1.0.0              standard        624bf11bdbe2ff0e819bc849 6b43f313f768ce8c1f38c75c running
48896cac62430f3dd207709a 2022-04-05T07:35:13.441Z product         1.0.0              standard        624bf11bdbe2ff0e819bc849 cb2e5e5363d6ba568c85af8b running
8301a38257848a797bf85f78 2022-04-05T07:00:10.113Z company         1.0.0              standard        624af1418fa5e24f38c418c8 6b43f313f768ce8c1f38c75c running
4c689b74ae904de38457020c 2022-04-05T07:00:10.105Z company         1.0.0              standard        624af1418fa5e24f38c418c8 cb2e5e5363d6ba568c85af8b running
e3e04e37bf423ec83c589803 2022-04-05T06:59:58.091Z monolith        1.1.0              standard        624ae4b48fa5e24f38c4071e 6b43f313f768ce8c1f38c75c running
be40469997ec1529dc12459f 2022-04-05T06:59:46.279Z monolith        1.1.0              standard        624ae4b48fa5e24f38c4071e cb2e5e5363d6ba568c85af8b running
9fa26e2ed4ae9868fc2ee369 2022-04-05T06:59:34.859Z frontend        1.0.0              standard        624ae0268fa5e24f38c400ce 6b43f313f768ce8c1f38c75c running
805515eeb87bea2521e2307b 2022-04-05T06:59:23.583Z frontend        1.0.0              standard        624ae0268fa5e24f38c400ce cb2e5e5363d6ba568c85af8b running
1c43c42be158e5db67c46651 2022-04-05T06:59:05.731Z nginx-edge      1.1.3              embedded        0a1769f83abc3224fe6cb477 cb2e5e5363d6ba568c85af8b running
6aa5b6a12bb5b952f3d836bd 2022-04-05T06:59:05.73Z  nginx-edge      1.1.3              embedded        0a1769f83abc3224fe6cb477 6b43f313f768ce8c1f38c75c running
285af2d5e54f34b365eb92a1 2022-04-05T06:58:59.501Z nginx-inner     1.1.3              embedded        157958ad0cdf14266501ad70 6b43f313f768ce8c1f38c75c running
41aa5a989cf9e7b8750ff828 2022-04-05T06:58:59.5Z   nginx-inner     1.1.3              embedded        157958ad0cdf14266501ad70 cb2e5e5363d6ba568c85af8b running
```

Report

First, copy the `monolith` directory and rename it to `report`.

Then go to pom.xml and change names. This time, don't remove the openpdf dependency.

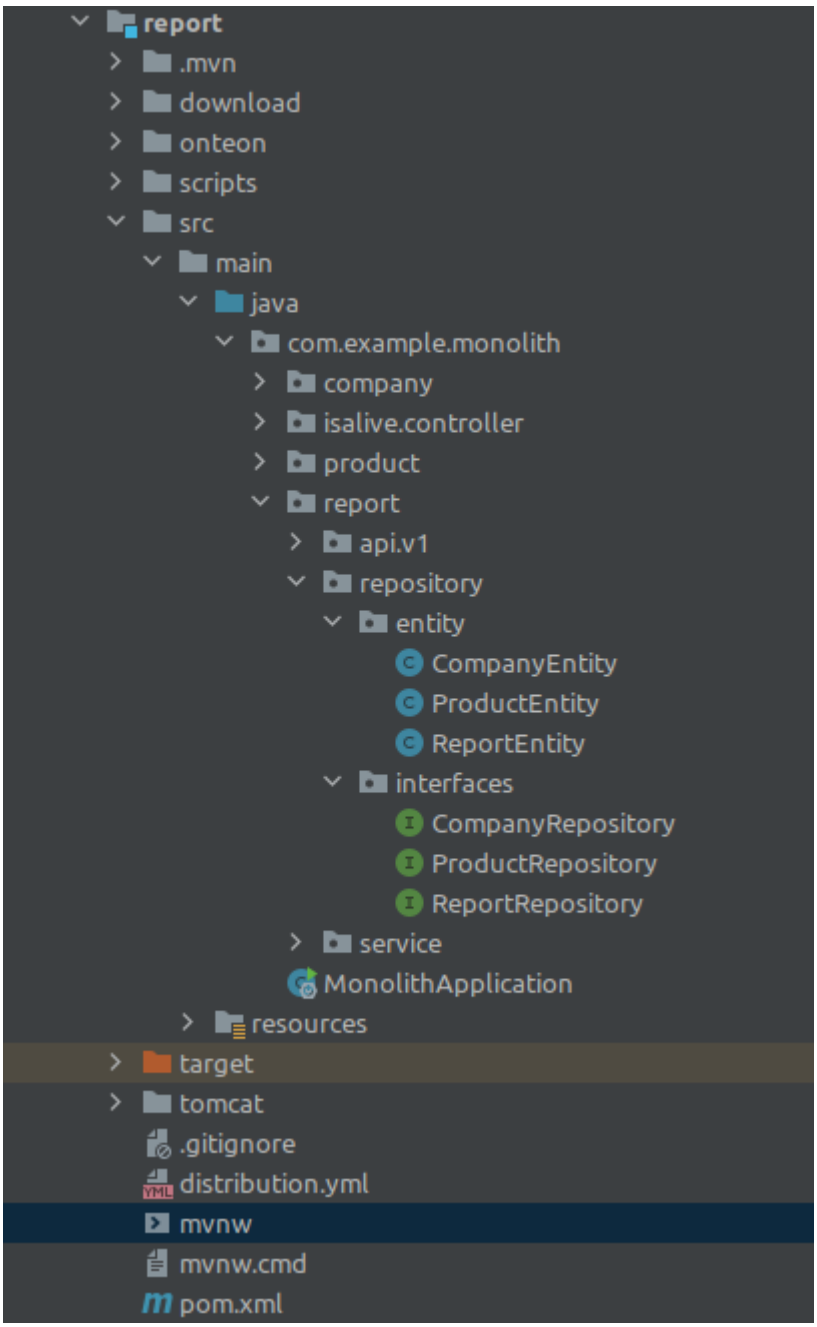
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.6.5</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>

    <groupId>com.example</groupId>
    <artifactId>report</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>report</name>
    <description>report</description>
    <packaging>war</packaging>

    <properties>
        <java.version>11</java.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-validation</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-tomcat</artifactId>
```

```
            <scope>provided</scope>
        </dependency>

        <dependency>
            <groupId>com.github.librepdf</groupId>
            <artifactId>openpdf</artifactId>
            <version>1.3.27</version>
        </dependency>

        <dependency>
            <groupId>org.postgresql</groupId>
            <artifactId>postgresql</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <finalName>${artifactId}</finalName>

        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <configuration>
                    <excludes>
                        <exclude>
                            <groupId>org.projectlombok</groupId>
                            <artifactId>lombok</artifactId>
                        </exclude>
                    </excludes>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

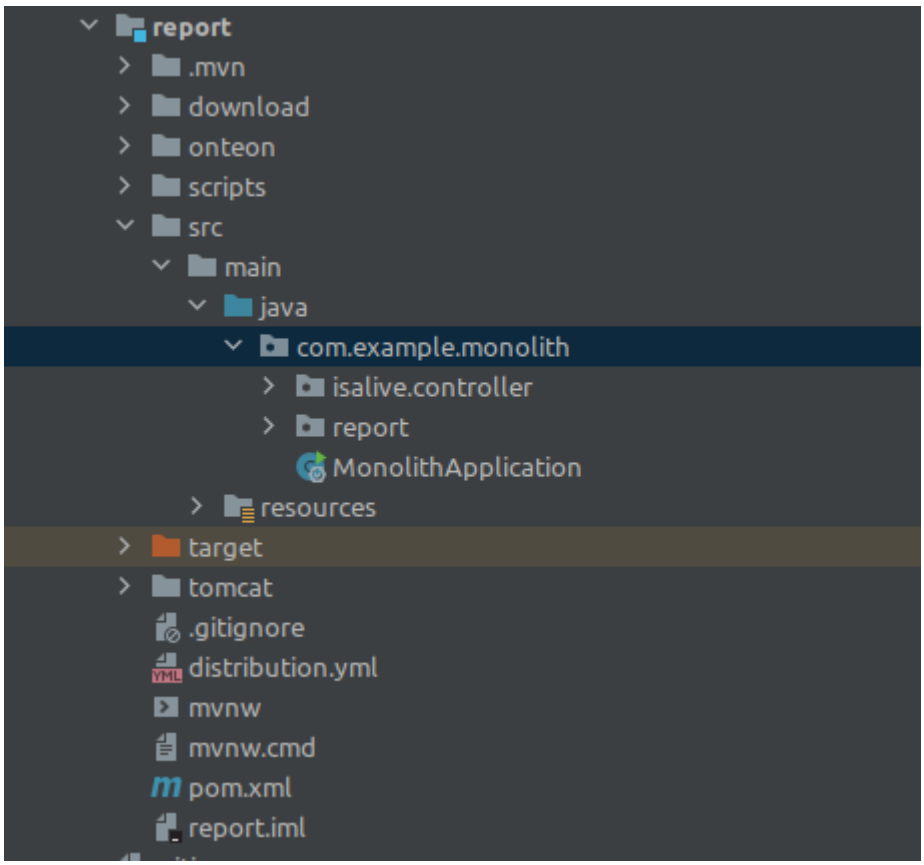
Now, let's open the `ReportServiceImpl`, and check if there are imports from other packages than `report`.

```
package com.example.monolith.product.service.impl;

import com.example.monolith.company.repository.entity.CompanyEntity;
import com.example.monolith.company.repository.interfaces.CompanyRepository;
import com.example.monolith.company.service.exception.NotFoundCompanyException;
import com.example.monolith.product.repository.entity.ProductEntity;
import com.example.monolith.product.repository.interfaces.ProductRepository;
import com.example.monolith.product.service.dto.ProductDTO;
import com.example.monolith.product.service.dto.converter.ProductConverter;
import com.example.monolith.product.service.exception.NotFoundProductException;
import com.example.monolith.product.service.interfaces.ProductService;
import lombok.NonNull;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.stream.Collectors;

/**
 * @author Patryk Borchowiec
 */
@Service
public class ProductServiceImpl implements ProductService {
    private final ProductRepository productRepository;
    private final CompanyRepository companyRepository;
    private final ProductConverter productConverter;

    public ProductServiceImpl(
            final ProductRepository productRepository,
            final CompanyRepository companyRepository,
            final ProductConverter productConverter
    ) {
        this.productRepository = productRepository;
        this.companyRepository = companyRepository;
        this.productConverter = productConverter;
    }
```

As you can see, ProductServiceImpl uses CompanyEntity, CompanyRepository from `company` and ProductEntity, ProductRepository from `product` package. We need to copy these classes to product report. Report package should look like this:

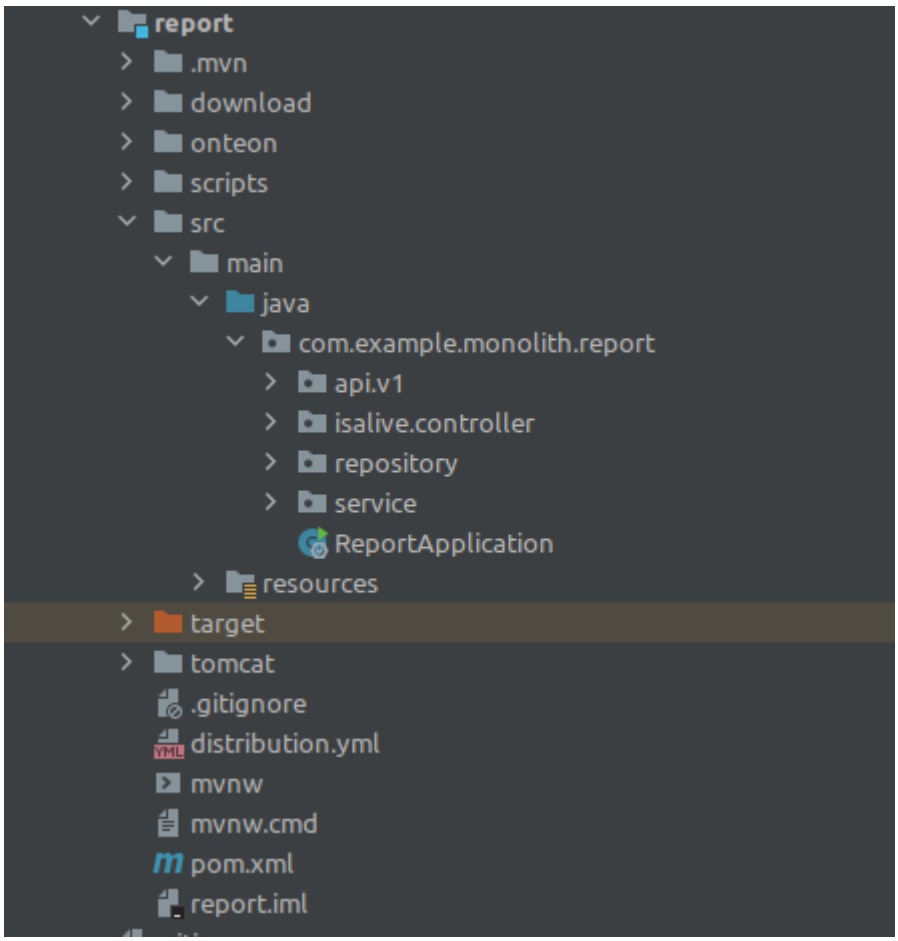


Next step is to remove the `company` and `product` packages. The packages should look like this:

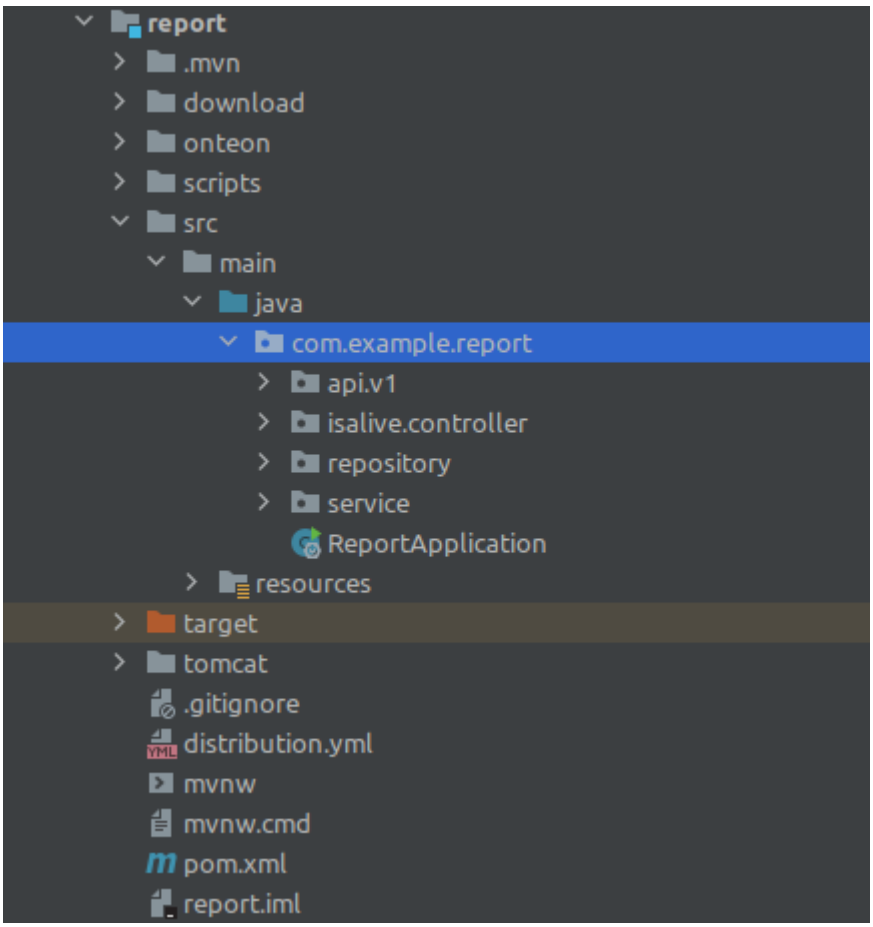


Here you can run `mvn clean package` to make sure that product project has everything.

Now, rename `MonolithApplication` to `ReportApplication` and, move `ReportApplication` and `isalive` package to `report` package.



Finally, rename `com.example.monolith.report` package to `com.example.report`.



Onteon Configuration

Let's create `report/onteon/conf.yml` file.

```
app:
  name: 'report'
  version: '1.0.0'
  appType: 'standard'
  procType: 'native'
  processProvider:
    name: 'GenericOsProcessProviderImpl'
    version: '1.0.0'
    executable:
      start:
        command: '${ont_app_path}/bin/bin/catalina.sh run'
        successLine: 'Server startup'
      stop:
        command: '${ont_app_path}/bin/bin/catalina.sh stop'
      terminate:
        command: '${ont_app_path}/bin/bin/catalina.sh stop -force'
  placeHolder:
    name: 'PlaceHolderManagerImpl'
    version: '1.0.0'
    filesToReplace:
      - '${ont_app_path}/bin/conf/server.xml'
    variables:
  serviceRepository:
    healthCheckUrl: 'http://localhost:${ont_port_2}/is-alive'
    entities:
      - entity:
          priority: 1
          port: ${ont_port_2}
          protocol:
            type: 'HTTP'
            version: '1.1'
          isExternal: false
          isInternal: true
```

The configuration is pretty similar to the previous one. This application will also be available only in internal network.

Distribution

Create `report/onteon/distribution.yml`.

```
application: report:1.0.0
numberOfInstances: 2
scripts:
  checkIfNodeCanAcceptNewApplicationInstance: defaultAvailableNodeOnlyCINCANAIV1
  selectNodeForNewApplicationInstance: defaultApplicationInstancesCountOnlySNFNAIV1
  selectApplicationInstanceToRemove: defaultApplicationInstancesCountOnlySAITRV1
```

Building with script

Let's modify a `report/scripts/build.sh` script.

```
#!/usr/bin/env bash

set -e

SOURCE="${BASH_SOURCE[0]}"
while [ -h "$SOURCE" ]; do
  DIR="$(cd -P "$( dirname "$SOURCE" )" >/dev/null 2>&1 && pwd)"
  SOURCE="$(readlink "$SOURCE")"
  [[ $SOURCE != /* ]] && SOURCE="$DIR/$SOURCE"
done
SCRIPT_DIR="$(cd -P "$(dirname "$SOURCE")" >/dev/null 2>&1 && pwd)"
ROOT_DIR="${SCRIPT_DIR}/.."
TARGET_DIR="${ROOT_DIR}/target"
DOWNLOAD_DIR="${ROOT_DIR}/download"
TOMCAT_SOURCE="${DOWNLOAD_DIR}/tomcat.tar.gz"
TOMCAT_DOWNLOAD_URL="http://archive.apache.org/dist/tomcat/tomcat-9/v9.0.60/bin/apache-tomcat-9.0.60.tar.gz"
TOMCAT_DIR_NAME="apache-tomcat-9.0.60"
APP_BIN_DIR="${TARGET_DIR}/app/bin"
APP_CONF_DIR="${TARGET_DIR}/app/conf"


mkdir -p "${DOWNLOAD_DIR}"
if [ ! -f "${TOMCAT_SOURCE}" ]; then
  wget -O "${TOMCAT_SOURCE}" "${TOMCAT_DOWNLOAD_URL}"
fi

cd "${ROOT_DIR}" && mvn clean package
tar -xvf "${TOMCAT_SOURCE}" -C "${TARGET_DIR}"
mkdir -p "${APP_BIN_DIR}"
mkdir -p "${APP_CONF_DIR}"
cp -r "${TARGET_DIR}/${TOMCAT_DIR_NAME}/"* "${APP_BIN_DIR}"
rm -rf "${APP_BIN_DIR}/webapps/"*
cp -r "${TARGET_DIR}/report.war" "${APP_BIN_DIR}/webapps/ROOT.war"
cp "${ROOT_DIR}/onteon/conf.yml" "${APP_CONF_DIR}"
cp -r "${ROOT_DIR}/tomcat/"* "${APP_BIN_DIR}"
cd "${TARGET_DIR}/app" && tar -zcvf "${TARGET_DIR}/report.tar.gz" *
```

and now you can run script. The package can be found in target directory.

```
$ ./report/scripts/build.sh
$ ls report/target/report.tar.gz
```

Start application

Let's upload application and create distribution.

```
$ onteoncli application-registry upload report/target/report.tar.gz
Starting upload session...
Sending part no 0...
Sending part no 1...
Sending part no 2...
Sending part no 3...
```

```
Sending part no 4...
Sending part no 5...
Sending part no 6...
Sending part no 7...
Sending part no 8...
Finishing upload session...
uploaded: true

$ onteoncli distribution create-from-file report/onteon/distribution.yml
id:                                          624bf812dbe2ff0e819bd83e
createdAT:                                   2022-04-05T08:04:34.299Z
updatedAt:                                   2022-04-05T08:04:34.299Z
applicationId:                               624bf7f3dbe2ff0e819bd7f8
numberOfInstances:                           2
checkIfNodeCanAcceptNewApplicationInstanceScriptId: 624ada7b8fa5e24f38c3fb35
selectNodeForNewApplicationInstanceScriptId:        624ada7b8fa5e24f38c3fb31
selectApplicationInstanceToRemoveScriptId:          624ada7b8fa5e24f38c3fb33

$ watch onteoncli application-instance list --per-page 20
id                       createdAt                applicationName applicationVersion applicationType applicationId            nodeId                   status
c023a855c66d5652af72570f 2022-04-05T08:04:55.742Z report          1.0.0              standard        624bf7f3dbe2ff0e819bd7f8 6b43f313f768ce8c1f38c75c running
1675c3d1a7c3c73328e08776 2022-04-05T08:04:43.466Z report          1.0.0              standard        624bf7f3dbe2ff0e819bd7f8 cb2e5e5363d6ba568c85af8b running
00df60104dfb3cd4bda8b709 2022-04-05T07:35:25.63Z  product         1.0.0              standard        624bf11bdbe2ff0e819bc849 6b43f313f768ce8c1f38c75c running
48896cac62430f3dd207709a 2022-04-05T07:35:13.441Z product         1.0.0              standard        624bf11bdbe2ff0e819bc849 cb2e5e5363d6ba568c85af8b running
8301a38257848a797bf85f78 2022-04-05T07:00:10.113Z company         1.0.0              standard        624af1418fa5e24f38c418c8 6b43f313f768ce8c1f38c75c running
4c689b74ae904de38457020c 2022-04-05T07:00:10.105Z company         1.0.0              standard        624af1418fa5e24f38c418c8 cb2e5e5363d6ba568c85af8b running
e3e04e37bf423ec83c589803 2022-04-05T06:59:58.091Z monolith        1.1.0              standard        624ae4b48fa5e24f38c4071e 6b43f313f768ce8c1f38c75c running
be40469997ec1529dc12459f 2022-04-05T06:59:46.279Z monolith        1.1.0              standard        624ae4b48fa5e24f38c4071e cb2e5e5363d6ba568c85af8b running
9fa26e2ed4ae9868fc2ee369 2022-04-05T06:59:34.859Z frontend        1.0.0              standard        624ae0268fa5e24f38c400ce 6b43f313f768ce8c1f38c75c running
805515eeb87bea2521e2307b 2022-04-05T06:59:23.583Z frontend        1.0.0              standard        624ae0268fa5e24f38c400ce cb2e5e5363d6ba568c85af8b running
1c43c42be158e5db67c46651 2022-04-05T06:59:05.731Z nginx-edge      1.1.3              embedded        0a1769f83abc3224fe6cb477 cb2e5e5363d6ba568c85af8b running
6aa5b6a12bb5b952f3d836bd 2022-04-05T06:59:05.73Z  nginx-edge      1.1.3              embedded        0a1769f83abc3224fe6cb477 6b43f313f768ce8c1f38c75c running
285af2d5e54f34b365eb92a1 2022-04-05T06:58:59.501Z nginx-inner     1.1.3              embedded        157958ad0cdf14266501ad70 6b43f313f768ce8c1f38c75c running
41aa5a989cf9e7b8750ff828 2022-04-05T06:58:59.5Z   nginx-inner     1.1.3              embedded        157958ad0cdf14266501ad70 cb2e5e5363d6ba568c85af8b running
```

Gateway

This time, we will create a gateway microservice, which will be the connection between microservices in inner network and external frontend. To do this, we will use a Spring Cloud Gateway.

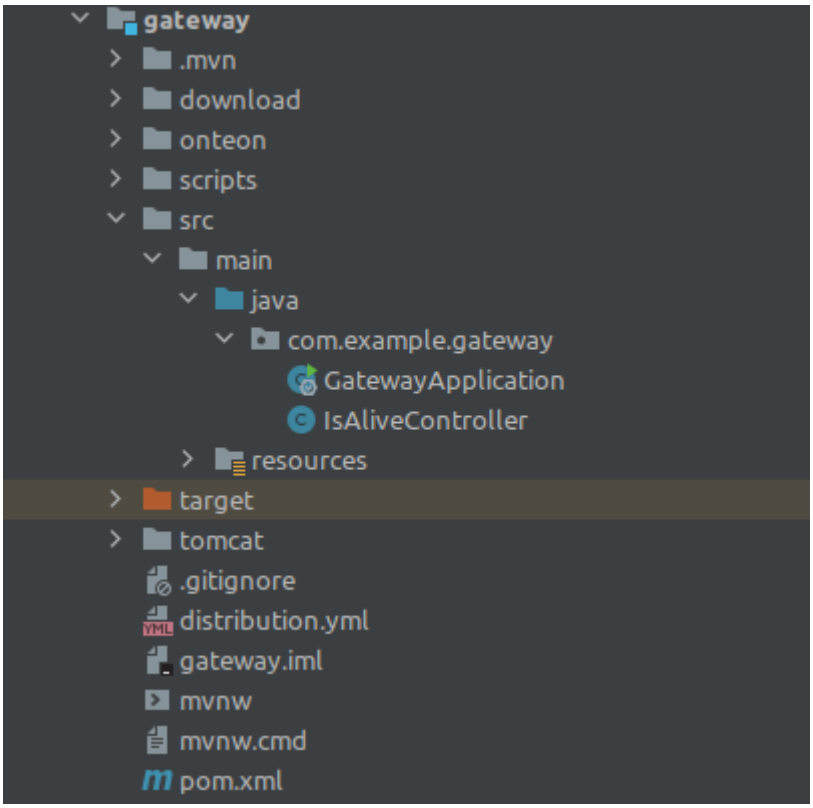
Let's copy a `monolith` and change it name to `gateway`.

Then, change the content of pom.xml file:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.6.5</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>

    <groupId>com.example</groupId>
    <artifactId>gateway</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>gateway</name>
    <description>gateway</description>

    <properties>
        <java.version>11</java.version>
        <spring-cloud.version>2021.0.1</spring-cloud.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-gateway</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>org.springframework.cloud</groupId>
```

```
                <artifactId>spring-cloud-dependencies</artifactId>
                <version>${spring-cloud.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>

    <build>
        <finalName>${artifactId}</finalName>

        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>
```

Now, let's create a `com.example.gateway` package, and copy here a MonolithApplication and IsAliveController. The rest of the classes need to be deleted. Finally, rename MonolithApplication to GatewayApplication.



Now, let's modify a GatewayApplication class. Here we will create a routing to our microservices.

```
package com.example.gateway;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.gateway.route.RouteLocator;
import org.springframework.cloud.gateway.route.builder.RouteLocatorBuilder;
import org.springframework.context.annotation.Bean;

@SpringBootApplication
public class GatewayApplication {

    public static void main(String[] args) {
        SpringApplication.run(GatewayApplication.class, args);
    }

    @Bean
    public RouteLocator routeLocator(final RouteLocatorBuilder builder) {
        final String innerEdgeAddress = "http://localhost:8021";

        return builder.routes()
```

```
                .route("product_api", r -> r.path("/api/v1/products")
                        .filters(f -> f.rewritePath(
                                "/api/v1/products",
                                "/_by_name/product/api/v1/products"
                        ))
                        .uri(innerEdgeAddress))
                .route("product_rewrite_api", r -> r.path("/api/v1/products/**")
                        .filters(f -> f.rewritePath(
                                "/api/v1/products/(?<segment>.*)",
                                "/_by_name/product/api/v1/products/${segment}"
                        ))
                        .uri(innerEdgeAddress))
                .route("company_api", r -> r.path("/api/v1/companies")
                        .filters(f -> f.rewritePath(
                                "/api/v1/companies",
                                "/_by_name/company/api/v1/companies"
                        ))
                        .uri(innerEdgeAddress))
                .route("company_rewrite_api", r -> r.path("/api/v1/companies/**")
                        .filters(f -> f.rewritePath(
                                "/api/v1/companies/(?<segment>.*)",
                                "/_by_name/company/api/v1/companies/${segment}"
                        ))
                        .uri(innerEdgeAddress))
                .route("report_api", r -> r.path("/api/v1/reports")
                        .filters(f -> f.rewritePath(
                                "/api/v1/reports",
                                "/_by_name/report/api/v1/reports"
                        ))
                        .uri(innerEdgeAddress))
                .route("report_rewrite_api", r -> r.path("/api/v1/reports/**")
                        .filters(f -> f.rewritePath(
                                "/api/v1/reports/(?<segment>.*)",
                                "/_by_name/report/api/v1/reports/${segment}"
                        ))
                        .uri(innerEdgeAddress))
                .build();
    }
}
```

Let's modify a few files. First, `gateway/onteon/conf.yml`

```
app:
  name: 'gateway'
  version: '1.0.0'
  appType: 'standard'
  procType: 'native'
  processProvider:
    name: 'JVMOsProcessProviderImpl'
    version: '1.0.0'
    executable:
      start:
        path: '${ont_app_path}/bin'
        startJvmCommand: 'java -jar -Dserver.port=${ont_port_1}'
        successLine: 'Started GatewayApplication'
        executableFileName: 'gateway.jar'
  placeHolder:
    name: 'PlaceHolderManagerImpl'
    version: '1.0.0'
    filesToReplace:
    variables:
  serviceRepository:
    healthCheckUrl: 'http://localhost:${ont_port_1}/is-alive'
    entities:
      - entity:
          priority: 1
          port: ${ont_port_1}
          protocol:
            type: 'HTTP'
            version: '1.1'
          isExternal: true
          isInternal: true
```

Gateway needs to be external to be accessible by frontend. It also uses `JVMOsProcessProviderImpl` because it will be started from jar.

Next file is `gateway/onteon/distribution.yml`

```
application: gateway:1.0.0
numberOfInstances: 2
scripts:
  checkIfNodeCanAcceptNewApplicationInstance: defaultAvailableNodeOnlyCINCANAIV1
  selectNodeForNewApplicationInstance: defaultApplicationInstancesCountOnlySNFNAIV1
  selectApplicationInstanceToRemove: defaultApplicationInstancesCountOnlySAITRV1
```

Finally, let's modify a `gateway/scripts/build.sh` script.

```
#!/usr/bin/env bash

set -e

SOURCE="${BASH_SOURCE[0]}"
while [ -h "$SOURCE" ]; do
  DIR="$(cd -P "$( dirname "$SOURCE" )" >/dev/null 2>&1 && pwd)"
  SOURCE="$(readlink "$SOURCE")"
  [[ $SOURCE != /* ]] && SOURCE="$DIR/$SOURCE"
done
SCRIPT_DIR="$(cd -P "$(dirname "$SOURCE")" >/dev/null 2>&1 && pwd)"
ROOT_DIR="${SCRIPT_DIR}/.."
TARGET_DIR="${ROOT_DIR}/target"
APP_BIN_DIR="${TARGET_DIR}/app/bin"
APP_CONF_DIR="${TARGET_DIR}/app/conf"

cd "${ROOT_DIR}" && mvn clean package
mkdir -p "${APP_BIN_DIR}"
mkdir -p "${APP_CONF_DIR}"
cp -r "${TARGET_DIR}/gateway.jar" "${APP_BIN_DIR}"
cp "${ROOT_DIR}/onteon/conf.yml" "${APP_CONF_DIR}"
cd "${TARGET_DIR}/app" && tar -zcvf "${TARGET_DIR}/gateway.tar.gz" *
```

and now you can run script. The package can be found in target directory.

```
$ ./gateway/scripts/build.sh
$ ls gateway/target/gateway.tar.gz
```

Start application

Let's upload application and create distribution.

```
$ onteoncli application-registry upload gateway/target/gateway.tar.gz
Starting upload session...
Sending part no 0...
Sending part no 1...
Sending part no 2...
Sending part no 3...
Sending part no 4...
Sending part no 5...
Finishing upload session...
uploaded: true

$ onteoncli distribution create-from-file gateway/onteon/distribution.yml
id:                                           624bfc46dbe2ff0e819be38c
createdAT:                                    2022-04-05T08:22:30.351Z
updatedAt:                                    2022-04-05T08:22:30.351Z
applicationId:                                624bfc3bdbe2ff0e819be365
numberOfInstances:                            2
checkIfNodeCanAcceptNewApplicationInstanceScriptId: 624ada7b8fa5e24f38c3fb35
selectNodeForNewApplicationInstanceScriptId:        624ada7b8fa5e24f38c3fb31
selectApplicationInstanceToRemoveScriptId:          624ada7b8fa5e24f38c3fb33

$ watch onteoncli application-instance list --per-page 20
id                       createdAt                applicationName applicationVersion applicationType applicationId            nodeId                   status
c023a855c66d5652af72570f 2022-04-05T08:04:55.742Z report          1.0.0              standard        624bf7f3dbe2ff0e819bd7f8 6b43f313f768ce8c1f38c75c running
1675c3d1a7c3c73328e08776 2022-04-05T08:04:43.466Z report          1.0.0              standard        624bf7f3dbe2ff0e819bd7f8 cb2e5e5363d6ba568c85af8b running
00df60104dfb3cd4bda8b709 2022-04-05T07:35:25.63Z  product         1.0.0              standard        624bf11bdbe2ff0e819bc849 6b43f313f768ce8c1f38c75c running
48896cac62430f3dd207709a 2022-04-05T07:35:13.441Z product         1.0.0              standard        624bf11bdbe2ff0e819bc849 cb2e5e5363d6ba568c85af8b running
8301a38257848a797bf85f78 2022-04-05T07:00:10.113Z company         1.0.0              standard        624af1418fa5e24f38c418c8 6b43f313f768ce8c1f38c75c running
4c689b74ae904de38457020c 2022-04-05T07:00:10.105Z company         1.0.0              standard        624af1418fa5e24f38c418c8 cb2e5e5363d6ba568c85af8b running
e3e04e37bf423ec83c589803 2022-04-05T06:59:58.091Z monolith        1.1.0              standard        624ae4b48fa5e24f38c4071e 6b43f313f768ce8c1f38c75c running
be40469997ec1529dc12459f 2022-04-05T06:59:46.279Z monolith        1.1.0              standard        624ae4b48fa5e24f38c4071e cb2e5e5363d6ba568c85af8b running
9fa26e2ed4ae9868fc2ee369 2022-04-05T06:59:34.859Z frontend        1.0.0              standard        624ae0268fa5e24f38c400ce 6b43f313f768ce8c1f38c75c running
805515eeb87bea2521e2307b 2022-04-05T06:59:23.583Z frontend        1.0.0              standard        624ae0268fa5e24f38c400ce cb2e5e5363d6ba568c85af8b running
1c43c42be158e5db67c46651 2022-04-05T06:59:05.731Z nginx-edge      1.1.3              embedded        0a1769f83abc3224fe6cb477 cb2e5e5363d6ba568c85af8b running
6aa5b6a12bb5b952f3d836bd 2022-04-05T06:59:05.73Z  nginx-edge      1.1.3              embedded        0a1769f83abc3224fe6cb477 6b43f313f768ce8c1f38c75c running
285af2d5e54f34b365eb92a1 2022-04-05T06:58:59.501Z nginx-inner     1.1.3              embedded        157958ad0cdf14266501ad70 6b43f313f768ce8c1f38c75c running
41aa5a989cf9e7b8750ff828 2022-04-05T06:58:59.5Z   nginx-inner     1.1.3              embedded        157958ad0cdf14266501ad70 cb2e5e5363d6ba568c85af8b running
```

Frontend

Now let's update the urls in frontend. Requests should go through gateway. Go to `frontend/app/bin/webapps/ROOT/index.html` and modify the value of `baseUrl` variable to `/_by_name/gateway`.

```
const baseUrl = '/_by_name/gateway';
const companyApiBaseUrl = `${baseUrl}/api/v1/companies`;
const productApiBaseUrl = `${baseUrl}/api/v1/products`;
const reportApiBaseUrl = `${baseUrl}/api/v1/reports`;

function initCompaniesForm() {
    $(".new-company-form").submit(e => {
        e.preventDefault(e);
        addCompany($("#company-name").val());
    })
}
```

Next step is to update the version of frontend to 1.0.1 in `frontend/app/conf/conf.yml`.

```
app:
  name: 'frontend'
  version: '1.0.1'
  appType: 'standard'
  procType: 'native'
  processProvider:
    name: 'GenericOsProcessProviderImpl'
    version: '1.0.0'
    executable:
      start:
        command: '${ont_app_path}/bin/bin/catalina.sh run'
        successLine: 'Server startup'
      stop:
        command: '${ont_app_path}/bin/bin/catalina.sh stop'
      terminate:
        command: '${ont_app_path}/bin/bin/catalina.sh stop -force'
  placeHolder:
    name: 'PlaceHolderManagerImpl'
    version: '1.0.0'
    filesToReplace:
      - '${ont_app_path}/bin/conf/server.xml'
    variables:
  serviceRepository:
    healthCheckUrl: 'http://localhost:${ont_port_2}'
    entities:
      - entity:
          priority: 1
          port: ${ont_port_2}
          protocol:
            type: 'HTTP'
            version: '2.0'
          isExternal: true
          isInternal: true
```

Now package and deploy application package.

```
$ ./frontend/scripts/build.sh
$ onteoncli application-registry upload ./frontend/target/frontend.tar.gz
$ onteoncli application list
id                       createdAt                updatedAt                name     version type     processType
624c0403dbe2ff0e819bfb8f 2022-04-05T08:55:31.834Z 2022-04-05T08:55:31.834Z frontend 1.0.1   standard native
624bfc3bdbe2ff0e819be365 2022-04-05T08:22:19.29Z  2022-04-05T08:22:19.29Z  gateway  1.0.0   standard native
624bf7f3dbe2ff0e819bd7f8 2022-04-05T08:04:03.947Z 2022-04-05T08:04:03.947Z report   1.0.0   standard native
624bf11bdbe2ff0e819bc849 2022-04-05T07:34:51.56Z  2022-04-05T07:34:51.56Z  product  1.0.0   standard native
624af1418fa5e24f38c418c8 2022-04-04T13:23:13.206Z 2022-04-04T13:23:13.206Z company  1.0.0   standard native
624ae4b48fa5e24f38c4071e 2022-04-04T12:29:40.566Z 2022-04-04T12:29:40.566Z monolith 1.1.0   standard native
624ae0268fa5e24f38c400ce 2022-04-04T12:10:14.809Z 2022-04-04T12:10:14.809Z frontend 1.0.0   standard native
624adaae8fa5e24f38c3fb5f 2022-04-04T11:46:54.782Z 2022-04-04T11:46:54.782Z monolith 1.0.0   standard native
```

Now let's upgrade frontend application. First, list all applications and choose the id of old frontend, and id of new frontend. Then, you need to execute `onteoncli application-change create <old-frontend-id> <new-frontend-id>` and then, you can check how it automatically replaces old application with new one.

```
frontend/app$ onteoncli application list
id                       createdAt                updatedAt                name     version type     processType
624c0403dbe2ff0e819bfb8f 2022-04-05T08:55:31.834Z 2022-04-05T08:55:31.834Z frontend 1.0.1   standard native
624bfc3bdbe2ff0e819be365 2022-04-05T08:22:19.29Z  2022-04-05T08:22:19.29Z  gateway  1.0.0   standard native
624bf7f3dbe2ff0e819bd7f8 2022-04-05T08:04:03.947Z 2022-04-05T08:04:03.947Z report   1.0.0   standard native
```

```
624bf11bdbe2ff0e819bc849 2022-04-05T07:34:51.56Z  2022-04-05T07:34:51.56Z  product  1.0.0   standard native
624af1418fa5e24f38c418c8 2022-04-04T13:23:13.206Z 2022-04-04T13:23:13.206Z company  1.0.0   standard native
624ae4b48fa5e24f38c4071e 2022-04-04T12:29:40.566Z 2022-04-04T12:29:40.566Z monolith 1.1.0   standard native
624ae0268fa5e24f38c400ce 2022-04-04T12:10:14.809Z 2022-04-04T12:10:14.809Z frontend 1.0.0   standard native
624adaae8fa5e24f38c3fb5f 2022-04-04T11:46:54.782Z 2022-04-04T11:46:54.782Z monolith 1.0.0   standard native

frontend/app$ onteoncli application-change create 624ae0268fa5e24f38c400ce 624c0403dbe2ff0e819bfb8f
id:               624c0492dbe2ff0e819bfd42
createdAt:        2022-04-05T08:57:54.36Z
updatedAt:        2022-04-05T08:57:54.36Z
oldApplicationId: 624ae0268fa5e24f38c400ce
newApplicationId: 624c0403dbe2ff0e819bfb8f

frontend/app$ watch onteoncli application-instance list --per-page 20
id                       createdAt                applicationName applicationVersion applicationType applicationId            nodeId                   status
f0549eaa190a5e575c287047 2022-04-05T08:59:01.891Z frontend        1.0.1              standard        624c0403dbe2ff0e819bfb8f cb2e5e5363d6ba568c85af8b running
ec3666e8b7dfea1c767c04c1 2022-04-05T08:58:02.832Z frontend        1.0.1              standard        624c0403dbe2ff0e819bfb8f 6b43f313f768ce8c1f38c75c running
a974138e7efa9303539da878 2022-04-05T08:22:54.182Z gateway         1.0.0              standard        624bfc3bdbe2ff0e819be365 6b43f313f768ce8c1f38c75c running
444f79ea15074220618497e7 2022-04-05T08:22:43.16Z  gateway         1.0.0              standard        624bfc3bdbe2ff0e819be365 cb2e5e5363d6ba568c85af8b running
c023a855c66d5652af72570f 2022-04-05T08:04:55.742Z report          1.0.0              standard        624bf7f3dbe2ff0e819bd7f8 6b43f313f768ce8c1f38c75c running
1675c3d1a7c3c73328e08776 2022-04-05T08:04:43.466Z report          1.0.0              standard        624bf7f3dbe2ff0e819bd7f8 cb2e5e5363d6ba568c85af8b running
00df60104dfb3cd4bda8b709 2022-04-05T07:35:25.63Z  product         1.0.0              standard        624bf11bdbe2ff0e819bc849 6b43f313f768ce8c1f38c75c running
48896cac62430f3dd207709a 2022-04-05T07:35:13.441Z product         1.0.0              standard        624bf11bdbe2ff0e819bc849 cb2e5e5363d6ba568c85af8b running
8301a38257848a797bf85f78 2022-04-05T07:00:10.113Z company         1.0.0              standard        624af1418fa5e24f38c418c8 6b43f313f768ce8c1f38c75c running
4c689b74ae904de38457020c 2022-04-05T07:00:10.105Z company         1.0.0              standard        624af1418fa5e24f38c418c8 cb2e5e5363d6ba568c85af8b running
e3e04e37bf423ec83c589803 2022-04-05T06:59:58.091Z monolith        1.1.0              standard        624ae4b48fa5e24f38c4071e 6b43f313f768ce8c1f38c75c running
be40469997ec1529dc12459f 2022-04-05T06:59:46.279Z monolith        1.1.0              standard        624ae4b48fa5e24f38c4071e cb2e5e5363d6ba568c85af8b running
1c43c42be158e5db67c46651 2022-04-05T06:59:05.731Z nginx-edge      1.1.3              embedded        0a1769f83abc3224fe6cb477 cb2e5e5363d6ba568c85af8b running
6aa5b6a12bb5b952f3d836bd 2022-04-05T06:59:05.73Z  nginx-edge      1.1.3              embedded        0a1769f83abc3224fe6cb477 6b43f313f768ce8c1f38c75c running
285af2d5e54f34b365eb92a1 2022-04-05T06:58:59.501Z nginx-inner     1.1.3              embedded        157958ad0cdf14266501ad70 6b43f313f768ce8c1f38c75c running
41aa5a989cf9e7b8750ff828 2022-04-05T06:58:59.5Z   nginx-inner     1.1.3              embedded        157958ad0cdf14266501ad70 cb2e5e5363d6ba568c85af8b running
```

Monolith

Finally, we can remove old monolith application. To do so, let's find the id of monolith application.

```
$ onteoncli application list
id                       createdAt                updatedAt                name     version type     processType
624c0403dbe2ff0e819bfb8f 2022-04-05T08:55:31.834Z 2022-04-05T08:55:31.834Z frontend 1.0.1   standard native
624bfc3bdbe2ff0e819be365 2022-04-05T08:22:19.29Z  2022-04-05T08:22:19.29Z  gateway  1.0.0   standard native
624bf7f3dbe2ff0e819bd7f8 2022-04-05T08:04:03.947Z 2022-04-05T08:04:03.947Z report   1.0.0   standard native
624bf11bdbe2ff0e819bc849 2022-04-05T07:34:51.56Z  2022-04-05T07:34:51.56Z  product  1.0.0   standard native
624af1418fa5e24f38c418c8 2022-04-04T13:23:13.206Z 2022-04-04T13:23:13.206Z company  1.0.0   standard native
624ae4b48fa5e24f38c4071e 2022-04-04T12:29:40.566Z 2022-04-04T12:29:40.566Z monolith 1.1.0   standard native
624ae0268fa5e24f38c400ce 2022-04-04T12:10:14.809Z 2022-04-04T12:10:14.809Z frontend 1.0.0   standard native
624adaae8fa5e24f38c3fb5f 2022-04-04T11:46:54.782Z 2022-04-04T11:46:54.782Z monolith 1.0.0   standard native
```

In my case it is a `624ae4b48fa5e24f38c4071e`. Now let's find a distribution that manages this application. Let's list distribution and find the one with the application id.

```
$ onteoncli distribution list
id                       createdAt                updatedAt                applicationId            numberOfInstances
624bfc46dbe2ff0e819be38c 2022-04-05T08:22:30.351Z 2022-04-05T08:22:30.351Z 624bfc3bdbe2ff0e819be365 2
624bf812dbe2ff0e819bd83e 2022-04-05T08:04:34.299Z 2022-04-05T08:04:34.299Z 624bf7f3dbe2ff0e819bd7f8 2
624bf12ddbe2ff0e819bc85d 2022-04-05T07:35:09.069Z 2022-04-05T07:35:09.069Z 624bf11bdbe2ff0e819bc849 2
624af1868fa5e24f38c4192b 2022-04-04T13:24:22.004Z 2022-04-04T13:24:22.004Z 624af1418fa5e24f38c418c8 2
624ae0428fa5e24f38c400ed 2022-04-04T12:10:42.037Z 2022-04-05T08:58:00.049Z 624c0403dbe2ff0e819bfb8f 2
624adab58fa5e24f38c3fb67 2022-04-04T11:47:01.212Z 2022-04-04T12:32:30.038Z 624ae4b48fa5e24f38c4071e 2
```

In my case, id of distribution is a `624adab58fa5e24f38c3fb67`. Now let's change the number of instances to 0, and wait for distribution to remove instances.

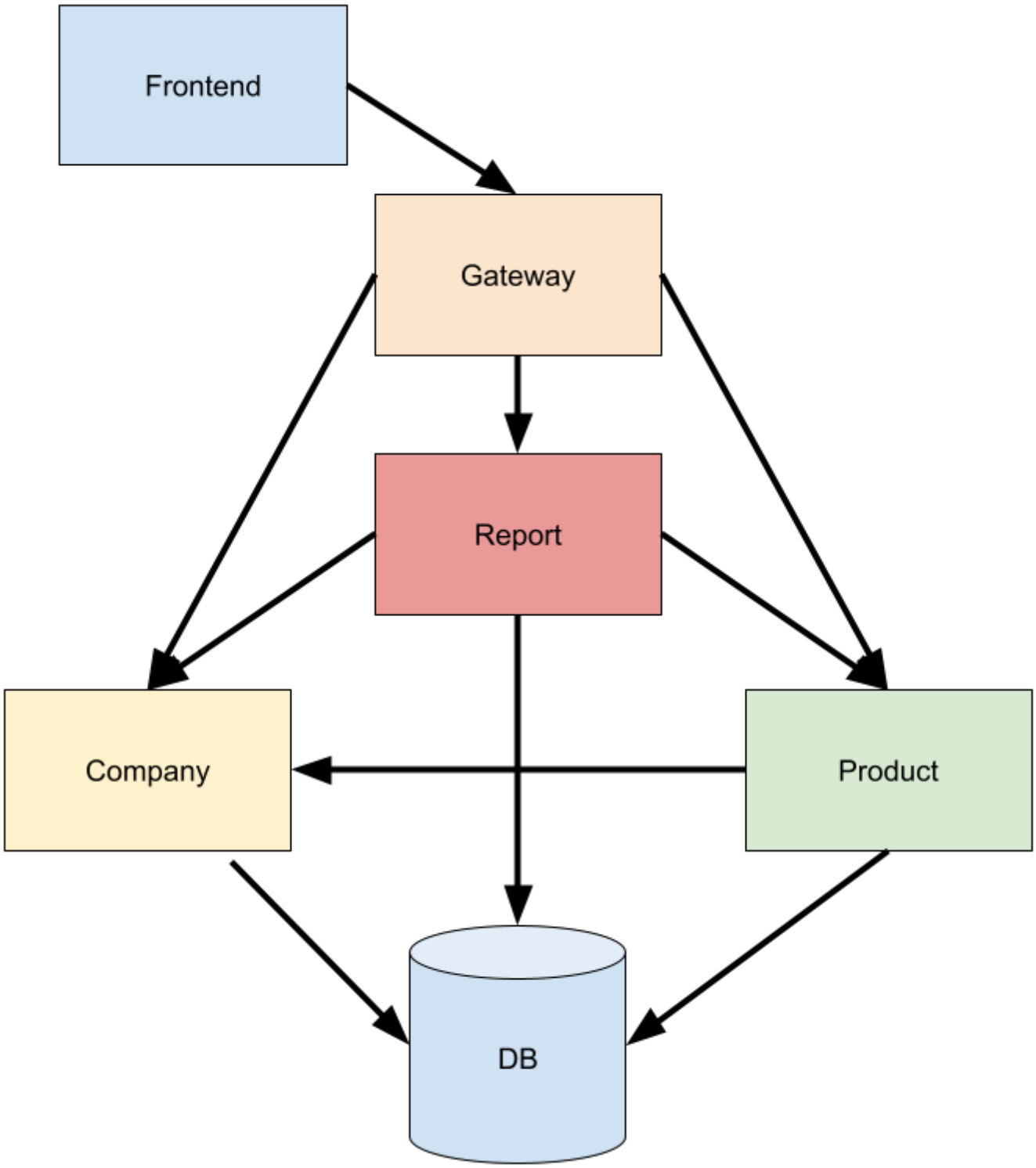
```
$ onteoncli distribution set-number-of-instances 624adab58fa5e24f38c3fb67 0
id:                                          624adab58fa5e24f38c3fb67
createdAT:                                   2022-04-04T11:47:01.212Z
updatedAt:                                   2022-04-05T09:02:39.314Z
applicationId:                               624ae4b48fa5e24f38c4071e
numberOfInstances:                           0
checkIfNodeCanAcceptNewApplicationInstanceScriptId: 624ada7b8fa5e24f38c3fb35
selectNodeForNewApplicationInstanceScriptId:        624ada7b8fa5e24f38c3fb31
selectApplicationInstanceToRemoveScriptId:          624ada7b8fa5e24f38c3fb33

$ watch onteoncli application-instance list --per-page 20
id                       createdAt                applicationName applicationVersion applicationType applicationId            nodeId                   status
f0549eaa190a5e575c287047 2022-04-05T08:59:01.891Z frontend        1.0.1              standard        624c0403dbe2ff0e819bfb8f cb2e5e5363d6ba568c85af8b running
ec3666e8b7dfea1c767c04c1 2022-04-05T08:58:02.832Z frontend        1.0.1              standard        624c0403dbe2ff0e819bfb8f 6b43f313f768ce8c1f38c75c running
a974138e7efa9303539da878 2022-04-05T08:22:54.182Z gateway         1.0.0              standard        624bfc3bdbe2ff0e819be365 6b43f313f768ce8c1f38c75c running
444f79ea15074220618497e7 2022-04-05T08:22:43.16Z  gateway         1.0.0              standard        624bfc3bdbe2ff0e819be365 cb2e5e5363d6ba568c85af8b running
c023a855c66d5652af72570f 2022-04-05T08:04:55.742Z report          1.0.0              standard        624bf7f3dbe2ff0e819bd7f8 6b43f313f768ce8c1f38c75c running
```

```
1675c3d1a7c3c73328e08776 2022-04-05T08:04:43.466Z report          1.0.0             standard        624bf7f3dbe2ff0e819bd7f8 cb2e5e5363d6ba568c85af8b running
00df60104dfb3cd4bda8b709 2022-04-05T07:35:25.63Z  product         1.0.0             standard        624bf11bdbe2ff0e819bc849 6b43f313f768ce8c1f38c75c running
48896cac62430f3dd207709a 2022-04-05T07:35:13.441Z product         1.0.0             standard        624bf11bdbe2ff0e819bc849 cb2e5e5363d6ba568c85af8b running
8301a38257848a797bf85f78 2022-04-05T07:00:10.113Z company         1.0.0             standard        624af1418fa5e24f38c418c8 6b43f313f768ce8c1f38c75c running
4c689b74ae904de38457020c 2022-04-05T07:00:10.105Z company         1.0.0             standard        624af1418fa5e24f38c418c8 cb2e5e5363d6ba568c85af8b running
1c43c42be158e5db67c46651 2022-04-05T06:59:05.731Z nginx-edge      1.1.3             embedded        0a1769f83abc3224fe6cb477 cb2e5e5363d6ba568c85af8b running
6aa5b6a12bb5b952f3d836bd 2022-04-05T06:59:05.73Z  nginx-edge      1.1.3             embedded        0a1769f83abc3224fe6cb477 6b43f313f768ce8c1f38c75c running
285af2d5e54f34b365eb92a1 2022-04-05T06:58:59.501Z nginx-inner     1.1.3             embedded        157958ad0cdf14266501ad70 6b43f313f768ce8c1f38c75c running
41aa5a989cf9e7b8750ff828 2022-04-05T06:58:59.5Z   nginx-inner     1.1.3             embedded        157958ad0cdf14266501ad70 cb2e5e5363d6ba568c85af8b running
```

The monolith application is removed. Now go to http://localhost:8020/_by_name/frontend and check if application is working properly.

Step 4 - Communication between microservices

In this step we will make microservices communicate with each other. After this step, they will read from database only data from specific domain e.g. product microservice will read from database only data about products. It will ask microservice `company` about companies.



Files

If you don't have files from previous step, you can:

- download **initial** files from here
- download **complete** files from here

Company

Let's start with `company` microservice, as it does not depend on other microservices. First let's see if company's api has everything that other microservices needs.

First, let's see the report's ReportServiceImpl.

```
    @Override
    public void create(final @NonNull ReportDTO reportDTO) {
        final ReportEntity entity = reportRepository.save(reportConverter.toNewEntity(reportDTO));
        final List<CompanyEntity> companies = companyRepository.findAll();
        final List<ProductEntity> products = productRepository.findAll();

        Document document = new Document();
        final ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
        final PdfWriter instance = PdfWriter.getInstance(document, byteArrayOutputStream);
```

Report needs to get a list of all companies, now let's see if company microservice has endpoint that returns list of companies.

```
    @GetMapping
    public List<CompanyV1Response> getAll() {
        return companyService.getAll().stream().map(companyV1Converter::toResponse).collect(Collectors.toList());
    }
```

As you can see, company microservice already has that functionality.

Now, let's check the product's ProductServiceImpl class.

```
    @Override
    public void create(final @NonNull ProductDTO productDTO) {
        final CompanyEntity companyEntity = companyRepository.findById(productDTO.getCompanyId())
                .orElseThrow(() -> new NotFoundCompanyException(
                        String.format("Not found company of id %s.", productDTO.getCompanyId())
                ));

        final ProductEntity productEntity = productConverter.toNewEntity(productDTO, companyEntity);
        productRepository.save(productEntity);
    }

    @Override
    public void update(final @NonNull ProductDTO productDTO) {
        final CompanyEntity companyEntity = companyRepository.findById(productDTO.getCompanyId())
                .orElseThrow(() -> new NotFoundCompanyException(
                        String.format("Not found company of id %s.", productDTO.getCompanyId())
                ));

        if (!productRepository.existsById(productDTO.getId())) {
            throw new NotFoundProductException(String.format("Not found product with id %s.", productDTO.getId()));
        }

        final ProductEntity productEntity = productConverter.toEntity(productDTO, companyEntity);
        productRepository.save(productEntity);
    }
```

Product microservice need to get company by id. Company microservice does not have an endpoint that returns a company by id.

Let's start with defining new `getById` method in CompanyService interface.

```
package com.example.company.service.interfaces;

import com.example.company.service.dto.CompanyDTO;
import com.example.company.service.exception.NotFoundCompanyException;
import lombok.NonNull;

import java.util.List;

/**
 * @author Patryk Borchowiec
 */tła
public interface CompanyService {
    void create(final @NonNull CompanyDTO companyDTO);
    void update(final @NonNull CompanyDTO companyDTO) throws NotFoundCompanyException;
    void deleteById(final @NonNull Long id) throws NotFoundCompanyException;
    CompanyDTO getById(final @NonNull Long id);
    List<CompanyDTO> getAll();
}
```

Now, let's implement the `getById` method in CompanyServiceImpl.

```
package com.example.company.service.impl;

import com.example.company.service.dto.CompanyDTO;
import com.example.company.service.dto.converter.CompanyConverter;
import com.example.company.repository.entity.CompanyEntity;
import com.example.company.repository.interfaces.CompanyRepository;
import com.example.company.service.exception.NotFoundCompanyException;
import com.example.company.service.interfaces.CompanyService;
import lombok.NonNull;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.stream.Collectors;

/**
 * @author Patryk Borchowiec
 */
@Service
public class CompanyServiceImpl implements CompanyService {
    private final CompanyRepository companyRepository;
    private final CompanyConverter companyConverter;

    public CompanyServiceImpl(final CompanyRepository companyRepository, final CompanyConverter companyConverter) {
        this.companyRepository = companyRepository;
        this.companyConverter = companyConverter;
    }

    @Override
    public void create(final @NonNull CompanyDTO companyDTO) {
        final CompanyEntity companyEntity = companyConverter.toNewEntity(companyDTO);
        companyRepository.save(companyEntity);
    }

    @Override
    public void update(final @NonNull CompanyDTO companyDTO) throws NotFoundCompanyException{
        if (!companyRepository.existsById(companyDTO.getId())) {
            throw new NotFoundCompanyException(String.format("Not found company with id %s", companyDTO.getId()));
        }

        final CompanyEntity companyEntity = companyConverter.toEntity(companyDTO);
        companyRepository.save(companyEntity);
    }

    @Override
    public void deleteById(final @NonNull Long id) throws NotFoundCompanyException {
        if (!companyRepository.existsById(id)) {
            throw new NotFoundCompanyException(String.format("Not found company with id %s", id));
        }

        companyRepository.deleteById(id);
    }

    @Override
    public CompanyDTO getById(@NonNull Long id) {
        return companyRepository.findById(id)
                .map(companyConverter::toDTO)
                .orElseThrow(() -> new NotFoundCompanyException(String.format("Not found company with id %s", id)));
    }

    @Override
    public List<CompanyDTO> getAll() {
        return companyRepository.findAll().stream().map(companyConverter::toDTO).collect(Collectors.toList());
    }
}
```

Finally, let's call this method in CompanyController from `getById` method.

```
package com.example.company.api.v1.controller;

import com.example.company.api.v1.pojo.converter.CompanyV1Converter;
import com.example.company.api.v1.pojo.request.CreateCompanyV1Request;
import com.example.company.api.v1.pojo.request.UpdateCompanyV1Request;
import com.example.company.api.v1.pojo.response.CompanyV1Response;
import com.example.company.service.dto.CompanyDTO;
import com.example.company.service.exception.NotFoundCompanyException;
import com.example.company.service.interfaces.CompanyService;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.server.ResponseStatusException;

import javax.validation.Valid;
import java.util.List;
import java.util.stream.Collectors;

/**
 * @author Patryk Borchowiec
 */
@RestController
@RequestMapping(path = "/api/v1/companies", produces = MediaType.APPLICATION_JSON_VALUE)
public class CompanyV1Controller {
    private final CompanyService companyService;
    private final CompanyV1Converter companyV1Converter;

    public CompanyV1Controller(final CompanyService companyService, final CompanyV1Converter companyV1Converter) {
        this.companyService = companyService;
        this.companyV1Converter = companyV1Converter;
    }

    @PostMapping
    public void create(final @RequestBody @Valid CreateCompanyV1Request request) {
        companyService.create(companyV1Converter.toDTO(request));
    }

    @GetMapping
    public List<CompanyV1Response> getAll() {
        return companyService.getAll().stream().map(companyV1Converter::toResponse).collect(Collectors.toList());
    }

    @GetMapping("/{id}")
    public CompanyV1Response getById(final @PathVariable long id) {
        final CompanyDTO companyDTO;
        try {
            companyDTO = companyService.getById(id);
        } catch (final NotFoundCompanyException e) {
            throw new ResponseStatusException(
                    HttpStatus.NOT_FOUND,
                    String.format("Cannot update company due to: %s", e.getMessage())
            );
        }

        return companyV1Converter.toResponse(companyDTO);
    }

    @PutMapping("/{id}")
    public void update(final @RequestBody @Valid UpdateCompanyV1Request request, final @PathVariable long id) {
        try {
            companyService.update(companyV1Converter.toDTO(request, id));
        } catch (final NotFoundCompanyException e) {
            throw new ResponseStatusException(
                    HttpStatus.NOT_FOUND,
                    String.format("Cannot update company due to: %s", e.getMessage())
            );
        }
    }

    @DeleteMapping("/{id}")
    public void delete(final @PathVariable long id) {
        try {
            companyService.deleteById(id);
        } catch (final NotFoundCompanyException e) {
            throw new ResponseStatusException(
                    HttpStatus.NOT_FOUND,
                    String.format("Cannot delete company due to: %s", e.getMessage())
            );
        }
    }
}
```

Now, let's update a version in `company/onteon/conf.yml` file to 1.1.0. We change the minor version because the API is changed.

```
app:
  name: 'company'
  version: '1.1.0'
  appType: 'standard'
  procType: 'native'
  processProvider:
    name: 'GenericOsProcessProviderImpl'
    version: '1.0.0'
    executable:
      start:
        command: '${ont_app_path}/bin/bin/catalina.sh run'
        successLine: 'Server startup'
      stop:
        command: '${ont_app_path}/bin/bin/catalina.sh stop'
      terminate:
        command: '${ont_app_path}/bin/bin/catalina.sh stop -force'
  placeHolder:
    name: 'PlaceHolderManagerImpl'
    version: '1.0.0'
    filesToReplace:
      - '${ont_app_path}/bin/conf/server.xml'
    variables:
  serviceRepository:
    healthCheckUrl: 'http://localhost:${ont_port_2}/is-alive'
    entities:
      - entity:
          priority: 1
          port: ${ont_port_2}
          protocol:
            type: 'HTTP'
            version: '1.1'
          isExternal: false
          isInternal: true
```

Let's build and upload application.

```
$ ./company/scripts/build.sh
$ onteoncli application-registry upload company/target/company.tar.gz
Starting upload session...
Sending part no 0...
Sending part no 1...
Sending part no 2...
Sending part no 3...
Sending part no 4...
Sending part no 5...
Sending part no 6...
Sending part no 7...
Sending part no 8...
Finishing upload session...
uploaded: true

$ onteoncli application list
id                       createdAt                updatedAt                name     version type     processType
624c1006dbe2ff0e819c1cb9 2022-04-05T09:46:46.512Z 2022-04-05T09:46:46.512Z company  1.1.0   standard native
624c0403dbe2ff0e819bfb8f 2022-04-05T08:55:31.834Z 2022-04-05T08:55:31.834Z frontend 1.0.1   standard native
624bfc3bdbe2ff0e819be365 2022-04-05T08:22:19.29Z  2022-04-05T08:22:19.29Z  gateway  1.0.0   standard native
624bf7f3dbe2ff0e819bd7f8 2022-04-05T08:04:03.947Z 2022-04-05T08:04:03.947Z report   1.0.0   standard native
624bf11bdbe2ff0e819bc849 2022-04-05T07:34:51.56Z  2022-04-05T07:34:51.56Z  product  1.0.0   standard native
624af1418fa5e24f38c418c8 2022-04-04T13:23:13.206Z 2022-04-04T13:23:13.206Z company  1.0.0   standard native
624ae4b48fa5e24f38c4071e 2022-04-04T12:29:40.566Z 2022-04-04T12:29:40.566Z monolith 1.1.0   standard native
624ae0268fa5e24f38c400ce 2022-04-04T12:10:14.809Z 2022-04-04T12:10:14.809Z frontend 1.0.0   standard native
624adaae8fa5e24f38c3fb5f 2022-04-04T11:46:54.782Z 2022-04-04T11:46:54.782Z monolith 1.0.0   standard native
```

Now, let's upgrade old company application with new company application by executing `onteoncli application-change create <old-company-id> <new-company-id>`.

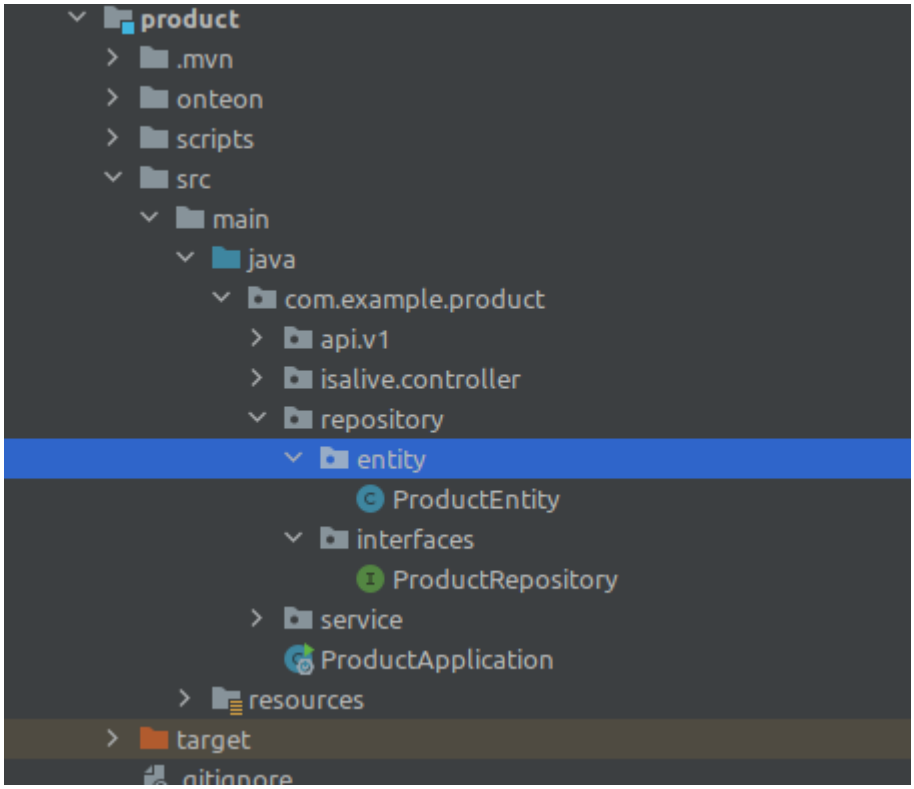
```
$ onteoncli application-change create 624af1418fa5e24f38c418c8 624c1006dbe2ff0e819c1cb9
id:               624c1080dbe2ff0e819c1dfe
createdAt:        2022-04-05T09:48:48.984Z
updatedAt:        2022-04-05T09:48:48.984Z
oldApplicationId: 624af1418fa5e24f38c418c8
newApplicationId: 624c1006dbe2ff0e819c1cb9

$ watch onteoncli application-instance list --per-page 20
id                       createdAt                applicationName applicationVersion applicationType applicationId            nodeId                   status
1f8884be35382c4b697a440d 2022-04-05T09:49:03.358Z company         1.1.0              standard        624c1006dbe2ff0e819c1cb9 cb2e5e5363d6ba568c85af8b running
ccf21b97022ac8da4218059c 2022-04-05T09:48:53.34Z  company         1.1.0              standard        624c1006dbe2ff0e819c1cb9 6b43f313f768ce8c1f38c75c running
f0549eaa190a5e575c287047 2022-04-05T08:59:01.891Z frontend        1.0.1              standard        624c0403dbe2ff0e819bfb8f cb2e5e5363d6ba568c85af8b running
ec3666e8b7dfea1c767c04c1 2022-04-05T08:58:02.832Z frontend        1.0.1              standard        624c0403dbe2ff0e819bfb8f 6b43f313f768ce8c1f38c75c running
a974138e7efa9303539da878 2022-04-05T08:22:54.182Z gateway         1.0.0              standard        624bfc3bdbe2ff0e819be365 6b43f313f768ce8c1f38c75c running
444f79ea15074220618497e7 2022-04-05T08:22:43.16Z  gateway         1.0.0              standard        624bfc3bdbe2ff0e819be365 cb2e5e5363d6ba568c85af8b running
c023a855c66d5652af72570f 2022-04-05T08:04:55.742Z report          1.0.0              standard        624bf7f3dbe2ff0e819bd7f8 6b43f313f768ce8c1f38c75c running
1675c3d1a7c3c73328e08776 2022-04-05T08:04:43.466Z report          1.0.0              standard        624bf7f3dbe2ff0e819bd7f8 cb2e5e5363d6ba568c85af8b running
00df60104dfb3cd4bda8b709 2022-04-05T07:35:25.63Z  product         1.0.0              standard        624bf11bdbe2ff0e819bc849 6b43f313f768ce8c1f38c75c running
48896cac62430f3dd207709a 2022-04-05T07:35:13.441Z product         1.0.0              standard        624bf11bdbe2ff0e819bc849 cb2e5e5363d6ba568c85af8b running
1c43c42be158e5db67c46651 2022-04-05T06:59:05.731Z nginx-edge      1.1.3              embedded        0a1769f83abc3224fe6cb477 cb2e5e5363d6ba568c85af8b running
6aa5b6a12bb5b952f3d836bd 2022-04-05T06:59:05.73Z  nginx-edge      1.1.3              embedded        0a1769f83abc3224fe6cb477 6b43f313f768ce8c1f38c75c running
```

```
285af2d5e54f34b365eb92a1 2022-04-05T06:58:59.501Z nginx-inner     1.1.3             embedded        157958ad0cdf14266501ad70 6b43f313f768ce8c1f38c75c running
41aa5a989cf9e7b8750ff828 2022-04-05T06:58:59.5Z   nginx-inner     1.1.3             embedded        157958ad0cdf14266501ad70 cb2e5e5363d6ba568c85af8b running
```

Product

Remove repositories

First, let's delete the CompanyEntity and CompanyRepository. The `product` directory should look like this:



Now, let's open the ProductEntity and modify the company field to long.

```
package com.example.product.repository.entity;

import lombok.Data;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;

/**
 * @author Patryk Borchowiec
 */
@Entity(name = "products")
@Data
public class ProductEntity {
    @Id
    @GeneratedValue
    private Long id;

    private String name;

    private int amount;

    private Long company;
}
```

Finally, modify the ProductConverter. Replace CompanyEntity from args with Long.

```
package com.example.product.service.dto.converter;

import com.example.product.repository.entity.ProductEntity;
import com.example.product.service.dto.ProductDTO;
import org.springframework.stereotype.Component;

import java.util.Objects;

/**
 * @author Patryk Borchowiec
 */
@Component
public class ProductConverter {
    public ProductEntity toNewEntity(final ProductDTO productDTO, final Long companyId) {
        if (Objects.isNull(productDTO)) {
            return null;
        }

        final ProductEntity productEntity = new ProductEntity();
        productEntity.setName(productDTO.getName());
        productEntity.setAmount(productDTO.getAmount());
        productEntity.setCompany(companyId);

        return productEntity;
    }

    public ProductEntity toEntity(final ProductDTO productDTO, final Long companyId) {
        if (Objects.isNull(productDTO)) {
            return null;
        }

        final ProductEntity productEntity = new ProductEntity();
        productEntity.setId(productDTO.getId());
        productEntity.setName(productDTO.getName());
        productEntity.setAmount(productDTO.getAmount());
        productEntity.setCompany(companyId);

        return productEntity;
    }

    public ProductDTO toDTO(final ProductEntity productEntity) {
        if (Objects.isNull(productEntity)) {
            return null;
        }

        return new ProductDTO(
                productEntity.getId(),
                productEntity.getName(),
                productEntity.getAmount(),
                productEntity.getCompany()
        );
    }
}
```

Remote Service

Now, we need to create remote service to communicate with Company microservice. We will use the OkHttp library.

First, go to pom.xml and add OkHttp dependency.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.6.5</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>

  <groupId>com.example</groupId>
  <artifactId>product</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>product</name>
  <description>product</description>
  <packaging>war</packaging>

  <properties>
    <java.version>11</java.version>
  </properties>

  <dependencies>
```

```
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-validation</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-tomcat</artifactId>
      <scope>provided</scope>
    </dependency>

    <dependency>
      <groupId>com.squareup.okhttp</groupId>
      <artifactId>okhttp</artifactId>
      <version>2.7.5</version>
    </dependency>

    <dependency>
      <groupId>org.postgresql</groupId>
      <artifactId>postgresql</artifactId>
      <scope>runtime</scope>
    </dependency>
    <dependency>
      <groupId>org.projectlombok</groupId>
      <artifactId>lombok</artifactId>
      <optional>true</optional>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
    </dependency>
  </dependencies>

  <build>
    <finalName>${artifactId}</finalName>

    <plugins>
      <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
        <configuration>
          <excludes>
            <exclude>
              <groupId>org.projectlombok</groupId>
              <artifactId>lombok</artifactId>
            </exclude>
          </excludes>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

Now, let's create a `com.example.product.remote.company.pojo.response` package and create here two classes:

CompanyResponse

```
package com.example.product.remote.company.pojo.response;

import lombok.Data;

/**
 * @author Patryk Borchowiec
 */
@Data
public class CompanyResponse {
    private Long id;
    private String name;
}
```

ErrorResponse

```
package com.example.product.remote.company.pojo.response;

import lombok.Data;

/**
 * @author Patryk Borchowiec
 */
```

```
@Data
public class ErrorResponse {
    private String timestamp;
    private String path;
    private int status;
    private String error;
    private String message;
    private String requestId;
}
```

Now create a `com.example.product.remote.exception` package and create a `RemoteServiceException` class.

```
package com.example.product.remote.exception;

/**
 * @author Patryk Borchowiec
 */
public class RemoteServiceException extends RuntimeException {
    public RemoteServiceException() {
    }

    public RemoteServiceException(String message) {
        super(message);
    }

    public RemoteServiceException(String message, Throwable cause) {
        super(message, cause);
    }
}
```

Now, let's create the actual Remote Service. Let's start with CompanyRemoteService interface in `com.example.product.remote.company.interfaces` package.

```
package com.example.product.remote.company.interfaces;

import com.example.product.remote.company.pojo.response.CompanyResponse;
import com.example.product.remote.exception.RemoteServiceException;
import lombok.NonNull;

/**
 * @author Patryk Borchowiec
 */
public interface CompanyRemoteService {
    CompanyResponse getById(final @NonNull Long id) throws RemoteServiceException;
}
```

Then, implement method in `CompanyRemoteServiceImpl` class in `com.example.product.remote.company.impl` package.

```
package com.example.product.remote.company.impl;

import com.example.product.remote.company.interfaces.CompanyRemoteService;
import com.example.product.remote.company.pojo.response.CompanyResponse;
import com.example.product.remote.company.pojo.response.ErrorResponse;
import com.example.product.remote.exception.RemoteServiceException;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.squareup.okhttp.OkHttpClient;
import com.squareup.okhttp.Request;
import com.squareup.okhttp.Response;
import lombok.NonNull;
import org.springframework.http.HttpStatus;
import org.springframework.stereotype.Service;
import org.springframework.web.server.ResponseStatusException;

import java.io.IOException;
import java.util.Objects;

/**
 * @author Patryk Borchowiec
 */
@Service
public class CompanyRemoteServiceImpl implements CompanyRemoteService {
    final String BASE_URL = "http://localhost:8021/_by_name/company/api/v1/companies";
    final OkHttpClient client;
    final ObjectMapper objectMapper;

    public CompanyRemoteServiceImpl() {
        this.client = new OkHttpClient();
        this.objectMapper = new ObjectMapper();
    }

    @Override
```

```
    public CompanyResponse getById(final @NonNull Long id) throws RemoteServiceException {
        final Request request = new Request.Builder()
                .get()
                .url(BASE_URL + "/" + id)
                .build();

        final String responseBody = executeRequest(request);
        try {
            return objectMapper.readValue(responseBody, CompanyResponse.class);
        } catch (final JsonProcessingException e) {
            throw new RemoteServiceException("Cannot parse response, due to: " + e.getMessage(), e);
        }
    }

    private String executeRequest(Request request) throws RemoteServiceException {
        final Response response;
        try {
            response = client.newCall(request).execute();
        } catch (IOException e) {
            throw new RemoteServiceException("Cannot get company by id, due to:" + e.getMessage(), e);
        }

        final String body;
        try {
            body = response.body().string();
        } catch (final IOException e) {
            throw new RemoteServiceException("Cannot get response body, due to:" + e.getMessage(), e);
        }

        if (!response.isSuccessful()) {
            throw new ResponseStatusException(HttpStatus.valueOf(response.code()), getErrorMessage(body));
        }

        return body;
    }

    private String getErrorMessage(final String responseBody) {
        final ErrorResponse errorResponse;
        try {
            errorResponse = objectMapper.readValue(responseBody, ErrorResponse.class);
        } catch (JsonProcessingException e) {
            return "";
        }

        if (Objects.isNull(errorResponse) || Objects.isNull(errorResponse.getMessage())) {
            return "";
        }
        return errorResponse.getMessage();
    }
}
```

Before using the remote service in ProductServiceImpl class, let's modify the `NotFoundCompanyException` by adding the constructor with throwable.

```
package com.example.product.service.exception;

/**
 * @author Patryk Borchowiec
 */
public class NotFoundCompanyException extends RuntimeException {
    public NotFoundCompanyException() {
    }

    public NotFoundCompanyException(String message) {
        super(message);
    }

    public NotFoundCompanyException(String message, Throwable cause) {
        super(message, cause);
    }
}
```

Now let's replace calling the CompanyRepository with calling the CompanyRemoteService. Go to ProductServiceImpl and modify the class.

```
package com.example.product.service.impl;

import com.example.product.remote.company.interfaces.CompanyRemoteService;
import com.example.product.remote.company.pojo.response.CompanyResponse;
import com.example.product.repository.entity.ProductEntity;
import com.example.product.repository.interfaces.ProductRepository;
import com.example.product.service.dto.ProductDTO;
import com.example.product.service.dto.converter.ProductConverter;
import com.example.product.service.exception.NotFoundCompanyException;
import com.example.product.service.exception.NotFoundProductException;
import com.example.product.service.interfaces.ProductService;
```

```
import lombok.NonNull;
import org.springframework.http.HttpStatus;
import org.springframework.stereotype.Service;
import org.springframework.web.server.ResponseStatusException;

import java.util.List;
import java.util.stream.Collectors;

/**
 * @author Patryk Borchowiec
 */
@Service
public class ProductServiceImpl implements ProductService {
    private final ProductRepository productRepository;
    private final ProductConverter productConverter;
    private final CompanyRemoteService companyRemoteService;

    public ProductServiceImpl(
            final ProductRepository productRepository,
            final ProductConverter productConverter,
            CompanyRemoteService companyRemoteService) {
        this.productRepository = productRepository;
        this.productConverter = productConverter;
        this.companyRemoteService = companyRemoteService;
    }

    @Override
    public void create(final @NonNull ProductDTO productDTO) {
        final CompanyResponse companyResponse;
        try {
            companyResponse = companyRemoteService.getById(productDTO.getCompanyId());
        } catch (final ResponseStatusException e) {
            if (e.getStatus().equals(HttpStatus.NOT_FOUND)) {
                throw new NotFoundCompanyException(
                        String.format("Not found company of id %s.", productDTO.getCompanyId()),
                        e
                );
            }
            throw e;
        }

        final ProductEntity productEntity = productConverter.toNewEntity(productDTO, companyResponse.getId());
        productRepository.save(productEntity);
    }

    @Override
    public void update(final @NonNull ProductDTO productDTO) {
        final CompanyResponse companyResponse;
        try {
            companyResponse = companyRemoteService.getById(productDTO.getCompanyId());
        } catch (final ResponseStatusException e) {
            if (e.getStatus().equals(HttpStatus.NOT_FOUND)) {
                throw new NotFoundCompanyException(
                        String.format("Not found company of id %s.", productDTO.getCompanyId()),
                        e
                );
            }
            throw e;
        }

        if (!productRepository.existsById(productDTO.getId())) {
            throw new NotFoundProductException(String.format("Not found product with id %s.", productDTO.getId()));
        }

        final ProductEntity productEntity = productConverter.toEntity(productDTO, companyResponse.getId());
        productRepository.save(productEntity);
    }

    @Override
    public void deleteById(final @NonNull Long id) {
        if (!productRepository.existsById(id)) {
            throw new NotFoundProductException(String.format("Not found product with id %s.", id));
        }

        productRepository.deleteById(id);
    }

    @Override
    public List<ProductDTO> getAll() {
        return productRepository.findAll().stream().map(productConverter::toDTO).collect(Collectors.toList());
    }
}
```

Configuration

Now, let's update a version in `product/onteon/conf.yml` file to 1.0.1.

```
app:
  name: 'product'
  version: '1.0.1'
  appType: 'standard'
  procType: 'native'
  processProvider:
    name: 'GenericOsProcessProviderImpl'
    version: '1.0.0'
    executable:
      start:
        command: '${ont_app_path}/bin/bin/catalina.sh run'
        successLine: 'Server startup'
      stop:
        command: '${ont_app_path}/bin/bin/catalina.sh stop'
      terminate:
        command: '${ont_app_path}/bin/bin/catalina.sh stop -force'
  placeHolder:
    name: 'PlaceHolderManagerImpl'
    version: '1.0.0'
    filesToReplace:
      - '${ont_app_path}/bin/conf/server.xml'
    variables:
  serviceRepository:
    healthCheckUrl: 'http://localhost:${ont_port_2}/is-alive'
    entities:
      - entity:
          priority: 1
          port: ${ont_port_2}
          protocol:
            type: 'HTTP'
            version: '1.1'
          isExternal: false
          isInternal: true
```

Build and upload

Let's build and upload application.

```
$ ./product/scripts/build.sh
$ onteoncli application-registry upload product/target/product.tar.gz
Starting upload session...
Sending part no 0...
Sending part no 1...
Sending part no 2...
Sending part no 3...
Sending part no 4...
Sending part no 5...
Sending part no 6...
Sending part no 7...
Sending part no 8...
Finishing upload session...
uploaded: true

$ onteoncli application list
id                       createdAt                updatedAt                name     version type     processType
624c1a5bdbe2ff0e819c39b8 2022-04-05T10:30:51.16Z  2022-04-05T10:30:51.16Z  product  1.0.1   standard native
624c1006dbe2ff0e819c1cb9 2022-04-05T09:46:46.512Z 2022-04-05T09:46:46.512Z company  1.1.0   standard native
624c0403dbe2ff0e819bfb8f 2022-04-05T08:55:31.834Z 2022-04-05T08:55:31.834Z frontend 1.0.1   standard native
624bfc3bdbe2ff0e819be365 2022-04-05T08:22:19.29Z  2022-04-05T08:22:19.29Z  gateway  1.0.0   standard native
624bf7f3dbe2ff0e819bd7f8 2022-04-05T08:04:03.947Z 2022-04-05T08:04:03.947Z report   1.0.0   standard native
624bf11bdbe2ff0e819bc849 2022-04-05T07:34:51.56Z  2022-04-05T07:34:51.56Z  product  1.0.0   standard native
624af1418fa5e24f38c418c8 2022-04-04T13:23:13.206Z 2022-04-04T13:23:13.206Z company  1.0.0   standard native
624ae4b48fa5e24f38c4071e 2022-04-04T12:29:40.566Z 2022-04-04T12:29:40.566Z monolith 1.1.0   standard native
624ae0268fa5e24f38c400ce 2022-04-04T12:10:14.809Z 2022-04-04T12:10:14.809Z frontend 1.0.0   standard native
624adaae8fa5e24f38c3fb5f 2022-04-04T11:46:54.782Z 2022-04-04T11:46:54.782Z monolith 1.0.0   standard native
```

Upgrade

Now, let's upgrade old product application with new product application by executing `onteoncli application-change create <old-product-id> <new-product-id>`.

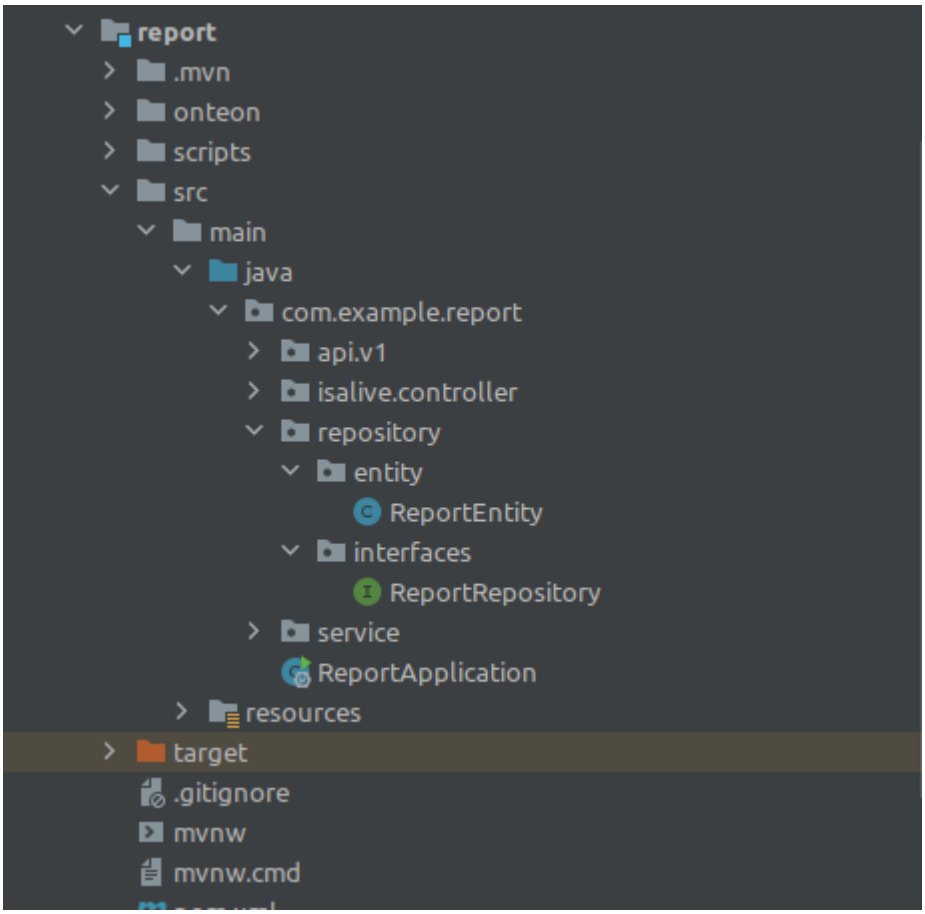
```
$ onteoncli application-change create 624bf11bdbe2ff0e819bc849 624c1a5bdbe2ff0e819c39b8
id:               624c1a8ddbe2ff0e819c3a45
createdAt:        2022-04-05T10:31:41.733Z
updatedAt:        2022-04-05T10:31:41.733Z
oldApplicationId: 624bf11bdbe2ff0e819bc849
newApplicationId: 624c1a5bdbe2ff0e819c39b8

$ watch onteoncli application-instance list --per-page 20
id                       createdAt                applicationName applicationVersion applicationType applicationId           nodeId                   status
```

```
5ab67d50d978bd9dff8e9c01 2022-04-05T10:32:48.522Z product         1.0.1             standard        624c1a5bdbe2ff0e819c39b8 cb2e5e5363d6ba568c85af8b running
a6a79e72b4094c0ea4f78808 2022-04-05T10:31:53.349Z product         1.0.1             standard        624c1a5bdbe2ff0e819c39b8 6b43f313f768ce8c1f38c75c running
1f8884be35382c4b697a440d 2022-04-05T09:49:03.358Z company         1.1.0             standard        624c1006dbe2ff0e819c1cb9 cb2e5e5363d6ba568c85af8b running
ccf21b97022ac8da4218059c 2022-04-05T09:48:53.34Z  company         1.1.0             standard        624c1006dbe2ff0e819c1cb9 6b43f313f768ce8c1f38c75c running
f0549eaa190a5e575c287047 2022-04-05T08:59:01.891Z frontend        1.0.1             standard        624c0403dbe2ff0e819bfb8f cb2e5e5363d6ba568c85af8b running
ec3666e8b7dfea1c767c04c1 2022-04-05T08:58:02.832Z frontend        1.0.1             standard        624c0403dbe2ff0e819bfb8f 6b43f313f768ce8c1f38c75c running
a974138e7efa9303539da878 2022-04-05T08:22:54.182Z gateway         1.0.0             standard        624bfc3bdbe2ff0e819be365 6b43f313f768ce8c1f38c75c running
444f79ea15074220618497e7 2022-04-05T08:22:43.16Z  gateway         1.0.0             standard        624bfc3bdbe2ff0e819be365 cb2e5e5363d6ba568c85af8b running
c023a855c66d5652af72570f 2022-04-05T08:04:55.742Z report          1.0.0             standard        624bf7f3dbe2ff0e819bd7f8 6b43f313f768ce8c1f38c75c running
1675c3d1a7c3c73328e08776 2022-04-05T08:04:43.466Z report          1.0.0             standard        624bf7f3dbe2ff0e819bd7f8 cb2e5e5363d6ba568c85af8b running
1c43c42be158e5db67c46651 2022-04-05T06:59:05.731Z nginx-edge      1.1.3             embedded        0a1769f83abc3224fe6cb477 cb2e5e5363d6ba568c85af8b running
6aa5b6a12bb5b952f3d836bd 2022-04-05T06:59:05.73Z  nginx-edge      1.1.3             embedded        0a1769f83abc3224fe6cb477 6b43f313f768ce8c1f38c75c running
285af2d5e54f34b365eb92a1 2022-04-05T06:58:59.501Z nginx-inner     1.1.3             embedded        157958ad0cdf14266501ad70 6b43f313f768ce8c1f38c75c running
41aa5a989cf9e7b8750ff828 2022-04-05T06:58:59.5Z   nginx-inner     1.1.3             embedded        157958ad0cdf14266501ad70 cb2e5e5363d6ba568c85af8b running
```

Report

Remove repositories

First, let's delete the CompanyEntity, ProductEntity, CompanyRepository and ProductRepository. The `report` package should look like this:



Remote Service

Before creating the remote services, let's add a `OkHttp` to pom.xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
```

```
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.6.5</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>

  <groupId>com.example</groupId>
  <artifactId>report</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>report</name>
  <description>report</description>
  <packaging>war</packaging>

  <properties>
    <java.version>11</java.version>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-validation</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-tomcat</artifactId>
      <scope>provided</scope>
    </dependency>

    <dependency>
      <groupId>com.github.librepdf</groupId>
      <artifactId>openpdf</artifactId>
      <version>1.3.27</version>
    </dependency>
    <dependency>
      <groupId>com.squareup.okhttp</groupId>
      <artifactId>okhttp</artifactId>
      <version>2.7.5</version>
    </dependency>

    <dependency>
      <groupId>org.postgresql</groupId>
      <artifactId>postgresql</artifactId>
      <scope>runtime</scope>
    </dependency>
    <dependency>
      <groupId>org.projectlombok</groupId>
      <artifactId>lombok</artifactId>
      <optional>true</optional>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
    </dependency>
  </dependencies>

  <build>
    <finalName>${artifactId}</finalName>

    <plugins>
      <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
        <configuration>
          <excludes>
            <exclude>
              <groupId>org.projectlombok</groupId>
              <artifactId>lombok</artifactId>
            </exclude>
          </excludes>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

Then, let's create a `com.example.report.remote.exception` package and create there a RemoteServiceException class.

```
package com.example.report.remote.exception;
```

```
/**
 * @author Patryk Borchowiec
 */
public class RemoteServiceException extends RuntimeException {
    public RemoteServiceException() {
    }

    public RemoteServiceException(String message) {
        super(message);
    }

    public RemoteServiceException(String message, Throwable cause) {
        super(message, cause);
    }
}
```

Company

Now, let's create a `com.example.report.remote.company.pojo.response` package and create here two classes:

CompanyResponse

```
package com.example.report.remote.company.pojo.response;

import lombok.Data;

/**
 * @author Patryk Borchowiec
 */
@Data
public class CompanyResponse {
    private Long id;
    private String name;
}
```

ErrorResponse

```
package com.example.report.remote.company.pojo.response;

import lombok.Data;

/**
 * @author Patryk Borchowiec
 */
@Data
public class ErrorResponse {
    private String timestamp;
    private String path;
    private int status;
    private String error;
    private String message;
    private String requestId;
}
```

Now, let's create the actual Remote Service. Let's start with CompanyRemoteService interface in `com.example.report.remote.company.interfaces` package.

```
package com.example.report.remote.company.interfaces;

import com.example.report.remote.company.pojo.response.CompanyResponse;
import com.example.report.remote.exception.RemoteServiceException;
import lombok.NonNull;

import java.util.List;

/**
 * @author Patryk Borchowiec
 */
public interface CompanyRemoteService {
    List<CompanyResponse> getAll() throws RemoteServiceException;
}
```

Then, implement method in `CompanyRemoteServiceImpl` class in `com.example.report.remote.company.impl` package.

```
package com.example.report.remote.company.impl;
```

```
import com.example.report.remote.company.interfaces.CompanyRemoteService;
import com.example.report.remote.company.pojo.response.CompanyResponse;
import com.example.report.remote.company.pojo.response.ErrorResponse;
import com.example.report.remote.exception.RemoteServiceException;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.core.type.TypeReference;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.squareup.okhttp.OkHttpClient;
import com.squareup.okhttp.Request;
import com.squareup.okhttp.Response;
import lombok.NonNull;
import org.springframework.http.HttpStatus;
import org.springframework.stereotype.Service;
import org.springframework.web.server.ResponseStatusException;

import java.io.IOException;
import java.util.List;
import java.util.Objects;

/**
 * @author Patryk Borchowiec
 */
@Service
public class CompanyRemoteServiceImpl implements CompanyRemoteService {
    final String BASE_URL = "http://localhost:8021/_by_name/company/api/v1/companies";
    final OkHttpClient client;
    final ObjectMapper objectMapper;

    public CompanyRemoteServiceImpl() {
        this.client = new OkHttpClient();
        this.objectMapper = new ObjectMapper();
    }

    @Override
    public List<CompanyResponse> getAll() {
        final Request request = new Request.Builder()
                .get()
                .url(BASE_URL)
                .build();

        final String responseBody = executeRequest(request);
        try {
            return objectMapper.readValue(responseBody, new TypeReference<>() {});
        } catch (final JsonProcessingException e) {
            throw new RemoteServiceException("Cannot parse response, due to: " + e.getMessage(), e);
        }
    }

    private String executeRequest(Request request) throws RemoteServiceException {
        final Response response;
        try {
            response = client.newCall(request).execute();
        } catch (IOException e) {
            throw new RemoteServiceException("Cannot get company by id, due to:" + e.getMessage(), e);
        }

        final String body;
        try {
            body = response.body().string();
        } catch (final IOException e) {
            throw new RemoteServiceException("Cannot get response body, due to:" + e.getMessage(), e);
        }

        if (!response.isSuccessful()) {
            throw new ResponseStatusException(HttpStatus.valueOf(response.code()), getErrorMessage(body));
        }

        return body;
    }

    private String getErrorMessage(final String responseBody) {
        final ErrorResponse errorResponse;
        try {
            errorResponse = objectMapper.readValue(responseBody, ErrorResponse.class);
        } catch (JsonProcessingException e) {
            return "";
        }

        if (Objects.isNull(errorResponse) || Objects.isNull(errorResponse.getMessage())) {
            return "";
        }
        return errorResponse.getMessage();
    }
}
```

Product

Now, let's create a `com.example.report.remote.product.response` package and create here two classes:

ProductResponse

```
package com.example.report.remote.product.response;

import lombok.Data;

/**
 * @author Patryk Borchowiec
 */
@Data
public class ProductResponse {
    private Long id;
    private String name;
    private int amount;
    private Long companyId;
}
```

ErrorResponse

```
package com.example.report.remote.product.response;

import lombok.Data;

/**
 * @author Patryk Borchowiec
 */
@Data
public class ErrorResponse {
    private String timestamp;
    private String path;
    private int status;
    private String error;
    private String message;
    private String requestId;
}
```

Now, let's create the actual Remote Service. Let's start with ProductRemoteService interface in `com.example.report.remote.product.interfaces` package.

```
package com.example.report.remote.product.interfaces;

import com.example.report.remote.exception.RemoteServiceException;
import com.example.report.remote.product.response.ProductResponse;

import java.util.List;

/**
 * @author Patryk Borchowiec
 */
public interface ProductRemoteService {
    List<ProductResponse> getAll() throws RemoteServiceException;
}
```

Then, implement method in `ProductRemoteServiceImpl` class in `com.example.report.remote.product.impl` package.

```
package com.example.report.remote.product.impl;

import com.example.report.remote.exception.RemoteServiceException;
import com.example.report.remote.product.interfaces.ProductRemoteService;
import com.example.report.remote.product.response.ProductResponse;
import com.example.report.remote.product.response.ErrorResponse;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.core.type.TypeReference;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.squareup.okhttp.OkHttpClient;
import com.squareup.okhttp.Request;
import com.squareup.okhttp.Response;
import org.springframework.http.HttpStatus;
import org.springframework.stereotype.Service;
import org.springframework.web.server.ResponseStatusException;

import java.io.IOException;
import java.util.List;
import java.util.Objects;

/**
 * @author Patryk Borchowiec
 */
@Service
public class ProductRemoteServiceImpl implements ProductRemoteService {
```

```
    final String BASE_URL = "http://localhost:8021/_by_name/product/api/v1/products";
    final OkHttpClient client;
    final ObjectMapper objectMapper;

    public ProductRemoteServiceImpl() {
        this.client = new OkHttpClient();
        this.objectMapper = new ObjectMapper();
    }

    @Override
    public List<ProductResponse> getAll() {
        final Request request = new Request.Builder()
                .get()
                .url(BASE_URL)
                .build();

        final String responseBody = executeRequest(request);
        try {
            return objectMapper.readValue(responseBody, new TypeReference<>() {});
        } catch (final JsonProcessingException e) {
            throw new RemoteServiceException("Cannot parse response, due to: " + e.getMessage(), e);
        }
    }

    private String executeRequest(Request request) throws RemoteServiceException {
        final Response response;
        try {
            response = client.newCall(request).execute();
        } catch (IOException e) {
            throw new RemoteServiceException("Cannot get company by id, due to:" + e.getMessage(), e);
        }

        final String body;
        try {
            body = response.body().string();
        } catch (final IOException e) {
            throw new RemoteServiceException("Cannot get response body, due to:" + e.getMessage(), e);
        }

        if (!response.isSuccessful()) {
            throw new ResponseStatusException(HttpStatus.valueOf(response.code()), getErrorMessage(body));
        }

        return body;
    }

    private String getErrorMessage(final String responseBody) {
        final ErrorResponse errorResponse;
        try {
            errorResponse = objectMapper.readValue(responseBody, ErrorResponse.class);
        } catch (JsonProcessingException e) {
            return "";
        }

        if (Objects.isNull(errorResponse) || Objects.isNull(errorResponse.getMessage())) {
            return "";
        }
        return errorResponse.getMessage();
    }
}
```

Integration

Now we will replace calling the repository classes with calling the remote services. Let's modify the ReportServiceImpl.

```
package com.example.report.service.impl;

import com.example.report.remote.company.interfaces.CompanyRemoteService;
import com.example.report.remote.company.pojo.response.CompanyResponse;
import com.example.report.remote.product.interfaces.ProductRemoteService;
import com.example.report.remote.product.response.ProductResponse;
import com.example.report.repository.entity.ReportEntity;
import com.example.report.repository.interfaces.ReportRepository;
import com.example.report.service.dto.ReportDTO;
import com.example.report.service.dto.ReportInfoDTO;
import com.example.report.service.dto.ReportStatus;
import com.example.report.service.dto.converter.ReportConverter;
import com.example.report.service.exception.NotFoundReportException;
import com.example.report.service.interfaces.ReportService;
import com.lowagie.text.Document;
import com.lowagie.text.Font;
import com.lowagie.text.Paragraph;
import com.lowagie.text.pdf.PdfName;
import com.lowagie.text.pdf.PdfPTable;
import com.lowagie.text.pdf.PdfString;
import com.lowagie.text.pdf.PdfWriter;
import lombok.NonNull;
import org.springframework.stereotype.Service;
```

```
import java.io.ByteArrayOutputStream;
import java.util.List;
import java.util.Objects;
import java.util.stream.Collectors;

/**
 * @author Patryk Borchowiec
 */
@Service
public class ReportServiceImpl implements ReportService {
    private final ReportRepository reportRepository;
    private final ReportConverter reportConverter;
    private final CompanyRemoteService companyRemoteService;
    private final ProductRemoteService productRemoteService;
    private final Font h1;
    private final Font h2;
    private final Font h3;

    public ReportServiceImpl(
            final ReportRepository reportRepository,
            final ReportConverter reportConverter,
            final CompanyRemoteService companyRemoteService,
            final ProductRemoteService productRemoteService
    ) {
        this.reportRepository = reportRepository;
        this.reportConverter = reportConverter;
        this.companyRemoteService = companyRemoteService;
        this.productRemoteService = productRemoteService;

        h1 = new Font();
        h1.setSize(22);
        h1.setStyle(Font.BOLD);

        h2 = new Font();
        h2.setSize(18);
        h2.setStyle(Font.BOLD);

        h3 = new Font();
        h3.setSize(16);
        h3.setStyle(Font.BOLD);
    }

    @Override
    public void create(final @NonNull ReportDTO reportDTO) {
        final ReportEntity entity = reportRepository.save(reportConverter.toNewEntity(reportDTO));
        final List<CompanyResponse> companies = companyRemoteService.getAll();
        final List<ProductResponse> products = productRemoteService.getAll();

        Document document = new Document();
        final ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
        final PdfWriter instance = PdfWriter.getInstance(document, byteArrayOutputStream);
        document.open();
        instance.getInfo().put(PdfName.CREATOR, new PdfString(Document.getVersion()));

        companies.forEach(company -> addCompanyPage(document, company, products));

        document.close();
        final byte[] content = byteArrayOutputStream.toByteArray();
        entity.setPdfContent(content);
        entity.setStatus(ReportStatus.READY.name());

        reportRepository.save(entity);
    }

    private void addCompanyPage(
            final Document document,
            final CompanyResponse company,
            final List<ProductResponse> products
    ) {
        addProducts(document, company, products);
        addSummary(document, company, products);
        document.newPage();
    }

    private void addProducts(
            final Document document,
            final CompanyResponse company,
            final List<ProductResponse> products
    ) {
        document.add(new Paragraph(company.getName(), h1));
        document.add(new Paragraph("Products", h2));
        document.add(new Paragraph(" ", h2));

        final PdfPTable table = new PdfPTable(3);
        table.addCell(new Paragraph("ID", h3));
        table.addCell(new Paragraph("Name", h3));
        table.addCell(new Paragraph("Amount", h3));

        products.stream()
                .filter(product -> Objects.equals(product.getCompanyId(), company.getId()))
                .forEach(product -> {
                    table.addCell(String.valueOf(product.getId()));
                    table.addCell(product.getName());
                    table.addCell(String.valueOf(product.getAmount()));
```

```
                });
        document.add(table);
    }

    private void addSummary(
            final Document document,
            final CompanyResponse company,
            final List<ProductResponse> products
    ) {
        document.add(new Paragraph("Summary", h2));
        final long numberOfProducts = products.stream()
                .filter(product -> Objects.equals(product.getCompanyId(), company.getId()))
                .count();

        final int summedAmounts = products.stream()
                .filter(product -> Objects.equals(product.getCompanyId(), company.getId()))
                .mapToInt(ProductResponse::getAmount)
                .sum();

        document.add(new Paragraph("Products: \t" + numberOfProducts));
        document.add(new Paragraph("SummedAmounts: \t" + summedAmounts));
    }

    @Override
    public List<ReportInfoDTO> getAll() {
        return reportRepository.findAll().stream().map(reportConverter::toInfoDTO).collect(Collectors.toList());
    }

    @Override
    public ReportDTO getById(final @NonNull Long id) {
        return reportRepository.findById(id)
                .map(reportConverter::toDTO)
                .orElseThrow(() -> new NotFoundReportException(String.format("Not found report with id %s.", id)));
    }

    @Override
    public void update(final @NonNull ReportDTO reportDTO) {
        final ReportEntity reportEntity = reportRepository.findById(reportDTO.getId())
                .orElseThrow(() ->
                        new NotFoundReportException(String.format("Not found report with id %s.", reportDTO.getId()))
                );
        reportEntity.setName(reportDTO.getName());
        reportRepository.save(reportEntity);
    }

    @Override
    public void deleteById(final @NonNull Long id) {
        if (!reportRepository.existsById(id)) {
            throw new NotFoundReportException(String.format("Not found report with id %s.", id));
        }

        reportRepository.deleteById(id);
    }
}
```

Configuration

Now, let's update a version in `report/onteon/conf.yml` file to 1.0.1.

```
app:
  name: 'report'
  version: '1.0.1'
  appType: 'standard'
  procType: 'native'
  processProvider:
    name: 'GenericOsProcessProviderImpl'
    version: '1.0.0'
    executable:
      start:
        command: '${ont_app_path}/bin/bin/catalina.sh run'
        successLine: 'Server startup'
      stop:
        command: '${ont_app_path}/bin/bin/catalina.sh stop'
      terminate:
        command: '${ont_app_path}/bin/bin/catalina.sh stop -force'
  placeHolder:
    name: 'PlaceHolderManagerImpl'
    version: '1.0.0'
    filesToReplace:
      - '${ont_app_path}/bin/conf/server.xml'
    variables:
  serviceRepository:
    healthCheckUrl: 'http://localhost:${ont_port_2}/is-alive'
    entities:
      - entity:
          priority: 1
          port: ${ont_port_2}
          protocol:
```

```
            type: 'HTTP'
            version: '1.1'
          isExternal: false
          isInternal: true
```

Build and upload

Let's build and upload application.

```
$ ./report/scripts/build.sh
$ onteoncli application-registry upload report/target/report.tar.gz
Starting upload session...
Sending part no 0...
Sending part no 1...
Sending part no 2...
Sending part no 3...
Sending part no 4...
Sending part no 5...
Sending part no 6...
Sending part no 7...
Finishing upload session...
uploaded: true

$ onteoncli application list
id                       createdAt                updatedAt                name     version type     processType
624c2f20dbe2ff0e819c73e7 2022-04-05T11:59:28.093Z 2022-04-05T11:59:28.093Z report   1.0.1   standard native
624c1a5bdbe2ff0e819c39b8 2022-04-05T10:30:51.16Z  2022-04-05T10:30:51.16Z  product  1.0.1   standard native
624c1006dbe2ff0e819c1cb9 2022-04-05T09:46:46.512Z 2022-04-05T09:46:46.512Z company  1.1.0   standard native
624c0403dbe2ff0e819bfb8f 2022-04-05T08:55:31.834Z 2022-04-05T08:55:31.834Z frontend 1.0.1   standard native
624bfc3bdbe2ff0e819be365 2022-04-05T08:22:19.29Z  2022-04-05T08:22:19.29Z  gateway  1.0.0   standard native
624bf7f3dbe2ff0e819bd7f8 2022-04-05T08:04:03.947Z 2022-04-05T08:04:03.947Z report   1.0.0   standard native
624bf11bdbe2ff0e819bc849 2022-04-05T07:34:51.56Z  2022-04-05T07:34:51.56Z  product  1.0.0   standard native
624af1418fa5e24f38c418c8 2022-04-04T13:23:13.206Z 2022-04-04T13:23:13.206Z company  1.0.0   standard native
624ae4b48fa5e24f38c4071e 2022-04-04T12:29:40.566Z 2022-04-04T12:29:40.566Z monolith 1.1.0   standard native
624ae0268fa5e24f38c400ce 2022-04-04T12:10:14.809Z 2022-04-04T12:10:14.809Z frontend 1.0.0   standard native
```

Upgrade

Now, let's upgrade old report application with new report application by executing `onteoncli application-change create <old-report-id> <new-report-id>`.

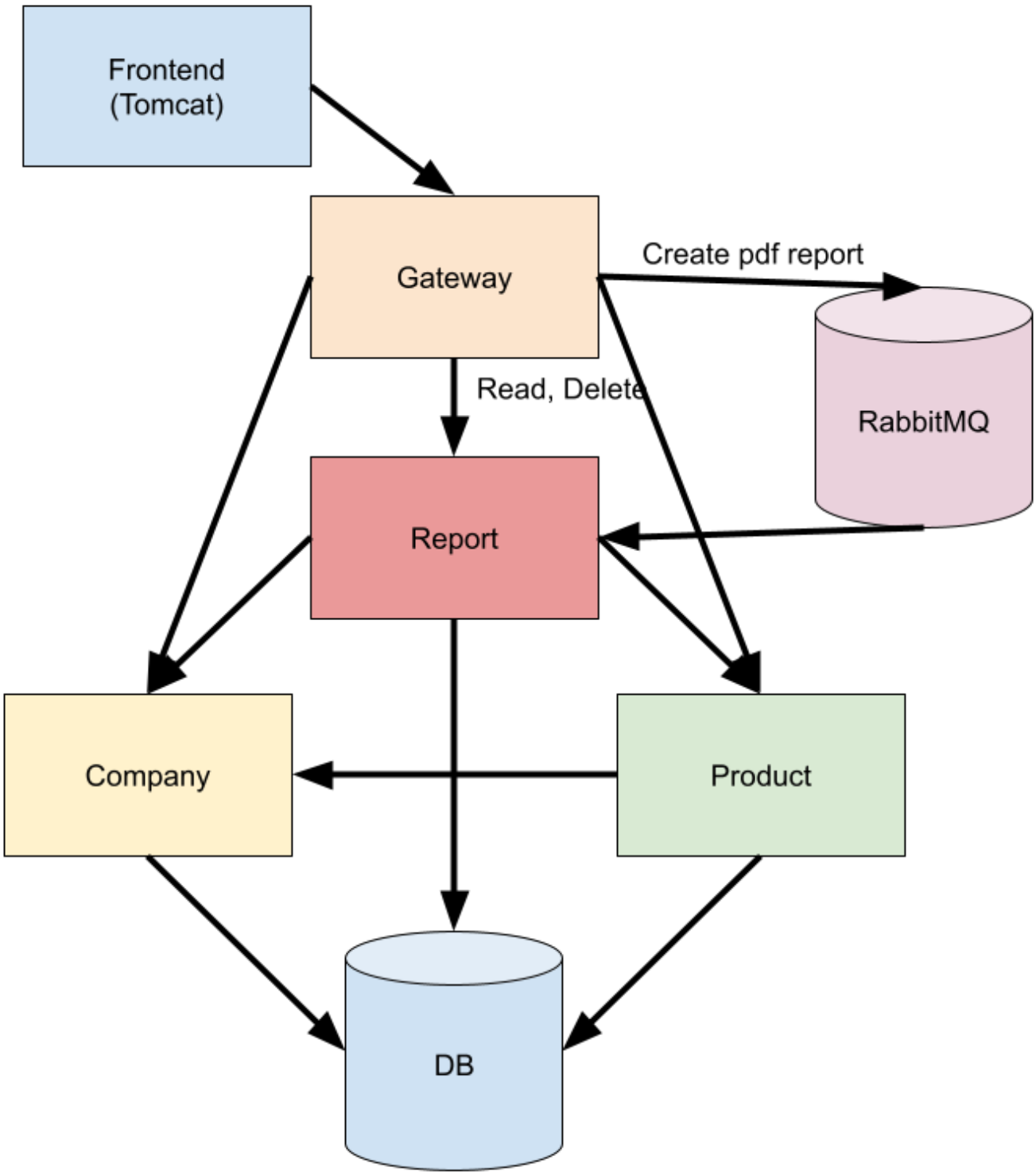
```
$ onteoncli application-change create 624bf7f3dbe2ff0e819bd7f8 624c2f20dbe2ff0e819c73e7
id:               624c2f61dbe2ff0e819c74ac
createdAt:        2022-04-05T12:00:33.162Z
updatedAt:        2022-04-05T12:00:33.162Z
oldApplicationId: 624bf7f3dbe2ff0e819bd7f8
newApplicationId: 624c2f20dbe2ff0e819c73e7

$ watch onteoncli application-instance list --per-page 20
id                       createdAt                applicationName applicationVersion applicationType applicationId            nodeId                   status
a70fce919f4a57e1de257436 2022-04-05T12:01:03.355Z report          1.0.1              standard        624c2f20dbe2ff0e819c73e7 6b43f313f768ce8c1f38c75c running
d4c93f7c3c51340ff5710a9f 2022-04-05T12:00:43.405Z report          1.0.1              standard        624c2f20dbe2ff0e819c73e7 cb2e5e5363d6ba568c85af8b running
5ab67d50d978bd9dff8e9c01 2022-04-05T10:32:48.522Z product         1.0.1              standard        624c1a5bdbe2ff0e819c39b8 cb2e5e5363d6ba568c85af8b running
a6a79e72b4094c0ea4f78808 2022-04-05T10:31:53.349Z product         1.0.1              standard        624c1a5bdbe2ff0e819c39b8 6b43f313f768ce8c1f38c75c running
1f8884be35382c4b697a440d 2022-04-05T09:49:03.358Z company         1.1.0              standard        624c1006dbe2ff0e819c1cb9 cb2e5e5363d6ba568c85af8b running
ccf21b97022ac8da4218059c 2022-04-05T09:48:53.34Z  company         1.1.0              standard        624c1006dbe2ff0e819c1cb9 6b43f313f768ce8c1f38c75c running
f0549eaa190a5e575c287047 2022-04-05T08:59:01.891Z frontend        1.0.1              standard        624c0403dbe2ff0e819bfb8f cb2e5e5363d6ba568c85af8b running
ec3666e8b7dfea1c767c04c1 2022-04-05T08:58:02.832Z frontend        1.0.1              standard        624c0403dbe2ff0e819bfb8f 6b43f313f768ce8c1f38c75c running
a974138e7efa9303539da878 2022-04-05T08:22:54.182Z gateway         1.0.0              standard        624bfc3bdbe2ff0e819be365 6b43f313f768ce8c1f38c75c running
444f79ea15074220618497e7 2022-04-05T08:22:43.16Z  gateway         1.0.0              standard        624bfc3bdbe2ff0e819be365 cb2e5e5363d6ba568c85af8b running
1c43c42be158e5db67c46651 2022-04-05T06:59:05.731Z nginx-edge      1.1.3              embedded        0a1769f83abc3224fe6cb477 cb2e5e5363d6ba568c85af8b running
6aa5b6a12bb5b952f3d836bd 2022-04-05T06:59:05.73Z  nginx-edge      1.1.3              embedded        0a1769f83abc3224fe6cb477 6b43f313f768ce8c1f38c75c running
285af2d5e54f34b365eb92a1 2022-04-05T06:58:59.501Z nginx-inner     1.1.3              embedded        157958ad0cdf14266501ad70 6b43f313f768ce8c1f38c75c running
41aa5a989cf9e7b8750ff828 2022-04-05T06:58:59.5Z   nginx-inner     1.1.3              embedded        157958ad0cdf14266501ad70 cb2e5e5363d6ba568c85af8b running
```

Test

Go to http://localhost:8020/_by_name/frontend/ and test if our application still works.

Step 5 - Asynchronous communication with RabbitMQ

In this step we will add asynchronous communication between gateway and report microservice, with RabbitMQ.



Files

If you don't have files from previous step, you can:

- download **initial** files from here
- download **complete** files from here

RabbitMQ

First, we need to start RabbitMQ. We need to do this with Docker image. To do so, stop the Docker containers and modify the `docker-compose.yml`, by adding the RabbitMQ service.

```
version: "3.8"

services:
  onteoncc-master:
    image: onteon/control-center:1.2.0
    ports:
      - "8050:8050"
      - "9096:9096"
      - "27017:27017"
      - "27018:27018"
      - "27019:27019"
    environment:
      ONTEON_MONITORING_ENABLED: "false"
    command: [ "./start-master.sh" ]
  onteon-node-manager-1:
    image: onteon/node-manager:1.2.0
    privileged: true
    tty: true
    depends_on:
      - "onteoncc-master"
    ports:
      - "8020:8020"
  onteon-node-manager-2:
    image: onteon/node-manager:1.2.0
    privileged: true
    tty: true
    depends_on:
      - "onteoncc-master"
    ports:
      - "8021:8021"
  db:
    image: postgres
    environment:
      POSTGRES_PASSWORD: password
    ports:
      - "5432:5432"
  rabbitmq:
    image: rabbitmq:3.9-management
    ports:
      - "5672:5672"
      - "15672:15672"
  onteoncli:
    profiles: [ "cli-only" ]
    image: onteon/onteoncli-with-demo-apps:1.2.0
    stdin_open: true
    tty: true
    volumes:
      - ".:/opt/onteon"
    depends_on:
      - "onteoncc-master"
```

Then, execute `docker-compose up` and wait a few minutes. Onteon containers creates volumes, so it should create the environment from the previous run.

```
$ watch onteoncli application-instance list --per-page 20
id                       createdAt                applicationName applicationVersion applicationType applicationId            nodeId                   status
14b00f2a969ef53c9128ff24 2022-04-05T12:37:28.847Z gateway         1.0.0              standard        624bfc3bdbe2ff0e819be365 efd21e63b71fe30c9248a873 running
bedf313c22db59189274244a 2022-04-05T12:37:00.154Z product         1.0.1              standard        624c1a5bdbe2ff0e819c39b8 8f493d39501607701243ba1f running
b6ae5fa9b8e4ab41f9221f21 2022-04-05T12:35:28.583Z gateway         1.0.0              standard        624bfc3bdbe2ff0e819be365 8f493d39501607701243ba1f running
4f14f6c9be99ca25433e3f71 2022-04-05T12:35:01.494Z report          1.0.1              standard        624c2f20dbe2ff0e819c73e7 efd21e63b71fe30c9248a873 running
f65e43bf517420b86111b261 2022-04-05T12:34:52.584Z report          1.0.1              standard        624c2f20dbe2ff0e819c73e7 efd21e63b71fe30c9248a873 running
a3ad01d1bcbc41a1b9ffca4c 2022-04-05T12:34:49.823Z company         1.1.0              standard        624c1006dbe2ff0e819c1cb9 8f493d39501607701243ba1f running
51a018364ca90d4c7bcf3e3f 2022-04-05T12:34:38.217Z company         1.1.0              standard        624c1006dbe2ff0e819c1cb9 efd21e63b71fe30c9248a873 running
```

```
2d6328fd02974b4f3de6c495 2022-04-05T12:34:26.544Z product       1.0.1             standard       624c1a5bdbe2ff0e819c39b8 8f493d39501607701243ba1f running
fb3049ad8af56dc1683ada2e 2022-04-05T12:34:03.051Z frontend      1.0.1             standard       624c0403dbe2ff0e819bfb8f 8f493d39501607701243ba1f running
a537b8f44d805d11d6fea6ab 2022-04-05T12:33:53.126Z frontend      1.0.1             standard       624c0403dbe2ff0e819bfb8f efd21e63b71fe30c9248a873 running
01e043ee199cc2574f416ef4 2022-04-05T12:33:33.596Z nginx-edge    1.1.3             embedded       0a1769f83abc3224fe6cb477 8f493d39501607701243ba1f running
3a4cc7fe4263634d3b2d3c1e 2022-04-05T12:33:33.27Z  nginx-edge    1.1.3             embedded       0a1769f83abc3224fe6cb477 efd21e63b71fe30c9248a873 running
e1ba5925a9630f69ac88a156 2022-04-05T12:33:27.422Z nginx-inner   1.1.3             embedded       157958ad0cdf14266501ad70 8f493d39501607701243ba1f running
731f0a31c8c667af1727fc04 2022-04-05T12:33:27.086Z nginx-inner   1.1.3             embedded       157958ad0cdf14266501ad70 efd21e63b71fe30c9248a873 running
```

Gateway

First, we need to add some dependencies to gateway's pom.xml. We need add lombok and spring-boot-starter-amqp.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.6.5</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>

    <groupId>com.example</groupId>
    <artifactId>gateway</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>gateway</name>
    <description>gateway</description>

    <properties>
        <java.version>11</java.version>
        <spring-cloud.version>2021.0.1</spring-cloud.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-gateway</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-amqp</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
        </dependency>
    </dependencies>

    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>org.springframework.cloud</groupId>
                <artifactId>spring-cloud-dependencies</artifactId>
                <version>${spring-cloud.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>

    <build>
        <finalName>${artifactId}</finalName>

        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>
```

Then, we need to specify the Rabbit's host and port in `gateway/src/main/resources/application.properties` file.

```
spring.rabbitmq.host=rabbitmq
spring.rabbitmq.port=5672
```

Next step is to create a RabbitMQ configuration class. Let's create a RabbitmqConfiguration class in `com.example.gateway` package.

```
package com.example.gateway;

import org.springframework.amqp.core.Queue;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

/**
 * @author Patryk Borchowiec
 */
@Configuration
public class RabbitmqConfiguration {
    @Bean
    public Queue queue() {
        return new Queue("createReportQueue");
    }
}
```

Configuration class contains bean with queue that will be used to send messages with new report data.

Now, let's create a CreateReportMessage class in `com.example.gateway.report.remote.message` package.

```
package com.example.gateway.report.remote.message;

import lombok.Value;

/**
 * @author Patryk Borchowiec
 */
@Value
public class CreateReportMessage {
    Long id;
    String name;
}
```

Then, create a ReportRemoteService interface in `com.example.gateway.report.remote.interfaces` package.

```
package com.example.gateway.report.remote.interfaces;

import com.example.gateway.report.remote.message.CreateReportMessage;
import com.fasterxml.jackson.core.JsonProcessingException;
import lombok.NonNull;

/**
 * @author Patryk Borchowiec
 */
public interface ReportRemoteService {
    void create(final @NonNull CreateReportMessage reportDTO) throws JsonProcessingException;
}
```

Then, create an implementation of ReportRemoteService, AsynchronousReportRemoteService in `com.example.gateway.report.remote.impl` package.

```
package com.example.gateway.report.remote.impl;

import com.example.gateway.report.remote.message.CreateReportMessage;
import com.example.gateway.report.remote.interfaces.ReportRemoteService;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.ObjectMapper;
import lombok.NonNull;
import org.springframework.amqp.core.Queue;
import org.springframework.amqp.rabbit.core.RabbitTemplate;
import org.springframework.stereotype.Service;

/**
 * @author Patryk Borchowiec
 */
@Service
public class AsynchronousReportRemoteService implements ReportRemoteService {
    private final RabbitTemplate rabbitTemplate;
    private final Queue queue;
    private final ObjectMapper objectMapper;
```

```
    public AsynchronousReportRemoteService(final RabbitTemplate rabbitTemplate, final Queue queue) {
        this.rabbitTemplate = rabbitTemplate;
        this.queue = queue;
        this.objectMapper = new ObjectMapper();
    }

    @Override
    public void create(final @NonNull CreateReportMessage reportDTO) throws JsonProcessingException {
        rabbitTemplate.convertAndSend(
                queue.getName(),
                objectMapper.writeValueAsString(reportDTO)
        );
    }
}
```

Now, we need to create controller that will call the ReportRemoteService method.

But first, let's create a request class - CreateReportV1Request, in `com.example.gateway.report.controller.pojo.request` package.

```
package com.example.gateway.report.controller.pojo.request;

import lombok.Data;

import javax.validation.constraints.NotBlank;

/**
 * @author Patryk Borchowiec
 */
@Data
public class CreateReportV1Request {
    @NotBlank
    private String name;
}
```

Then create a ReportV1Converter class in `com.example.gateway.report.controller.pojo.converter` package.

```
package com.example.gateway.report.controller.pojo.converter;

import com.example.gateway.report.controller.pojo.request.CreateReportV1Request;
import com.example.gateway.report.remote.message.CreateReportMessage;
import org.springframework.stereotype.Component;

import java.util.Objects;

/**
 * @author Patryk Borchowiec
 */
@Component
public class ReportV1Converter {
    public CreateReportMessage toDTO(final CreateReportV1Request request) {
        if (Objects.isNull(request)) {
            return null;
        }

        return new CreateReportMessage(null, request.getName());
    }
}
```

Finally, let's create a controller class called ReportV1Controller in `com.example.gateway.report.controller` package.

```
package com.example.gateway.report.controller;

import com.example.gateway.report.controller.pojo.converter.ReportV1Converter;
import com.example.gateway.report.controller.pojo.request.CreateReportV1Request;
import com.example.gateway.report.remote.interfaces.ReportRemoteService;
import com.fasterxml.jackson.core.JsonProcessingException;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import javax.validation.Valid;

/**
 * @author Patryk Borchowiec
 */
@RestController
public class ReportV1Controller {
    private final ReportV1Converter reportV1Converter;
    private final ReportRemoteService reportRemoteService;
```

```
    public ReportV1Controller(final ReportV1Converter reportV1Converter, final ReportRemoteService reportRemoteService) {
        this.reportV1Converter = reportV1Converter;
        this.reportRemoteService = reportRemoteService;
    }

    @PostMapping("/reports")
    public void create(final @RequestBody @Valid CreateReportV1Request request) throws JsonProcessingException {
        reportRemoteService.create(reportV1Converter.toDTO(request));
    }
}
```

The last step will be to modify the router, to redirect the "create" requests to controller. Let's modify the GatewayApplication class.

```
package com.example.gateway;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.gateway.route.RouteLocator;
import org.springframework.cloud.gateway.route.builder.RouteLocatorBuilder;
import org.springframework.context.annotation.Bean;
import org.springframework.http.HttpMethod;

@SpringBootApplication
public class GatewayApplication {

    public static void main(String[] args) {
        SpringApplication.run(GatewayApplication.class, args);
    }

    @Bean
    public RouteLocator routeLocator(final RouteLocatorBuilder builder) {
        final String innerEdgeAddress = "http://localhost:8021";

        return builder.routes()
                .route("product_api", r -> r.path("/api/v1/products")
                        .filters(f -> f.rewritePath(
                                "/api/v1/products",
                                "/_by_name/product/api/v1/products"
                        ))
                        .uri(innerEdgeAddress))
                .route("product_rewrite_api", r -> r.path("/api/v1/products/**")
                        .filters(f -> f.rewritePath(
                                "/api/v1/products/(?<segment>.*)",
                                "/_by_name/product/api/v1/products/${segment}"
                        ))
                        .uri(innerEdgeAddress))
                .route("company_api", r -> r.path("/api/v1/companies")
                        .filters(f -> f.rewritePath(
                                "/api/v1/companies",
                                "/_by_name/company/api/v1/companies"
                        ))
                        .uri(innerEdgeAddress))
                .route("company_rewrite_api", r -> r.path("/api/v1/companies/**")
                        .filters(f -> f.rewritePath(
                                "/api/v1/companies/(?<segment>.*)",
                                "/_by_name/company/api/v1/companies/${segment}"
                        ))
                        .uri(innerEdgeAddress))
                .route("create_report_api", r -> r.method(HttpMethod.POST).and().path("/api/v1/reports")
                        .filters(f -> f.rewritePath(
                                "/api/v1/reports",
                                "/_by_name/gateway/reports"
                        ))
                        .uri(innerEdgeAddress))
                .route("report_api", r -> r.method(HttpMethod.POST).negate().and().path("/api/v1/reports")
                        .filters(f -> f.rewritePath(
                                "/api/v1/reports",
                                "/_by_name/report/api/v1/reports"
                        ))
                        .uri(innerEdgeAddress))
                .route("report_rewrite_api", r -> r.path("/api/v1/reports/*/**")
                        .filters(f -> f.rewritePath(
                                "/api/v1/reports/(?<segment>.*)",
                                "/_by_name/report/api/v1/reports/${segment}"
                        ))
                        .uri(innerEdgeAddress))
                .build();
    }
}
```

Configuration

Now, let's update a version in `gateway/onteon/conf.yml` file to 1.0.1.

```
app:
  name: 'gateway'
  version: '1.0.1'
  appType: 'standard'
  procType: 'native'
  processProvider:
    name: 'JVMOsProcessProviderImpl'
    version: '1.0.0'
    executable:
      start:
        path: '${ont_app_path}/bin'
        startJvmCommand: 'java -jar -Dserver.port=${ont_port_1}'
        successLine: 'Started GatewayApplication'
        executableFileName: 'gateway.jar'
  placeHolder:
    name: 'PlaceHolderManagerImpl'
    version: '1.0.0'
    filesToReplace:
    variables:
  serviceRepository:
    healthCheckUrl: 'http://localhost:${ont_port_1}/is-alive'
    entities:
      - entity:
          priority: 1
          port: ${ont_port_1}
          protocol:
            type: 'HTTP'
            version: '1.1'
          isExternal: true
          isInternal: true
```

Build and upload

Let's build and upload application.

```
$ ./gateway/scripts/build.sh
$ onteoncli application-registry upload gateway/target/gateway.tar.gz
Starting upload session...
Sending part no 0...
Sending part no 1...
Sending part no 2...
Sending part no 3...
Sending part no 4...
Sending part no 5...
Finishing upload session...
uploaded: true

$ onteoncli application list
id                       createdAt                updatedAt                name     version type     processType
624c3e47bfc5c34b20a5985b 2022-04-05T13:04:07.884Z 2022-04-05T13:04:07.884Z gateway  1.0.1   standard native
624c2f20dbe2ff0e819c73e7 2022-04-05T11:59:28.093Z 2022-04-05T11:59:28.093Z report   1.0.1   standard native
624c1a5bdbe2ff0e819c39b8 2022-04-05T10:30:51.16Z  2022-04-05T10:30:51.16Z  product  1.0.1   standard native
624c1006dbe2ff0e819c1cb9 2022-04-05T09:46:46.512Z 2022-04-05T09:46:46.512Z company  1.1.0   standard native
624c0403dbe2ff0e819bfb8f 2022-04-05T08:55:31.834Z 2022-04-05T08:55:31.834Z frontend 1.0.1   standard native
624bfc3bdbe2ff0e819be365 2022-04-05T08:22:19.29Z  2022-04-05T08:22:19.29Z  gateway  1.0.0   standard native
624bf7f3dbe2ff0e819bd7f8 2022-04-05T08:04:03.947Z 2022-04-05T08:04:03.947Z report   1.0.0   standard native
624bf11bdbe2ff0e819bc849 2022-04-05T07:34:51.56Z  2022-04-05T07:34:51.56Z  product  1.0.0   standard native
624af1418fa5e24f38c418c8 2022-04-04T13:23:13.206Z 2022-04-04T13:23:13.206Z company  1.0.0   standard native
624ae4b48fa5e24f38c4071e 2022-04-04T12:29:40.566Z 2022-04-04T12:29:40.566Z monolith 1.1.0   standard native
```

Upgrade

Now, let's upgrade old product application with new product application by executing `onteoncli application-change create <old-gateway-id> <new-gateway-id>`.

```
$ onteoncli application-change create 624bfc3bdbe2ff0e819be365 624c3e47bfc5c34b20a5985b
id:               624c3e64bfc5c34b20a598aa
createdAt:        2022-04-05T13:04:36.652Z
updatedAt:        2022-04-05T13:04:36.652Z
oldApplicationId: 624bfc3bdbe2ff0e819be365
newApplicationId: 624c3e47bfc5c34b20a5985b

$ watch onteoncli application-instance list --per-page 20
id                       createdAt                applicationName applicationVersion applicationType applicationId           nodeId                   status
5ab67d50d978bd9dff8e9c01 2022-04-05T10:32:48.522Z product         1.0.1              standard        624c1a5bdbe2ff0e819c39b8 cb2e5e5363d6ba568c85af8b running
a6a79e72b4094c0ea4f78808 2022-04-05T10:31:53.349Z product         1.0.1              standard        624c1a5bdbe2ff0e819c39b8 6b43f313f768ce8c1f38c75c running
1f8884be35382c4b697a440d 2022-04-05T09:49:03.358Z company         1.1.0              standard        624c1006dbe2ff0e819c1cb9 cb2e5e5363d6ba568c85af8b running
ccf21b97022ac8da4218059c 2022-04-05T09:48:53.34Z  company         1.1.0              standard        624c1006dbe2ff0e819c1cb9 6b43f313f768ce8c1f38c75c running
f0549eaa190a5e575c287047 2022-04-05T08:59:01.891Z frontend        1.0.1              standard        624c0403dbe2ff0e819bfb8f cb2e5e5363d6ba568c85af8b running
ec3666e8b7dfea1c767c04c1 2022-04-05T08:58:02.832Z frontend        1.0.1              standard        624c0403dbe2ff0e819bfb8f 6b43f313f768ce8c1f38c75c running
```

```
a974138e7efa9303539da878 2022-04-05T08:22:54.182Z gateway         1.0.0             standard        624bfc3bdbe2ff0e819be365 6b43f313f768ce8c1f38c75c running
444f79ea15074220618497e7 2022-04-05T08:22:43.16Z  gateway         1.0.0             standard        624bfc3bdbe2ff0e819be365 cb2e5e5363d6ba568c85af8b running
c023a855c66d5652af72570f 2022-04-05T08:04:55.742Z report          1.0.0             standard        624bf7f3dbe2ff0e819bd7f8 6b43f313f768ce8c1f38c75c running
1675c3d1a7c3c73328e08776 2022-04-05T08:04:43.466Z report          1.0.0             standard        624bf7f3dbe2ff0e819bd7f8 cb2e5e5363d6ba568c85af8b running
1c43c42be158e5db67c46651 2022-04-05T06:59:05.731Z nginx-edge      1.1.3             embedded        0a1769f83abc3224fe6cb477 cb2e5e5363d6ba568c85af8b running
6aa5b6a12bb5b952f3d836bd 2022-04-05T06:59:05.73Z  nginx-edge      1.1.3             embedded        0a1769f83abc3224fe6cb477 6b43f313f768ce8c1f38c75c running
285af2d5e54f34b365eb92a1 2022-04-05T06:58:59.501Z nginx-inner     1.1.3             embedded        157958ad0cdf14266501ad70 6b43f313f768ce8c1f38c75c running
41aa5a989cf9e7b8750ff828 2022-04-05T06:58:59.5Z   nginx-inner     1.1.3             embedded        157958ad0cdf14266501ad70 cb2e5e5363d6ba568c85af8b running
```

Report

First, let's add a spring-boot-starter-amqp dependency to pom.xml.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.6.5</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>

    <groupId>com.example</groupId>
    <artifactId>report</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>report</name>
    <description>report</description>
    <packaging>war</packaging>

    <properties>
        <java.version>11</java.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-validation</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-amqp</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-tomcat</artifactId>
            <scope>provided</scope>
        </dependency>

        <dependency>
            <groupId>com.github.librepdf</groupId>
            <artifactId>openpdf</artifactId>
            <version>1.3.27</version>
        </dependency>
        <dependency>
            <groupId>com.squareup.okhttp</groupId>
            <artifactId>okhttp</artifactId>
            <version>2.7.5</version>
        </dependency>

        <dependency>
            <groupId>org.postgresql</groupId>
            <artifactId>postgresql</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>
```

```
    <build>
        <finalName>${artifactId}</finalName>

        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <configuration>
                    <excludes>
                        <exclude>
                            <groupId>org.projectlombok</groupId>
                            <artifactId>lombok</artifactId>
                        </exclude>
                    </excludes>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

Then, we need to specify the Rabbit's host and port in `report/src/main/resources/application.properties` file.

```
spring.datasource.url= jdbc:postgresql://db:5432/postgres
spring.datasource.username=postgres
spring.datasource.password=password
spring.jpa.hibernate.ddl-auto=update
spring.jpa.database-platform = org.hibernate.dialect.PostgreSQL94Dialect
spring.rabbitmq.host=rabbitmq
spring.rabbitmq.port=5672
```

Since we will use a RabbitMQ to create a reports, we can delete an endpoint from controller. Let's go to ReportV1Controller and delete the `create` method.

```
package com.example.report.api.v1.controller;

import com.example.report.api.v1.pojo.converter.ReportV1Converter;
import com.example.report.api.v1.pojo.request.UpdateReportV1Request;
import com.example.report.api.v1.pojo.response.ReportV1Response;
import com.example.report.service.dto.ReportDTO;
import com.example.report.service.exception.NotFoundReportException;
import com.example.report.service.interfaces.ReportService;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.server.ResponseStatusException;

import javax.validation.Valid;
import java.util.List;
import java.util.stream.Collectors;

/**
 * @author Patryk Borchowiec
 */
@RestController
@RequestMapping("/api/v1/reports")
public class ReportV1Controller {
    private final ReportService reportService;
    private final ReportV1Converter reportV1Converter;

    public ReportV1Controller(final ReportService reportService, final ReportV1Converter reportV1Converter) {
        this.reportService = reportService;
        this.reportV1Converter = reportV1Converter;
    }

    @GetMapping(produces = MediaType.APPLICATION_JSON_VALUE)
    public List<ReportV1Response> getAll() {
        return reportService.getAll().stream().map(reportV1Converter::toResponse).collect(Collectors.toList());
    }

    @GetMapping(path = "/{id}", produces = MediaType.APPLICATION_PDF_VALUE)
    public ResponseEntity<byte[]> getById(final @PathVariable Long id) {
        final ReportDTO reportDTO;
        try {
            reportDTO = reportService.getById(id);
        } catch (final NotFoundReportException e) {
            throw new ResponseStatusException(
                    HttpStatus.NOT_FOUND,
                    String.format("Cannot return report due to: %s", e.getMessage())
            );
        }

        final String filename = reportDTO.getName().replaceAll(" ", "_") + ".pdf";
        final HttpHeaders headers = new HttpHeaders();
```

```
        headers.setContentType(MediaType.APPLICATION_PDF);
        headers.setContentDispositionFormData(filename, filename);
        headers.setCacheControl("must-revalidate, post-check=0, pre-check=0");

        return new ResponseEntity<>(reportDTO.getPdfContent(), headers, HttpStatus.OK);
    }

    @PutMapping(path = "/{id}")
    public void update(final @RequestBody @Valid UpdateReportV1Request request, final @PathVariable long id) {
        try {
            reportService.update(reportV1Converter.toDTO(request, id));
        } catch (final NotFoundReportException e) {
            throw new ResponseStatusException(
                    HttpStatus.NOT_FOUND,
                    String.format("Cannot update report due to: %s", e.getMessage())
            );
        }
    }

    @DeleteMapping("/{id}")
    public void delete(final @PathVariable long id) {
        try {
            reportService.deleteById(id);
        } catch (final NotFoundReportException e) {
            throw new ResponseStatusException(
                    HttpStatus.NOT_FOUND,
                    String.format("Cannot delete report due to: %s", e.getMessage())
            );
        }
    }
}
```

You can also delete the CreateReportV1Request class.

Now, go to ReportV1Converter and delete the `toDTO` method.

```
package com.example.report.api.v1.pojo.converter;

import com.example.report.api.v1.pojo.request.UpdateReportV1Request;
import com.example.report.api.v1.pojo.response.ReportV1Response;
import com.example.report.service.dto.ReportDTO;
import com.example.report.service.dto.ReportInfoDTO;
import org.springframework.stereotype.Component;

import java.util.Objects;

/**
 * @author Patryk Borchowiec
 */
@Component
public class ReportV1Converter {

    public ReportV1Response toResponse(final ReportInfoDTO reportInfoDTO) {
        if (Objects.isNull(reportInfoDTO)) {
            return null;
        }

        return new ReportV1Response(reportInfoDTO.getId(), reportInfoDTO.getName(), reportInfoDTO.getStatus().name());
    }

    public ReportDTO toDTO(final UpdateReportV1Request request, final long id) {
        if (Objects.isNull(request)) {
            return null;
        }

        return new ReportDTO(id, request.getName(), null, null);
    }
}
```

Now, we will create a receiver that will receive the messages from queue. But first, let's create CreateReportMessage in `com.example.report.api.queue.message` package.

```
package com.example.report.api.queue.message;

import lombok.Data;

/**
 * @author Patryk Borchowiec
 */
@Data
public class CreateReportMessage {
    private Long id;
    private String name;
}
```

Then, create a ReportMessageConverter class in `com.example.report.api.queue.message.converter` package.

```
package com.example.report.api.queue.message.converter;

import com.example.report.api.queue.message.CreateReportMessage;
import com.example.report.service.dto.ReportDTO;
import org.springframework.stereotype.Component;

import java.util.Objects;

/**
 * @author Patryk Borchowiec
 */
@Component
public class ReportMessageConverter {
    public ReportDTO toDTO(final CreateReportMessage message) {
        if (Objects.isNull(message)) {
            return null;
        }

        return new ReportDTO(null, message.getName(), null, null);
    }
}
```

Now, let's create a receiver class called CreateReportReceiver, in `com.example.report.api.queue.receiver` package.

```
package com.example.report.api.queue.receiver;

import com.example.report.api.queue.message.CreateReportMessage;
import com.example.report.api.queue.message.converter.ReportMessageConverter;
import com.example.report.service.interfaces.ReportService;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.ObjectMapper;
import org.springframework.stereotype.Component;

/**
 * @author Patryk Borchowiec
 */
@Component
public class CreateReportReceiver {
    private final ReportService reportService;
    private final ReportMessageConverter reportMessageConverter;
    private final ObjectMapper objectMapper;

    public CreateReportReceiver(
            final ReportService reportService,
            final ReportMessageConverter reportMessageConverter
    ) {
        this.reportService = reportService;
        this.reportMessageConverter = reportMessageConverter;
        this.objectMapper = new ObjectMapper();
    }

    public void receiveMessage(final String message) throws JsonProcessingException {
        final CreateReportMessage reportMessage = objectMapper.readValue(message, CreateReportMessage.class);
        reportService.create(reportMessageConverter.toDTO(reportMessage));
    }
}
```

Finally, let's create a configuration class called RabbitmqConfiguration, where you will be able to register the receiver. Class can be created in `com.example.report.configuration` package.

```
package com.example.report.configuration;

import com.example.report.api.queue.receiver.CreateReportReceiver;
import org.springframework.amqp.core.Queue;
import org.springframework.amqp.rabbit.connection.ConnectionFactory;
import org.springframework.amqp.rabbit.listener.SimpleMessageListenerContainer;
import org.springframework.amqp.rabbit.listener.adapter.MessageListenerAdapter;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

/**
 * @author Patryk Borchowiec
 */
@Configuration
public class RabbitmqConfiguration {
    @Bean
    public Queue queue() {
        return new Queue("createReportQueue");
    }

    @Bean
```

```
    MessageListenerAdapter listenerAdapter(final CreateReportReceiver createReportReceiver) {
        return new MessageListenerAdapter(createReportReceiver, "receiveMessage");
    }

    @Bean
    SimpleMessageListenerContainer container(
            final ConnectionFactory connectionFactory,
            final MessageListenerAdapter listenerAdapter,
            final Queue queue
    ) {
        final SimpleMessageListenerContainer container = new SimpleMessageListenerContainer();
        container.setConnectionFactory(connectionFactory);
        container.setQueueNames(queue.getName());
        container.setMessageListener(listenerAdapter);
        return container;
    }
}
```

Configuration

Now, let's update a version in `report/onteon/conf.yml` file to 1.1.0.

```
app:
  name: 'report'
  version: '1.1.0'
  appType: 'standard'
  procType: 'native'
  processProvider:
    name: 'JVMOsProcessProviderImpl'
    version: '1.0.0'
    executable:
      start:
        path: '${ont_app_path}/bin'
        startJvmCommand: 'java -jar -Dserver.port=${ont_port_1}'
        successLine: 'Started ReportApplication'
        executableFileName: 'report.jar'
  placeHolder:
    name: 'PlaceHolderManagerImpl'
    version: '1.0.0'
    filesToReplace:
    variables:
  serviceRepository:
    healthCheckUrl: 'http://localhost:${ont_port_1}/is-alive'
    entities:
      - entity:
          priority: 1
          port: ${ont_port_1}
          protocol:
            type: 'HTTP'
            version: '1.1'
          isExternal: false
          isInternal: true
```

Build and upload

Let's build and upload application.

```
$ ./report/scripts/build.sh
$ onteoncli application-registry upload report/target/report.tar.gz
Starting upload session...
Sending part no 0...
Sending part no 1...
Sending part no 2...
Sending part no 3...
Sending part no 4...
Sending part no 5...
Sending part no 6...
Sending part no 7...
Finishing upload session...
uploaded: true

$ onteoncli application list
id                       createdAt                updatedAt                name     version type     processType
624c426ebfc5c34b20a5a39e 2022-04-05T13:21:50.233Z 2022-04-05T13:21:50.233Z report   1.1.0   standard native
624c3e47bfc5c34b20a5985b 2022-04-05T13:04:07.884Z 2022-04-05T13:04:07.884Z gateway  1.0.1   standard native
624c2f20dbe2ff0e819c73e7 2022-04-05T11:59:28.093Z 2022-04-05T11:59:28.093Z report   1.0.1   standard native
624c1a5bdbe2ff0e819c39b8 2022-04-05T10:30:51.16Z  2022-04-05T10:30:51.16Z  product  1.0.1   standard native
624c1006dbe2ff0e819c1cb9 2022-04-05T09:46:46.512Z 2022-04-05T09:46:46.512Z company  1.1.0   standard native
624c0403dbe2ff0e819bfb8f 2022-04-05T08:55:31.834Z 2022-04-05T08:55:31.834Z frontend 1.0.1   standard native
624bfc3bdbe2ff0e819be365 2022-04-05T08:22:19.29Z  2022-04-05T08:22:19.29Z  gateway  1.0.0   standard native
624bf7f3dbe2ff0e819bd7f8 2022-04-05T08:04:03.947Z 2022-04-05T08:04:03.947Z report   1.0.0   standard native
```

```
624bf11bdbe2ff0e819bc849 2022-04-05T07:34:51.56Z  2022-04-05T07:34:51.56Z  product  1.0.0   standard native
624af1418fa5e24f38c418c8 2022-04-04T13:23:13.206Z 2022-04-04T13:23:13.206Z company  1.0.0   standard native
```

Upgrade

Now, let's upgrade old report application with new report application by executing `onteoncli application-change create <old-report-id> <new-report-id>`.

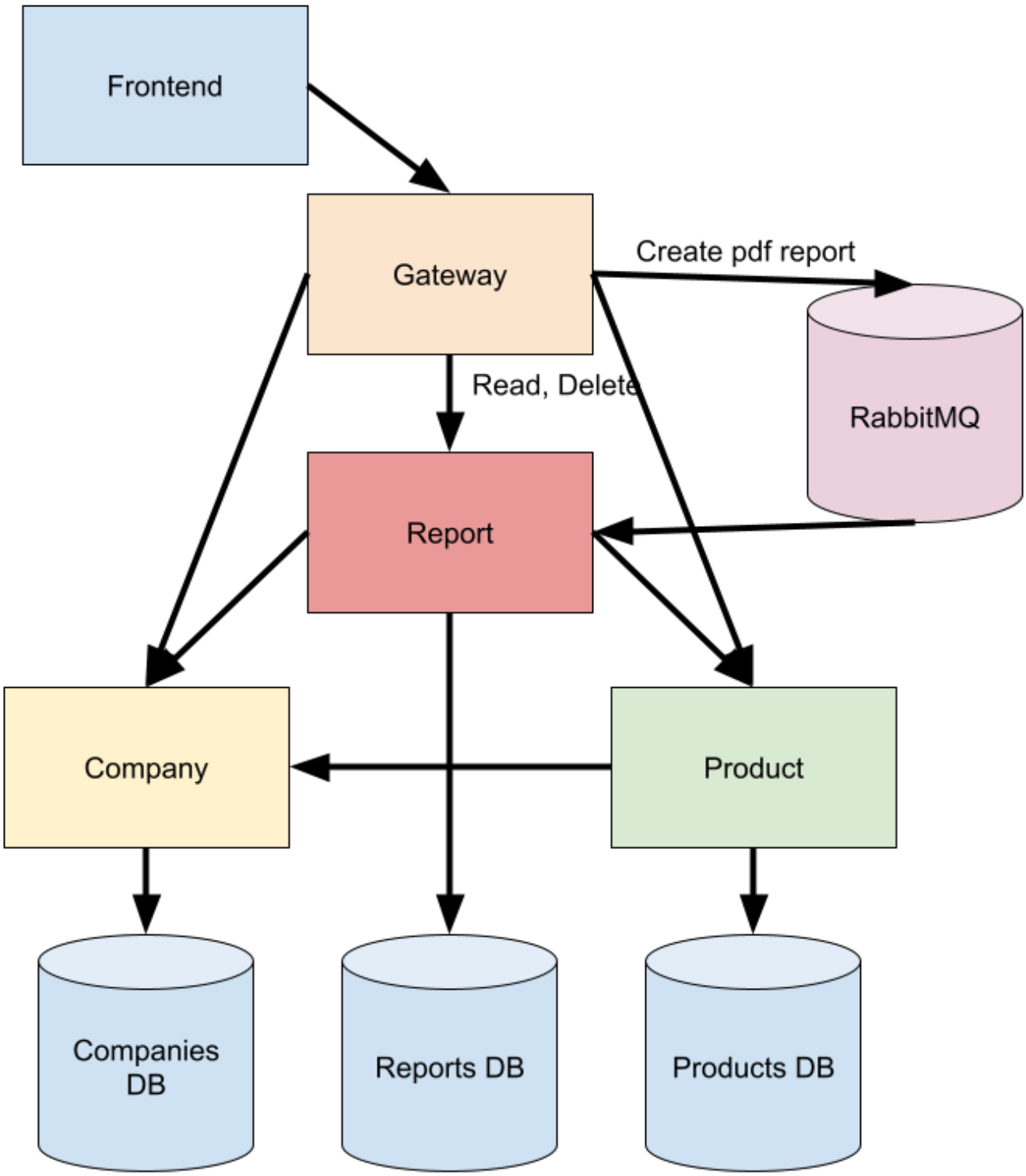
```
$ onteoncli application-change create 624c2f20dbe2ff0e819c73e7 624c426ebfc5c34b20a5a39e
id:               624c42c9bfc5c34b20a5a4a2
createdAt:        2022-04-05T13:23:21.018Z
updatedAt:        2022-04-05T13:23:21.018Z
oldApplicationId: 624c2f20dbe2ff0e819c73e7
newApplicationId: 624c426ebfc5c34b20a5a39e

$ watch onteoncli application-instance list --per-page 20
id                       createdAt                applicationName applicationVersion applicationType applicationId            nodeId                   status
11e927e5550939c0cd9e2822 2022-04-05T13:24:03.477Z report          1.1.0              standard        624c426ebfc5c34b20a5a39e 8f493d39501607701243ba1f running
85182682625d1c52dbe24ef7 2022-04-05T13:23:33.46Z  report          1.1.0              standard        624c426ebfc5c34b20a5a39e efd21e63b71fe30c9248a873 running
5be0239c94fe9b9f35ef7b1e 2022-04-05T13:05:03.31Z  gateway         1.0.1              standard        624c3e47bfc5c34b20a5985b 8f493d39501607701243ba1f running
ae888ce8968d08b5819594aa 2022-04-05T13:04:43.253Z gateway         1.0.1              standard        624c3e47bfc5c34b20a5985b efd21e63b71fe30c9248a873 running
bedf313c22db59189274244a 2022-04-05T12:37:00.154Z product         1.0.1              standard        624c1a5bdbe2ff0e819c39b8 8f493d39501607701243ba1f running
a3ad01d1bcbc41a1b9ffca4c 2022-04-05T12:34:49.823Z company         1.1.0              standard        624c1006dbe2ff0e819c1cb9 8f493d39501607701243ba1f running
51a018364ca90d4c7bcf3e3f 2022-04-05T12:34:38.217Z company         1.1.0              standard        624c1006dbe2ff0e819c1cb9 efd21e63b71fe30c9248a873 running
2d6328fd02974b4f3de6c495 2022-04-05T12:34:26.544Z product         1.0.1              standard        624c1a5bdbe2ff0e819c39b8 8f493d39501607701243ba1f running
fb3049ad8af56dc1683ada2e 2022-04-05T12:34:03.051Z frontend        1.0.1              standard        624c0403dbe2ff0e819bfb8f 8f493d39501607701243ba1f running
a537b8f44d805d11d6fea6ab 2022-04-05T12:33:53.126Z frontend        1.0.1              standard        624c0403dbe2ff0e819bfb8f efd21e63b71fe30c9248a873 running
01e043ee199cc2574f416ef4 2022-04-05T12:33:33.596Z nginx-edge      1.1.3              embedded        0a1769f83abc3224fe6cb477 8f493d39501607701243ba1f running
3a4cc7fe4263634d3b2d3c1e 2022-04-05T12:33:33.27Z  nginx-edge      1.1.3              embedded        0a1769f83abc3224fe6cb477 efd21e63b71fe30c9248a873 running
e1ba5925a9630f69ac88a156 2022-04-05T12:33:27.422Z nginx-inner     1.1.3              embedded        157958ad0cdf14266501ad70 8f493d39501607701243ba1f running
731f0a31c8c667af1727fc04 2022-04-05T12:33:27.086Z nginx-inner     1.1.3              embedded        157958ad0cdf14266501ad70 efd21e63b71fe30c9248a873 running
```

Test

Go to http://localhost:8020/_by_name/frontend/ and test if our application still works.

Step 6 - Extracting databases for each microservice

In this step we will connect microservices to different databases.



Files

If you don't have files from previous step, you can:

- download **initial** files from here
- download **complete** files from here

Databases

First, we need to run more databases. For this, stop current docker containers and modify the docker-compose.yml file.

```
version: "3.8"

services:
  onteoncc-master:
    image: onteon/control-center:1.2.0
    ports:
      - "8050:8050"
      - "9096:9096"
      - "27017:27017"
      - "27018:27018"
      - "27019:27019"
    environment:
      ONTEON_MONITORING_ENABLED: "false"
    command: ["./start-master.sh"]
  onteon-node-manager-1:
    image: onteon/node-manager:1.2.0
    privileged: true
    tty: true
    depends_on:
      - "onteoncc-master"
    ports:
      - "8020:8020"
  onteon-node-manager-2:
    image: onteon/node-manager:1.2.0
    privileged: true
    tty: true
    depends_on:
      - "onteoncc-master"
    ports:
      - "8021:8021"
  company-db:
    image: postgres
    environment:
      POSTGRES_PASSWORD: password
    ports:
      - "5432:5432"
  product-db:
    image: postgres
    environment:
      POSTGRES_PASSWORD: password
    ports:
      - "5433:5432"
  report-db:
    image: postgres
    environment:
      POSTGRES_PASSWORD: password
    ports:
      - "5434:5432"
  rabbitmq:
    image: rabbitmq:3.9-management
    ports:
      - "5672:5672"
      - "15672:15672"
  onteoncli:
    profiles: ["cli-only"]
    image: onteon/onteoncli-with-demo-apps:1.2.0
    stdin_open: true
    tty: true
    volumes:
      - ".:/opt/onteon"
    depends_on:
      - "onteoncc-master"
```

Then, execute `docker-compose up`. For now, there will be problems with applications because we changed the database. But it will be fixed after the upgrade of applications.

Company

Let's modify the communication data to database. To do so, modify the `application.properties` file.

```
spring.datasource.url=jdbc:postgresql://company-db:5432/postgres
spring.datasource.username=postgres
spring.datasource.password=password
spring.jpa.hibernate.ddl-auto=update
spring.jpa.database-platform=org.hibernate.dialect.PostgreSQL94Dialect
```

Let's change version in the conf.yml file to 1.1.1.

```
app:
  name: 'company'
  version: '1.1.1'
  appType: 'standard'
  procType: 'native'
  processProvider:
    name: 'GenericOsProcessProviderImpl'
    version: '1.0.0'
    executable:
      start:
        command: '${ont_app_path}/bin/bin/catalina.sh run'
        successLine: 'Server startup'
      stop:
        command: '${ont_app_path}/bin/bin/catalina.sh stop'
      terminate:
        command: '${ont_app_path}/bin/bin/catalina.sh stop -force'
  placeHolder:
    name: 'PlaceHolderManagerImpl'
    version: '1.0.0'
    filesToReplace:
      - '${ont_app_path}/bin/conf/server.xml'
    variables:
  serviceRepository:
    healthCheckUrl: 'http://localhost:${ont_port_2}/is-alive'
    entities:
      - entity:
          priority: 1
          port: ${ont_port_2}
          protocol:
            type: 'HTTP'
            version: '1.1'
          isExternal: false
          isInternal: true
```

Then build, upload and upgrade application.

```
$ ./company/scripts/build.sh
$ onteoncli application-registry upload company/target/company.tar.gz
Starting upload session...
Sending part no 0...
Sending part no 1...
Sending part no 2...
Sending part no 3...
Sending part no 4...
Sending part no 5...
Sending part no 6...
Finishing upload session...
uploaded: true

$ onteoncli application list
id                       createdAt                updatedAt                name     version type     processType
624c476059719200b1724d71 2022-04-05T13:42:56.211Z 2022-04-05T13:42:56.211Z company  1.1.1   standard native
624c426ebfc5c34b20a5a39e 2022-04-05T13:21:50.233Z 2022-04-05T13:21:50.233Z report   1.1.0   standard native
624c3e47bfc5c34b20a5985b 2022-04-05T13:04:07.884Z 2022-04-05T13:04:07.884Z gateway  1.0.1   standard native
624c2f20dbe2ff0e819c73e7 2022-04-05T11:59:28.093Z 2022-04-05T11:59:28.093Z report   1.0.1   standard native
624c1a5bdbe2ff0e819c39b8 2022-04-05T10:30:51.16Z  2022-04-05T10:30:51.16Z  product  1.0.1   standard native
624c1006dbe2ff0e819c1cb9 2022-04-05T09:46:46.512Z 2022-04-05T09:46:46.512Z company  1.1.0   standard native
624c0403dbe2ff0e819bfb8f 2022-04-05T08:55:31.834Z 2022-04-05T08:55:31.834Z frontend 1.0.1   standard native
624bfc3bdbe2ff0e819be365 2022-04-05T08:22:19.29Z  2022-04-05T08:22:19.29Z  gateway  1.0.0   standard native
624bf7f3dbe2ff0e819bd7f8 2022-04-05T08:04:03.947Z 2022-04-05T08:04:03.947Z report   1.0.0   standard native
624bf11bdbe2ff0e819bc849 2022-04-05T07:34:51.56Z  2022-04-05T07:34:51.56Z  product  1.0.0   standard native

$ onteoncli application-change create 624c1006dbe2ff0e819c1cb9 624c476059719200b1724d71
id:               624c477f59719200b1724da4
```

```
createdAt:        2022-04-05T13:43:27.417Z
updatedAt:        2022-04-05T13:43:27.417Z
oldApplicationId: 624c1006dbe2ff0e819c1cb9
newApplicationId: 624c476059719200b1724d71

$ watch onteoncli application-instance list --per-page 20
id                       createdAt                applicationName applicationVersion applicationType applicationId           nodeId                   status
aec0940d4ecda476c9be4fb0 2022-04-05T13:45:41.026Z report          1.1.0              standard        624c426ebfc5c34b20a5a39e 65508fa6ecdbbdb77029f61a created
766af96e021e3727a93498e0 2022-04-05T13:45:40.121Z product         1.0.1              standard        624c1a5bdbe2ff0e819c39b8 c934603cc1cd848c61700ba3 created
7ff05279e2f78858d6605436 2022-04-05T13:45:30.134Z product         1.0.1              standard        624c1a5bdbe2ff0e819c39b8 65508fa6ecdbbdb77029f61a created
aad8c4ba63531eb081e1dd89 2022-04-05T13:44:30.94Z  company         1.1.1              standard        624c476059719200b1724d71 65508fa6ecdbbdb77029f61a running
33b22e040722b09a89cfcaa0 2022-04-05T13:44:30.916Z company         1.1.1              standard        624c476059719200b1724d71 65508fa6ecdbbdb77029f61a running
cc3e908461df9ea6f418e70e 2022-04-05T13:37:32.218Z gateway         1.0.1              standard        624c3e47bfc5c34b20a5985b c934603cc1cd848c61700ba3 running
b8bfae714b00251dd61bb1f1 2022-04-05T13:37:23.888Z gateway         1.0.1              standard        624c3e47bfc5c34b20a5985b c934603cc1cd848c61700ba3 running
0483bb3fcbc718e959b6e80a 2022-04-05T13:35:31.784Z frontend        1.0.1              standard        624c0403dbe2ff0e819bfb8f c934603cc1cd848c61700ba3 running
c491fc58a72c3465a7c806f1 2022-04-05T13:35:23.18Z  frontend        1.0.1              standard        624c0403dbe2ff0e819bfb8f c934603cc1cd848c61700ba3 running
a12213168c6081955c1ba421 2022-04-05T13:35:04.608Z nginx-edge      1.1.3              embedded        0a1769f83abc3224fe6cb477 65508fa6ecdbbdb77029f61a running
bd4e83eaa93b4f59eb05350e 2022-04-05T13:35:04.446Z nginx-edge      1.1.3              embedded        0a1769f83abc3224fe6cb477 c934603cc1cd848c61700ba3 running
0af31412317ca01b73744b92 2022-04-05T13:34:58.49Z  nginx-inner     1.1.3              embedded        157958ad0cdf14266501ad70 65508fa6ecdbbdb77029f61a running
c8abbce78274f9cf5f492784 2022-04-05T13:34:58.283Z nginx-inner     1.1.3              embedded        157958ad0cdf14266501ad70 c934603cc1cd848c61700ba3 running
```

Product

Let's modify the communication data to database. To do so, modify the `application.properties` file.

```
spring.datasource.url= jdbc:postgresql://product-db:5432/postgres
spring.datasource.username=postgres
spring.datasource.password=password
spring.jpa.hibernate.ddl-auto=update
spring.jpa.database-platform = org.hibernate.dialect.PostgreSQL94Dialect
```

Let's change version in the conf.yml file to 1.0.2.

```
app:
  name: 'product'
  version: '1.0.2'
  appType: 'standard'
  procType: 'native'
  processProvider:
    name: 'GenericOsProcessProviderImpl'
    version: '1.0.0'
    executable:
      start:
        command: '${ont_app_path}/bin/bin/catalina.sh run'
        successLine: 'Server startup'
      stop:
        command: '${ont_app_path}/bin/bin/catalina.sh stop'
      terminate:
        command: '${ont_app_path}/bin/bin/catalina.sh stop -force'
  placeHolder:
    name: 'PlaceHolderManagerImpl'
    version: '1.0.0'
    filesToReplace:
      - '${ont_app_path}/bin/conf/server.xml'
    variables:
  serviceRepository:
    healthCheckUrl: 'http://localhost:${ont_port_2}/is-alive'
    entities:
      - entity:
          priority: 1
          port: ${ont_port_2}
          protocol:
            type: 'HTTP'
            version: '1.1'
          isExternal: false
          isInternal: true
```

Then build, upload and upgrade application.

```
$ ./product/scripts/build.sh
$ onteoncli application-registry upload product/target/product.tar.gz
Starting upload session...
Sending part no 0...
Sending part no 1...
Sending part no 2...
Sending part no 3...
Sending part no 4...
Sending part no 5...
Sending part no 6...
Finishing upload session...
uploaded: true
```

```
$ onteoncli application list
id                       createdAt                updatedAt                name     version type     processType
624c482659719200b1724eda 2022-04-05T13:46:14.073Z 2022-04-05T13:46:14.073Z product  1.0.2   standard native
624c476059719200b1724d71 2022-04-05T13:42:56.211Z 2022-04-05T13:42:56.211Z company  1.1.1   standard native
624c426ebfc5c34b20a5a39e 2022-04-05T13:21:50.233Z 2022-04-05T13:21:50.233Z report   1.1.0   standard native
624c3e47bfc5c34b20a5985b 2022-04-05T13:04:07.884Z 2022-04-05T13:04:07.884Z gateway  1.0.1   standard native
624c2f20dbe2ff0e819c73e7 2022-04-05T11:59:28.093Z 2022-04-05T11:59:28.093Z report   1.0.1   standard native
624c1a5bdbe2ff0e819c39b8 2022-04-05T10:30:51.16Z  2022-04-05T10:30:51.16Z  product  1.0.1   standard native
624c1006dbe2ff0e819c1cb9 2022-04-05T09:46:46.512Z 2022-04-05T09:46:46.512Z company  1.1.0   standard native
624c0403dbe2ff0e819bfb8f 2022-04-05T08:55:31.834Z 2022-04-05T08:55:31.834Z frontend 1.0.1   standard native
624bfc3bdbe2ff0e819be365 2022-04-05T08:22:19.29Z  2022-04-05T08:22:19.29Z  gateway  1.0.0   standard native
624bf7f3dbe2ff0e819bd7f8 2022-04-05T08:04:03.947Z 2022-04-05T08:04:03.947Z report   1.0.0   standard native

$ onteoncli application-change create 624c1a5bdbe2ff0e819c39b8 624c482659719200b1724eda
id:               624c484259719200b1724f15
createdAt:        2022-04-05T13:46:42.578Z
updatedAt:        2022-04-05T13:46:42.578Z
oldApplicationId: 624c1a5bdbe2ff0e819c39b8
newApplicationId: 624c482659719200b1724eda

$ watch onteoncli application-instance list --per-page 20
id                       createdAt                applicationName applicationVersion applicationType applicationId            nodeId                   status
6bcc8a689d1cb082462ca823 2022-04-05T13:48:21.108Z report          1.1.0              standard        624c426ebfc5c34b20a5a39e 65508fa6ecdbbdb77029f61a created
fa3a2b62d3f90d2f9e4d41ad 2022-04-05T13:47:22.573Z product         1.0.2              standard        624c482659719200b1724eda 65508fa6ecdbbdb77029f61a running
726b19dc81ba63cf0897efdb 2022-04-05T13:47:13.612Z product         1.0.2              standard        624c482659719200b1724eda 65508fa6ecdbbdb77029f61a running
aad8c4ba63531eb081e1dd89 2022-04-05T13:44:30.94Z  company         1.1.1              standard        624c476059719200b1724d71 65508fa6ecdbbdb77029f61a running
33b22e040722b09a89cfcaa0 2022-04-05T13:44:30.916Z company         1.1.1              standard        624c476059719200b1724d71 65508fa6ecdbbdb77029f61a running
cc3e908461df9ea6f418e70e 2022-04-05T13:37:32.218Z gateway         1.0.1              standard        624c3e47bfc5c34b20a5985b c934603cc1cd848c61700ba3 running
b8bfae714b00251dd61bb1f1 2022-04-05T13:37:23.888Z gateway         1.0.1              standard        624c3e47bfc5c34b20a5985b c934603cc1cd848c61700ba3 running
0483bb3fcbc718e959b6e80a 2022-04-05T13:35:31.784Z frontend        1.0.1              standard        624c0403dbe2ff0e819bfb8f c934603cc1cd848c61700ba3 running
c491fc58a72c3465a7c806f1 2022-04-05T13:35:23.18Z  frontend        1.0.1              standard        624c0403dbe2ff0e819bfb8f c934603cc1cd848c61700ba3 running
a12213168c6081955c1ba421 2022-04-05T13:35:04.608Z nginx-edge      1.1.3              embedded        0a1769f83abc3224fe6cb477 65508fa6ecdbbdb77029f61a running
bd4e83eaa93b4f59eb05350e 2022-04-05T13:35:04.446Z nginx-edge      1.1.3              embedded        0a1769f83abc3224fe6cb477 c934603cc1cd848c61700ba3 running
0af31412317ca01b73744b92 2022-04-05T13:34:58.49Z  nginx-inner     1.1.3              embedded        157958ad0cdf14266501ad70 65508fa6ecdbbdb77029f61a running
c8abbce78274f9cf5f492784 2022-04-05T13:34:58.283Z nginx-inner     1.1.3              embedded        157958ad0cdf14266501ad70 c934603cc1cd848c61700ba3 running
```

Report

Let's modify the communication data to database. To do so, modify the `application.properties` file.

```
spring.datasource.url= jdbc:postgresql://report-db:5432/postgres
spring.datasource.username=postgres
spring.datasource.password=password
spring.jpa.hibernate.ddl-auto=update
spring.jpa.database-platform = org.hibernate.dialect.PostgreSQL94Dialect
spring.rabbitmq.host=rabbitmq
spring.rabbitmq.port=5672
```

Let's change version in the conf.yml file to 1.1.1.

```
app:
  name: 'report'
  version: '1.1.1'
  appType: 'standard'
  procType: 'native'
  processProvider:
    name: 'GenericOsProcessProviderImpl'
    version: '1.0.0'
    executable:
      start:
        command: '${ont_app_path}/bin/bin/catalina.sh run'
        successLine: 'Server startup'
      stop:
        command: '${ont_app_path}/bin/bin/catalina.sh stop'
      terminate:
        command: '${ont_app_path}/bin/bin/catalina.sh stop -force'
  placeHolder:
    name: 'PlaceHolderManagerImpl'
    version: '1.0.0'
    filesToReplace:
      - '${ont_app_path}/bin/conf/server.xml'
    variables:
  serviceRepository:
    healthCheckUrl: 'http://localhost:${ont_port_2}/is-alive'
    entities:
      - entity:
          priority: 1
          port: ${ont_port_2}
          protocol:
            type: 'HTTP'
            version: '1.1'
          isExternal: false
          isInternal: true
```

Then build, upload and upgrade application.

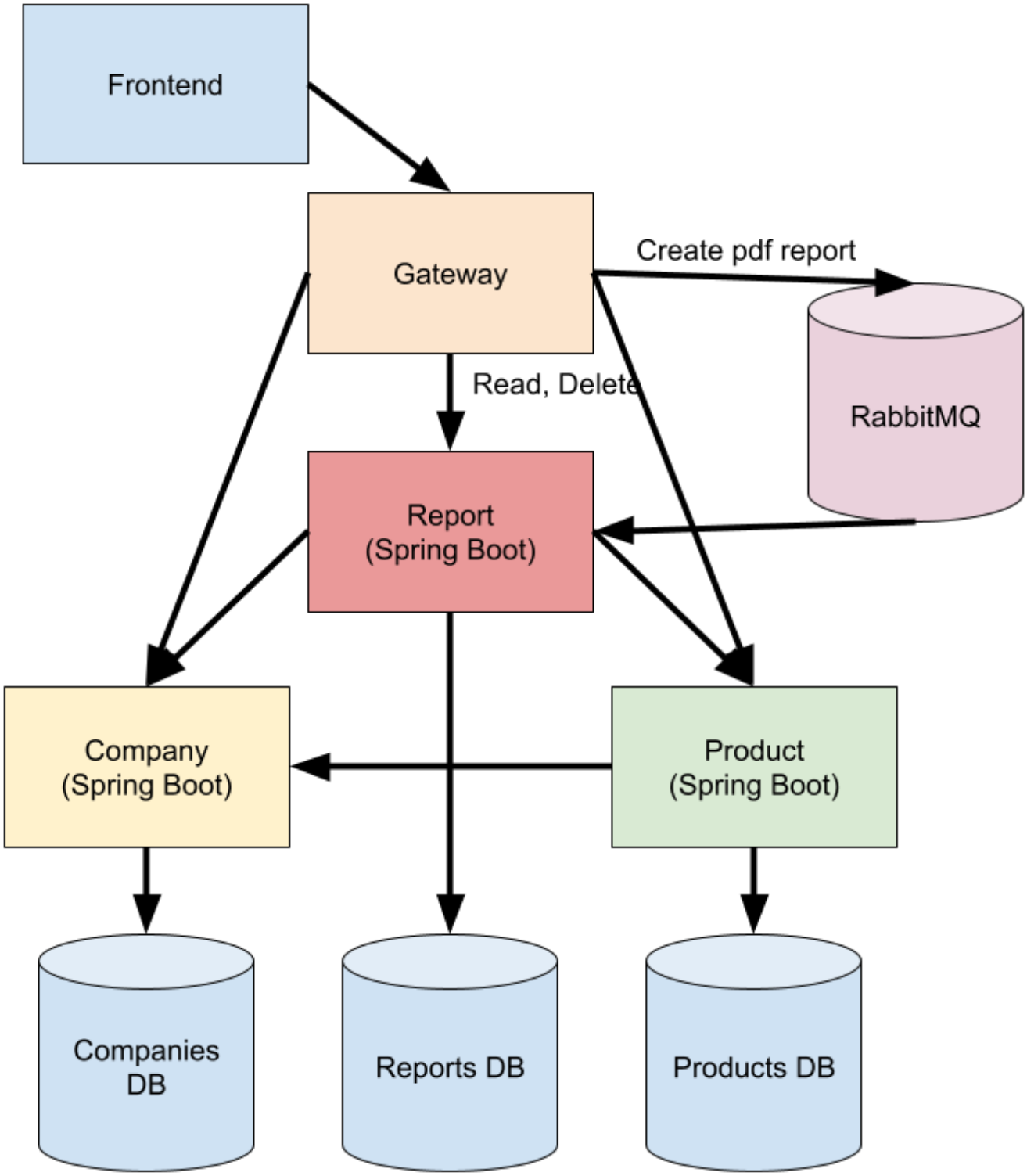
```
$ ./report/scripts/build.sh
$ onteoncli application-registry upload report/target/report.tar.gz
Starting upload session...
Sending part no 0...
Sending part no 1...
Sending part no 2...
Sending part no 3...
Sending part no 4...
Sending part no 5...
Sending part no 6...
Sending part no 7...
Finishing upload session...
uploaded: true

$ onteoncli application list
id                       createdAt                updatedAt                name     version type     processType
624c48e959719200b1725072 2022-04-05T13:49:29.343Z 2022-04-05T13:49:29.343Z report   1.1.1   standard native
624c482659719200b1724eda 2022-04-05T13:46:14.073Z 2022-04-05T13:46:14.073Z product  1.0.2   standard native
624c476059719200b1724d71 2022-04-05T13:42:56.211Z 2022-04-05T13:42:56.211Z company  1.1.1   standard native
624c426ebfc5c34b20a5a39e 2022-04-05T13:21:50.233Z 2022-04-05T13:21:50.233Z report   1.1.0   standard native
624c3e47bfc5c34b20a5985b 2022-04-05T13:04:07.884Z 2022-04-05T13:04:07.884Z gateway  1.0.1   standard native
624c2f20dbe2ff0e819c73e7 2022-04-05T11:59:28.093Z 2022-04-05T11:59:28.093Z report   1.0.1   standard native
624c1a5bdbe2ff0e819c39b8 2022-04-05T10:30:51.16Z  2022-04-05T10:30:51.16Z  product  1.0.1   standard native
624c1006dbe2ff0e819c1cb9 2022-04-05T09:46:46.512Z 2022-04-05T09:46:46.512Z company  1.1.0   standard native
624c0403dbe2ff0e819bfb8f 2022-04-05T08:55:31.834Z 2022-04-05T08:55:31.834Z frontend 1.0.1   standard native
624bfc3bdbe2ff0e819be365 2022-04-05T08:22:19.29Z  2022-04-05T08:22:19.29Z  gateway  1.0.0   standard native

$ onteoncli application-change create 624c426ebfc5c34b20a5a39e 624c48e959719200b1725072
id:               624c490659719200b17250b8
createdAt:        2022-04-05T13:49:58.913Z
updatedAt:        2022-04-05T13:49:58.913Z
oldApplicationId: 624c426ebfc5c34b20a5a39e
newApplicationId: 624c48e959719200b1725072

$ watch onteoncli application-instance list --per-page 20
id                       createdAt                applicationName applicationVersion applicationType applicationId            nodeId                   status
82e3974c288c00f24d5332bf 2022-04-05T13:51:10.08Z  report          1.1.1              standard        624c48e959719200b1725072 c934603cc1cd848c61700ba3 running
8a73951c5c6ee2afe2e30f80 2022-04-05T13:51:03.303Z report          1.1.1              standard        624c48e959719200b1725072 65508fa6ecdbbdb77029f61a running
9d44cab0d72a6ba098abcd93 2022-04-05T13:49:00.078Z product         1.0.2              standard        624c482659719200b1724eda 65508fa6ecdbbdb77029f61a running
726b19dc81ba63cf0897efdb 2022-04-05T13:47:13.612Z product         1.0.2              standard        624c482659719200b1724eda 65508fa6ecdbbdb77029f61a running
aad8c4ba63531eb081e1dd89 2022-04-05T13:44:30.94Z  company         1.1.1              standard        624c476059719200b1724d71 65508fa6ecdbbdb77029f61a running
33b22e040722b09a89cfcaa0 2022-04-05T13:44:30.916Z company         1.1.1              standard        624c476059719200b1724d71 65508fa6ecdbbdb77029f61a running
cc3e908461df9ea6f418e70e 2022-04-05T13:37:32.218Z gateway         1.0.1              standard        624c3e47bfc5c34b20a5985b c934603cc1cd848c61700ba3 running
b8bfae714b00251dd61bb1f1 2022-04-05T13:37:23.888Z gateway         1.0.1              standard        624c3e47bfc5c34b20a5985b c934603cc1cd848c61700ba3 running
0483bb3fcbc718e959b6e80a 2022-04-05T13:35:31.784Z frontend        1.0.1              standard        624c0403dbe2ff0e819bfb8f c934603cc1cd848c61700ba3 running
c491fc58a72c3465a7c806f1 2022-04-05T13:35:23.18Z  frontend        1.0.1              standard        624c0403dbe2ff0e819bfb8f c934603cc1cd848c61700ba3 running
a12213168c6081955c1ba421 2022-04-05T13:35:04.608Z nginx-edge      1.1.3              embedded        0a1769f83abc3224fe6cb477 65508fa6ecdbbdb77029f61a running
bd4e83eaa93b4f59eb05350e 2022-04-05T13:35:04.446Z nginx-edge      1.1.3              embedded        0a1769f83abc3224fe6cb477 c934603cc1cd848c61700ba3 running
0af31412317ca01b73744b92 2022-04-05T13:34:58.49Z  nginx-inner     1.1.3              embedded        157958ad0cdf14266501ad70 65508fa6ecdbbdb77029f61a running
c8abbce78274f9cf5f492784 2022-04-05T13:34:58.283Z nginx-inner     1.1.3              embedded        157958ad0cdf14266501ad70 c934603cc1cd848c61700ba3 running
```

Test

Go to http://localhost:8020/_by_name/frontend/ and test if our application still works.

Step 7 - Moving microservices from Tomcat to Spring Boot

In this step we move from Tomcat microservices to Spring Boot microservices.



Files

If you don't have files from previous step, you can:

- download **initial** files from here
- download **complete** files from here

Company

First, let's remove the packaging and Tomcat dependency from pom.xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.6.5</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>

    <groupId>com.example</groupId>
    <artifactId>company</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>company</name>
    <description>company</description>

    <properties>
        <java.version>11</java.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-validation</artifactId>
        </dependency>

        <dependency>
            <groupId>org.postgresql</groupId>
            <artifactId>postgresql</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <finalName>${artifactId}</finalName>

        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <configuration>
                    <excludes>
                        <exclude>
                            <groupId>org.projectlombok</groupId>
                            <artifactId>lombok</artifactId>
                        </exclude>
                    </excludes>
                </configuration>
            </plugin>
        </plugins>
```

```
    </build>
</project>
```

Then, go to CompanyApplication class, remove extending the SpringBootServletInitializer class.

```
package com.example.company;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class CompanyApplication {

    public static void main(String[] args) {
        SpringApplication.run(CompanyApplication.class, args);
    }

}
```

Then, remove the `tomcat` dir. We won't need Tomcat files.

Now, let's modify a `conf.yml` file. We need to change the process provider to JVM.

```
app:
  name: 'company'
  version: '1.1.2'
  appType: 'standard'
  procType: 'native'
  processProvider:
    name: 'JVMOsProcessProviderImpl'
    version: '1.0.0'
    executable:
      start:
        path: '${ont_app_path}/bin'
        startJvmCommand: 'java -jar -Dserver.port=${ont_port_1}'
        successLine: 'Started CompanyApplication'
        executableFileName: 'company.jar'
  placeHolder:
    name: 'PlaceHolderManagerImpl'
    version: '1.0.0'
    filesToReplace:
    variables:
  serviceRepository:
    healthCheckUrl: 'http://localhost:${ont_port_1}/is-alive'
    entities:
      - entity:
          priority: 1
          port: ${ont_port_1}
          protocol:
            type: 'HTTP'
            version: '1.1'
          isExternal: false
          isInternal: true
```

Then, modify the build.sh script.

```
#!/usr/bin/env bash

set -e

SOURCE="${BASH_SOURCE[0]}"
while [ -h "$SOURCE" ]; do
  DIR="$(cd -P "$( dirname "$SOURCE" )" >/dev/null 2>&1 && pwd)"
  SOURCE="$(readlink "$SOURCE")"
  [[ $SOURCE != /* ]] && SOURCE="$DIR/$SOURCE"
done
SCRIPT_DIR="$(cd -P "$(dirname "$SOURCE")" >/dev/null 2>&1 && pwd)"
ROOT_DIR="${SCRIPT_DIR}/.."
TARGET_DIR="${ROOT_DIR}/target"
APP_BIN_DIR="${TARGET_DIR}/app/bin"
APP_CONF_DIR="${TARGET_DIR}/app/conf"

cd "${ROOT_DIR}" && mvn clean package
mkdir -p "${APP_BIN_DIR}"
mkdir -p "${APP_CONF_DIR}"
cp -r "${TARGET_DIR}/company.jar" "${APP_BIN_DIR}"
cp "${ROOT_DIR}/onteon/conf.yml" "${APP_CONF_DIR}"
cd "${TARGET_DIR}/app" && tar -zcvf "${TARGET_DIR}/company.tar.gz" *
```

Finally, let's build, upload and upgrade application.

```
$ ./company/scripts/build.sh
$ onteoncli application-registry upload company/target/company.tar.gz
Starting upload session...
Sending part no 0...
Sending part no 1...
Sending part no 2...
Sending part no 3...
Sending part no 4...
Sending part no 5...
Sending part no 6...
Finishing upload session...
uploaded: true

$ onteoncli application list
id                       createdAt                updatedAt                name     version type     processType
6253ec5d1eb077160b73f724 2022-04-11T08:52:45.228Z 2022-04-11T08:52:45.228Z company  1.1.2   standard native
624c48e959719200b1725072 2022-04-05T13:49:29.343Z 2022-04-05T13:49:29.343Z report   1.1.1   standard native
624c482659719200b1724eda 2022-04-05T13:46:14.073Z 2022-04-05T13:46:14.073Z product  1.0.2   standard native
624c476059719200b1724d71 2022-04-05T13:42:56.211Z 2022-04-05T13:42:56.211Z company  1.1.1   standard native
624c426ebfc5c34b20a5a39e 2022-04-05T13:21:50.233Z 2022-04-05T13:21:50.233Z report   1.1.0   standard native
624c3e47bfc5c34b20a5985b 2022-04-05T13:04:07.884Z 2022-04-05T13:04:07.884Z gateway  1.0.1   standard native
624c2f20dbe2ff0e819c73e7 2022-04-05T11:59:28.093Z 2022-04-05T11:59:28.093Z report   1.0.1   standard native
624c1a5bdbe2ff0e819c39b8 2022-04-05T10:30:51.16Z  2022-04-05T10:30:51.16Z  product  1.0.1   standard native
624c1006dbe2ff0e819c1cb9 2022-04-05T09:46:46.512Z 2022-04-05T09:46:46.512Z company  1.1.0   standard native
624c0403dbe2ff0e819bfb8f 2022-04-05T08:55:31.834Z 2022-04-05T08:55:31.834Z frontend 1.0.1   standard native
624bfc3bdbe2ff0e819be365 2022-04-05T08:22:19.29Z  2022-04-05T08:22:19.29Z  gateway  1.0.0   standard native

$ onteoncli application-change create 624c476059719200b1724d71 6253ec5d1eb077160b73f724
id:               624c490659719200b17250b8
createdAt:        2022-04-05T13:49:58.913Z
updatedAt:        2022-04-05T13:49:58.913Z
oldApplicationId: 624c426ebfc5c34b20a5a39e
newApplicationId: 624c48e959719200b1725072

$ watch onteoncli application-instance list --per-page 20
id                       createdAt                applicationName applicationVersion applicationType applicationId            nodeId                   status
e454a4863c211f1efcbdfa89 2022-04-11T08:54:03.405Z company         1.1.2              standard        6253ec5d1eb077160b73f724 7478c5da25458f9ebb538fb7 running
ab077048449e6bd40deee7dc 2022-04-11T08:53:33.341Z company         1.1.2              standard        6253ec5d1eb077160b73f724 3ed62c0a38f52cf1efe94e67 running
82e3974c288c00f24d5332bf 2022-04-05T13:51:10.08Z  report          1.1.1              standard        624c48e959719200b1725072 c934603cc1cd848c61700ba3 running
8a73951c5c6ee2afe2e30f80 2022-04-05T13:51:03.303Z report          1.1.1              standard        624c48e959719200b1725072 65508fa6ecdbbdb77029f61a running
9d44cab0d72a6ba098abcd93 2022-04-05T13:49:00.078Z product         1.0.2              standard        624c482659719200b1724eda 65508fa6ecdbbdb77029f61a running
726b19dc81ba63cf0897efdb 2022-04-05T13:47:13.612Z product         1.0.2              standard        624c482659719200b1724eda 65508fa6ecdbbdb77029f61a running
cc3e908461df9ea6f418e70e 2022-04-05T13:37:32.218Z gateway         1.0.1              standard        624c3e47bfc5c34b20a5985b c934603cc1cd848c61700ba3 running
b8bfae714b00251dd61bb1f1 2022-04-05T13:37:23.888Z gateway         1.0.1              standard        624c3e47bfc5c34b20a5985b c934603cc1cd848c61700ba3 running
0483bb3fcbc718e959b6e80a 2022-04-05T13:35:31.784Z frontend        1.0.1              standard        624c0403dbe2ff0e819bfb8f c934603cc1cd848c61700ba3 running
c491fc58a72c3465a7c806f1 2022-04-05T13:35:23.18Z  frontend        1.0.1              standard        624c0403dbe2ff0e819bfb8f c934603cc1cd848c61700ba3 running
a12213168c6081955c1ba421 2022-04-05T13:35:04.608Z nginx-edge      1.1.3              embedded        0a1769f83abc3224fe6cb477 65508fa6ecdbbdb77029f61a running
bd4e83eaa93b4f59eb05350e 2022-04-05T13:35:04.446Z nginx-edge      1.1.3              embedded        0a1769f83abc3224fe6cb477 c934603cc1cd848c61700ba3 running
0af31412317ca01b73744b92 2022-04-05T13:34:58.49Z  nginx-inner     1.1.3              embedded        157958ad0cdf14266501ad70 65508fa6ecdbbdb77029f61a running
c8abbce78274f9cf5f492784 2022-04-05T13:34:58.283Z nginx-inner     1.1.3              embedded        157958ad0cdf14266501ad70 c934603cc1cd848c61700ba3 running
```

Product

First, let's remove the packaging and Tomcat dependency from pom.xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.6.5</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>

    <groupId>com.example</groupId>
    <artifactId>product</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>product</name>
    <description>product</description>

    <properties>
        <java.version>11</java.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
```

```
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-validation</artifactId>
        </dependency>

        <dependency>
            <groupId>com.squareup.okhttp</groupId>
            <artifactId>okhttp</artifactId>
            <version>2.7.5</version>
        </dependency>

        <dependency>
            <groupId>org.postgresql</groupId>
            <artifactId>postgresql</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <finalName>${artifactId}</finalName>

        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <configuration>
                    <excludes>
                        <exclude>
                            <groupId>org.projectlombok</groupId>
                            <artifactId>lombok</artifactId>
                        </exclude>
                    </excludes>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

Then, go to ProductApplication class, remove extending the SpringBootServletInitializer class.

```
package com.example.product;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class ProductApplication {

    public static void main(String[] args) {
        SpringApplication.run(ProductApplication.class, args);
    }

}
```

Then, remove the `tomcat` dir. We won't need Tomcat files.

Now, let's modify a `conf.yml` file. We need to change the process provider to JVM.

```
app:
  name: 'product'
  version: '1.0.3'
  appType: 'standard'
  procType: 'native'
  processProvider:
    name: 'JVMOsProcessProviderImpl'
    version: '1.0.0'
    executable:
      start:
        path: '${ont_app_path}/bin'
        startJvmCommand: 'java -jar -Dserver.port=${ont_port_1}'
        successLine: 'Started ProductApplication'
        executableFileName: 'product.jar'
  placeHolder:
    name: 'PlaceHolderManagerImpl'
    version: '1.0.0'
    filesToReplace:
    variables:
  serviceRepository:
```

```
    healthCheckUrl: 'http://localhost:${ont_port_1}/is-alive'
    entities:
      - entity:
          priority: 1
          port: ${ont_port_1}
          protocol:
            type: 'HTTP'
            version: '1.1'
          isExternal: false
          isInternal: true
```

Then, modify the build.sh script.

```
#!/usr/bin/env bash

set -e

SOURCE="${BASH_SOURCE[0]}"
while [ -h "$SOURCE" ]; do
  DIR="$(cd -P "$( dirname "$SOURCE" )" >/dev/null 2>&1 && pwd)"
  SOURCE="$(readlink "$SOURCE")"
  [[ $SOURCE != /* ]] && SOURCE="$DIR/$SOURCE"
done
SCRIPT_DIR="$(cd -P "$(dirname "$SOURCE")" >/dev/null 2>&1 && pwd)"
ROOT_DIR="${SCRIPT_DIR}/.."
TARGET_DIR="${ROOT_DIR}/target"
APP_BIN_DIR="${TARGET_DIR}/app/bin"
APP_CONF_DIR="${TARGET_DIR}/app/conf"

cd "${ROOT_DIR}" && mvn clean package
mkdir -p "${APP_BIN_DIR}"
mkdir -p "${APP_CONF_DIR}"
cp -r "${TARGET_DIR}/product.jar" "${APP_BIN_DIR}"
cp "${ROOT_DIR}/onteon/conf.yml" "${APP_CONF_DIR}"
cd "${TARGET_DIR}/app" && tar -zcvf "${TARGET_DIR}/product.tar.gz" *
```

Finally, let's build, upload and upgrade application.

```
$ ./product/scripts/build.sh
$ onteoncli application-registry upload product/target/product.tar.gz
Starting upload session...
Sending part no 0...
Sending part no 1...
Sending part no 2...
Sending part no 3...
Sending part no 4...
Sending part no 5...
Sending part no 6...
Finishing upload session...
uploaded: true

$ onteoncli application list
id                       createdAt                updatedAt                name     version type     processType
6253edbe1eb077160b73fafe 2022-04-11T08:58:38.32Z  2022-04-11T08:58:38.32Z  product  1.0.3   standard native
6253ec5d1eb077160b73f724 2022-04-11T08:52:45.228Z 2022-04-11T08:52:45.228Z company  1.1.2   standard native
624c48e959719200b1725072 2022-04-05T13:49:29.343Z 2022-04-05T13:49:29.343Z report   1.1.1   standard native
624c482659719200b1724eda 2022-04-05T13:46:14.073Z 2022-04-05T13:46:14.073Z product  1.0.2   standard native
624c476059719200b1724d71 2022-04-05T13:42:56.211Z 2022-04-05T13:42:56.211Z company  1.1.1   standard native
624c426ebfc5c34b20a5a39e 2022-04-05T13:21:50.233Z 2022-04-05T13:21:50.233Z report   1.1.0   standard native
624c3e47bfc5c34b20a5985b 2022-04-05T13:04:07.884Z 2022-04-05T13:04:07.884Z gateway  1.0.1   standard native
624c2f20dbe2ff0e819c73e7 2022-04-05T11:59:28.093Z 2022-04-05T11:59:28.093Z report   1.0.1   standard native
624c1a5bdbe2ff0e819c39b8 2022-04-05T10:30:51.16Z  2022-04-05T10:30:51.16Z  product  1.0.1   standard native
624c1006dbe2ff0e819c1cb9 2022-04-05T09:46:46.512Z 2022-04-05T09:46:46.512Z company  1.1.0   standard native
624c0403dbe2ff0e819bfb8f 2022-04-05T08:55:31.834Z 2022-04-05T08:55:31.834Z frontend 1.0.1   standard native
624bfc3bdbe2ff0e819be365 2022-04-05T08:22:19.29Z  2022-04-05T08:22:19.29Z  gateway  1.0.0   standard native

$ onteoncli application-change create 624c482659719200b1724eda 6253edbe1eb077160b73fafe
id:               624c490659719200b17250b8
createdAt:        2022-04-05T13:49:58.913Z
updatedAt:        2022-04-05T13:49:58.913Z
oldApplicationId: 624c482659719200b1724eda
newApplicationId: 6253edbe1eb077160b73fafe

$ watch onteoncli application-instance list --per-page 20
id                       createdAt                applicationName applicationVersion applicationType applicationId            nodeId                   status
458e2e568f8f0c6f5f218a6a 2022-04-11T09:00:00.65Z  product         1.0.3              standard        6253edbe1eb077160b73fafe 3ed62c0a38f52cf1efe94e67 running
e2b35134e61499fe143ebb17 2022-04-11T08:59:43.298Z product         1.0.3              standard        6253edbe1eb077160b73fafe 3ed62c0a38f52cf1efe94e67 running
e454a4863c211f1efcbdfa89 2022-04-11T08:54:03.405Z company         1.1.2              standard        6253ec5d1eb077160b73f724 7478c5da25458f9ebb538fb7 running
ab077048449e6bd40deee7dc 2022-04-11T08:53:33.341Z company         1.1.2              standard        6253ec5d1eb077160b73f724 3ed62c0a38f52cf1efe94e67 running
82e3974c288c00f24d5332bf 2022-04-05T13:51:10.08Z  report          1.1.1              standard        624c48e959719200b1725072 c934603cc1cd848c61700ba3 running
8a73951c5c6ee2afe2e30f80 2022-04-05T13:51:03.303Z report          1.1.1              standard        624c48e959719200b1725072 65508fa6ecdbbdb77029f61a running
cc3e908461df9ea6f418e70e 2022-04-05T13:37:32.218Z gateway         1.0.1              standard        624c3e47bfc5c34b20a5985b c934603cc1cd848c61700ba3 running
b8bfae714b00251dd61bb1f1 2022-04-05T13:37:23.888Z gateway         1.0.1              standard        624c3e47bfc5c34b20a5985b c934603cc1cd848c61700ba3 running
0483bb3fcbc718e959b6e80a 2022-04-05T13:35:31.784Z frontend        1.0.1              standard        624c0403dbe2ff0e819bfb8f c934603cc1cd848c61700ba3 running
c491fc58a72c3465a7c806f1 2022-04-05T13:35:23.18Z  frontend        1.0.1              standard        624c0403dbe2ff0e819bfb8f c934603cc1cd848c61700ba3 running
a12213168c6081955c1ba421 2022-04-05T13:35:04.608Z nginx-edge      1.1.3              embedded        0a1769f83abc3224fe6cb477 65508fa6ecdbbdb77029f61a running
bd4e83eaa93b4f59eb05350e 2022-04-05T13:35:04.446Z nginx-edge      1.1.3              embedded        0a1769f83abc3224fe6cb477 c934603cc1cd848c61700ba3 running
0af31412317ca01b73744b92 2022-04-05T13:34:58.49Z  nginx-inner     1.1.3              embedded        157958ad0cdf14266501ad70 65508fa6ecdbbdb77029f61a running
c8abbce78274f9cf5f492784 2022-04-05T13:34:58.283Z nginx-inner     1.1.3              embedded        157958ad0cdf14266501ad70 c934603cc1cd848c61700ba3 running
```

Report

First, let's remove the packaging and Tomcat dependency from pom.xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.6.5</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>

    <groupId>com.example</groupId>
    <artifactId>report</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>report</name>
    <description>report</description>

    <properties>
        <java.version>11</java.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
```

```
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-validation</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-amqp</artifactId>
        </dependency>

        <dependency>
            <groupId>com.github.librepdf</groupId>
            <artifactId>openpdf</artifactId>
            <version>1.3.27</version>
        </dependency>
        <dependency>
            <groupId>com.squareup.okhttp</groupId>
            <artifactId>okhttp</artifactId>
            <version>2.7.5</version>
        </dependency>

        <dependency>
            <groupId>org.postgresql</groupId>
            <artifactId>postgresql</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <finalName>${artifactId}</finalName>

        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <configuration>
                    <excludes>
                        <exclude>
                            <groupId>org.projectlombok</groupId>
                            <artifactId>lombok</artifactId>
                        </exclude>
                    </excludes>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

Then, go to ReportApplication class, remove extending the SpringBootServletInitializer class.

```
package com.example.report;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class ReportApplication {

    public static void main(String[] args) {
        SpringApplication.run(ReportApplication.class, args);
    }

}
```

Then, remove the `tomcat` dir. We won't need Tomcat files.

Now, let's modify a `conf.yml` file. We need to change the process provider to JVM.

```
app:
  name: 'report'
  version: '1.1.2'
  appType: 'standard'
  procType: 'native'
  processProvider:
    name: 'JVMOsProcessProviderImpl'
    version: '1.0.0'
    executable:
      start:
```

```
        path: '${ont_app_path}/bin'
        startJvmCommand: 'java -jar -Dserver.port=${ont_port_1}'
        successLine: 'Started ReportApplication'
        executableFileName: 'report.jar'
  placeHolder:
    name: 'PlaceHolderManagerImpl'
    version: '1.0.0'
    filesToReplace:
    variables:
  serviceRepository:
    healthCheckUrl: 'http://localhost:${ont_port_1}/is-alive'
    entities:
      - entity:
          priority: 1
          port: ${ont_port_1}
          protocol:
            type: 'HTTP'
            version: '1.1'
          isExternal: false
          isInternal: true
```

Then, modify the build.sh script.

```
#!/usr/bin/env bash

set -e

SOURCE="${BASH_SOURCE[0]}"
while [ -h "$SOURCE" ]; do
  DIR="$(cd -P "$( dirname "$SOURCE" )" >/dev/null 2>&1 && pwd)"
  SOURCE="$(readlink "$SOURCE")"
  [[ $SOURCE != /* ]] && SOURCE="$DIR/$SOURCE"
done
SCRIPT_DIR="$(cd -P "$(dirname "$SOURCE")" >/dev/null 2>&1 && pwd)"
ROOT_DIR="${SCRIPT_DIR}/.."
TARGET_DIR="${ROOT_DIR}/target"
APP_BIN_DIR="${TARGET_DIR}/app/bin"
APP_CONF_DIR="${TARGET_DIR}/app/conf"

cd "${ROOT_DIR}" && mvn clean package
mkdir -p "${APP_BIN_DIR}"
mkdir -p "${APP_CONF_DIR}"
cp -r "${TARGET_DIR}/report.jar" "${APP_BIN_DIR}"
cp "${ROOT_DIR}/onteon/conf.yml" "${APP_CONF_DIR}"
cd "${TARGET_DIR}/app" && tar -zcvf "${TARGET_DIR}/report.tar.gz" *
```

Finally, let's build, upload and upgrade application.

```
$ ./report/scripts/build.sh
$ onteoncli application-registry upload report/target/report.tar.gz
Starting upload session...
Sending part no 0...
Sending part no 1...
Sending part no 2...
Sending part no 3...
Sending part no 4...
Sending part no 5...
Sending part no 6...
Sending part no 7...
Finishing upload session...
uploaded: true

$ onteoncli application list
id                       createdAt                updatedAt                name     version type     processType
6253ef3e1eb077160b73ff72 2022-04-11T09:05:02.558Z 2022-04-11T09:05:02.558Z report   1.1.2   standard native
6253edbe1eb077160b73fafe 2022-04-11T08:58:38.32Z  2022-04-11T08:58:38.32Z  product  1.0.3   standard native
6253ec5d1eb077160b73f724 2022-04-11T08:52:45.228Z 2022-04-11T08:52:45.228Z company  1.1.2   standard native
624c48e959719200b1725072 2022-04-05T13:49:29.343Z 2022-04-05T13:49:29.343Z report   1.1.1   standard native
624c482659719200b1724eda 2022-04-05T13:46:14.073Z 2022-04-05T13:46:14.073Z product  1.0.2   standard native
624c476059719200b1724d71 2022-04-05T13:42:56.211Z 2022-04-05T13:42:56.211Z company  1.1.1   standard native
624c426ebfc5c34b20a5a39e 2022-04-05T13:21:50.233Z 2022-04-05T13:21:50.233Z report   1.1.0   standard native
624c3e47bfc5c34b20a5985b 2022-04-05T13:04:07.884Z 2022-04-05T13:04:07.884Z gateway  1.0.1   standard native
624c2f20dbe2ff0e819c73e7 2022-04-05T11:59:28.093Z 2022-04-05T11:59:28.093Z report   1.0.1   standard native
624c1a5bdbe2ff0e819c39b8 2022-04-05T10:30:51.16Z  2022-04-05T10:30:51.16Z  product  1.0.1   standard native
624c1006dbe2ff0e819c1cb9 2022-04-05T09:46:46.512Z 2022-04-05T09:46:46.512Z company  1.1.0   standard native
624c0403dbe2ff0e819bfb8f 2022-04-05T08:55:31.834Z 2022-04-05T08:55:31.834Z frontend 1.0.1   standard native
624bfc3bdbe2ff0e819be365 2022-04-05T08:22:19.29Z  2022-04-05T08:22:19.29Z  gateway  1.0.0   standard native

$ onteoncli application-change create 624c48e959719200b1725072 6253ef3e1eb077160b73ff72
id:               624c490659719200b17250b8
createdAt:        2022-04-05T13:49:58.913Z
updatedAt:        2022-04-05T13:49:58.913Z
oldApplicationId: 624c48e959719200b1725072
newApplicationId: 6253ef3e1eb077160b73ff72

$ watch onteoncli application-instance list --per-page 20
id                       createdAt                applicationName applicationVersion applicationType applicationId            nodeId                   status
863208b400779047057aa4db 2022-04-11T09:07:00.295Z report          1.1.2              standard        6253ef3e1eb077160b73ff72 7478c5da25458f9ebb538fb7 running
dd00e9b27393b80dda2c1be8 2022-04-11T09:06:03.422Z report          1.1.2              standard        6253ef3e1eb077160b73ff72 7478c5da25458f9ebb538fb7 running
```

```
458e2e568f8f0c6f5f218a6a 2022-04-11T09:00:00.65Z  product         1.0.3             standard        6253edbe1eb077160b73fafe 3ed62c0a38f52cf1efe94e67 running
e2b35134e61499fe143ebb17 2022-04-11T08:59:43.298Z product         1.0.3             standard        6253edbe1eb077160b73fafe 3ed62c0a38f52cf1efe94e67 running
e454a4863c211f1efcbdfa89 2022-04-11T08:54:03.405Z company         1.1.2             standard        6253ec5d1eb077160b73f724 7478c5da25458f9ebb538fb7 running
ab077048449e6bd40deee7dc 2022-04-11T08:53:33.341Z company         1.1.2             standard        6253ec5d1eb077160b73f724 3ed62c0a38f52cf1efe94e67 running
cc3e908461df9ea6f418e70e 2022-04-05T13:37:32.218Z gateway         1.0.1             standard        624c3e47bfc5c34b20a5985b c934603cc1cd848c61700ba3 running
b8bfae714b00251dd61bb1f1 2022-04-05T13:37:23.888Z gateway         1.0.1             standard        624c3e47bfc5c34b20a5985b c934603cc1cd848c61700ba3 running
0483bb3fcbc718e959b6e80a 2022-04-05T13:35:31.784Z frontend        1.0.1             standard        624c0403dbe2ff0e819bfb8f c934603cc1cd848c61700ba3 running
c491fc58a72c3465a7c806f1 2022-04-05T13:35:23.18Z  frontend        1.0.1             standard        624c0403dbe2ff0e819bfb8f c934603cc1cd848c61700ba3 running
a12213168c6081955c1ba421 2022-04-05T13:35:04.608Z nginx-edge      1.1.3             embedded        0a1769f83abc3224fe6cb477 65508fa6ecdbbdb77029f61a running
bd4e83eaa93b4f59eb05350e 2022-04-05T13:35:04.446Z nginx-edge      1.1.3             embedded        0a1769f83abc3224fe6cb477 c934603cc1cd848c61700ba3 running
0af31412317ca01b73744b92 2022-04-05T13:34:58.49Z  nginx-inner     1.1.3             embedded        157958ad0cdf14266501ad70 65508fa6ecdbbdb77029f61a running
c8abbce78274f9cf5f492784 2022-04-05T13:34:58.283Z nginx-inner     1.1.3             embedded        157958ad0cdf14266501ad70 c934603cc1cd848c61700ba3 running
```

Test

Go to http://localhost:8020/_by_name/frontend/ and test if our application still works.

> In case of questions and remarks please send email to contact@onteon.com.


let's try
onteon