|  |  |  |
| --- | --- | --- |
| **ЗАТВЕРДЖУЮ**  КП “Головний інформаційно-обчислювальний центр” | **ЗАТВЕРДЖУЮ**  ПрАТ “НДІ прикладних інформаційних технологій” |  |
| Директор | Директор |  |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. М. Козубський | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ю.В. Єсаулов |  |
| “\_\_\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2018 р. | “\_\_\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2018 р. |  |

**Послуги з розвитку платформи надання електронних послуг, у тому числі адміністративних, 3 черга**

**Створення програмного модуля “ Реєстр суб’єктів, що надають послуги”**

**Шифр роботи: ПМ «РСНП»**

**Загальна інструкція з розгортання та налаштування**

**Програмного модулю «Реєстр суб’єктів, що надають послуги»**

На \_\_ аркушах

діє з “\_\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_2018 року

|  |  |  |
| --- | --- | --- |
| **Від Замовника:** |  | **Від Виконавця:** |
| Начальник департаменту  впровадження та супроводу  інформаційно-комунікаційних систем |  | Керівник проекту |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_О. П. Перевозник |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| В.о. начальника департаменту  впровадження та технологічного супроводження обліково-фінансових систем |  |  |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А. П. Гусаревич |  |  |
| Провідний інженер відділу впровадження та розвитку інформаційно-комунікаційних систем |  |  |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. О. Мусієнко |  |  |

Київ 2018

**ЗМІСТ**

[1. ЗАГАЛЬНІ ПОЛОЖЕННЯ 3](#_Toc527442909)

[2. НОРМАТИВНІ ПОСИЛАННЯ 3](#_Toc527442910)

[3. ПРИЗНАЧЕННЯ 5](#_Toc527442911)

[4. ВИМОГИ ДО КВАЛІФІКАЦІЇ ПЕРСОНАЛУ 6](#_Toc527442912)

[5. ПОРЯДОК РОЗГОРТАННЯ ОПЕРАЦІЙНОГО СЕРЕДОВИЩА 6](#_Toc527442913)

[5.1. Компоненти системи 6](#_Toc527442914)

[5.2. Пререквізити 7](#_Toc527442915)

[5.3. Конфігурація додатку для кластеру Openshift 7](#_Toc527442916)

[5.3.1. Завантаження Docker-образів в registry 7](#_Toc527442917)

[5.3.2. Створення файлової структури для файлів конфігурації додатку 9](#_Toc527442918)

[5.3.3. Зміст файлів конфігурації додатку 10](#_Toc527442919)

[5.3.4. Розгортання додатку 48](#_Toc527442920)

[6. ПЕРЕВІРКА ВІРНОГО РОЗГОРТАННЯ ОПЕРАЦІЙНОГО СЕРЕДОВИЩА 49](#_Toc527442921)

# ЗАГАЛЬНІ ПОЛОЖЕННЯ

Даний документ містить інструкцію з розгортання та налаштування програмного модулю «Реєстр суб’єктів, що надають послуги» (далі – Модуль).

Дана інструкція розроблена для розгортання операційного середовища Модулю на комплексі технічних засобів Замовника.

# НОРМАТИВНІ ПОСИЛАННЯ

Створення програмного модуля «Реєстр суб’єктів, що надають послуги» здійснена на підставі Договору про надання послуг № від 2018 року.

Технічне проектування виконано розробниками згідно з Технічним завданням на створення програмного модулю «Реєстр суб’єктів, що надають послуги», у рамках розвитку платформи надання електронних послуг, у тому числі адміністративних, 3 черга.

Розробка здійснювалася у відповідності до наступних нормативних документів:

1. Закону України «Про інформацію»;
2. Закону України «Про електронні документи та електронний документообіг»;
3. Закону України «Про основні засади розвитку інформаційного суспільства в Україні на 2007-2015 роки»;
4. Закону України «Про адміністративні послуги»;
5. Закону України «Про звернення громадян»;
6. Закону України «Про доступ до публічної інформації»;
7. Закону України «Про захист інформації в інформаційно-телекомунікаційних системах»;
8. Закону України «Про електронний цифровий підпис»;
9. Закону України «Про захист персональних даних»;
10. Постанови Кабінету Міністрів України від 04.02.1998 № 121 «Про затвердження переліку обов’язкових етапів робіт під час проектування, впровадження та експлуатації систем і засобів автоматизованої обробки та передачі даних»;
11. Постанови Кабінету Міністрів України від 12.04.2002 №522 «Про затвердження Порядку підключення до глобальних мереж передачі даних»;
12. Постанови Кабінету Міністрів України від 10.09.2003 № 1433 «Про затвердження Порядку використання комп’ютерних програм в органах виконавчої влади»;
13. Постанови Кабінету Міністрів України від 28.10.2004 № 1452 «Про затвердження Порядку застосування електронного цифрового підпису органами державної влади, органами місцевого самоврядування, підприємствами, установами та організаціями державної форми власності»;
14. Постанови Кабінету Міністрів України від 29.03.2006 №373 «Про затвердження Правил забезпечення захисту інформації в інформаційних, телекомунікаційних та інформаційно-телекомунікаційних системах»;
15. ДСТУ 2394 - 94 «Інформація та документація. Терміни та визначення»;
16. НД ТЗІ 1.1-003-99. Термінологія в галузі захисту інформації у комп’ютерних системах від несанкціонованого доступу;
17. НД ТЗІ 1.4-001-2000. Типове положення про службу захисту інформації в автоматизованій системі;
18. НД ТЗІ 2.5-004-99. Критерії оцінки захищеності інформації у комп’ютерних системах від несанкціонованого доступу;
19. НД ТЗІ 2.5-005-99. Класифікація автоматизованих систем і стандартні функціональні профілі захищеності оброблюваної інформації від несанкціонованого доступу;
20. НД ТЗІ 3.6-001-2000. Технічний захист інформації. Комп’ютерні системи. Порядок створення, впровадження, супроводження та модернізації засобів технічного захисту інформації від несанкціонованого доступу;
21. НД ТЗІ 3.7-001-99. Методичні вказівки щодо розробки технічного завдання на створення комплексної системи захисту інформації в автоматизованій системі;
22. НД ТЗІ 3.7-003-05. Порядок проведення робіт із створення комплексної системи захисту інформації в інформаційно-телекомунікаційній системі;
23. ГОСТ 34.201-89. Інформаційна технологія. Комплекс стандартів на автоматизовані системи. Види, комплектність і позначення документів при створенні автоматизованих систем;
24. ГОСТ 34.601-90. Інформаційна технологія. Комплекс стандартів на автоматизовані системи. Автоматизовані системи. Стадії створення;
25. ГОСТ 34.602-89. Інформаційна технологія. Комплекс стандартів на автоматизовані системи. Технічне завдання на створення автоматизованої системи;
26. ГОСТ 34.603.-92. Інформаційна технологія. Види випробувань автоматизованих систем;
27. ГОСТ 19.101.-77. Єдина система програмної документації. Загальні положення;
28. ГОСТ 19.102.-77. Єдина система програмної документації. Стадії розробки;
29. ГОСТ 19.105.-78. Єдина система програмної документації. Загальні вимоги до текстових програмних документів;
30. ГОСТ 19.201.-78. Єдина система програмної документації. Технічне завдання. Вимоги до змісту та оформлення;
31. ГОСТ 19.301.-79. Єдина система програмної документації. Програма та методика випробувань. Вимоги до змісту та оформлення;
32. ГОСТ 19.401.-78. Єдина система програмної документації. Текст програми. Вимоги до змісту та оформлення;
33. ГОСТ 19.402.-78. Єдина система програмної документації. Опис програми;
34. ГОСТ 19.507.-79). Єдина система програмної документації. Відомість експлуатаційних документів;
35. ДСТУ 2394-94 «Інформація та документація. Терміни та визначення»;
36. ДСТУ 3396.0-96 «Захист інформації»;
37. ДК 010-98 «Державний класифікатор управлінської документації»;
38. ДСТУ 3918-1999 (ІБО/ІЕС 12207:1995) «Процеси життєвого циклу програмного забезпечення».

# ПРИЗНАЧЕННЯ

Реєстр суб’єктів, що надають послуги – це програмний ресурс територіальної громади міста Києва.

Призначення розробки РСНП – автоматизація процесу обліку інформації про перелік послуг та суб’єктів, що їх надають та, як результат, можливість для жителів міста у зручний спосіб отримувати інформацію про існуючі доступні послуги та про суб’єктів, що їх надають.

# ВИМОГИ ДО КВАЛІФІКАЦІЇ ПЕРСОНАЛУ

Співробітники, які безпосередньо відповідають за розгортання операційного середовища повинні мати рівень системного адміністратора з обов’язковими знаннями:

1. Linux;
2. Технології Docker;
3. Openshift / Kubernetes.

# ПОРЯДОК РОЗГОРТАННЯ ОПЕРАЦІЙНОГО СЕРЕДОВИЩА

## Компоненти системи

1. Front-end;
2. Back-end:
   1. OAuth2;
   2. Forms;
   3. Entities;
   4. Documents-data;
   5. Counterparty;
   6. Access;
   7. Alfresco:
      1. LibreOffice (Alfresco's dependency);
      2. Document Service (Alfresco's dependency).

## Пререквізити

1. Дана інструкція не торкається теми розгортання середовища RedHat Openshift. Вважається, що середовище Openshift встановлене та налаштоване належним чином.
2. Вважається, що система має достатній ресурс для запуску всіх необхідних docker-контейнерів.
3. Вважається, що спеціальна утілита "oc" для управління кластером Openshift встановлена та налаштована, а процедура автентифікації і авторизації виконана.
4. Вважається, що є доступний для кластеру docker-сумісний реєстр образів docker-registry.example.com, а всі необхідні docker-образи всіх потрібних мікросервісів з цієї інструкції передані команді, відповідальній за розгортання додатку в кластері, на якомусь носії даних (диск, USB-flash, або ін.).
5. В кластері Openshift налаштовано автоматичне виділення volume потрібного обсягу при створенні Persistent Volume Claim в розділі Storage, або Persistent Volumes в достатньому обсязі створені і доступні для сервісів додатку.

## Конфігурація додатку для кластеру Openshift

### Завантаження Docker-образів в registry

Всі docker-образи з носія потрібно скопіювати на будь-яку локальну робочу станцію або сервер з Linux в окремий каталог. На цьому сервері повинний бути встановлений Docker ([www.docker.com](https://www.docker.com/)).

Для завантаження файлів образів в регістрі потрібно виконати наступні команди:

|  |
| --- |
| $ cd service-provider-registry/docker-images/  $ docker load -q -i forms.tar.gz  $ docker load -q -i documents-data.tar.gz  $ docker load -q -i frontend.tar.gz  $ docker load -q -i document-service.tar.gz  $ docker load -q -i entities.tar.gz  $ docker load -q -i counterparty.tar.gz  $ docker load -q -i access.tar.gz  $ docker load -q -i oaut2.tar.gz |

Після цього потрібно впевнитись, що всі ці образи завантажені в локальний Docker:

|  |
| --- |
| $ docker images |

Далі потрібно підготовити ці образи для вивантаження на зовнішній регістрі docker-registry.example.com

|  |
| --- |
| $ docker tag forms:latest docker-registry.example.com/service-provider-registry/forms:latest  $ docker tag documents-data:latest docker-registry.example.com/service-provider-registry/documents-data:latest  $ docker tag frontend:latest docker-registry.example.com/service-provider-registry/frontend:latest  $ docker tag document-service:latest docker-registry.example.com/service-provider-registry/document-service:latest  $ docker tag entities:latest docker-registry.example.com/service-provider-registry/entities:latest  $ docker tag counterparty:latest docker-registry.example.com/service-provider-registry/counterparty:latest  $ docker tag access:latest docker-registry.example.com/service-provider-registry/access:latest  $ docker tag oauth2\_authorization\_server:latest docker-registry.example.com/service-provider-registry/oauth2\_authorization\_server:latest |

Після цього потрібно знову впевнитись, що копії образів з новими назвами з'явилися в локальному Docker:

|  |
| --- |
| $ docker images |

Після цього необхідно виконати команди для вивантаження цих образів на docker-registry.example.com

|  |
| --- |
| $ docker login docker-registry.example.com  $ docker push docker-registry.example.com/service-provider-registry/forms:latest  $ docker push docker-registry.example.com/service-provider-registry/documents-data:latest  $ docker push docker-registry.example.com/service-provider-registry/frontend:latest  $ docker push docker-registry.example.com/service-provider-registry/document-service:latest  $ docker push docker-registry.example.com/service-provider-registry/entities:latest  $ docker push docker-registry.example.com/service-provider-registry/counterparty:latest  $ docker push docker-registry.example.com/service-provider-registry/access:latest  $ docker push docker-registry.example.com/service-provider-registry/oauth2\_authorization\_server:latest |

Через деякий час залежно від можливостей каналу зв'язку і потужностей зовнішнього регістру ці образи з'являться на ньому.

### Створення файлової структури для файлів конфігурації додатку

В домашньому каталозі користувача створимо файли конфігурації додатку для розгортання його в кластері Openshift. Після створення файлова структура проекту буде мати вигляд:

|  |
| --- |
| service-provider-registry/  └── openshift/      ├── access-img.yaml      ├── alfresco-properties-cfg.yaml      ├── counterparty-img.yaml      ├── docker-sec.yaml      ├── docsvc-img.yaml      ├── docsvc-properties-cfg.yaml      ├── documents-data-img.yaml      ├── entities-img.yaml      ├── forms-img.yaml      ├── frontend-img.yaml      ├── oauth2-img.yaml      ├── front-end-dep.yaml      ├── counterparty-dep.yaml      ├── forms-dep.yaml      ├── documents-data-dep.yaml      ├── entities-dep.yaml      ├── access-dep.yaml      ├── oauth2-dep.yaml      ├── docsvc-dep.yaml      ├── alfresco-dep.yaml      └── libreoffice-dep.yaml          svc/          ├── frontend-svc.yaml          ├── forms-svc.yaml          ├── documents-data-svc.yaml          ├── entities-svc.yaml          ├── counterparty-svc.yaml          ├── access-svc.yaml          ├── oauth2-svc.yaml          ├── docsvc-svc.yaml          ├── alfresco-svc.yaml          ├── libreoffice-svc.yaml          ├── frontend-rt.yaml          ├── forms-rt.yaml          ├── documents-rt.yaml          ├── entities-rt.yaml          ├── counterparty-rt.yaml          ├── access-rt.yaml          ├── docsvc-rt.yaml          └── alfresco-rt.yaml |

Далі буде описано зміст вказаних файлів.

### Зміст файлів конфігурації додатку

#### Secrets

##### docker-sec.yaml

|  |
| --- |
| apiVersion: v1  data:    .dockerconfigjson: >-      [закодований\_Base64\_текст\_файлу\_конфігурації\_docker-registry]  kind: Secret  metadata:    name: docker-registry    namespace: service-provider-registry    selfLink: /api/v1/namespaces/service-provider-registry/secrets/docker-registry  type: kubernetes.io/dockerconfigjson |

Файл конфігурації docker-registry створюється або вручну за спеціальним форматом або автоматично при першій вдалій спробі підключення до docker-registry за допомогою команди "docker login" та має назву ~/.docker/config.json. Цей файл має вигляд:

|  |
| --- |
| {          "auths": {                  "docker-registry.example.com": {                          "auth": "YWtyeTytjsri6cGUTYeeEyer4Y="                  },                  "[https://docker-registry.example.com"](https://docker-registry.example.com/): {                          "auth": "YWtyeTytjsri6cGUTYeeEyer4Y="                  }          }  } |

Щоб дізнатися про точний формат створення цього файлу зверніться до документації Docker.

Для кодування цього файлу в формат Base64 потрібно виконати наступну команду:

|  |
| --- |
| cat ~/.docker/config.json | base64 |

#### Configuration Maps

##### alfresco-properties-cfg.yaml

|  |
| --- |
| apiVersion: v1  data:    alfresco-global.properties: >-      ###############################        ## Common Alfresco Properties #        ###############################          #        # Sample custom content and index data location        #        dir.root=/usr/local/alfresco/alf\_data        #dir.keystore=${dir.root}/keystore          #        # Sample database connection properties        #        db.username=service-provider-registry\_alfresco\_user        db.password=alfresco\_password          #        # External locations        #-------------        ooo.exe=soffice.bin        ooo.enabled=true        ooo.port=8997        ooo.host=libreoffice        #jodconverter.officeHome=./OpenOffice.org        #jodconverter.portNumbers=8101        jodconverter.enabled=false        #img.root=./ImageMagick        img.exe=/usr/bin/convert          #        # Windows specific environment variables (should not be set on unix) for ImageMagick        #        #img.coders=${img.root}/modules/coders        #img.config=${img.root}/config        #img.gslib=${img.root}/lib          #        # Property to control whether schema updates are performed automatically.        # Updates must be enabled during upgrades as, apart from the static upgrade scripts,      # there are also auto-generated update scripts that will need to be executed.  After      # upgrading to a new version, this can be disabled.        #        db.schema.update=true          #        # PostgreSQL connection (requires postgresql-8.2-504.jdbc3.jar or equivalent)        #        db.driver=org.postgresql.Driver        db.url=jdbc:postgresql://10.132.3.2:5432/service-provider-registry\_test\_alfresco          #        # Index Recovery Mode        #-------------        #index.recovery.mode=AUTO          #        # Outbound Email Configuration        #-------------        #mail.host=        #mail.port=25        #mail.username=anonymous        #mail.password=        #mail.encoding=UTF-8        #mail.from.default=alfresco@alfresco.org        #mail.smtp.auth=false          #        # Alfresco Email Service and Email Server        #-------------          # Enable/Disable the inbound email service.  The service could be used by      processes other than        # the Email Server (e.g. direct RMI access) so this flag is independent of      the Email Service.        #-------------        #email.inbound.enabled=true          # Email Server properties        #-------------        #email.server.enabled=true        #email.server.port=25        #email.server.domain=alfresco.com        #email.inbound.unknownUser=anonymous          # A comma separated list of email REGEX patterns of allowed senders.        # If there are any values in the list then all sender email addresses        # must match. For example:        #   .\*\@alfresco\.com, .\*\@alfresco\.org        # Allow anyone:        #-------------        #email.server.allowed.senders=.\*          #        # The default authentication chain        # To configure external authentication subsystems see:        # http://wiki.alfresco.com/wiki/Alfresco\_Authentication\_Subsystems        #-------------        #authentication.chain=alfrescoNtlm1:alfrescoNtlm          #        # URL Generation Parameters (The ${localname} token is replaced by the local server name)        #-------------        alfresco.context=alfresco        alfresco.host=${localname}        alfresco.port=8080        alfresco.protocol=http        #        share.context=share        share.host=${localname}        share.port=8080        share.protocol=http          imap.server.enabled=false        #imap.server.port=143        #imap.server.host=localhost          ftp.enabled=false        cifs.enabled=false          # Default value of alfresco.rmi.services.host is 0.0.0.0 which means 'listen on all adapters'.        # This allows connections to JMX both remotely and locally.        #        alfresco.rmi.services.host=0.0.0.0          #        #        # Assign individual ports for each service for best performance        # or run several services on the same port. You can even run everything on 50500 if needed.        #        # Select 0 to use a random unused port.        #        #monitor.rmi.service.port=50508          #Smart Folders Config Properties          smart.folders.enabled=false        smart.folders.model=alfresco/model/smartfolder-model.xml        smart.folders.model.labels=alfresco/messages/smartfolder-model          #Smart reference config          #smart.reference.classpath.hash=${smart.folders.config.vanilla.processor.classpath}->1,${smart.folders.config.system.templates.classpath}->2          #Smart store config          #Company home relative download associations of smart entries        #smart.download.associations.folder=${spaces.dictionary.childname}/${spaces.smartdownloads.childname}          #Generic virtualization methods config          #Vanilla JSON templates javascript processor classpath. A java script processor used to      #covert JSON templates to internal smart folder definitions.          #smart.folders.config.vanilla.processor.classpath=/org/alfresco/repo/virtual/node/vanilla.js          #System virtualization method config          #System virtualization method aspect.        #smart.folders.config.system.aspect=smf:systemConfigSmartFolder        #System virtualization method aspect defined template location property.        #smart.folders.config.system.aspect.template.location.property=smf:system-template-location        #Classpath to be explored for \*.json entries defining system templates.        #smart.folders.config.system.templates.classpath=/org/alfresco/repo/virtual/node        #A company home relative name or qname path location of repository system templates.        #smart.folders.config.system.templates.path=${spaces.dictionary.childname}/${spaces.smartfolders.childname}        #Content sub type of repository system templates.        #smart.folders.config.system.templates.template.type=smf:smartFolderTemplate          #Custom virtualization method config          #Custom virtualization method aspect.        #smart.folders.config.custom.aspect=smf:customConfigSmartFolder        #Custom virtualization method aspect template content association.        #smart.folders.config.custom.aspect.template.association=smf:custom-template-association            #Type virtualization method config          #A company home relative name or qname path location of the type mapped templates.        #smart.folders.config.type.templates.path=${spaces.dictionary.childname}/${spaces.smartfolders.childname}        #Type and aspect qname regular expression filter.        #smart.folders.config.type.templates.qname.filter=none          ### Solr indexing ###        index.subsystem.name=solr4        solr.host=localhost        solr.port=8080        solr.secureComms=none  kind: ConfigMap  metadata:    name: alfresco-global-properties    namespace: service-provider-registry    selfLink: /api/v1/namespaces/service-provider-registry/configmaps/alfresco-global-properties |

##### docsvc-properties-cfg.yaml

|  |
| --- |
| apiVersion: v1  data:    application.properties: |-      #alfresco config      #for docker image ip address must have view as alfresco variable      alfresco.url=http://alfresco:8080      #admin login in alfresco      alfresco.admin.password=admin      #admin password in alfresco      alfresco.admin.login=admin      #alfresco user group in which will be create new users      #current user group must have update file permissions      alfresco.admin.group=GROUP\_ALFRESCO\_ADMINISTRATORS      #server port      server.port=8080      # swagger parameter      swagger.enable=true      #multipart configuration      spring.http.multipart.enabled=true      spring.http.multipart.max-request-size=50MB      spring.http.multipart.max-file-size=50MB      #version of application which will shows in swagger from maven version      api.version=@version@      api.major.version=@parsedVersion.majorVersion@      #alfresco response date format      alfresco.date.format=yyyy-MM-dd'T'HH:mm:ss.SSS'+0000'      #AsyncTaskExecutor task executor configuration for async controllers      pool.size=5      pool.capacity=10      pool.size.max=5      #for docker application.properties file must contain:      #current directory must be thrown as docker volume      #logging.config=      #logging.path=  kind: ConfigMap  metadata:    name: docsvc-app-properties    namespace: service-provider-registry    selfLink: /api/v1/namespaces/service-provider-registry/configmaps/docsvc-app-properties |

#### Images

##### frontend-img.yaml

|  |
| --- |
| apiVersion: v1  kind: ImageStream  metadata:    name: frontend    namespace: service-provider-registry    selfLink: /oapi/v1/namespaces/service-provider-registry/imagestreams/frontend  spec:    dockerImageRepository: >-      docker-registry.example.com/service-provider-registry\_frontend/service-provider-registry\_fr    lookupPolicy:      local: false    tags:      - annotations: null        from:          kind: DockerImage          name: >-            docker-registry.example.com/service-provider-registry\_frontend/service-provider-registry\_fr:latest        importPolicy:          scheduled: true        name: latest        referencePolicy:          type: Source |

##### counterparty-img.yaml

|  |
| --- |
| apiVersion: v1  kind: ImageStream  metadata:    name: counterparty    namespace: service-provider-registry    selfLink: /oapi/v1/namespaces/service-provider-registry/imagestreams/counterparty  spec:    dockerImageRepository: 'docker-registry.example.com/service-provider-registry/counterparty'    lookupPolicy:      local: false    tags:      - annotations: null        from:          kind: DockerImage          name: >-            docker-registry.example.com/service-provider-registry/counterparty:latest        importPolicy:          scheduled: true        name: latest        referencePolicy:          type: Source |

##### forms-img.yaml

|  |
| --- |
| apiVersion: v1  kind: ImageStream  metadata:    name: forms    namespace: service-provider-registry    selfLink: /oapi/v1/namespaces/service-provider-registry/imagestreams/forms  spec:    dockerImageRepository: 'docker-registry.example.com/service-provider-registry/forms'    lookupPolicy:      local: false    tags:      - annotations: null        from:          kind: DockerImage          name: >-            docker-registry.example.com/service-provider-registry/forms:latest        importPolicy:          scheduled: true        name: latest        referencePolicy:          type: Source |

##### documents-data-img.yaml

|  |
| --- |
| apiVersion: v1  kind: ImageStream  metadata:    name: documents-data    namespace: service-provider-registry    selfLink: /oapi/v1/namespaces/service-provider-registry/imagestreams/documents-data  spec:    dockerImageRepository: 'docker-registry.example.com/service-provider-registry/documents-data'    lookupPolicy:      local: false    tags:      - annotations: null        from:          kind: DockerImage          name: >-            docker-registry.example.com/service-provider-registry/documents-data:latest        importPolicy:          scheduled: true        name: latest        referencePolicy:          type: Source |

##### entities-img.yaml

|  |
| --- |
| apiVersion: v1  kind: ImageStream  metadata:    name: entities    namespace: service-provider-registry    selfLink: /oapi/v1/namespaces/service-provider-registry/imagestreams/entities  spec:    dockerImageRepository: 'docker-registry.example.com/service-provider-registry/entities'    lookupPolicy:      local: false    tags:      - annotations: null        from:          kind: DockerImage          name: >-            docker-registry.example.com/service-provider-registry/entities:latest        importPolicy:          scheduled: true        name: latest        referencePolicy:          type: Source |

##### access-img.yaml

|  |
| --- |
| apiVersion: v1  kind: ImageStream  metadata:    name: access    namespace: service-provider-registry    selfLink: /oapi/v1/namespaces/service-provider-registry/imagestreams/access  spec:    dockerImageRepository: 'docker-registry.example.com/service-provider-registry/access'    lookupPolicy:      local: false    tags:      - annotations: null        from:          kind: DockerImage          name: >-            docker-registry.example.com/service-provider-registry/access:latest        importPolicy:          scheduled: true        name: latest        referencePolicy:          type: Source |

##### oauth2-img.yaml

|  |
| --- |
| apiVersion: v1  kind: ImageStream  metadata:    name: oauth2\_authorization\_server    namespace: service-provider-registry    selfLink: /oapi/v1/namespaces/service-provider-registry/imagestreams/oauth2\_authorization\_server  spec:    dockerImageRepository: >-      docker-registry.example.com/service-provider-registry/oauth2\_authorization\_server    lookupPolicy:      local: false    tags:      - annotations: null        from:          kind: DockerImage          name: >-            docker-registry.example.com/service-provider-registry/oauth2\_authorization\_server:latest        importPolicy:          scheduled: true        name: latest        referencePolicy:          type: Source |

##### docsvc-img.yaml

|  |
| --- |
| apiVersion: v1  kind: ImageStream  metadata:    name: document\_service    namespace: service-provider-registry    selfLink: /oapi/v1/namespaces/service-provider-registry/imagestreams/document\_service  spec:    dockerImageRepository: 'docker-registry.example.com/document-services/document\_service'    lookupPolicy:      local: false    tags:      - annotations: null        from:          kind: DockerImage          name: >-            docker-registry.example.com/document-services/document\_service:latest        generation: 5        importPolicy:          scheduled: true        name: latest        referencePolicy:          type: Source |

#### Deployments

##### front-end-dep.yaml

|  |
| --- |
| apiVersion: v1  kind: DeploymentConfig  metadata:    labels:      app: service-provider-registry      service: frontend    name: frontend    namespace: service-provider-registry    selfLink: /oapi/v1/namespaces/service-provider-registry/deploymentconfigs/frontend  spec:    replicas: 1    selector:      app: service-provider-registry      service: frontend      deploymentconfig: frontend    strategy:      activeDeadlineSeconds: 21600      resources: {}      rollingParams:        intervalSeconds: 1        maxSurge: 25%        maxUnavailable: 25%        timeoutSeconds: 600        updatePeriodSeconds: 1      type: Rolling    template:      metadata:        labels:          app: service-provider-registry          deploymentconfig: frontend          service: frontend      spec:        containers:          - image: >-              docker-registry.example.com/service-provider-registry\_frontend/service-provider-registry\_fr:latest            imagePullPolicy: Always            name: frontend            ports:              - containerPort: 80                protocol: TCP              - containerPort: 8080                protocol: TCP            resources: {}            terminationMessagePath: /dev/termination-log            terminationMessagePolicy: File        dnsPolicy: ClusterFirst        imagePullSecrets:          - name: docker-registry        restartPolicy: Always        schedulerName: default-scheduler        securityContext: {}        terminationGracePeriodSeconds: 30    test: false    triggers:      - type: ConfigChange      - imageChangeParams:          automatic: true          containerNames:            - frontend          from:            kind: ImageStreamTag            name: 'frontend:latest'            namespace: service-provider-registry-test          lastTriggeredImage: >-            docker-registry.example.com/service-provider-registry\_frontend/service-provider-registry\_fr:latest        type: ImageChange |

##### counterparty-dep.yaml

|  |
| --- |
| apiVersion: v1  kind: DeploymentConfig  metadata:    labels:      app: service-provider-registry      service: counterparty    name: counterparty    namespace: service-provider-registry    selfLink: /oapi/v1/namespaces/service-provider-registry/deploymentconfigs/counterparty  spec:    replicas: 1    selector:      app: service-provider-registry      service: counterparty      deploymentconfig: counterparty    strategy:      activeDeadlineSeconds: 21600      resources: {}      rollingParams:        intervalSeconds: 1        maxSurge: 25%        maxUnavailable: 25%        timeoutSeconds: 600        updatePeriodSeconds: 1      type: Rolling    template:      metadata:        labels:          app: service-provider-registry          deploymentconfig: counterparty          service: counterparty      spec:        containers:          - env:              - name: spring\_datasource\_url                value: 'jdbc:postgresql://postgresql:5432/service-provider-registry\_test\_counterparty'              - name: spring\_datasource\_username                value: service-provider-registry\_counterparty\_user              - name: spring\_datasource\_password                value: service-provider-registry\_counterparty\_password              - name: server\_port                value: '8081'              - name: server\_context\_path                value: /counterparty              - name: oauth\_token\_server                value: '[http://oauth2:8080'](http://oauth2:8080'/)              - name: swagger\_enable                value: 'true'              - name: flyway\_enabled                value: 'false'            image: >-              docker-registry.example.com/service-provider-registry/counterparty:latest            imagePullPolicy: Always            name: counterparty            resources: {}            terminationMessagePath: /dev/termination-log            terminationMessagePolicy: File            volumeMounts:              - mountPath: /logdir                name: counterparty-1              - mountPath: /properties                name: counterparty-2        dnsPolicy: ClusterFirst        imagePullSecrets:          - name: gitlab        restartPolicy: Always        schedulerName: default-scheduler        securityContext: {}        terminationGracePeriodSeconds: 30        volumes:          - emptyDir: {}            name: counterparty-1          - emptyDir: {}            name: counterparty-2    test: false    triggers:      - type: ConfigChange      - imageChangeParams:          automatic: true          containerNames:            - counterparty          from:            kind: ImageStreamTag            name: 'counterparty:latest'            namespace: service-provider-registry          lastTriggeredImage: >-            docker-registry.example.com/service-provider-registry/counterparty:latest        type: ImageChange |

##### forms-dep.yaml

|  |
| --- |
| apiVersion: v1  kind: DeploymentConfig  metadata:    labels:      app: service-provider-registry      service: forms    name: forms    namespace: service-provider-registry    selfLink: /oapi/v1/namespaces/service-provider-registry/deploymentconfigs/forms  spec:    replicas: 1    selector:      app: forms      deploymentconfig: forms    strategy:      activeDeadlineSeconds: 21600      resources: {}      rollingParams:        intervalSeconds: 1        maxSurge: 25%        maxUnavailable: 25%        timeoutSeconds: 600        updatePeriodSeconds: 1      type: Rolling    template:      metadata:        labels:          app: service-provider-registry          deploymentconfig: forms          service: forms      spec:        containers:          - env:              - name: forms\_datasource\_url                value: 'jdbc:postgresql://postgresql:5432/service-provider-registry\_test\_forms'              - name: forms\_datasource\_username                value: forms\_datasource\_username              - name: forms\_datasource\_password                value: forms\_datasource\_password              - name: spring\_profiles\_include                value: flyway              - name: flyway\_enabled                value: 'true'              - name: oauth\_token\_server                value: '[http://oauth2:8080'](http://oauth2:8080'/)              - name: api\_service\_url                value: '[http://docsvc-app:9090'](http://docsvc-app:9090'/)              - name: server\_context\_path                value: /forms              - name: swagger\_enable                value: 'true'              - name: address\_space\_service\_url                value: 'http://addressapi-swagger.example.com'              - name: address\_kiev\_uuid                value: 538d7492-371b-11e7-b112-000c29ff5864              - name: address\_space\_login                value: address\_space\_login              - name: address\_space\_password                value: address\_space\_passord              - name: server\_port                value: '8080'            image: >-              docker-registry.example.com/service-provider-registry/forms:latest            imagePullPolicy: Always            name: forms            resources: {}            terminationMessagePath: /dev/termination-log            terminationMessagePolicy: File            volumeMounts:              - mountPath: /logdir                name: forms-1              - mountPath: /properties                name: forms-2        dnsPolicy: ClusterFirst        imagePullSecrets:          - name: docker-registry        restartPolicy: Always        schedulerName: default-scheduler        securityContext: {}        terminationGracePeriodSeconds: 30        volumes:          - emptyDir: {}            name: forms-1          - emptyDir: {}            name: forms-2    test: false    triggers:      - type: ConfigChange      - imageChangeParams:          automatic: true          containerNames:            - forms          from:            kind: ImageStreamTag            name: 'forms:latest'            namespace: service-provider-registry          lastTriggeredImage: >-            docker-registry.example.com/service-provider-registry/forms:latest        type: ImageChange |

##### documents-data-dep.yaml

|  |
| --- |
| apiVersion: v1  kind: DeploymentConfig  metadata:    labels:      app: service-provider-registry      service: documents-data    name: documents-data    namespace: service-provider-registry    selfLink: /oapi/v1/namespaces/service-provider-registry/deploymentconfigs/documents-data  spec:    replicas: 1    selector:      app: service-provider-registry      service: documents-data      deploymentconfig: documents-data    strategy:      activeDeadlineSeconds: 21600      resources: {}      rollingParams:        intervalSeconds: 1        maxSurge: 25%        maxUnavailable: 25%        timeoutSeconds: 600        updatePeriodSeconds: 1      type: Rolling    template:      metadata:        labels:          app: service-provider-registry          deploymentconfig: documents-data          service: documents-data      spec:        containers:          - env:              - name: docdata\_datasource\_url                value: 'jdbc:postgresql://postgresql:5432/service-provider-registry\_test\_docdata'              - name: docdata\_datasource\_username                value: docdata\_datasource\_username              - name: docdata\_datasource\_password                value: docdata\_datasource\_password              - name: oauth\_token\_server                value: '[http://oauth2:8080'](http://oauth2:8080'/)              - name: server\_port                value: '8080'              - name: spring\_profiles\_include                value: flyway              - name: flyway\_enabled                value: 'true'              - name: api\_service\_url                value: '[http://docsvc-app:9090'](http://docsvc-app:9090'/)              - name: server\_context\_path                value: /documents-data              - name: swagger\_enable                value: 'true'            image: >-              docker-registry.example.com/service-provider-registry/documents-data:latest            imagePullPolicy: Always            name: documents-data            resources: {}            terminationMessagePath: /dev/termination-log            terminationMessagePolicy: File            volumeMounts:              - mountPath: /logdir                name: documents-data-1              - mountPath: /properties                name: documents-data-2        dnsPolicy: ClusterFirst        imagePullSecrets:          - name: gitlab        restartPolicy: Always        schedulerName: default-scheduler        securityContext: {}        terminationGracePeriodSeconds: 30        volumes:          - emptyDir: {}            name: documents-data-1          - emptyDir: {}            name: documents-data-2    test: false    triggers:      - type: ConfigChange      - imageChangeParams:          automatic: true          containerNames:            - documents-data          from:            kind: ImageStreamTag            name: 'documents-data:latest'            namespace: service-provider-registry          lastTriggeredImage: >-            docker-registry.example.com/service-provider-registry/documents-data:latest        type: ImageChange |

##### entities-dep.yaml

|  |
| --- |
| apiVersion: v1  kind: DeploymentConfig  metadata:    labels:      app: service-provider-registry      service: entities    name: entities    namespace: service-provider-registry    selfLink: /oapi/v1/namespaces/service-provider-registry/deploymentconfigs/entities  spec:    replicas: 1    selector:      app: service-provider-registry      service: entities      deploymentconfig: entities    strategy:      activeDeadlineSeconds: 21600      resources: {}      rollingParams:        intervalSeconds: 1        maxSurge: 25%        maxUnavailable: 25%        timeoutSeconds: 600        updatePeriodSeconds: 1      type: Rolling    template:      metadata:        labels:          app: service-provider-registry          deploymentconfig: entities          service: entities      spec:        containers:          - env:              - name: entities\_datasource\_url                value: 'jdbc:postgresql://postgresql:5432/service-provider-registry\_test\_entities'              - name: entities\_datasource\_username                value: entities\_datasource\_username              - name: entities\_datasource\_password                value: entities\_datasource\_password              - name: server\_port                value: '8080'              - name: spring\_profiles\_include                value: flyway              - name: server\_context\_path                value: /entities              - name: oauth\_token\_server                value: '[http://oauth2:8080'](http://oauth2:8080'/)              - name: api\_service\_url                value: '[http://docsvc-app:9090'](http://docsvc-app:9090'/)              - name: flyway\_enabled                value: 'true'            image: >-              docker-registry.example.com/service-provider-registry/entities:latest            imagePullPolicy: Always            name: entities            resources: {}            terminationMessagePath: /dev/termination-log            terminationMessagePolicy: File            volumeMounts:              - mountPath: /logdir                name: entities-1              - mountPath: /properties                name: entities-2        dnsPolicy: ClusterFirst        imagePullSecrets:          - name: gitlab        restartPolicy: Always        schedulerName: default-scheduler        securityContext: {}        terminationGracePeriodSeconds: 30        volumes:          - emptyDir: {}            name: entities-1          - emptyDir: {}            name: entities-2    test: false    triggers:      - type: ConfigChange      - imageChangeParams:          automatic: true          containerNames:            - entities          from:            kind: ImageStreamTag            name: 'entities:latest'            namespace: service-provider-registry          lastTriggeredImage: >-            docker-registry.example.com/service-provider-registry/entities:latest        type: ImageChange |

##### access-dep.yaml

|  |
| --- |
| apiVersion: v1  kind: DeploymentConfig  metadata:    labels:      app: service-provider-registry      service: access    name: access    namespace: service-provider-registry    selfLink: /oapi/v1/namespaces/service-provider-registry/deploymentconfigs/access  spec:    replicas: 1    selector:      app: service-provider-registry      service: access      deploymentconfig: access    strategy:      activeDeadlineSeconds: 21600      resources: {}      rollingParams:        intervalSeconds: 1        maxSurge: 25%        maxUnavailable: 25%        timeoutSeconds: 600        updatePeriodSeconds: 1      type: Rolling    template:      metadata:        labels:          app: service-provider-registry          deploymentconfig: access          service: access      spec:        containers:          - env:              - name: oauth\_token\_server                value: '[http://oauth2:8080'](http://oauth2:8080'/)              - name: access\_jdbc\_url                value: >-                  jdbc:postgresql://postgresql:5432/service-provider-registry\_test\_access?currentSchema=access              - name: access\_jdbc\_user                value: entities\_access\_username              - name: access\_jdbc\_password                value: entities\_access\_password              - name: access\_audit\_jdbc\_url                value: >-                  jdbc:postgresql://postgresql:5432/service-provider-registry\_test\_access?currentSchema=access\_audit              - name: access\_audit\_jdbc\_user                value: entities\_access\_username              - name: access\_audit\_jdbc\_password                value: entities\_access\_password              - name: server\_port                value: '8080'              - name: server\_context\_path                value: /access              - name: flyway\_enabled                value: 'true'              - name: spring\_profiles\_include                value: flyway              - name: swagger\_enable                value: 'true'            image: >-              docker-registry.example.com/service-provider-registry/access:latest            imagePullPolicy: Always            name: access            resources: {}            terminationMessagePath: /dev/termination-log            terminationMessagePolicy: File            volumeMounts:              - mountPath: /logdir                name: access-1              - mountPath: /properties                name: access-2        dnsPolicy: ClusterFirst        imagePullSecrets:          - name: docker-registry        restartPolicy: Always        schedulerName: default-scheduler        securityContext: {}        terminationGracePeriodSeconds: 30        volumes:          - emptyDir: {}            name: access-1          - emptyDir: {}            name: access-2    test: false    triggers:      - type: ConfigChange      - imageChangeParams:          automatic: true          containerNames:            - access          from:            kind: ImageStreamTag            name: 'access:latest'            namespace: service-provider-registry          lastTriggeredImage: >-            docker-registry.example.com/service-provider-registry/access:latest        type: ImageChange |

##### oauth2-dep.yaml

|  |
| --- |
| apiVersion: v1  kind: DeploymentConfig  metadata:    labels:      app: service-provider-registry      service: oauth2    name: oauth2    namespace: service-provider-registry    selfLink: /oapi/v1/namespaces/service-provider-registry/deploymentconfigs/oauth2  spec:    replicas: 1    selector:      app: service-provider-registry      service: oauth2      deploymentconfig: oauth2    strategy:      activeDeadlineSeconds: 21600      resources: {}      rollingParams:        intervalSeconds: 1        maxSurge: 25%        maxUnavailable: 25%        timeoutSeconds: 600        updatePeriodSeconds: 1      type: Rolling    template:      metadata:        labels:          app: service-provider-registry          deploymentconfig: oauth2          service: oauth2      spec:        containers:          - env:              - name: tokenstore\_db\_url                value: 'jdbc:postgresql://postgresql:5432/service-provider-registry\_test\_access'              - name: tokenstore\_db\_user                value: tokenstore\_db\_user              - name: tokenstore\_db\_password                value: tokenstore\_db\_password            image: >-              docker-registry.example.com/service-provider-registry/oauth2\_authorization\_server:latest            imagePullPolicy: Always            name: oauth2            resources: {}            terminationMessagePath: /dev/termination-log            terminationMessagePolicy: File            volumeMounts:              - mountPath: /logdir                name: oauth2-1              - mountPath: /properties                name: oauth2-2        dnsPolicy: ClusterFirst        imagePullSecrets:          - name: docker-registry        restartPolicy: Always        schedulerName: default-scheduler        securityContext: {}        terminationGracePeriodSeconds: 30        volumes:          - emptyDir: {}            name: oauth2-1          - emptyDir: {}            name: oauth2-2    test: false    triggers:      - type: ConfigChange      - imageChangeParams:          automatic: true          containerNames:            - oauth2          from:            kind: ImageStreamTag            name: 'oauth2\_authorization\_server:latest'            namespace: service-provider-registry          lastTriggeredImage: >-            docker-registry.example.com/service-provider-registry/oauth2\_authorization\_server:latest        type: ImageChange |

##### docsvc-dep.yaml

|  |
| --- |
| apiVersion: v1  kind: DeploymentConfig  metadata:    labels:      app: service-provider-registry      service: docsvc-app    name: docsvc-app    namespace: service-provider-registry    selfLink: /oapi/v1/namespaces/service-provider-registry/deploymentconfigs/docsvc-app  spec:    replicas: 1    selector:      service: docsvc-app    strategy:      activeDeadlineSeconds: 21600      resources: {}      rollingParams:        intervalSeconds: 1        maxSurge: 25%        maxUnavailable: 25%        timeoutSeconds: 600        updatePeriodSeconds: 1      type: Rolling    template:      metadata:        creationTimestamp: null        labels:          app: service-provider-registry          service: docsvc-app      spec:        containers:          - image: >-              docker-registry.example.com/service-provider-registry/document\_service:latest            imagePullPolicy: IfNotPresent            name: docsvc-app            ports:              - containerPort: 8080                protocol: TCP            resources: {}            terminationMessagePath: /dev/termination-log            terminationMessagePolicy: File            volumeMounts:              - mountPath: /properties                name: docsvc-app-config              - mountPath: /logdir                name: docsvc-claim        dnsPolicy: ClusterFirst        restartPolicy: Always        schedulerName: default-scheduler        securityContext: {}        terminationGracePeriodSeconds: 30        volumes:          - configMap:              defaultMode: 420              name: docsvc-app-properties            name: docsvc-app-config          - name: docsvc-claim            persistentVolumeClaim:              claimName: docsvc-app-claim    test: false    triggers:      - type: ConfigChange      - imageChangeParams:          automatic: true          containerNames:            - docsvc-app          from:            kind: ImageStreamTag            name: 'document\_service:latest'            namespace: service-provider-registry          lastTriggeredImage: >-            docker-registry.example.com/service-provider-registry/document\_service:latest        type: ImageChange |

##### alfresco-dep.yaml

|  |
| --- |
| apiVersion: apps/v1beta1  kind: Deployment  metadata:    annotations:      deployment.kubernetes.io/revision: '1'      kompose.cmd: kompose convert    labels:      app: service-provider-registry      service: alfresco    name: alfresco    namespace: service-provider-registry    selfLink: /apis/apps/v1beta1/namespaces/service-provider-registry/deployments/alfresco  spec:    progressDeadlineSeconds: 600    replicas: 1    revisionHistoryLimit: 2    selector:      matchLabels:        app: service-provider-registry        service: alfresco    strategy:      type: Recreate    template:      metadata:        labels:          app: service-provider-registry          service: alfresco      spec:        containers:          - image: mikelasla/alfresco-standalone            imagePullPolicy: Always            name: alfresco            ports:              - containerPort: 8080                protocol: TCP            resources: {}            terminationMessagePath: /dev/termination-log            terminationMessagePolicy: File            volumeMounts:              - mountPath: /usr/local/alfresco/alf\_data                name: alfresco-claim              - mountPath: /usr/local/alfresco/tomcat/shared/classes                name: alfresco-global-properties        dnsPolicy: ClusterFirst        restartPolicy: Always        schedulerName: default-scheduler        securityContext: {}        terminationGracePeriodSeconds: 30        volumes:          - name: alfresco-claim            persistentVolumeClaim:              claimName: alfresco-claim          - configMap:              defaultMode: 420              name: alfresco-global-properties            name: alfresco-global-properties |

##### libreoffice-dep.yaml

|  |
| --- |
| apiVersion: apps/v1beta1  kind: Deployment  metadata:    annotations:      deployment.kubernetes.io/revision: '1'      kompose.cmd: kompose convert      kompose.version: 1.1.0 ()    labels:      app: service-provider-registry      service: libreoffice    name: libreoffice    namespace: service-provider-registry    selfLink: /apis/apps/v1beta1/namespaces/service-provider-registry/deployments/libreoffice  spec:    progressDeadlineSeconds: 600    replicas: 1    revisionHistoryLimit: 2    selector:      matchLabels:        app: service-provider-registry        service: libreoffice    strategy:      rollingUpdate:        maxSurge: 1        maxUnavailable: 1      type: RollingUpdate    template:      metadata:        creationTimestamp: null        labels:          app: service-provider-registry          service: libreoffice      spec:        containers:          - image: xcgd/libreoffice            imagePullPolicy: Always            name: libreoffice            ports:              - containerPort: 8997                protocol: TCP            resources: {}            terminationMessagePath: /dev/termination-log            terminationMessagePolicy: File        dnsPolicy: ClusterFirst        restartPolicy: Always        schedulerName: default-scheduler        securityContext: {}        terminationGracePeriodSeconds: 30 |

#### Services

##### frontend-svc.yaml

|  |
| --- |
| apiVersion: v1  kind: Service  metadata:    labels:      app: service-provider-registry      service: frontend    name: frontend    namespace: service-provider-registry    selfLink: /api/v1/namespaces/service-provider-registry/services/frontend  spec:    ports:      - name: 80-tcp        port: 80        protocol: TCP        targetPort: 80      - name: 8080-tcp        port: 8080        protocol: TCP        targetPort: 8080    selector:      deploymentconfig: frontend    sessionAffinity: None    type: ClusterIP  status:    loadBalancer: {} |

##### forms-svc.yaml

|  |
| --- |
| apiVersion: v1  kind: Service  metadata:    labels:      app: service-provider-registry      service: forms    name: forms    namespace: service-provider-registry    selfLink: /api/v1/namespaces/service-provider-registry/services/forms  spec:    ports:      - name: '8080'        port: 8080        protocol: TCP        targetPort: 8080    selector:      service: forms    sessionAffinity: None    type: ClusterIP  status:    loadBalancer: {} |

##### documents-data-svc.yaml

|  |
| --- |
| apiVersion: v1  kind: Service  metadata:    labels:      app: service-provider-registry      service: documents-data    name: documents-data    namespace: service-provider-registry    selfLink: /api/v1/namespaces/service-provider-registry/services/documents-data  spec:    ports:      - name: '8080'        port: 8080        protocol: TCP        targetPort: 8080    selector:      service: documents-data    sessionAffinity: None    type: ClusterIP  status:    loadBalancer: {} |

##### entities-svc.yaml

|  |
| --- |
| apiVersion: v1  kind: Service  metadata:    labels:      app: service-provider-registry      service: entities    name: entities    namespace: service-provider-registry    selfLink: /api/v1/namespaces/service-provider-registry/services/entities  spec:    ports:      - name: '8080'        port: 8080        protocol: TCP        targetPort: 8080    selector:      service: entities    sessionAffinity: None    type: ClusterIP  status:    loadBalancer: {} |

##### counterparty-svc.yaml

|  |
| --- |
| apiVersion: v1  kind: Service  metadata:    labels:      app: service-provider-registry      service: counterparty    name: counterparty    namespace: service-provider-registry    selfLink: /api/v1/namespaces/service-provider-registry/services/counterparty  spec:    ports:      - name: '8081'        port: 8081        protocol: TCP        targetPort: 8081    selector:      service: counterparty    sessionAffinity: None    type: ClusterIP  status:    loadBalancer: {} |

##### access-svc.yaml

|  |
| --- |
| apiVersion: v1  kind: Service  metadata:    labels:      app: service-provider-registry      service: access    name: access    namespace: service-provider-registry    selfLink: /api/v1/namespaces/service-provider-registry/services/access  spec:    ports:      - name: '8080'        port: 8080        protocol: TCP        targetPort: 8080    selector:      service: access    sessionAffinity: None    type: ClusterIP  status:    loadBalancer: {} |

##### oauth2-svc.yaml

|  |
| --- |
| apiVersion: v1  kind: Service  metadata:    labels:      app: service-provider-registry      service: oauth2    name: oauth2    namespace: service-provider-registry    selfLink: /api/v1/namespaces/service-provider-registry/services/oauth2  spec:    ports:      - name: '8080'        port: 8080        protocol: TCP        targetPort: 8080    selector:      service: oauth2    sessionAffinity: None    type: ClusterIP  status:    loadBalancer: {} |

##### docsvc-svc.yaml

|  |
| --- |
| apiVersion: v1  kind: Service  metadata:    labels:      app: service-provider-registry      service: docsvc-app    name: docsvc-app    namespace: service-provider-registry    selfLink: /api/v1/namespaces/service-provider-registry/services/docsvc-app  spec:    ports:      - name: '9090'        port: 9090        protocol: TCP        targetPort: 8080    selector:      service: docsvc-app    sessionAffinity: None    type: ClusterIP  status:    loadBalancer: {} |

##### alfresco-svc.yaml

|  |
| --- |
| apiVersion: v1  kind: Service  metadata:    annotations:      kompose.cmd: kompose convert    labels:      app: service-provider-registry      name: alfresco    name: alfresco    namespace: service-provider-registry    selfLink: /api/v1/namespaces/service-provider-registry/services/alfresco  spec:    ports:      - name: '8080'        port: 8080        protocol: TCP        targetPort: 8080    selector:      name: alfresco    sessionAffinity: None    type: ClusterIP  status:    loadBalancer: {} |

##### libreoffice-svc.yaml

|  |
| --- |
| apiVersion: v1  kind: Service  metadata:    annotations:      kompose.cmd: kompose convert      kompose.version: 1.1.0 ()    labels:      app: service-provider-registry      name: libreoffice    name: libreoffice    namespace: service-provider-registry    selfLink: /api/v1/namespaces/service-provider-registry/services/libreoffice  spec:    ports:      - name: '8997'        port: 8997        protocol: TCP        targetPort: 8997    selector:      name: libreoffice    sessionAffinity: None    type: ClusterIP  status:    loadBalancer: {} |

#### Routes

##### frontend-rt.yaml

|  |
| --- |
| apiVersion: v1  kind: Route  metadata:    labels:      app: service-provider-registry      service: frontend    name: frontend    namespace: service-provider-registry    selfLink: /oapi/v1/namespaces/service-provider-registry/routes/frontend  spec:    host: service-provider-registry.example.com    port:      targetPort: 80-tcp    to:      kind: Service      name: frontend      weight: 100    wildcardPolicy: None |

##### forms-rt.yaml

|  |
| --- |
| apiVersion: v1  kind: Route  metadata:    labels:      app: service-provider-registry      service: forms    name: forms    namespace: service-provider-registry    selfLink: /oapi/v1/namespaces/service-provider-registry/routes/forms  spec:    host: service-provider-registry.example.com    path: /forms    port:      targetPort: '8080'    to:      kind: Service      name: forms      weight: 100    wildcardPolicy: None |

##### documents-data-rt.yaml

|  |
| --- |
| apiVersion: v1  kind: Route  metadata:    labels:      app: service-provider-registry      service: documents-data    name: documents-data    namespace: service-provider-registry    selfLink: /oapi/v1/namespaces/service-provider-registry/routes/documents-data  spec:    host: service-provider-registry.example.com    path: /documents-data    port:      targetPort: '8080'    to:      kind: Service      name: documents-data      weight: 100    wildcardPolicy: None |

##### entities-rt.yaml

|  |
| --- |
| apiVersion: v1  kind: Route  metadata:    labels:      app: service-provider-registry      service: entities    name: entities    namespace: service-provider-registry    selfLink: /oapi/v1/namespaces/service-provider-registry/routes/entities  spec:    host: service-provider-registry.example.com    path: /entities    port:      targetPort: '8080'    to:      kind: Service      name: entities      weight: 100    wildcardPolicy: None |

##### counterparty-rt.yaml

|  |
| --- |
| apiVersion: v1  kind: Route  metadata:    labels:      app: service-provider-registry      service: counterparty    name: counterparty    namespace: service-provider-registry    selfLink: /oapi/v1/namespaces/service-provider-registry/routes/counterparty  spec:    host: service-provider-registry.example.com    path: /counterparty    port:      targetPort: '8081'    to:      kind: Service      name: counterparty      weight: 100    wildcardPolicy: None |

##### access-rt.yaml

|  |
| --- |
| apiVersion: v1  kind: Route  metadata:    labels:      app: service-provider-registry      service: access    name: access    namespace: service-provider-registry    selfLink: /oapi/v1/namespaces/service-provider-registry/routes/access  spec:    host: service-provider-registry.example.com    path: /access    port:      targetPort: '8080'    to:      kind: Service      name: access      weight: 100    wildcardPolicy: None |

##### docsvc-rt.yaml

|  |
| --- |
| apiVersion: v1  kind: Route  metadata:    labels:      app: service-provider-registry      service: doc-svc    name: docsvc-app    namespace: service-provider-registry    selfLink: /oapi/v1/namespaces/service-provider-registry/routes/docsvc-app  spec:    host: doc-mon.example.com    port:      targetPort: '9090'    to:      kind: Service      name: docsvc-app      weight: 100    wildcardPolicy: None |

##### alfresco-rt.yaml

|  |
| --- |
| apiVersion: v1  kind: Route  metadata:    labels:      app: service-provider-registry      service: alfresco    name: alfresco    namespace: service-provider-registry    selfLink: /oapi/v1/namespaces/service-provider-registry/routes/alfresco  spec:    host: alfresco-mon.example.com    port:      targetPort: '8080'    to:      kind: Service      name: alfresco      weight: 100    wildcardPolicy: None |

### Розгортання додатку

Розгортання додатку має вигляд послідовного завантаження файлів конфігурації сервісів в кластер за допомогою утиліти oc.

**Увага!** Послідовність виконання команд має значення!

Наприклад, конфігурації Services повинні бути завантажені в останню чергу!

Для завантаження сервісів у кластер виконуємо такі команди:

|  |
| --- |
| $ cd service-provider-registry/openshift/  $ oc create -f docker-sec.yaml  $  $ oc create -f alfresco-properties-cfg.yaml  $ oc create -f docsvc-properties-cfg.yaml  $  $ oc create -f access-img.yaml  $ oc create -f counterparty-img.yaml  $ oc create -f docsvc-img.yaml  $ oc create -f documents-data-img.yaml  $ oc create -f entities-img.yaml  $ oc create -f forms-img.yaml  $ oc create -f frontend-img.yaml  $ oc create -f oauth2-img.yaml  $  $ oc create -f front-end-dep.yaml  $ oc create -f counterparty-dep.yaml  $ oc create -f forms-dep.yaml  $ oc create -f documents-data-dep.yaml  $ oc create -f entities-dep.yaml  $ oc create -f access-dep.yaml  $ oc create -f oauth2-dep.yaml  $ oc create -f docsvc-dep.yaml  $ oc create -f alfresco-dep.yaml  $ oc create -f libreoffice-dep.yaml  $  $ oc create -f svc |

Остання команда створює об'єкти кластера з усіх файлів конфігурації, які знаходяться в каталозі service-provider-registry/openshift/svc/.

Після того як всі сервіси будуть створені в кластері й стартують, додаток почне працювати у відповідності до заявленої функціональності у документації. Дістатися його можна буде за адресою **service-provider-registry.example.com.**

# ПЕРЕВІРКА ВІРНОГО РОЗГОРТАННЯ ОПЕРАЦІЙНОГО СЕРЕДОВИЩА

Перевірити вірність розгортання операційного середовища можливо за такими ознаками:

* 1. Відсутні помилок на етапах завантаження конфігурації в кластер.
  2. Після завантаження кожному файлу з переліку (п. 5.3.2.) в кластері, у відповідних розділах, створився об’єкт (віртуальна машина), який співвідноситься з файлом.
  3. У створеному об’єкті при встановлені додатку відсутні помилки.
  4. Після завантаження при переході за адресою *service-provider-registry.kyivcity.gov.ua* відкривається сторінка входу у програмний модуль «Реєстр суб’єктів, що надають послуги».

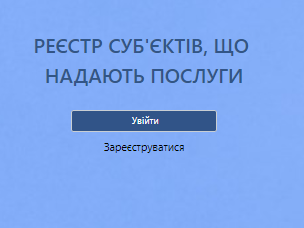


Рисунок 1. Головна сторінка програмого модулю «РСНП»