



Politechnika Łódzka

Instytut Informatyki

PRACA DYPLOMOWA INŻYNIERSKA

Budowa prywatnej chmury PaaS przy użyciu otwartego oprogramowania

Wydział Fizyki Technicznej, Informatyki i Matematyki Stosowanej

Promotor: dr inż. Michał Karbowańczyk

Dyplomant: Mateusz Maciejewski

Nr albumu: 165467

Kierunek: Informatyka

Specjalność: Sieci Komputerowe i Systemy Rozproszone

Łódź (20.09.2014)



Instytut Informatyki

90-924 Łódź, ul. Wólczańska 215, **budynek B9**

tel. 042 631 27 97, 042 632 97 57, fax 042 630 34 14 email: office@ics.p.lodz.pl

Spis treści

1. Wstęp.....	5
1.1. Wprowadzenie.....	5
1.2. Cel pracy.....	5
1.3. Układ pracy.....	5
1.4. Słownik pojęć.....	6
2. Teoretyczne podstawy chmur obliczeniowych.....	7
2.1. Technologie będące kamieniami milowymi w rozwoju chmur.....	7
2.2. Definicja chmury.....	7
2.3. Cechy charakterystyczne rozwiązań chmurowych.....	8
2.4. Modele wdrożeniowe.....	9
2.4.1. Zalety chmur prywatnych.....	10
2.5. Modele dostawcze.....	11
2.5.1. Zalety chmury typu Platform-as-a-Service.....	11
3. Przegląd dostępnych rozwiązań.....	13
3.1. Cloud Foundry.....	13
3.2. Paasmaker.....	14
3.3. Tsuru.....	14
3.4. OpenShift Origin.....	15
3.5. Wybór rozwiązania.....	16
4. Teoretyczne podstawy wybranego rozwiązania.....	18
4.1. Architektura.....	18
4.1.1. Zarządca.....	19
4.1.2. Węzeł.....	20
4.1.3. Kontener.....	20
4.1.4. Kasetta.....	22
4.1.5. Rejon.....	23
4.2. Aplikacje.....	23
4.2.1. Proces tworzenia aplikacji.....	24
4.2.2. Aplikacje nieskalowalne.....	26
4.2.3. Aplikacje skalowalne.....	26
4.3. Wykorzystywane technologie.....	27
4.3.1. SSH.....	27
4.3.2. Git.....	29
4.3.3. PAM.....	30
4.3.4. Control Groups.....	32
4.3.5. SELinux.....	34
4.3.6. BIND.....	36
4.3.7. MongoDB.....	37
4.3.8. MCollective.....	38
4.3.9. HAProxy.....	39
4.3.10. Lokkit.....	40
5. Uruchomienie serwera chmurowego.....	41
5.1. Konfiguracja testowa.....	41
5.2. Instalacja oprogramowania.....	41
5.2.1. Przygotowanie systemów.....	42
5.2.2. Zarządca oraz węzeł pierwszy.....	43

5.2.2.1. DNS.....	43
5.2.2.2. MongoDB.....	46
5.2.2.3. ActiveMQ.....	48
5.2.2.4. Klient MCollective.....	50
5.2.2.5. Zarządca.....	50
5.2.2.6. Wtyczki wykorzystywane przez zarządcę.....	53
5.2.2.7. Konsola administracyjna.....	54
5.2.2.8. Konsola użytkownika.....	55
5.2.2.9. Konfiguracja kluczy SSH w systemie węzła.....	56
5.2.2.10. Serwer MCollective.....	57
5.2.2.11. Węzeł.....	59
5.2.2.12. Instalacja kaset.....	60
5.2.2.13. Konfiguracja współdzielenia zasobów węzła.....	61
5.2.2.14. Konfiguracja aplikacji węzła.....	64
5.2.3. Węzeł drugi.....	64
5.2.3.1. Rejestracja systemu węzła w systemie DNS.....	65
5.2.3.2. Konfiguracja kluczy SSH w systemie węzła.....	65
5.2.3.3. Konfiguracja węzła.....	66
5.2.4. Konfiguracja rejonu.....	67
5.3. Wdrożenie aplikacji testowych.....	68
5.3.1.1. Błąd związany z błędną konfiguracją zapory sieciowej.....	72
5.4. Przenoszenie sesji HTTP pomiędzy instancjami aplikacji.....	73
5.4.1. Dystrybucja żądań pomiędzy kontenerami.....	73
5.4.2. Przenoszenie sesji pomiędzy kontenerami.....	74
5.4.3. Przenoszenie sesji HTTP za pośrednictwem bazy danych.....	78
6. Podsumowanie.....	81
7. Bibliografia.....	82
8. Spis rysunków.....	83
9. Spis tabel.....	85

1. Wstęp

1.1. Wprowadzenie

Obecnie wiele firm oraz osób prywatnych decyduje się na wykorzystanie chmury obliczeniowej. Główną korzyścią, przemawiającą za użyciem wspomnianego rozwiązania, jest duża oszczędność zasobów finansowych.. Przeniesienie części infrastruktury informatycznej firmy do chmury zmniejsza wydatki związane z obsługą działu IT poprzez zmniejszenie zapotrzebowania na sprzęt komputerowy oraz wyszkolony personel zdolny do stworzenia i utrzymania infrastruktury informatycznej firmy.

Zgodnie z badaniami przeprowadzonymi w latach 2009-2012, w 2012 roku z chmury obliczeniowej w Polsce korzystało 6,2% przedsiębiorstw. Usługa ta była wykorzystywana najczęściej w firmach prowadzących swoją działalność w zakresie konserwacji sprzętu komunikacyjnego oraz informacji i komunikacji (odpowiednio 22,6% oraz 20,8%)[1]. Chmura obliczeniowa zyskała również i wciąż zyskuje uznanie na świecie. Zgodnie z przewidywaniami, dochód uzyskany poprzez wykorzystanie tej usługi na świecie wyniesie w roku 2015, w zależności od źródeł, od 43 do 94 miliardów dolarów[2].

Niniejsza praca opisuje możliwość uruchomienia prywatnej chmury obliczeniowej opisywanej przez specjalistów jako chmura Platform-as-a-Service (PaaS). Przedstawione rozwiązanie można uruchomić, wykorzystując standardową konfigurację komputera wykorzystywaną przez zwykłych użytkowników.

1.2. Cel pracy

Realizowana praca ma na celu przedstawienie budowy oraz procesu wdrożenia prywatnej chmury obliczeniowej Platform-as-a-Service. Kolejnym ważnym celem jest przedstawienie możliwości stworzonej infrastruktury, z wyszczególnieniem jej mocnych i słabych stron. Do stworzenia omawianego rozwiązania zostanie wykorzystane wolne oprogramowanie, co pozwoli na w pełni legalne korzystanie z oprogramowania. Chmura zostanie uruchomiona na standardowej konfiguracji sprzętowej, co pozwoli na wdrożenie rozwiązania bez dodatkowych nakładów finansowych związanych z

zakupem sprzętu.

1.3. Układ pracy

Niniejszy rozdział zawiera wstęp, cele pracy oraz słownik pojęć, który będzie wykorzystywany w dalszej części pracy. W rozdziale drugim opisane zostały podstawy teoretyczne, leżące u podstaw rozwiązań opisywanych jako chmury obliczeniowe. Rozdział trzeci zawiera przegląd dostępnych rozwiązań. W rozdziałach czwartym oraz piątym zostały opisane odpowiednio: architektura wybranego rozwiązania oraz proces wdrożenia serwera, wraz z testem możliwości. W podsumowaniu przedstawione zostały wnioski wynikające z przeprowadzonych testów.

1.4. Słownik pojęć

Zdecydowana większość materiałów dotyczących tematu pracy jest dostępna w języku angielskim. Większość najważniejszych określeń nie posiada odpowiedników w języku polskim. W celu zachowania spójności stylistycznej pracy, zostały zaproponowane własne odpowiedniki określeń anglojęzycznych. Wykaz określeń oryginalnych oraz ich odpowiedników zawiera Tabela 1.

Tabela 1. Polskie odpowiedniki sformułowań występujących w literaturze anglojęzycznej

Określenie angielskie	Proponowany odpowiednik
Broker	Zarządca
Node	Węzeł
Cartridge	Kaseta
Gear	Kontener
District	Rejon

2. Teoretyczne podstawy chmur obliczeniowych

2.1. Technologie będące kamieniami milowymi w rozwoju chmur

Chmura obliczeniowa nie jest zupełnie nowym rozwiązaniem. Wykorzystuje ona wiele mechanizmów, które pojawiły się na świecie już wcześniej. Pojawienie się chmury jest efektem ciągłego rozwoju rozwiązań stosowanych w informatyce.

Technologii, które położyły podwaliny pod rozwiązania chmurowe, jest wiele. Powszechnie uważa się, że największy wpływ na kształt i rozwój chmur obliczeniowych miały następujące rozwiązania[3]:

- klaster (ang. *cluster*) – jest to zestaw niezależnych jednostek obliczeniowych połączonych ze sobą za pomocą przeznaczonych specjalnie do tego celu łączów komunikacyjnych, zapewniających możliwość szybkiej synchronizacji urządzeń wchodzących w skład klastra. Urządzenia te funkcjonują jako pojedynczy system obliczeniowy. Elementy składowe klastra powinny być zbliżone pod względem mocy obliczeniowej oraz wykorzystywanego oprogramowania. Takie rozwiązanie pozwala na zwiększenie niezawodności systemu, gdyż zadania uszkodzonego elementu mogą zostać przejęte przez inne, zdolne do pracy urządzenie.
- sieć obliczeniowa (ang. *computing grid*) – podobnie jak klaster, składa się z wielu niezależnych jednostek. Jednostki te są zorganizowane w pule logiczne, którymi można wspólnie zarządzać. Umożliwia to stworzenie wydajnej sieci obliczeniowej, w skład której mogą wchodzić jednostki o zróżnicowanej mocy obliczeniowej, umieszczone w różnych lokacjach geograficznych.
- wirtualizacja (ang. *virtualization*) – technologia umożliwiająca uruchomienie wielu wirtualnych jednostek obliczeniowych umieszczonych na jednej jednostce fizycznej. Umożliwia ona wykorzystywanie możliwości fizycznej jednostki obliczeniowej przez wielu użytkowników w sposób zapewniający izolację danych i procesów należących od innych użytkowników. Takie podejście pozwala wykorzystać w większym stopniu możliwości posiadanego sprzętu.

2.2. Definicja chmury

W ciągu ostatnich lat pojawiło się kilka definicji chmury, opublikowanych przez różne instytucje. Według raportu firmy Gartner[3], chmura to styl wykorzystywania komputerów, w którym zasoby obliczeniowe dostarczane są do klientów za pośrednictwem Internetu. Zasoby wchodzące w skład chmury są traktowane w sposób elastyczny – można zwiększać i zmniejszać ich możliwości w zależności od potrzeb klienta. Inną definicję zaproponowała firma Forrester Research[3]. Zgodnie z nią, chmura obliczeniowa to ujednolicone możliwości przetwarzania informacji (rozumiane jako usługi, oprogramowanie, bądź infrastruktura) udostępniane w sposób samoobsługowy za pośrednictwem Internetu. Wspomniane możliwości opłacane są według wykorzystania – klienci płacą za rzeczywiście wykorzystane zasoby (czas procesora, przestrzeń dyskowa oraz inne)[3].

Powszechną akceptację w środowiskach przemysłowych uzyskała definicja zaproponowana przez amerykański Narodowy Instytut Standaryzacji i Technologii (ang. *National Institute of Standards and Technology*, w skrócie NIST), opublikowana po raz pierwszy w roku 2009[3]. Obecne brzmienie uzyskała ona w roku 2011, kiedy to NIST wydał jej uaktualnioną wersję[3]. Według omawianej definicji, chmura obliczeniowa to model umożliwiający powszechny i wygodny dostęp do współdzielonej puli zasobów (takich jak serwery, aplikacje, usługi oraz innych), która może być udostępniona klientowi na jego żądanie, przy minimalnym nakładzie pracy ze strony dostawcy takiej usługi. W ramach modelu opisane jest pięć cech charakterystycznych dla chmur, cztery modele wdrożeniowe oraz trzy modele dostawcze[4].

W tej pracy za definicję chmury przyjęta została propozycja NIST. Wspomniane cechy oraz modele zostały opisane w dalszej części rozdziału.

2.3. Cechy charakterystyczne rozwiązań chmurowych

Każde rozwiązanie informatyczne posiada swoje charakterystyczne cechy, wyznaczające jego funkcjonalność oraz możliwości. W przypadku rozwiązań chmurowych cechy te zostały zdefiniowane przez NIST. Każda z nich ma istotny wpływ na funkcjonalność chmury obliczeniowej[3].

Pierwszą istotną cechą chmur jest możliwość automatycznego wykorzystywania zasobów udostępnianych w ramach usługi. Zasoby, raz skonfigurowane przez dostawcę usługi, mogą być oferowane użytkownikom na ich żądanie. Przydzielenie zasobów odbywa się w sposób zautomatyzowany, niewymagający interwencji oraz nadzoru ze strony człowieka. Cecha ta pozwala na wykorzystywanie zasobów chmury w sposób definiowany przez użytkowników rozwiązania chmurowego.

Kolejną cechą charakterystyczną, blisko związaną z opisaną poprzednio, jest zdolność chmury do mierzenia stopnia wykorzystywania udostępnianych przez nią zasobów. Dzięki temu mechanizmowi, użytkownik usługi chmurowej płaci jedynie za zasoby, które realnie wykorzystywał w pewnej jednostce czasu. Możliwość zbierania informacji dotyczących wykorzystywania zasobów jest cennym narzędziem również dla osób odpowiedzialnych za utrzymanie usługi. Umożliwia bowiem rozsądne zaplanowanie rozbudowy bądź konserwacji sprzętu.

Rozwiązanie chmurowe powinno być również powszechnie dostępne. Pod tym pojęciem kryje się wsparcie dla różnych sposobów dostępu do zasobów oferowanych przez chmurę. Jako sposoby dostępu rozumiane tu są różne urządzenia elektroniczne, protokoły komunikacyjne oraz metody autoryzacji. Chmura powinna być dostosowana do potrzeb poszczególnych grup jej użytkowników.

Niezwykle istotną cechą chmur jest również współdzielenie zasobów. Istotą tej funkcjonalności jest obsługa pewnej liczby użytkowników przez pojedynczą instancję zasobu udostępnianego przez chmurę (na przykład procesora, pamięci operacyjnej, przestrzeni dyskowej). Informacje i procesy użytkowników współdzielących daną instancję są od siebie odizolowane – pojedynczy użytkownik może wykorzystywać zasób tak, jakby był przeznaczony wyłącznie dla niego. Współdzielenie zasobów jednej instancji zasobu umożliwia również ich dynamiczne przydzielanie – część zasobów, niewykorzystywana obecnie przez jednego z użytkowników, może zostać przydzielona temu użytkownikowi, który potrzebuje ich w tej chwili.

Za najważniejszą cechę rozwiązań chmurowych uważana jest, wspomniana już wcześniej, ich elastyczność. Może dojść do sytuacji, że zasoby przydzielone początkowo użytkownikowi okażą się niewystarczające. W takim przypadku chmura przydziela użytkownikowi dodatkowe zasoby. W chwili, gdy nie są one już potrzebne,

zostają zwolnione i powracają do puli zasobów dostępnych dla wszystkich użytkowników. Najczęściej proces zarządzania dynamicznym przydziałem zasobów jest zautomatyzowany, dzięki czemu może skutecznie wspierać potrzeby użytkowników przez całą dobę[3].

2.4. Modele wdrożeniowe

Modele wdrożeniowe (ang. *deployment models*) dzielą rozwiązania chmurowe według ich dostępności oraz właściciela. Rozróżniane są cztery podstawowe modele wdrożeniowe, chociaż nic nie wyklucza powstawania innych, bardziej specjalistycznych modeli.

Chmura publiczna (ang. *public cloud*) jest najbardziej znanym modelem wdrożeniowym. Właścicielem chmury jest firma trzecia, udostępniająca usługę w sieci. Właściciel jest odpowiedzialny za uruchomienie i utrzymanie infrastruktury informatycznej. Usługa ta jest dostępna dla każdego zainteresowanego za odpowiednią opłatą, uzależnioną od ilości wykorzystywanych zasobów.

Chmura społecznościowa (ang. *community cloud*) to model, w którym usługa jest dostępna tylko dla pewnej określonej grupy odbiorców, posiadających wspólne cele bądź oczekiwania (na przykład wymagania dotyczące bezpieczeństwa). Właścicielami chmury mogą być członkowie społeczności z niej korzystającej.

Kolejnym modelem wdrożeniowym jest chmura prywatna (ang. *private cloud*). W przypadku tego modelu ta sama organizacja jest zarówno dostawcą usługi, jak i jej klientem. Chmura taka nie jest dostępna dla szerokiego grona odbiorców – jest wykorzystywana przez właściciela do usprawnienia wewnętrznych działań.

Ostatnim z podstawowych modeli jest chmura hybrydowa (ang. *hybrid cloud*). Łączy on cechy różnych modeli. Najczęściej spotykanym zastosowaniem tego modelu jest podejście znane jako „cloud-bursting” - w przypadku, gdy zasoby chmury prywatnej okazują się niewystarczające, dodatkowa pula zasobów jest pozyskiwana z chmury publicznej. Ze względu na różnorodność dostępnych rozwiązań, budowa chmury hybrydowej może stanowić duże wyzwanie[3][4].

2.4.1. Zalety chmur prywatnych

Korzyści płynące z wykorzystywania chmur prywatnych nie wynikają z oszczędności zasobów finansowych, jak ma to miejsce w przypadku chmur publicznych. Zalety tego rodzaju chmur ujawniają się, gdy rozpatrzy się pewne negatywne skutki wykorzystywania chmur publicznych.

Pierwszy z nich dotyczy bezpieczeństwa danych. Po przeniesieniu do chmury publicznej, informacje nie są już chronione przez mechanizmy bezpieczeństwa działające w firmie. Za bezpieczeństwo danych w chmurze odpowiada bowiem dostawca usługi chmurowej. Jeżeli stosowane przez niego mechanizmy zawiodą, dane firmy mogą zostać zmodyfikowane przez nieupoważnione do tego osoby bądź wykradzione.

Kolejnym skutkiem przeniesienia działalności do chmury jest utrata pełnej kontroli nad sposobem jej działania. W przypadku, gdy chmura przestanie funkcjonować, usługi firmy opierające się na niej również nie będą spełniać swojej roli. Może to doprowadzić do tego, że firma utraci zaufanie swoich klientów.

Niejednokrotnie zdarza się, że infrastruktura usługi chmurowej jest rozmieszczona na kilku kontynentach. Firma, przenosząc dane do chmury, nie ma zazwyczaj kontroli nad tym, gdzie zostaną one fizycznie zapisane. Może to mieć pewne konsekwencje prawne – na przykład w Wielkiej Brytanii prawo wymaga, żeby dane osobowe obywateli były przechowywane na terenie Zjednoczonego Królestwa[3].

Wykorzystanie chmury prywatnej pozwala znacznie ograniczyć wyżej opisane problemy. Dzięki temu firma może korzystać z zalet chmur obliczeniowych, nie tracąc kontroli nad danymi oraz usługami istotnymi dla jej prawidłowego funkcjonowania na rynku.

2.5. Modele dostawcze

Modele dostawcze określają zasoby, które usługa chmurowa udostępnia swoim użytkownikom. Trzy z nich zostały dobrze opisane i uznane za podstawowe.

Pierwszym z nich jest model Infrastructure-as-a-Service (IaaS). Chmura tego typu udostępnia swoim użytkownikom „czyste” zasoby komputerowe - takie jak moc obliczeniowa, przestrzeń dyskowa czy łącza sieciowe. Wybór oprogramowania,

począwszy od systemu operacyjnego, należy do użytkownika usługi. Tego typu rozwiązania chmurowe wymagają najwięcej wysiłku oraz wiedzy ze strony użytkownika, w zamian zapewniając mu największą kontrolę nad wykorzystywanymi zasobami.

Inne podejście prezentują chmury typu Platform-as-a-Service (PaaS). Chmury te udostępniają już skonfigurowane i gotowe do pracy zasoby. Zawierają one zestaw narzędzi służący do tworzenia i rozwoju aplikacji. W skład oferowanych zasobów wchodzi takie elementy jak kompilatory i interpretery różnych języków programowania, bazy danych, serwery aplikacyjne i inne. Ceną za te udogodnienia jest mniejsza niż w przypadku modelu IaaS kontrola użytkownika nad zasobami. Najczęściej ogranicza się ona do wyboru narzędzi służących do rozwoju aplikacji.

Ostatnim z opisywanych modeli jest model Software-as-a-Service (SaaS). Użytkownikom udostępniane jest gotowe oprogramowanie, przeznaczone dla pewnej grupy odbiorców (na przykład aplikacja obsługująca sklep internetowy). Kontrola użytkownika w tym przypadku ogranicza się do administrowania otrzymaną aplikacją[3][4].

2.5.1. Zalety chmury typu Platform-as-a-Service

Chmury typu PaaS, ze względu na swoją charakterystykę, są najbardziej użyteczne dla programistów. Dzięki nim programista może skoncentrować się na swoim głównym zadaniu – tworzeniu kodu aplikacji. Nie musi już zajmować się uruchomieniem i utrzymaniem takich elementów jak baza danych czy serwer aplikacyjny. Pozwala to skrócić czas tworzenia aplikacji.

Chmura ta posiada również zalety z punktu widzenia projektantów aplikacji oraz zarządu firmy. Projektanci mogą w swobodny sposób testować rozwiązania wykorzystujące różne technologie, nie potrzebując specjalistów potrafiących owe technologie uruchomić. Prototypy aplikacji mogą zostać uruchomione w ciągu kilku dni, zamiast tygodni czy miesięcy. Zmianie ulegną również koszty tworzenia nowych projektów. Dzięki wykorzystaniu chmury typu PaaS koszty projektów mogą zostać w znaczący sposób ograniczone. Pozwala to firmie na swobodniejsze rozwijanie nowych projektów, niż miało to miejsce w przypadku tradycyjnej organizacji działu IT[5].

3. Przegląd dostępnych rozwiązań

Bardzo ważnym krokiem podczas procesu wdrażania chmury prywatnej jest wybór odpowiedniego rozwiązania. Wybór niewłaściwego serwera chmurowego może wymusić konieczność wprowadzenia poważnych zmian w aplikacji przeznaczonej do uruchomienia w chmurze – w skrajnych przypadkach może zaistnieć konieczność całkowitego przepisania kodu aplikacji. Podobne zjawisko ma miejsce w przypadku bazy danych – w przypadku zmiany serwera bazodanowego należy liczyć się z koniecznością pewnych zmian w strukturze bazy. Serwer chmurowy powinien więc oferować wsparcie dla języków programowania, usług oraz technologii, z których korzysta aplikacja przeznaczona do uruchomienia w środowisku chmurowym.

Należy również pamiętać, że dotychczas nie zostały ustalone żadne powszechnie uznawane standardy dotyczące komunikacji pomiędzy rozwiązaniami chmurowymi. Ten brak standaryzacji dotyczy również samej budowy rozwiązań chmurowych[3]. Z tego powodu przeniesienie aplikacji z jednej chmury do innej, opartej na innym rozwiązaniu, może wymagać wiele pracy. Problemem również może się okazać ewentualne przejście do modelu hybrydowego, pozwalającego na zwiększenie możliwości chmury.

3.1. Cloud Foundry

Jest to rozwiązanie, nad którego rozwojem czuwa grupa o nazwie The Cloud Foundry Foundation. Jej członkami są takie firmy, jak Hewlett Packard, Accenture, Intel czy Pitaval[6].

Cloud Foundry wykorzystuje system paczek (ang. *buildpacks*) w celu zapewnienia aplikacjom wsparcia w postaci odpowiednich rozszerzeń oraz usług. Serwer chmurowy w chwili umieszczenia na nim kodu aplikacji automatycznie wykrywa, które paczki należy zapewnić w celu uruchomienia aplikacji. Listę standardowo dostępnych paczek przedstawia Tabela 2.

Tabela 2. Lista paczek dostępnych w standardowej instancji serwera Cloud Foundry

Nazwa paczki	Wspierane oprogramowanie
Go	Go
Java	Java, Grails, Spring oraz każdy język programowania wykorzystujący maszynę wirtualną Javy
Node.js	Node.js oraz JavaScript
PHP	PHP
Python	Python
Ruby	Ruby, Rack, Rails, Sinatra

Możliwości serwera nie są ograniczone do oprogramowania wspieranego przez wyżej wymienione paczki. W przypadku wykorzystywania języka nie posiadającego oficjalnego wsparcia, użytkownicy mogą wykorzystać również paczki stworzone przez społeczność skoncentrowaną wokół projektu. Doświadczeni programiści mogą również samodzielnie stworzyć paczkę bądź przystosować jedną z istniejących do własnych potrzeb[7].

3.2. Paasmaker

Paasmaker jest rozwiązaniem tworzonym przez zespół prowadzony przez Daniela Foote'a. Pierwsza wersja serwera została wydana w kwietniu 2013 roku[8].

Zgodnie z założeniami projektowymi, serwer ten będzie posiadał wygodny interfejs dostępny przez przeglądarkę internetową. Oprócz tego zapewni również pełne API, dzięki czemu będzie mógł zostać zintegrowany z już istniejącymi usługami. Kolejnym kluczowym założeniem jest możliwość rozbudowy serwera za pomocą wtyczek (ang. *plugins*) – pozwoli to programistom oraz administratorom na wygodne rozwijanie możliwości chmury. Proces wdrożenia chmury zostanie uproszczony, jednak wciąż będzie zapewniał kontrolę nad konfiguracją serwera[9].

Obecnie, Paasmaker wspiera następujące języki oraz platformy:

- PHP
- Ruby

- Python
- Node.js

Programiści mogą tworzyć własne wtyczki, rozszerzając zakres języków wspieranych przez serwer[10].

3.3. Tsuru

Tsuru to otwarta chmura typu PaaS rozwijana przez zespół pod kierownictwem Francisco Souzy. Pierwsza wersja serwera została opublikowana w 2012 roku. Najnowsza wersja ma numer 0.5.2 i została wydana w lipcu 2014 roku[11].

Tsuru umożliwia szybkie wdrożenie aplikacji za pomocą podstawowych operacji oferowanych przez repozytorium Git. Dzięki temu użytkownik może w wygodny sposób tworzyć i wdrażać różne wersje swojej aplikacji. Serwer pozwala również na szybkie utworzenie instancji serwera bazodanowego i powiązanie go z aplikacją użytkownika. Oprócz tego, Tsuru umożliwia wykorzystywanie istniejących usług do obsługi aplikacji uruchomionej w chmurze. Jest to możliwe dzięki wykorzystywaniu do konfiguracji zmiennych środowiskowych[12] .

W chwili obecnej Tsuru wspiera następujące języki oraz środowiska:

- Node.js
- PHP
- Python
- Ruby
- Go
- Java

Oprócz języków, Tsuru oferuje wsparcie dla kilku serwerów baz danych:

- MySQL
- MongoDB
- Cassandra Memcached

- Redis

Istnieje również możliwość wykorzystania usług niewspieranych bezpośrednio przez serwer. W tym celu użytkownik musi dostarczyć odpowiednie API, które będzie wykorzystywane przez serwer do komunikacji z usługą[13].

3.4. OpenShift Origin

Rozwiązanie to powstało w wyniku udostępnienia przez firmę Red Hat kodu źródłowego serwera chmurowego OpenShift Enterprise opracowanego przez tę firmę[14]. Zarząd firmy uznał, że uwolnienie kodu przyczyni się do jego dalszego, dynamicznego rozwoju. Udostępnienie serwera chmurowego pod postacią wolnego oprogramowania oferuje korzyści również dla jego użytkowników, między innymi:

- możliwość uruchomienia chmury na komputerze lokalnymi bądź za własną zaporą sieciową
- możliwość zintegrowania ulubionych narzędzi programistycznych z chmurą
- możliwość budowy własnych rozwiązań na podstawie technologii chmurowych[14].

Serwer OpenShift Origin wykorzystuje mechanizm kaset (ang. *cartridge*). Dzięki niemu możliwe jest zwiększanie puli wspieranych języków programowania oraz technologii bez konieczności zmian w architekturze serwera. Obecnie, OpenShift Origin zapewnia wsparcie dla następujących języków programowania:

- Ruby
- Perl
- PHP
- Python

Użytkownicy mogą również wykorzystać inne języki programowania, wykorzystując w tym celu kasety Do-It-Yourself. Do dyspozycji programistów wykorzystujących serwer OpenShift twórcy oddali również dodatkowe oprogramowanie oraz usługi:

- Cron
- Jboss
- Tomcat
- Jenkins
- MariaDB
- PostgreSQL
- MongoDB
- phpMyAdmin

W tym przypadku nie istnieje dedykowana kasetę, za pomocą której programiści mogliby uruchamiać oprogramowanie niewspierane przez serwer. Doświadczeni programiści mogą jednak samodzielnie stworzyć odpowiednią kasetę, rozwijając w ten sposób możliwości opisywanego serwera chmurowego[15].

3.5. Wybór rozwiązania

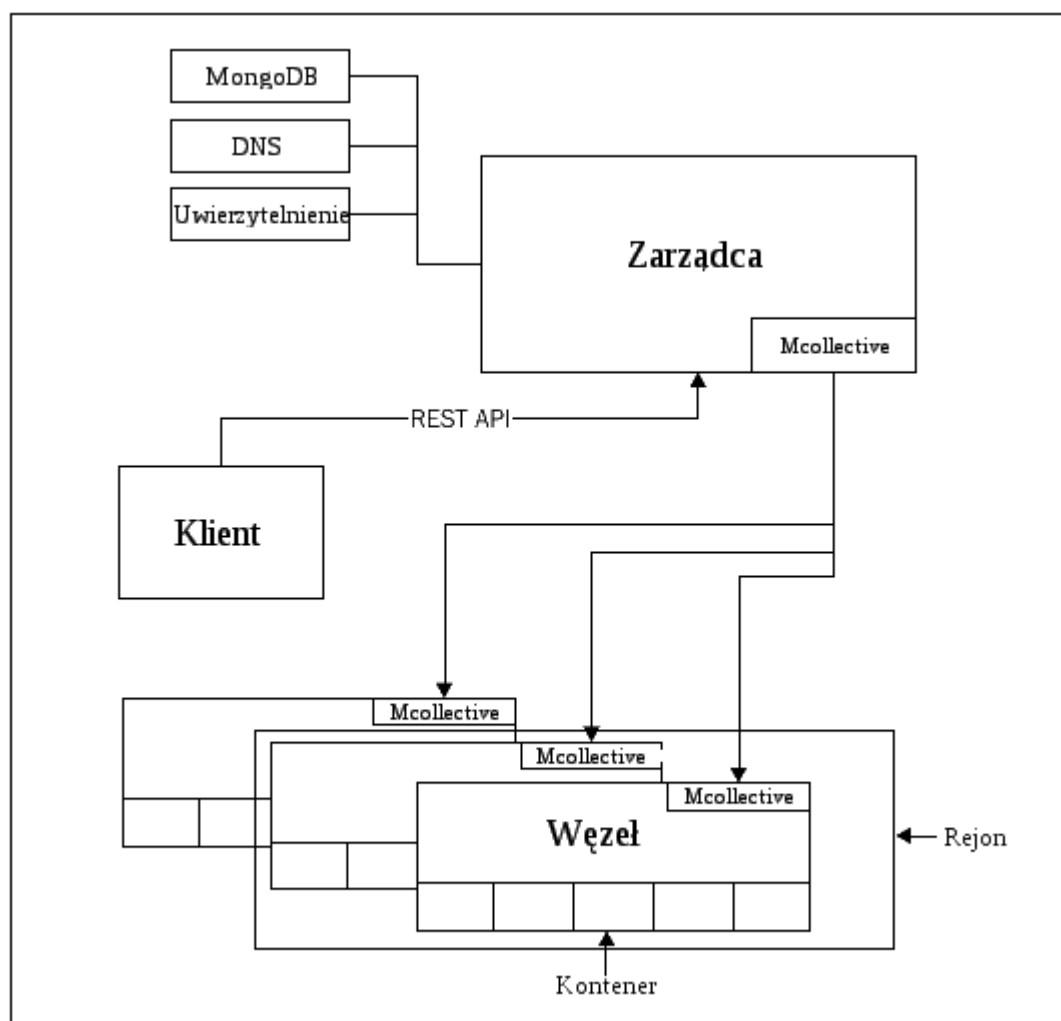
Wszystkie omówione rozwiązania posiadają zalety i wady. Ich wspólną cechą jest możliwość zwiększania możliwości za pomocą różnego rodzaju wtyczek i rozszerzeń. Jest to ich niewątpliwą zaletą, żaden z serwerów nie oferuje wbudowanego wsparcia dla wszystkich języków programowania – co nie jest oczywiście faktem zaskakującym.

Po przeanalizowaniu dostępnych rozwiązań do uruchomienia serwera chmurowego wybrany został serwer OpenShift Origin. Projekt ten powstał na podstawie komercyjnego rozwiązania opracowanego przez firmę Red Hat. W razie trudności podczas procesu wdrażania serwera będzie można szukać informacji nie tylko w dokumentacji projektu OpenShift Origin, ale również w dokumentach udostępnionych przez Red Hat. Wybór technologii OpenShift Origin daje również realną szansę na ewentualną rozbudowę chmury. Użytkownicy, dla których możliwości prywatnej chmury okazały się niewystarczające, będą mogli przejść do modelu hybrydowego. Taka rozbudowa mogłaby okazać się kłopotliwa w przypadku innych rozwiązań chmurowych, ze względu na wspomniane już w tym rozdziale różnice pomiędzy poszczególnymi rozwiązaniami.

4. Teoretyczne podstawy wybranego rozwiązania

4.1. Architektura

OpenShift Origin jest rozwiązaniem służącym do budowy chmury typu Platform-as-a-Service. W jego skład wchodzi kilka elementów, które współpracują ze sobą w celu realizacji zadań związanych z obsługą funkcjonalności oferowanych przez chmurę. Wspomniane elementy oraz ich powiązania przedstawione są na Rysunku 1[16].



Rysunek 1. Przegląd architektury OpenShift Origin

Zrozumienie sposobu działania wdrażanego rozwiązania jest bardzo istotne z punktu widzenia osoby odpowiedzialnej za ten proces. Bez znajomości sposobu funkcjonowania serwera chmurowego bardzo trudno jest nie tylko zdiagnozować

ewentualne problemy z uruchomieniem usługi, ale również dostosować konfigurację do własnych potrzeb (na przykład rozdzielić funkcje pomiędzy różne komputery).

4.1.1. Zarządca

Zarządca (ang. *broker*) jest to program odpowiedzialny za zarządzanie chmurą. Do jego zadań należą:

- uwierzytelnianie użytkowników
- przyjmowanie poleceń od uwierzytelnionych użytkowników
- zarządzanie informacjami dotyczącymi użytkowników oraz utworzonych przez nich aplikacji
- zarządzanie wpisami w systemie DNS

Zarządca stanowi główny punkt, z którego przeprowadzane są prace administracyjne związane z funkcjonowaniem chmury. Za pomocą klienta usługi MCollective[17] zarządca wysyła polecenia do węzłów (ang. *nodes*), które przechowują aplikacje utworzone przez użytkowników chmury.

Komunikacja z zarządcą odbywa się za pośrednictwem REST API (*Representational State Transfer Application Programming Interface*)[16]. Wydanie polecenia odbywa się poprzez wygenerowanie odpowiedniego żądania HTTP. Po jego otrzymaniu, zarządca wykonuje zlecone zadanie, po czym zwraca odpowiedź, przedstawiającą wynik polecenia[16][18].

OpenShift Origin oferuje następujące narzędzia służące do komunikacji z zarządcą:

- konsola dostępna przez przeglądarkę internetową
- narzędzie *rhc*, uruchamiane z linii poleceń
- wtyczka do środowiska Eclipse

Możliwe jest również wykorzystanie dowolnego narzędzia, umożliwiającego generowanie żądań HTTP. Na Rysunku 2 przedstawiony został przykład użycia narzędzia *curl*[19] do nawiązania komunikacji z zarządcą.

umożliwia skonfigurowanie różnych profili kontenerów, co pozwala użytkownikom na wybór wielkości kontenera podczas tworzenia aplikacji[18].

Kontener jest tworzony przez węzeł na polecenie wydawane przez zarządcę. Podczas procesu tworzenia kontenera węzeł przydziela mu odpowiednią ilość zasobów, zdefiniowaną w konfiguracji chmury. Kontener zostaje odpowiednio zabezpieczony – aplikacja umieszczona w kontenerze nie ma dostępu do reszty systemu operacyjnego węzła. Również inne aplikacje, umieszczone w kontenerach umiejscowionych na tym samym węźle, nie mają dostępu do aplikacji. Umożliwia to współdzielenie węzła przez wielu użytkowników chmury[16].



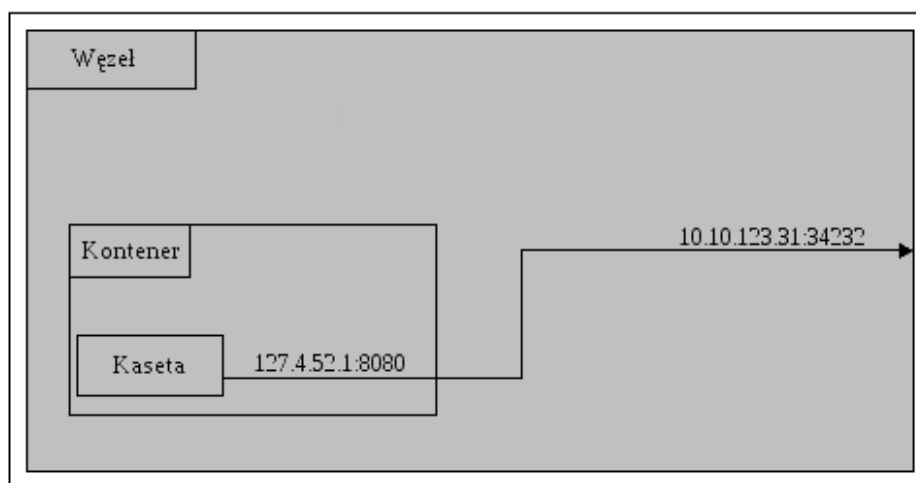
Rysunek 3. Struktura katalogów w kontenerze

Podczas tworzenia kontenera tworzona jest odpowiednia struktura katalogów. Strukturę tę przedstawia Rysunek 3[18], zaś zawartość poszczególnych katalogów znajduje się w Tabeli 3[18].

Tabela 3. Zawartość katalogów kontenera

Numer katalogu z Rysunku 3	Znaczenie katalogu
1	Zmienne środowiskowe aplikacji
2	Miejsce przechowywania danych aplikacji
3	Kod aplikacji
4	Wsparcie dla rozwiązywania zależności
5	Skrypty, katalog należy do użytkownika root
6	Skrypty uruchamiające proces budowania aplikacji

Każdy z kontenerów może wykorzystać adresy z puli 127.0.0.0/8 dla potrzeb umieszczonych w nich aplikacji. W celu zapewnienia światu zewnętrznemu dostępu do aplikacji umieszczonej w chmurze, wewnętrzny adres kontenera jest przypisywany do numeru portu przynależnego do węzła. Schemat znajduje się na Rysunku 4[16].



Rysunek 4. Powiązanie portu kontenera z portem węzła

Użytkownicy chmury, tworzący własne aplikacje, nie muszą się zajmować szczegółami powyższego procesu – jest on przeprowadzany automatycznie przez serwer.

4.1.4. Kaseta

Kasety (ang. *cartridge*) reprezentują komponenty, które mogą zostać wykorzystane do skonstruowania aplikacji. Wykorzystując je, można wybrać język programowania, bazę danych bądź narzędzia do zarządzania. W najprostszym przypadku, aplikacja może wykorzystywać jedynie kasety zapewniającą wsparcie dla wybranego języka programowania bądź środowiska. Większość aplikacji będzie jednak wymagać użycia kasety zapewniającej dostęp do wybranego serwera bazodanowego.

OpenShift Origin zapewnia kasety oparte na najpopularniejszych językach programowania oraz bazach danych. Serwer pozwala również na wykorzystanie kasety utworzonych przez niezależnych programistów – zarówno konsola dostępna przez przeglądarkę, jak i narzędzie związane z linią poleceń pozwalają na wykorzystanie kasety wprost z repozytorium Git.

Kasety są instalowane na węzłach. Obecny sposób funkcjonowania serwera wymaga, aby na wszystkich węzłach był zainstalowany ten sam zestaw kaset[18].

4.1.5. Rejon

Rejony (ang. *districts*) nie mają bezpośredniego wpływu na aplikacje użytkownika. Są natomiast bardzo użytecznym narzędziem dla administratora. Wspomagają one proces przenoszenia kontenerów pomiędzy węzłami. Rejony powinny zostać skonfigurowane, zanim serwer zostanie udostępniony użytkownikom.

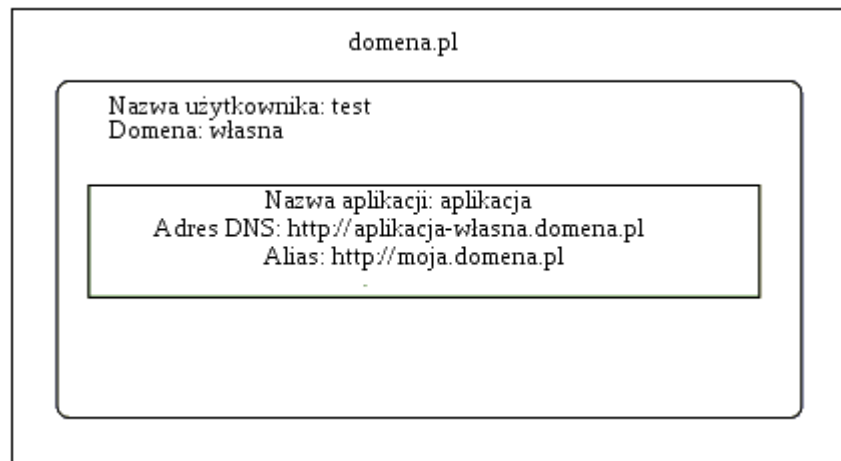
Podczas tworzenia kontenera, węzeł przydziela mu pewien identyfikator – będący jednocześnie identyfikatorem użytkownika w systemie (UID). Identyfikator ten wykorzystywany jest do wyznaczenia zakresów adresów IP oraz portów. Kontener może więc zostać przeniesiony tylko na taki węzeł, który nie przydzielił jeszcze identyfikatora przenoszonoego kontenera innemu, lokalnie utworzonemu kontenerowi.

Rejon rezerwuje przydzielony identyfikator na wszystkich węzłach, które wchodzą w jego skład. Dzięki temu kontener może być swobodnie przenoszony pomiędzy węzłami należącymi do tego samego rejonu, bez konieczności zmiany identyfikatora. Omawiany mechanizm pozwala również oznaczyć węzeł jako nieaktywny – wówczas zarządca nie będzie na nim umieszczał nowych kontenerów. Kontenery już umieszczone na tym węźle będą nadal działać.

Chcąc wykorzystać rejony w swojej instalacji, administrator musi pamiętać o kilku istotnych kwestiach. Po pierwsze, domyślna konfiguracja serwera nie wykorzystuje tego mechanizmu – musi on zostać uaktywniony przez administratora. Należy również zwrócić uwagę na fakt, że do rejonu można dołączyć jedynie te węzły, które nie przechowują żadnych kontenerów. Dlatego zalecane jest aktywowanie opcji, która uniemożliwia tworzenie kontenerów na węzłach, które nie należą do żadnego rejonu. W rejonie nie należy umieszczać zbyt dużej ilości węzłów – powodem jest fakt współdzielenia puli identyfikatorów przez wszystkie węzły w rejonie. W chwili, gdy pula ta zostanie wyczerpana, rejon nie przyjmie już żadnych kontenerów niezależnie od tego, ile wolnych zasobów zostało na węzłach[20].

4.2. Aplikacje

OpenShift Origin umożliwia użytkownikom utworzenie i rozwijanie własnych aplikacji. Aplikacje te umieszczane są w kontenerach, które zapewniają im zasoby niezbędne do prawidłowego funkcjonowania. Podstawowe informacje o aplikacji przedstawione zostały na Rysunku 5[18].

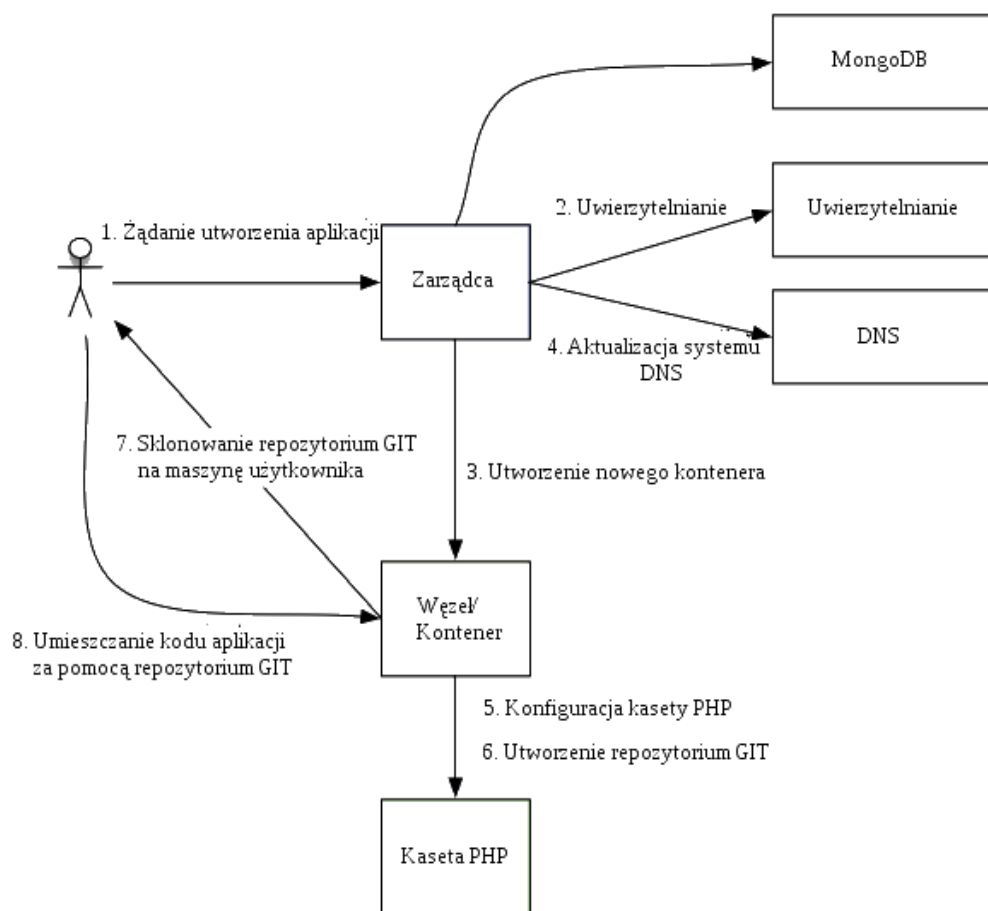


Rysunek 5. Aplikacja w systemie OpenShift Origin

Na powyższym rysunku widać podstawowe informacje na temat aplikacji. Domena zapewnia unikalną przestrzeń nazw dla aplikacji tworzonych przez określonego użytkownika. Wraz z nazwą aplikacji tworzy ona adres aplikacji, tworzony według schematu `http://[nazwa aplikacji]-[domena].[domena serwera chmurowego]` (w powyższym przykładzie domeną serwera jest „domena.pl”). Użytkownik może również zdefiniować własny adres, wykorzystując mechanizm aliasów.

4.2.1. Proces tworzenia aplikacji

Proces tworzenia aplikacji rozpoczyna się na żądanie użytkownika. Użytkownik podaje nazwę tworzonej aplikacji oraz nazwę kasety, która ma zostać zainstalowana w nowo utworzonym kontenerze. Aktualizowany jest również system DNS, w którym umieszczany jest wpis dotyczący nowo utworzonej aplikacji. Rysunek 6[18] przedstawia proces tworzenia prostej aplikacji, korzystającej z kasety zapewniającej wsparcie dla języka PHP.



Rysunek 6. Tworzenie aplikacji

Na lokalną maszynę użytkownika klonowane jest repozytorium, zawierające strukturę katalogów służącą do rozwoju aplikacji. Struktura ta zależna jest od kasety, którą użytkownik wskazał podczas tworzenia aplikacji. Opis przykładowej struktury znajduje się w Tabeli 4[21].

Pewne dane, wykorzystywane przez aplikacje użytkownika, mogą zostać ustalone dopiero na etapie wykonywania. W związku z tym, OpenShift zestaw zmiennych środowiskowych, umożliwiających odwoływanie się do tych informacji w kodzie aplikacji. Opis podstawowych zmiennych środowiskowych, dostępnych niezależnie od wykorzystywanych kaset, znajduje się w Tabeli 5[21].

Tabela 4. Struktura katalogów w repozytorium

Katalog	Zawartość katalogu
.git	Informacje na temat repozytorium GIT, wykorzystywanego do wdrażania kodu
.openshift	Skrypty użytkownika, markery, cykliczne zadania. Podkatalogi: action_hooks – skrypty, wykonywane w różnych fazach wdrażania aplikacji cron – zadania wykonywane cyklicznie markers – ustawienia, na przykład odpowiednia wersja Javy do wykorzystania, bądź wyłączenie automatycznego skalowania
libs	Biblioteki oraz inne zależności, które nie mogą zostać rozwiązane automatycznie.
misc	Kod nieprzeznaczony do publicznego ujawnienia.
php	Kod aplikacji.

Tabela 5. Podstawowe zmienne środowiskowe

Zmienna środowiskowa	Wartość
HOME	Alias dla OPENSIFT_HOMEDIR
HISTFILE	Plik historii dla powłoki bash
OPENSIFT_APP_DNS	Pełna nazwa kwalifikowana aplikacji
OPENSIFT_APP_NAME	Nazwa aplikacji, spełniająca wymagania systemu
OPENSIFT_APP_UUID	UUID aplikacji
OPENSIFT_DATA_DIR	Katalog, gdzie aplikacja może przechowywać dane
OPENSIFT_GEAR_DNS	Pełna nazwa kwalifikowana kontenera.
OPENSIFT_GEAR_NAME	Nazwa kontenera
OPENSIFT_GEAR_UUID	UUID kontenera
OPENSIFT_HOMEDIR	Katalog kontenera
OPENSIFT_REPO_DIR	Katalog zawierający kod aplikacji, uruchamiany przez OpenShift
OPENSIFT_TMP_DIR	Katalog, gdzie aplikacja może przechowywać tymczasowe dane
TMPDIR	Alias dla OPENSIFT_TMP_DIR
TMP	Alias dla OPENSIFT_TMP_DIR

Oprócz powyższych zmiennych, każda kasetka może udostępniać inne, specyficzne dla wspieranego przez nią oprogramowania[21].

OpenShift Origin pozwala użytkownikom na tworzenie dwóch typów aplikacji: skalowalnych i nieskalowalnych. Każda z nich posiada swoje specyficzne cechy oraz zastosowania. Użytkownik dokonuje wyboru w chwili tworzenia aplikacji. OpenShift Origin nie oferuje możliwości zmiany typu aplikacji po jej utworzeniu – z tego powodu wybór typu aplikacji musi zostać dokładnie przemyślany już na etapie projektowania.

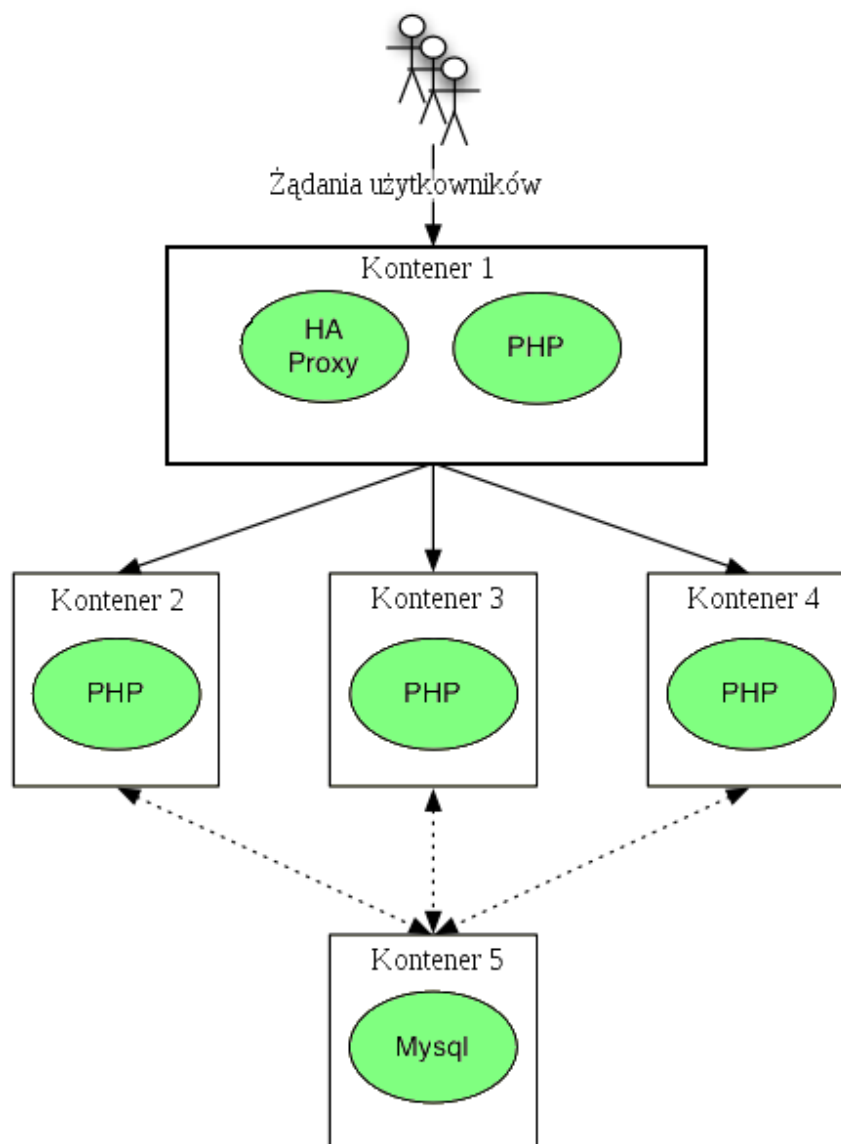
4.2.2. Aplikacje nieskalowalne

Aplikacje nieskalowalne wykorzystują jeden kontener. Umieszczane są w nim wszystkie wykorzystywane przez użytkownika kasety[21]. Oznacza to, że wszystkie technologie wykorzystywane przez aplikację, takie jak baza danych czy serwer aplikacyjny, współdzielą zasoby zapewniane przez kontener. Kontener ten obsługuje wszystkie żądania skierowane do aplikacji użytkownika.

4.2.3. Aplikacje skalowalne

Aplikacje skalowalne pozwalają użytkownikowi na wykorzystanie większej ilości kontenerów. Aplikacja użytkownika może zostać umieszczona w kilku kontenerach, które będą obsługiwać żądania skierowane do aplikacji użytkownika. O tym, który z kontenerów zawierających aplikację obsłuży konkretne żądanie, decyduje load balancer – program mający za zadanie rozdzielanie żądań pomiędzy różne instancje aplikacji. Jest on instalowany w pierwszym utworzonym kontenerze zawierającym aplikację użytkownika – tym, który zostaje stworzony podczas tworzenia aplikacji. Inne kasety, zawierające bazę danych, umieszczane są we własnych, dedykowanych kontenerach[21]. Schemat budowy aplikacji skalowalnej przedstawia Rysunek 7[18].

Zarządzaniem żądaniami oraz ilością kontenerów zawierających aplikację użytkownika zajmuje się load balancer. W chwili, gdy wzrasta liczba żądań kierowanych do aplikacji, powiadamia on serwer o konieczności przydzielenia dodatkowych zasobów. Jeżeli przydział ten jest możliwy, serwer tworzy nowy kontener i kopiuje do niego kod aplikacji użytkownika. Po uruchomieniu nowego kontenera, jest on dołączany do konfiguracji load balancera i zaczyna obsługiwać przychodzące żądania[21]. Jeżeli liczba żądań spadnie, dodatkowy kontener zostanie usunięty, zwalniając zasoby węzła.



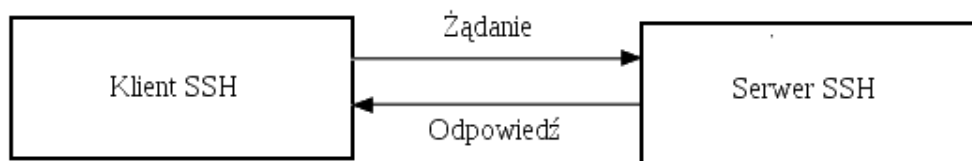
Rysunek 7. Schemat aplikacji skalowalnej

Podczas korzystania z aplikacji skalowalnych należy pamiętać o tym, że kontenery zawierające aplikację nie współdzielą żadnych zasobów. Podczas tworzenia nowego kontenera kopiowana jest wyłącznie zawartość katalogu zawierającego kod aplikacji – każdy nowy kontener, zaczynając pracę, posiada pusty katalog danych. Jeżeli istnieje konieczność współdzielenia danych pomiędzy kontenerami, użytkownik musi samodzielnie zapewnić stosowne rozwiązanie. Może do tego wykorzystać bazę danych – każdy kontener ma do niej dostęp i może wykorzystywać zapisane w niej dane.

4.3. Wykorzystywane technologie

4.3.1. SSH

SSH[22] jest protokołem zapewniającym bezpieczne połączenie pomiędzy dwoma komputerami. Umożliwia bezpieczną autoryzację, zdalne wykonywanie poleceń oraz przesyłanie plików. SSH jest protokołem działającym w oparciu o architekturę klient-serwer. Serwer SSH przyjmuje bądź odrzuca połączenia przychodzące do komputera, na którym serwer został uruchomiony. Zadaniem klienta SSH jest generowanie żądań skierowanych do serwera. Schemat architektury przedstawiony został na Rysunku 8.



Rysunek 8. Architektura SSH

Głównymi cechami zapewnianymi przez protokół SSH są[22]:

- prywatność – SSH zapewnia prywatność poprzez szyfrowanie wysyłanych danych. Do procesu szyfrowania wykorzystywane są klucze uzgadniane przez protokół SSH podczas tworzenia połączenia. Klucze te zostają zniszczone po zakończeniu połączenia. SSH wspiera różne algorytmy szyfrujące, między innymi AES oraz triple-DES.
- integralność – protokół SSH sprawdza, czy dane dotarły przez sieć w stanie nienaruszonym, to znaczy nie zostały po drodze zmodyfikowane. Wykorzystuje do tego sprawdzone algorytmy haszujące: MD5 oraz SHA-1.
- uwierzytelnianie – SSH wspiera kilka metod uwierzytelniania. Najprostszą z nich jest podanie hasła: użytkownik chcący nawiązać połączenie podaje swoje hasło. W porównaniu do usług takich jak telnet, hasło zostaje wysłane w formie zaszyfrowanej. Oprócz tego, SSH oferuje silniejsze metody, takie jak uwierzytelnianie za pomocą kluczy publicznych.
- autoryzacja – SSH pozwala na ograniczenie uprawnień użytkowników

logujących się zdalnie do systemu. Zakres możliwości jest jednak różny, w zależności od wykorzystywanej implementacji.

- tunelowanie – protokół SSH pozwala na opakowanie innych usług wykorzystujących protokół TCP, co umożliwia bezpiecznie korzystanie z tych usług. Jest to szczególnie przydatne w przypadku wykorzystywania usług przesyłających dane w sposób otwarty, takich jak Telnet

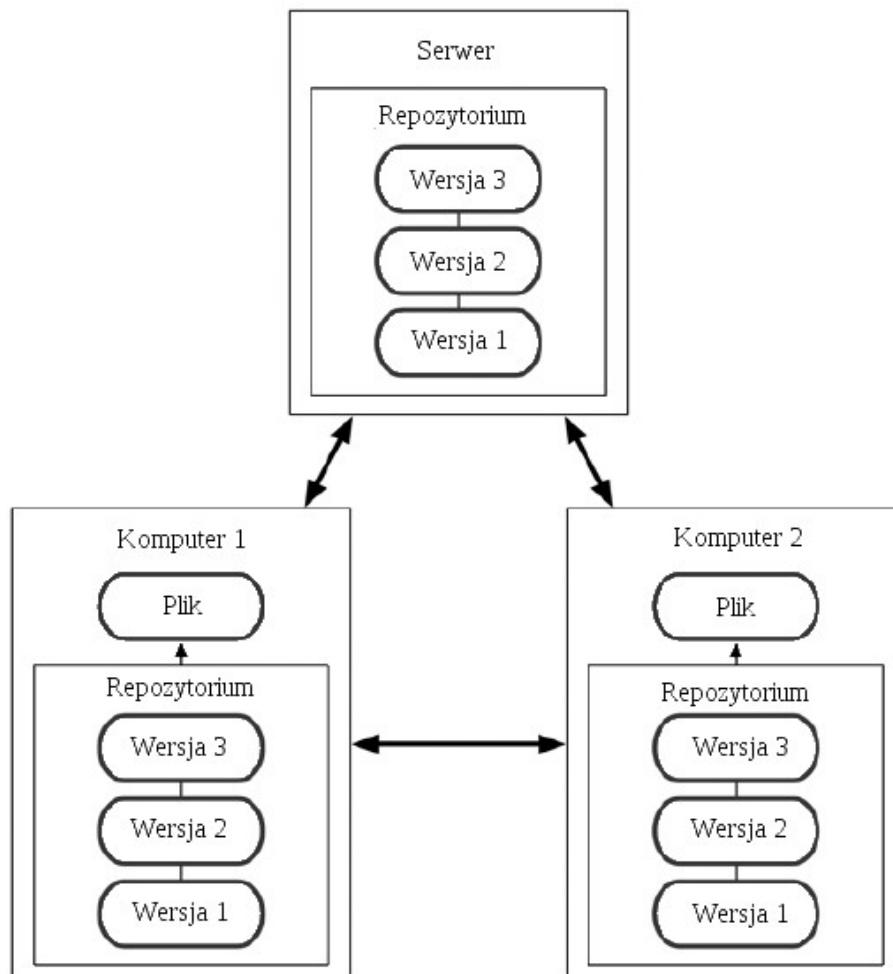
OpenShift Origin wykorzystuje protokół SSH w celu zapewnienia bezpiecznego połączenia pomiędzy użytkownikiem chmury, a jego aplikacją, umieszczoną w chmurze. Umożliwia to użytkownikowi dostęp do powłoki, dzięki któremu może, na przykład, skonfigurować bazę danych, z której korzysta jego aplikacja. Implementacją SSH wykorzystywaną przez OpenShift jest OpenSSH[16].

Do uwierzytelniania użytkownika OpenShift wykorzystuje mechanizm kluczy publicznych. Klucz to zestaw danych binarnych, służący do identyfikacji użytkownika. Omawiany mechanizm wykorzystuje dwa klucze: publiczny i prywatny. Klucz publiczny przechowywany jest na serwerze, klucz prywatny zostaje lokalnie zapisany na komputerze użytkownika. Ta para kluczy wykorzystywana jest podczas nawiązywania połączenia w celu stwierdzenia tożsamości użytkownika[22].

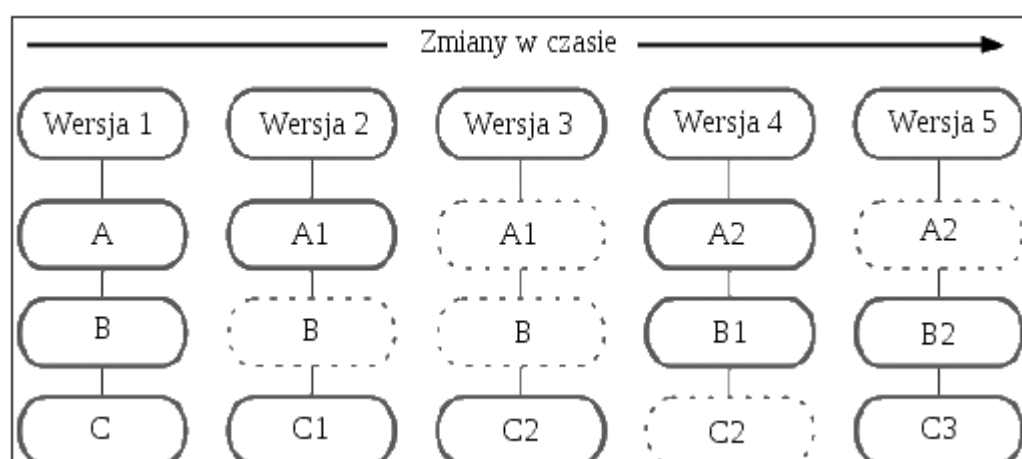
4.3.2. Git

Git[23] jest rozproszonym systemem kontroli wersji (ang. *distributed version control system*). Każdy użytkownik, ściągając dane z repozytorium, pobiera kompletną historię projektu – tworząc na swojej maszynie lokalnej kompletną kopię repozytorium. W przypadku awarii serwera udostępniającego repozytorium w sieci, do odtworzenia danych można wykorzystać dowolną kopię znajdującą się u któregośkolwiek z użytkowników. Schemat takiego systemu pokazuje Rysunek 9[23].

Większość znanych systemów kontroli wersji przechowuje dane w postaci jednego zestawu plików oraz zmian, które zostały wprowadzone do tych plików. Git działa w inny sposób. Podczas zapisywania danych w repozytorium, Git zapisuje całą zawartość plików oraz tworzy odnośnik do tak stworzonej migawki (ang. *snapshot*). Schemat ten ukazuje Rysunek 10[23].



Rysunek 9. Rozproszony system kontroli wersji



Rysunek 10. Zapis wersji pliku w Gicie

W przypadku, gdy plik nie został zmieniony, Git nie zapisuje go ponownie, tylko wykorzystuje odnośnik do ostatnio zapisanej wersji pliku.

Fakt, że Git tworzy na maszynie lokalną kopię repozytorium, ma znaczący wpływ na sposób pracy z tym systemem. Większość operacji może być wykonana lokalnie, bez potrzeby połączenia z serwerem udostępniającym repozytorium. Powoduje to, że operacje wykonywane są bardzo szybko[23]. Podczas pracy z Gitem, dostęp do sieci niezbędny jest tylko w dwóch przypadkach: podczas klonowania repozytorium (ściągnięcia go z serwera) oraz do wysłania lokalnie dokonanych zmian do zdalnego repozytorium.

Git jest jedną z najistotniejszych z punktu widzenia użytkownika technologii wykorzystywanych przez OpenShift Origin. Za jego pomocą użytkownik zarządza kodem swojej aplikacji. Dzięki wykorzystaniu tego systemu kontroli wersji, użytkownik może swobodnie modyfikować kod aplikacji, nawet nie mając dostępu do sieci. Wszystkie zmiany może wprowadzić lokalnie i po upewnieniu się co do ich poprawności, wdrożyć jej w chmurze szybko i bez trudności.

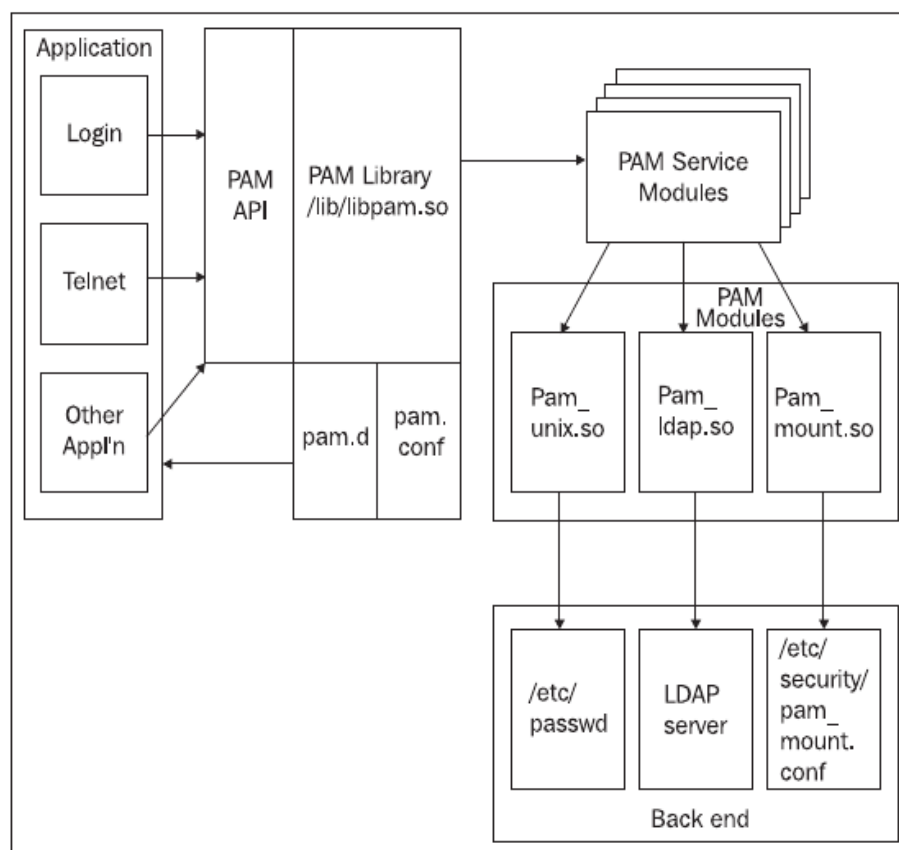
4.3.3. PAM

PAM[24] (Pluggable Authentication Modules) jest rozwiązaniem zapewniającym jednolity i spójny system uwierzytelniania użytkowników dla systemu Linux. Umożliwia ono nie tylko uwierzytelnianie, ale również przygotowanie środowiska pracy dla użytkownika (na przykład zamontowanie odpowiednich systemów plików). Schemat PAM ukazany jest na Rysunku 11[24].

Działanie PAM opiera się na modułach (ang. *modules*), które umożliwiają uwierzytelnianie użytkownika w oparciu o informacje zapisane w różnych miejscach – takich jak pliki systemowe (`/etc/passwd` oraz `/etc/shadow`), serwer LDAP czy relacyjna baza danych. Moduły te mogą być ładowane w trakcie działania systemu – zmiana sposobu uwierzytelniania nie wymaga zresetowania systemu ani ponownej kompilacji programów wykorzystujących PAM do uwierzytelniania[24].

Drugim istotnym elementem PAM są moduły usług (ang. *service modules*). Są to biblioteki, które zapewniają odpowiednie usługi aplikacjom korzystającym z PAM. Aplikacje te porozumiewają się z PAM za pomocą API oferowanego przez omawiane

rozwiązanie.



Rysunek 11. Schemat działania PAM

PAM umożliwia zdefiniowanie bardzo elastycznego systemu uwierzytelniania. Administrator systemu może, na przykład, wykorzystać kilka źródeł danych, nadając poszczególnym źródłom zróżnicowane priorytety[24].

OpenShift Origin wykorzystuje PAM w celu zapewnienia możliwości współdzielenia zasobów. OpenShift wykorzystuje w tym celu własny moduł PAM oraz moduł pam_namespace. Drugi z wymienionych modułów pozwala na wykorzystywanie wielu instancji tego samego katalogu. Katalog taki udostępnia instancję samego siebie na podstawie nazwy użytkownika bądź, w przypadku wykorzystywania technologii SELinux[25], kontekstu bezpieczeństwa (możliwe jest również wykorzystanie obu)[16].

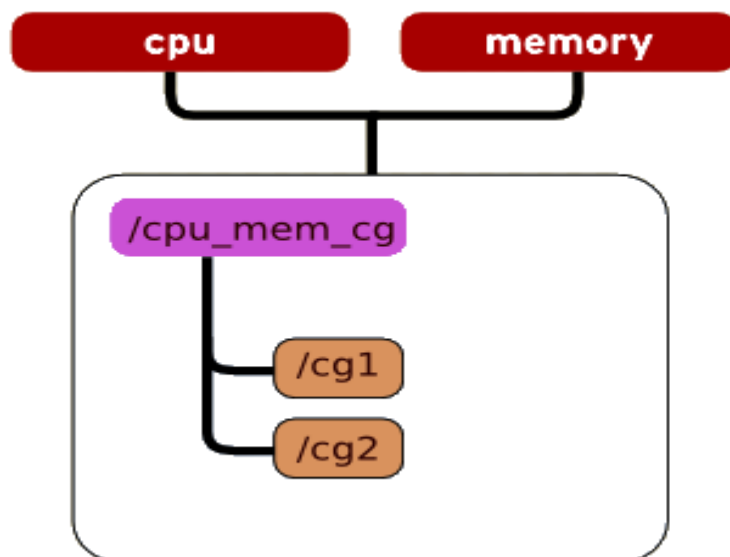
4.3.4. Control Groups

Grupy kontrolne[26] (ang. *control groups*) są jedną z funkcjonalności oferowanych przez jądro systemu Linux. Umożliwiają one ograniczenie zasobów dostępnych dla procesów, określanych w tym przypadku jako zadania (ang. *task*). Są one wykorzystywane przez serwer OpenShift do podziału zasobów węzła pomiędzy aplikacje na nim uruchamiane[16].

Grupy kontrolne, podobnie jak procesy w systemie Linux, zorganizowane są w sposób hierarchiczny: każda grupa ma swojego rodzica (ang. *parent*), po której dziedziczy pewne atrybuty. W systemie może być zdefiniowanych wiele takich hierarchii. Hierarchie te są przyłączane do co najmniej jednego podsystemu (ang. *subsystem*). Podsystemy reprezentują zasoby komputera, takie jak czas procesora czy pamięć operacyjna[26].

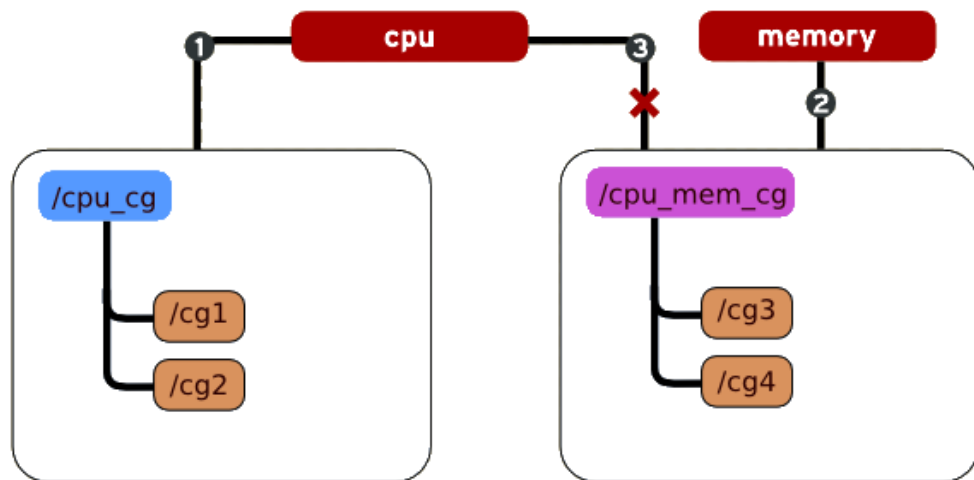
Relacje pomiędzy poszczególnymi elementami (podsystemami, hierarchiami, grupami i zadaniami) określone są przez poniższe reguły[26]:

1. Z pojedynczą hierarchią może być związany więcej niż jeden podsystem (Rysunek 12).



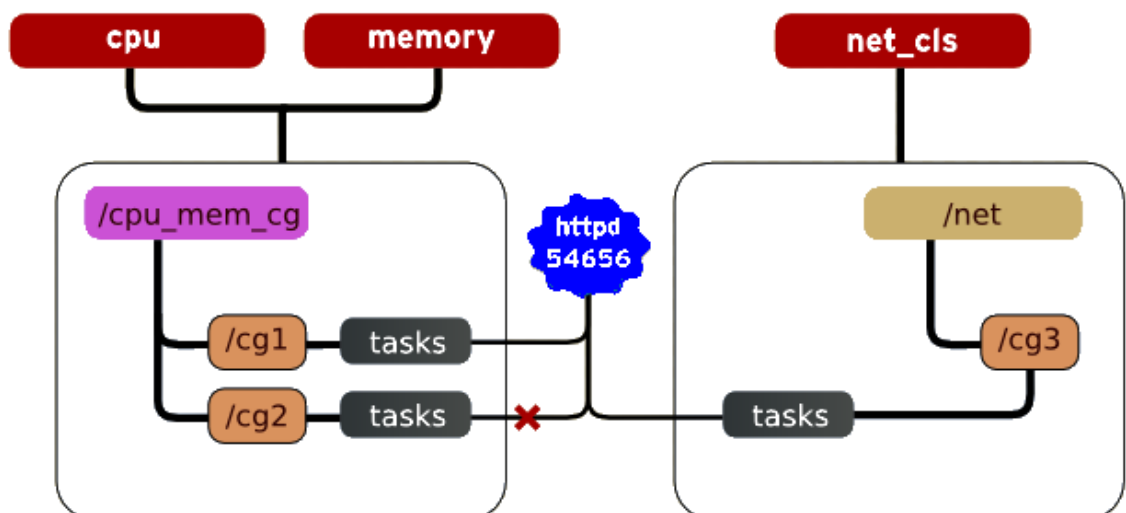
Rysunek 12. Przypisanie podsystemów do hierarchii

- Podsystem nie może być przyłączony do hierarchii, która ma już podłączony inny podsystem (Rysunek 13).



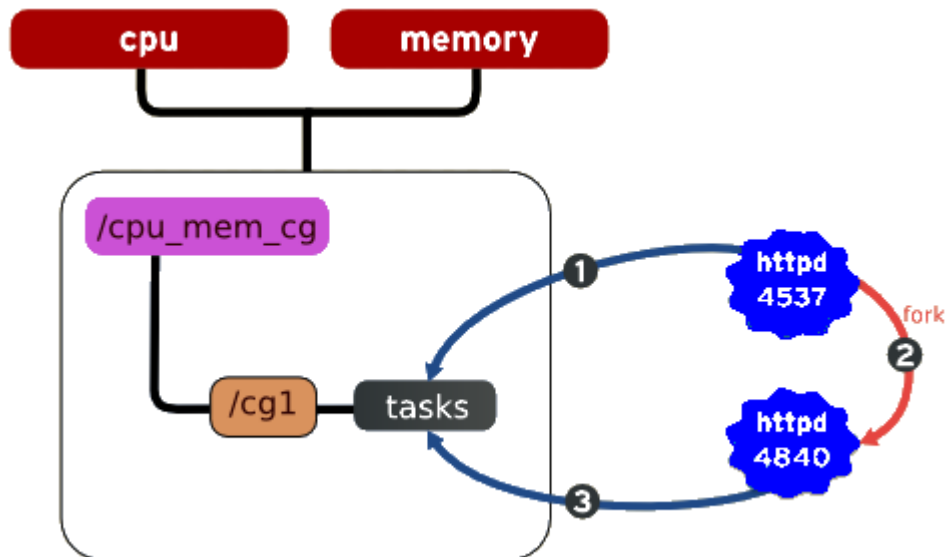
Rysunek 13. Przypisanie podsystemu do hierarchii związanej z innym podsystemem

- Każde zadanie może być umieszczone w dokładnie jednej grupie należącej do pojedynczej hierarchii. Zadanie może należeć do większej ilości grup, jeżeli każda z grup należy do innej hierarchii (Rysunek 14).



Rysunek 14. Przynależność zadania do grup

4. Zadanie będące potomkiem zadania należącego do grupy kontrolnej dziedziczy przynależność do tej grupy. Po utworzeniu zadania potomnego przynależność obu procesów może być zmieniana niezależnie (Rysunek 15).



Rysunek 15. Przynależność zadania potomnego

4.3.5. SELinux

SELinux[25] został opracowany przez Narodową Agencję Bezpieczeństwa USA, w celu wprowadzenia do jądra systemu Linux mechanizmów zapewniających zwiększony poziom bezpieczeństwa. Z pomocą tej technologii serwer OpenShift ogranicza możliwości kontenerów. Dzięki niej kontener nie ma dostępu, na przykład, do danych należących do innego kontenera umieszczonego na tym samym węźle[16].

Podstawowym elementem wykorzystywanym przez SELinuxa jest kontekst (ang. *context*). W systemie wykorzystującym omawianą technologię, każdy obiekt w systemie (na przykład proces, port czy plik) posiada swój kontekst. Kontekst obiektu można prosto sprawdzić za pomocą przełącznika `-Z` (jest on powszechnie uznawany za przełącznik przeznaczony do wypisywania informacji związanych z SELinuxem). Przykładowo, aby sprawdzić kontekst obiektów znajdujących się w katalogu `/home`, wystarczy wydać polecenie `ls -lZ /home`.

Przykładowy kontekst został przedstawiony na Rysunku 16[25].

unconfined_u	unconfined_r	unconfined_t	s0-s0:c0.c1023
użytkownik	rola	typ	poziom dostępności

Rysunek 16. Przykładowy kontekst SELinuxa

Najważniejszym elementem kontekstu jest typ (ang. *SELinux type*). Za jego pomocą, SELinux może kontrolować uprawnienia procesu w zależności od tego, w jaki sposób proces został uruchomiony. Proces uruchomiony przez użytkownika otrzyma inny typ, niż proces uruchomiony przez system. Procesy te będą miały inne uprawnienia, mimo że wykorzystują one te same pakiety binarne. Typy SELinuxa zwyczajowo zakończone są przyrostkiem „_t”, nie jest to jednak obowiązkowe.

Oczywiste jest, że nie wszyscy użytkownicy mogą uruchamiać wszystkie typy procesów. O tym, które typy mogą zostać przypisane do procesów uruchamianych przez użytkownika, decyduje rola użytkownika (ang. *SELinux role*). Najczęściej spotykane role zostały opisane w Tabeli 6[25]. Zwyczajowo role zakończone są przyrostkiem „_r”.

Tabela 6. Najczęściej spotykane role SELinuxa

Rola SELinuxa	Opis roli
user_r	Pozwala na uruchamianie procesów z typami przeznaczonymi dla aplikacji użytkowych. Przeznaczona dla użytkowników systemu.
staff_r	Dopuszcza te same typy, co user_r, oraz niewielką ilość bardziej uprzywilejowanych typów. Przeznaczona dla operatorów systemu.
sysadm_r	Przeznaczona do zadań administracyjnych. Pozwala na wykorzystywanie szerokiego zakresu typów. Nie dopuszcza wykorzystania pewnych typów związanych z aplikacjami użytkowymi, w celu zapewnienia bezpieczeństwa systemu.
system_r	Przeznaczona dla demonów (procesów uruchamianych w tle). Umożliwia dostęp do wielu uprzywilejowanych typów, nie dopuszcza jednak do wykorzystania typów związanych z aplikacjami użytkowymi oraz zadaniami administracyjnymi.
unconfined_r	Przeznaczona dla użytkowników. Wykorzystywana, gdy administrator systemu chce chronić za pomocą SELinuxa wybrane procesy, a resztę systemu pozostawić bez większych zmian.

Kolejnym elementem kontekstu jest informacja o użytkowniku (ang. *SELinux user*). Użytkownik SELinuxa nie jest tożsamy z użytkownikiem linuksowym. Użytkownik linuksowy może po zalogowaniu zmienić swoją tożsamość. SELinux zapewnia, że informacja o użytkowniku SELinuxa nie zmieni się, nawet jeżeli nastąpi zmiana informacji o użytkowniku linuksowym, na przykład za pomocą narzędzia `su`. Definicja użytkownika SELinuxa ogranicza role, w które może się wcielić użytkownik linuksowy.

Ostatnią informacją zawartą w kontekście jest poziom dostępności (ang. *sensitivity level*). Pozwala on na klasyfikację zasobów oraz ograniczenie dostępu do nich za pomocą odpowiednich poświadczeń. Poziom dostępności definiowany jest poprzez dwa elementy: poufność (ang. *confidentiality value*) oraz kategorię (ang. *category value*). Poufność (oznaczona przedrostkiem „s”) oceniana jest od 0 do wartości zdefiniowanej przez administratora systemu. Gdy kontrola poufności jest aktywna, proces nie może czytać danych z zasobów o wyższej wartości poufności niż ta, którą posiada proces. Nie może również komunikować się z zasobami, które posiadają niższy poziom poufności. Kategorie (oznaczone przedrostkiem „c”) pozwalają na podział zasobów na pewne grupy, dla których również jest możliwa kontrola dostępu. Proces ma dostęp tylko do tych zasobów, których wartość kategorii jest taka sama, jak wartość kategorii przypisana do procesu[25].

Kolejnym istotnym elementem SELinuxa jest polityka (ang. *policy*). Definiuje ona, do których zasobów mają dostęp poszczególne procesy. Kontrola dostępu odbywa się poprzez analizę kontekstów procesu oraz zasobu, do którego proces chce uzyskać dostęp. Za ową analizę odpowiadają reguły dostępu (ang. *allow rules*). Reguła dostępu skonstruowana jest według poniższej składni[25]:

```
allow <source> <destination> : <class> <permissions>;
```

gdzie:

- <source> - typ, jaki jest przypisany do procesu chcącego uzyskać dostęp do zasobu
- <destination> - typ, który jest przypisany do zasobu
- <class> - pozwala na kontrolę uprawnień w zależności od typu zasobu

(zasobem może być plik, katalog, gniazdo TCP i tak dalej)

- <permissions> - uprawnienia, jakie zostają przydzielone procesowi

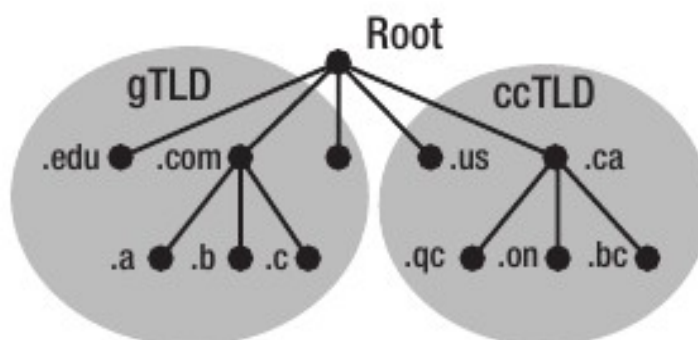
4.3.6. BIND

BIND[27] (Berkeley Internet Name Domain) jest otwartą implementacją protokołu DNS (Domain Name System). Ze względu na wysoką jakość kodu oraz stabilność działania, jest najczęściej wykorzystywanym oprogramowaniem tego rodzaju na świecie.

DNS pozwala na translację adresów domenowych (łatwych do zapamiętania dla człowieka) na adresy IP, wykorzystywane w sieciach komputerowych. System nazw domenowych jest systemem hierarchicznym. Na szczycie hierarchii znajduje się główny węzeł (ang. *root node*). Stopień niżej znajdują się domeny najwyższego poziomu (ang. *Top-Level Domains, TLDs*). Domeny te zostały podzielone na dwa typy:

- domeny generyczne (ang. *Generic Top-Level Domains, gTLDs*) – takie jak .com, .edu, .net
- domeny związane z krajami (ang. *Country Code Top-Level Domains, ccTLDs*) – takie jak .pl, .tv, .uk

Dalej w hierarchii znajdują się kolejne poziomy domeny. W adresie domenowym każda z domen oddzielona jest kropką. Hierarchia nazw domenowych przedstawiona jest na Rysunku 17[27].



Rysunek 17. Hierarchia domen w systemie DNS

Każdy węzeł w hierarchii przypisany jest do instytucji, która nim zarządza (może to być organizacja, firma, lub nawet osoba prywatna). Instytucja ta może oddelegować zarządzanie domenami niższego poziomu innym instytucjom[27]. Pozwala to na odciążenie serwerów i umożliwia łatwą rozbudowę hierarchii o domeny niższego poziomu (w przypadku dodania kolejnego poziomu hierarchii, nie ma potrzeby aktualizowania danych we wszystkich serwerach w hierarchii).

OpenShift wykorzystuje BIND do zarządzania nazwami domenowymi aplikacji tworzonych przez użytkowników. Pozwala to na wygodnie korzystanie z aplikacji. Umożliwia również dostęp do aplikacji za pośrednictwem Internetu.

4.3.7. MongoDB

MongoDB[28] jest skalowalnym systemem zarządzania bazą danych o ogólnym przeznaczeniu. W celu ułatwienia procesu skalowania, projektanci zdecydowali się na odejście od modelu relacyjnego. Podstawową jednostką danych stosowaną w omawianym systemie jest dokument. Dokument jest uporządkowanym zbiorem par – para składa się z klucza oraz przypisanej mu wartości. Kluczami w dokumentach są łańcuchy znaków. W kluczu dopuszczalne są wszystkie znaki UTF-8, z wyjątkiem znaku NULL (oznaczanego jako \0), „.” oraz „?”. Wartości przypisywane do kluczy mogą być różnych typów. Najczęściej spotykanymi typami są[28]:

- null (typ pusty)
- boolean
- number (liczby zmiennoprzecinkowe, zapisywane za pomocą 64 bitów)
- string (kodowanie UTF-8)
- date (data przechowywana jako milisekundy od początku epoki)
- regular expression (wyrażenia regularne)
- embedded document (wartościami mogą być również dokumenty)

Kolejnym z podstawowych elementów wykorzystywanych w MongoDB jest kolekcja. Kolekcją jest zbiór dokumentów. W omawianym systemie, dokumenty

umieszczane w kolekcji mogą mieć dowolną budowę. Nie istnieje więc potrzeba wykorzystywania większej ilości kolekcji – wszystkie dokumenty mogą być przechowywane w jednej kolekcji. Nie jest to jednak zalecane ze względów wydajnościowych. Kolekcja jest identyfikowana poprzez jej nazwę. Nazwą kolekcji może być dowolny ciąg znaków, z kilkoma wyjątkami[28]:

- nazwą nie może być pusty łańcuch znaków
- nazwa nie może zawierać znaku NULL („\0”) ani znaku „\$”
- nazwa kolekcji nie powinna zaczynać się od ciągu „system.” (przedrostek ten zarezerwowany jest dla wewnętrznych kolekcji MongoDB)

Kolekcje są przechowywane w bazie danych. Każda baza ma własne uprawnienia oraz jest zapisywana w oddzielnych plikach. Podobnie jak kolekcje, bazy danych są identyfikowane na podstawie nazwy. Nazwa bazy danych musi spełniać następujące warunki[28]:

- nazwą nie może być pusty łańcuch znaków
- nazwa nie może zawierać znaków specjalnych, pojedynczej spacji ani znaku NULL
- długość nazwy ograniczona jest do 64 bajtów

OpenShift wykorzystuje MongoDB do przechowywania wewnętrznych informacji. Są tam przechowywane informacje o węzłach i rejonach obecnych w systemie. Znajdują się tam również informacje o użytkownikach chmury, ich domenach oraz utworzonych aplikacjach.

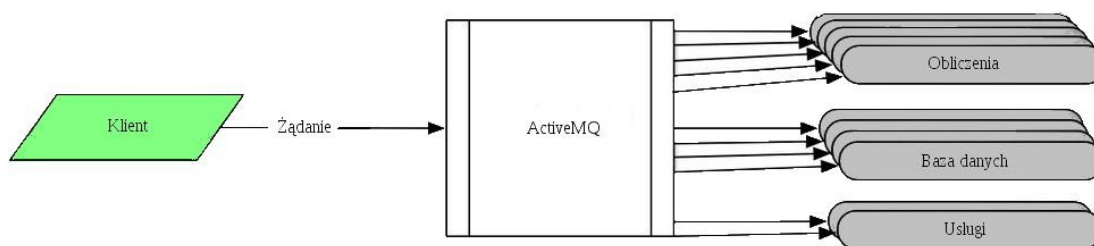
4.3.8. MCollective

MCollective[17] jest technologią zapewniającą możliwość zdalnego zarządzania serwerami oraz równoległego wykonywania zadań. Jest zazwyczaj wykorzystywana do przeprowadzania synchronicznych zmian na pewnej ilości komputerów w czasie porównywalnym do czasu rzeczywistego.

MCollective został w celu zapewnienia możliwości równoległego wykonywania zadań w sposób zapewniający powtarzalne rezultaty. Nie korzysta on z modelu

zawierającego system określany jako master. Komunikaty nie są wysyłane do serwerów za pomocą uporządkowanej pętli. Powyższe założenia sprawiają, że MCollective unika problemów znanych z rozwiązań wykorzystujących scentralizowany model działania[17].

MCollective wykorzystuje dodatkowe oprogramowanie do przekazywania żądań do zarządzanych serwerów – obecnie najczęściej wykorzystywanym oprogramowaniem jest ActiveMQ. Oprogramowanie to jest instalowane na komputerze, z którego administrator systemu wydaje polecenia. Komputer ten nazywany jest klientem. Komputery, które odbierają żądania od klienta, nazywane są serwerami. Kiedy klient wysyła żądanie do serwerów, każdy serwer przetwarza żądanie, niezależnie od innych serwerów, które odebrały to żądanie. Schemat ten ukazany jest na Rysunku 18[17].



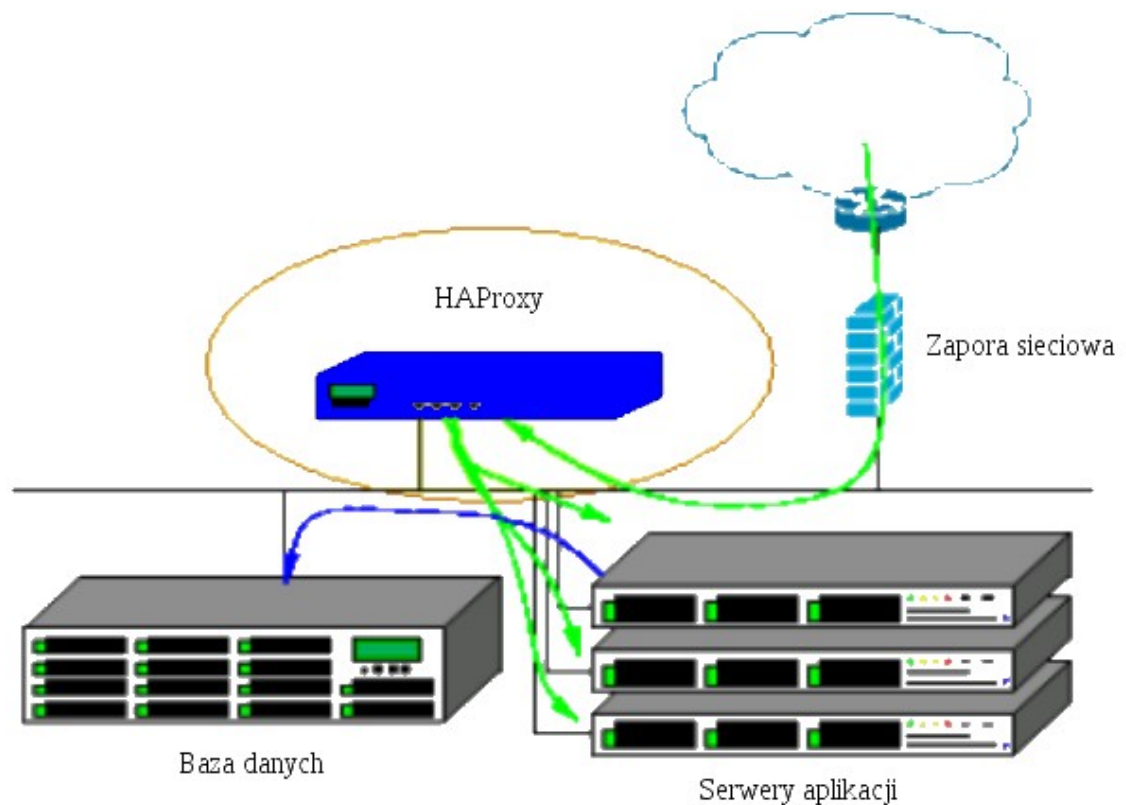
Rysunek 18. Model komunikacji wykorzystywany przez MCollective

Na serwerach instalowani są tak zwani agenci – są to odpowiednie programy, które odpowiadają na żądania przesyłane przez klientów oraz wykonuje zlecane przez nie zadania. Agenci zwracają informacje o tym, czy zadanie zostało poprawnie wykonane czy też nie, wraz z informacjami o wykonywanym procesie. Agenci zajmują się również tłumaczeniem żądań przysyłanych od klienta – dzięki temu administrator systemu może wydać jedno polecenie, niezależnie od tego, pod kontrolą których systemów operacyjnych działają poszczególne serwery[17].

OpenShift wykorzystuje MCollective do zorganizowania komunikacji pomiędzy zarządcą a węzłami. Zarządcą, pełniąc rolę klienta, wysyła do węzłów żądania zlecające wykonanie zadań związanych z działaniem chmury, takich jak utworzenie czy zniszczenie kontenera.

4.3.9. HAProxy

HAProxy[29] jest szybkim i sprawdzonym rozwiązaniem, wykorzystywanym najczęściej w celu rozdzielenia żądań pomiędzy kilka serwerów (ang. *load balancing*). HAProxy rozdziela ruch pomiędzy zdefiniowane przez administratora serwery. Oferuje szeroki wybór algorytmów sterujących procesem podziału żądań[29]. Umieszczenie tej technologii w architekturze przedstawia Rysunek 19.



Rysunek 19. Położenie HAProxy w architekturze

HAProxy jest wykorzystywany przez aplikacje skalowalne w chmurze OpenShift Origin. Jest instalowany w pierwszym kontenerze tworzonym dla aplikacji skalowalnej. Odpowiada za podział żądań oraz za generowanie żądań do zarządcy, dotyczących konieczności przydzielenia bądź zwolnienia dodatkowych kontenerów.

4.3.10. Lokkit

Lokkit[30] jest programem służącym do konfigurowania zapory sieciowej. Pozwala on na szybkie skonfigurowanie zapory sieciowej opartej na programie iptables, poprzez automatyczne wygenerowanie odpowiednich reguł. Lokkit umożliwia

skonfigurowanie zapory sieciowej bez konieczności ręcznej konfiguracji iptables, co zmniejsza prawdopodobieństwo popełnienia błędu.

5. Uruchomienie serwera chmurowego

5.1. Konfiguracja testowa

Do stworzenia konfiguracji testowej został wykorzystany laptop Dell Inspiron N5010 z procesorem Intel Core i3 oraz pamięcią operacyjną 4 GB, pracujący pod kontrolą systemu Fedora 19 w wersji 64-bitowej.

Testowa chmura składa się z dwóch maszyn wirtualnych, utworzonych i zarządzanych za pomocą programu Oracle VM VirtualBox. Opis konfiguracji maszyn przedstawiają Tabele 7 i 8. Do testów została wybrana trzecia wersja oprogramowania OpenShift Origin.

Domeną chmury jest domena.pl. Nie jest ona oddelegowana przez żadną instytucję, wobec czego będzie ona dostępna jedynie w sieci lokalnej. Chmura została uruchomiona w sieci lokalnej wykorzystującej adresy nierutowalne klasy C.

Tabela 7. Opis konfiguracji pierwszej maszyny wirtualnej

Nazwa komputera (ang. <i>hostname</i>)	broker
Rola w chmurze	Zarządca, węzeł
System operacyjny	Fedora 19 wersja 64-bitowa
Ilość przydzielonych procesorów	1
Ilość przydzielonej pamięci operacyjnej	1100 MB
Adapter sieciowy	Połączenie mostkowe (ang. <i>bridged adapter</i>)
Adres IP maszyny	192.168.1.125/24 (adres statyczny)

Tabela 8. Opis konfiguracji drugiej maszyny wirtualnej

Nazwa komputera (ang. <i>hostname</i>)	node
Rola w chmurze	Węzeł
System operacyjny	Fedora 19 wersja 64-bitowa
Ilość przydzielonych procesorów	1
Ilość przydzielonej pamięci operacyjnej	1024 MB
Adapter sieciowy	Połączenie mostkowe (ang. <i>bridged adapter</i>)
Adres IP maszyny	192.168.1.126/24 (adres statyczny)

5.2. Instalacja oprogramowania

Do instalacji chmury można wykorzystać wiele procedur oraz narzędzi. Chmura testowa została skonfigurowana ręcznie, bez użycia narzędzi automatyzujących proces. Pozwoliło to na lepsze zrozumienie konfiguracji oraz dostosowanie jej do indywidualnych potrzeb.

Opisana niżej procedura instalacji opiera się na informacjach zaczerpniętych z [31]. Podczas prezentowania poleceń przeznaczonych do wpisania w konsoli, znak zgłoszenia będzie reprezentowany przez symbol [#] .

5.2.1. Przygotowanie systemów

Przed rozpoczęciem instalacji należy wykonać niżej opisane czynności. Dotyczą one obu systemów, wchodzących w skład chmury.

Pierwszym krokiem jest zainstalowanie odpowiednich repozytoriów, w których znajdują się pakiety zawierające instalowane oprogramowanie. W celu ich zainstalowania, należy wydać następujące polecenia:

```
[#] cat <<EOF> /etc/yum.repos.d/openshift-origin-deps.repo
[openshift-origin-deps]
name=openshift-origin-deps
baseurl=http://mirror.openshift.com/pub/origin-server/release/3/fedora-19/dependencies/x86_64/
gpgcheck=0
enabled=1
EOF
[#] cat <<EOF> /etc/yum.repos.d/openshift-origin.repo
[openshift-origin]
name=openshift-origin
baseurl=http://mirror.openshift.com/pub/origin-server/release/3/fedora-19/packages/x86_64/
gpgcheck=0
enabled=1
EOF
```

Następnie należy zaktualizować system – istotne jest, żeby w systemie była obecna najnowsza wersja SELinuxa. W tym celu wydajemy polecenia:

```
[#] yum clean all
[#] yum -y update
```

Należy również zapewnić synchronizację zegarów we wszystkich systemach wchodzących w skład chmury. Każde żądanie wysyłane przez MCollective zawiera

znacznik czasowy. Jeżeli wskazania zegarów będą różnić się za bardzo, MCollective będzie porzucał zapytania, zamiast je przetwarzać.

```
[#] yum install -y ntpdate ntp
[#] ntpdate clock.redhat.com
[#] systemctl enable ntpd.service
[#] systemctl start ntpd.service
```

Ostatnim krokiem jest zainstalowanie odpowiedniego oprogramowania do zarządzania regułami zapory sieciowej:

```
yum erase -y firewalld
yum install -y lokkit
```

5.2.2. Zarządca oraz węzeł pierwszy

5.2.2.1. DNS

Pierwszym krokiem jest skonfigurowanie systemu DNS, służącego do zarządzania nazwami aplikacji tworzonych w chmurze. Należy stworzyć zmienne środowiskowe zawierające potrzebne informacje oraz nazwę domeny oraz zainstalować serwer BIND:

```
[#] domain=domena.pl
[#] keyfile=/var/named/${domain}.key
[#] yum install -y bind bind-utils
```

Należy wygenerować parę kluczy wykorzystywanych przez mechanizm DNSSEC oraz klucz rndc wykorzystywany do sprawdzania statusu usługi

```
[#] pushd /var/named
[#] rm K${domain}*
[#] dnssec-keygen -a HMAC-MD5 -b 512 -n USER -r /dev/urandom ${domain}
[#] KEY="$(grep Key: K${domain}*.private | cut -d ' ' -f 2)"
[#] popd
[#] rndc-confgen -a -r /dev/urandom
```

Należy skonfigurować prawa dostępu oraz kontekst SELinuxa dla nowo utworzonych kluczy:

```
[#] restorecon -v /etc/rndc.* /etc/named.*
[#] chown -v root:named /etc/rndc.key
[#] chmod -v 640 /etc/rndc.key
```

Kolejnym krokiem jest wskazanie serwerów DNS, które konfigurowany serwer będzie odpytywał w przypadku, w którym nie będzie mógł sam udzielić odpowiedzi. Serwery te znajdują się w pliku /var/named/forwarders.conf. Należy również zadbać o

konfigurację SELinuxa oraz prawa dostępu do pliku:

```
[#] echo "forwarders { 8.8.8.8; 8.8.4.4; } ;" >> /var/named/forwarders.conf
[#] restorecon -v /var/named/forwarders.conf
[#] chmod -v 640 /var/named/forwarders.conf
```

Należy również utworzyć plik strefowy, będący źródłem danych dla serwera DNS oraz zainstalować klucz DNSSEC dla tworzonej domeny.

```
[#] cat <<EOF > /var/named/dynamic/${domain}.db
\$ORIGIN .
\$TTL 1 ; 1 seconds (for testing only)
${domain} IN SOA ns1.${domain}. hostmaster.${domain}. (
    2011112904 ; serial
    60      ; refresh (1 minute)
    15      ; retry (15 seconds)
    1800    ; expire (30 minutes)
    10      ; minimum (10 seconds)
)
NS ns1.${domain}.
MX 10 mail.${domain}.
\$ORIGIN ${domain}.
ns1 A 127.0.0.1
EOF
[#] cat <<EOF > /var/named/${domain}.key
key ${domain} {
    algorithm HMAC-MD5;
    secret "${KEY}";
};
EOF
```

Istotne jest oczywiście zadbanie o odpowiednią konfigurację uprawnień oraz kontekstu:

```
[#] chown -Rv named:named /var/named
[#] restorecon -rv /var/named
```

Ostatnim plikiem do stworzenia jest plik konfiguracyjny serwera BIND, /etc/named.conf, oraz skonfigurować uprawnienia:

```
cat <<EOF > /etc/named.conf
// named.conf
//
// Provided by Red Hat bind package to configure the ISC BIND named(8) DNS
// server as a caching only nameserver (as a localhost DNS resolver only).
//
// See /usr/share/doc/bind*/sample/ for example named configuration files.
//

options {
    listen-on port 53 { any; };
    directory "/var/named";
    dump-file "/var/named/data/cache_dump.db";
    statistics-file "/var/named/data/named_stats.txt";
    memstatistics-file "/var/named/data/named_mem_stats.txt";
    allow-query { any; };
}
```



```

recursion yes;

/* Path to ISC DLV key */
bindkeys-file "/etc/named.iscdlv.key";

// set forwarding to the next nearest server (from DHCP response
forward only;
include "forwarders.conf";
};

logging {
    channel default_debug {
        file "data/named.run";
        severity dynamic;
    };
};

// use the default rndc key
include "/etc/rndc.key";

controls {
    inet 127.0.0.1 port 953
    allow { 127.0.0.1; } keys { "rndc-key"; };
};

include "/etc/named.rfc1912.zones";

include "${domain}.key";

zone "${domain}" IN {
    type master;
    file "dynamic/${domain}.db";
    allow-update { key ${domain}; };
};
EOF
[#] chown -v root:named /etc/named.conf
[#] restorecon /etc/named.conf

```

Po wykonaniu wyżej opisanych czynności należy uruchomić serwer BIND. Jeżeli uruchomienie usługi nie powiedzie się, należy wrócić do powyższych kroków i zweryfikować ich poprawność:

```
[#] systemctl start named.service
```

Jeżeli usługa uruchamia się poprawnie, należy skonfigurować system tak, aby automatycznie uruchamiał serwer DNS. Trzeba również zapewnić odpowiednią konfigurację zapory sieciowej:

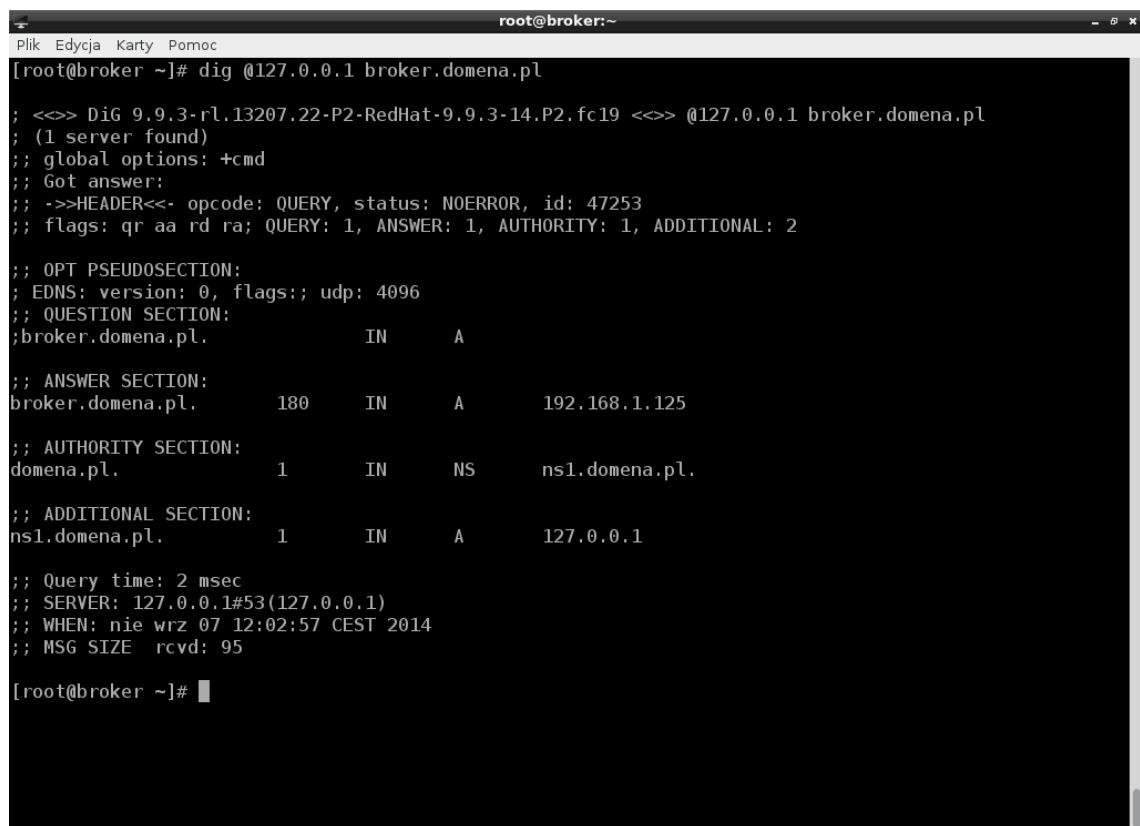
```
[#] systemctl enable named.service
[#] lokkit --service=dns
```

Następnie należy dodać lokalny serwer DNS do pliku /etc/resolv.conf. Należy pamiętać

o tym, żeby wpis był pierwszym w pliku. Ostatnim krokiem jest dodanie wpisu do bazy serwera BIND, pozwalającego poprawnie rozwiązać nazwę lokalnej maszyny:

```
[#] nsupdate -k ${keyfile}
> server 127.0.0.1
> update add broker.domena.pl 180 A 192.168.1.125
> send
```

W tym momencie serwer DNS poprawnie rozwiązywać nazwę broker.domena.pl. Należy to zweryfikować, na przykład za pomocą polecenia dig. Poprawny rezultat wykonania polecenia pokazany jest na Rysunku 20.



```
root@broker:~
Plik Edycja Karty Pomoc
[root@broker ~]# dig @127.0.0.1 broker.domena.pl

; <<>> DiG 9.9.3-rl.13207.22-P2-RedHat-9.9.3-14.P2.fc19 <<>> @127.0.0.1 broker.domena.pl
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->HEADER<<- opcode: QUERY, status: NOERROR, id: 47253
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;broker.domena.pl.                IN      A

;; ANSWER SECTION:
broker.domena.pl.                180     IN      A      192.168.1.125

;; AUTHORITY SECTION:
domena.pl.                       1       IN      NS      ns1.domena.pl.

;; ADDITIONAL SECTION:
ns1.domena.pl.                   1       IN      A      127.0.0.1

;; Query time: 2 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: nie wrz 07 12:02:57 CEST 2014
;; MSG SIZE rcvd: 95

[root@broker ~]#
```

Rysunek 20. Weryfikacja konfiguracji serwera DNS

Po zweryfikowaniu poprawności działania serwera DNS, należy zmienić nazwę lokalnego komputera poprzez edycję pliku /etc/hostname:

```
[#] echo "broker.domena.pl" > /etc/hostname
```

5.2.2.2. MongoDB

W celu zainstalowania pakietów zawierających MongoDB należy wydać następujące polecenie:

```
[#] yum install -y mongodb-server mongodb libmongodb
```

Następnie należy zmodyfikować plik `/etc/mongodb.conf`. W tym pliku umieścić należy wpisy `smallfiles=true` oraz `auth=true`:

```
##
### Basic Defaults
##
bind_ip = 127.0.0.1
port = 27017
fork = true
pidfilepath = /var/run/mongodb/mongodb.pid
logpath = /var/log/mongodb/mongodb.log
dbpath = /var/lib/mongodb
journal = true

# Enables periodic logging of CPU utilization and I/O wait
#cpu = true

# Turn on/off security. Off is currently the default
#noauth = true
auth = true

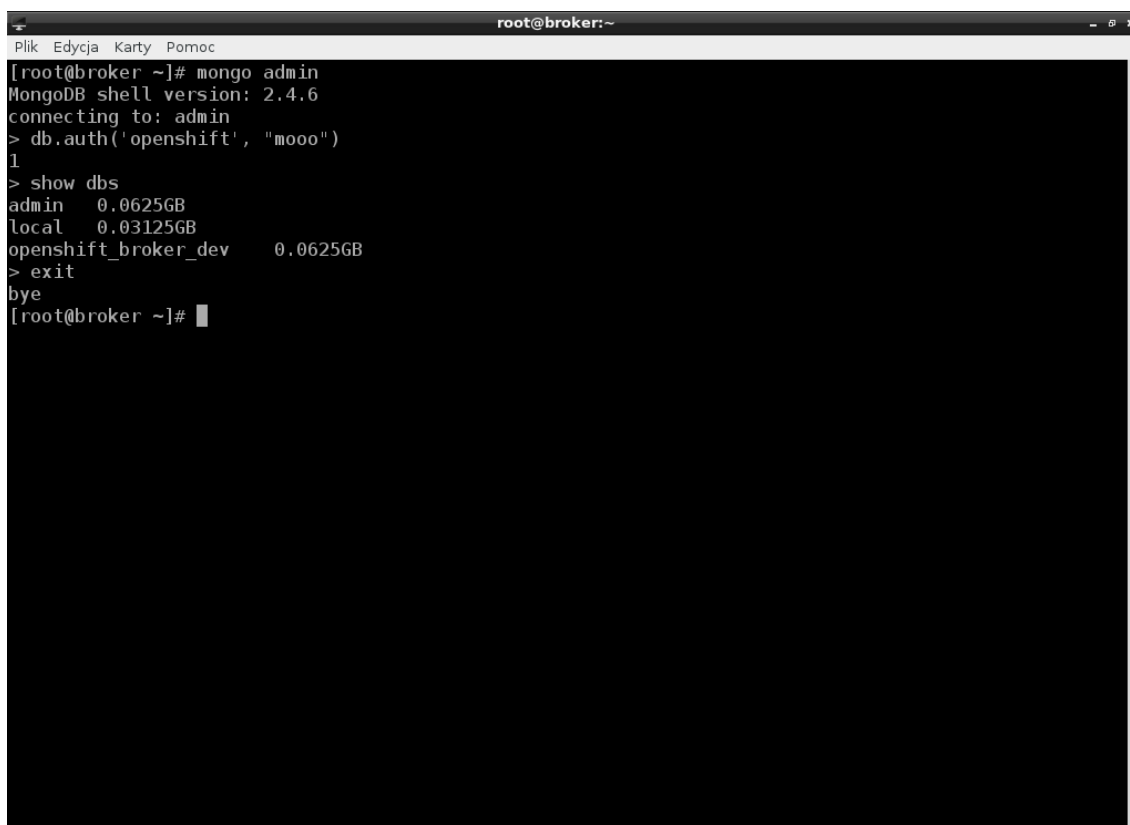
smallfiles = true
# Verbose logging output.
#verbose = true
<reszta pliku została pominięta>
```

Konieczne jest utworzenie użytkownika, za pomocą którego serwer będzie mógł posługiwać się bazą danych. Należy również umożliwić MongoDB korzystanie z odpowiedniego portu. W tym celu trzeba wydać następujące polecenia:

```
[#] systemctl start mongod.service
[#] /usr/bin/mongo localhost/openshift_broker_dev --eval 'db.addUser("openshift", "mo00")'
[#] /usr/bin/mongo localhost/admin --eval 'db.addUser("openshift", "mo00")'
[#] systemctl stop mongod.service
[#] lokkit --port=27017:tcp
```

Następnie należy zapewnić uruchomienie serwera MongoDB podczas startu systemu oraz zweryfikować poprawność działania serwera. Poprawny rezultat przedstawiony jest na Rysunku 21.

```
[#] systemctl enable mongod.service
[#] systemctl start mongod.service
```

A terminal window titled 'root@broker:~' with a menu bar 'Plik Edycja Karty Pomoc'. The terminal shows the following commands and output:

```
[root@broker ~]# mongo admin
MongoDB shell version: 2.4.6
connecting to: admin
> db.auth('openshift', "mo00")
1
> show dbs
admin    0.0625GB
local    0.03125GB
openshift_broker_dev  0.0625GB
> exit
bye
[root@broker ~]#
```

Rysunek 21. Weryfikacja konfiguracji MongoDB

5.2.2.3. ActiveMQ

ActiveMQ jest oprogramowaniem służącym do wysyłania wiadomości. Jest wykorzystywany przez MCollective, przez co konieczne jest jego zainstalowanie w systemie zarządcy. W tym celu należy wydać polecenie:

```
[#] yum install -y activemq activemq-client
```

Następnie należy zmodyfikować pliki konfiguracyjne. OpenShift Origin dostarcza szablony plików konfiguracyjnych dla ActiveMQ, które można wykorzystać w celu szybkiego skonfigurowania omawianego oprogramowania. Pliki oraz źródła odpowiednich szablonów znajdują się w Tabeli 9.

Przed modyfikacją pliku /etc/activemq/activemq.xml należy stworzyć jego kopię:

```
[#] cd /etc/activemq
[#] mv activemq.xml activemq.orig
```

Tabela 9. Szablony plików konfiguracyjnych dla ActiveMQ

Plik konfiguracyjny	Szablon
/etc/activemq/activemq.xml	https://github.com/openshift/origin-server/blob/openshift-origin-release-3/documentation/files/activemq.xml
/etc/activemq/jetty.xml	https://github.com/openshift/origin-server/blob/openshift-origin-release-3/documentation/files/jetty.xml
/etc/activemq/jetty-realm.properties	https://github.com/openshift/origin-server/blob/openshift-origin-release-3/documentation/files/jetty-realm.properties

Po ściągnięciu szablonów należy przystąpić do ich modyfikacji. W pliku /etc/activemq/activemq.xml należy umieścić nazwę lokalnego komputera oraz hasło, z którego będzie korzystał MCollective (uwzględnione jedynie te linijki, w których należy wprowadzić zmiany):

```
<!--
  The <broker> element is used to configure the ActiveMQ broker.
-->
  <broker xmlns="http://activemq.apache.org/schema/core" brokerName="broker.domena.pl"
dataDirectory="${activemq.data}">

    <simpleAuthenticationPlugin>
      <users>

        <authenticationUser username="mcollective" password="marionette"
groups="mcollective,everyone"/>
        <authenticationUser username="admin" password="marionette"
groups="mcollective,admin,everyone"/>

      </broker>
```

Następnie należy uzupełnić hasło administratora w pliku /etc/activemq/jetty-realm.properties:

```
# Defines users that can access the web (console, demo, etc.)
# username: password [,rolename ...]
admin: marionette, admin
```

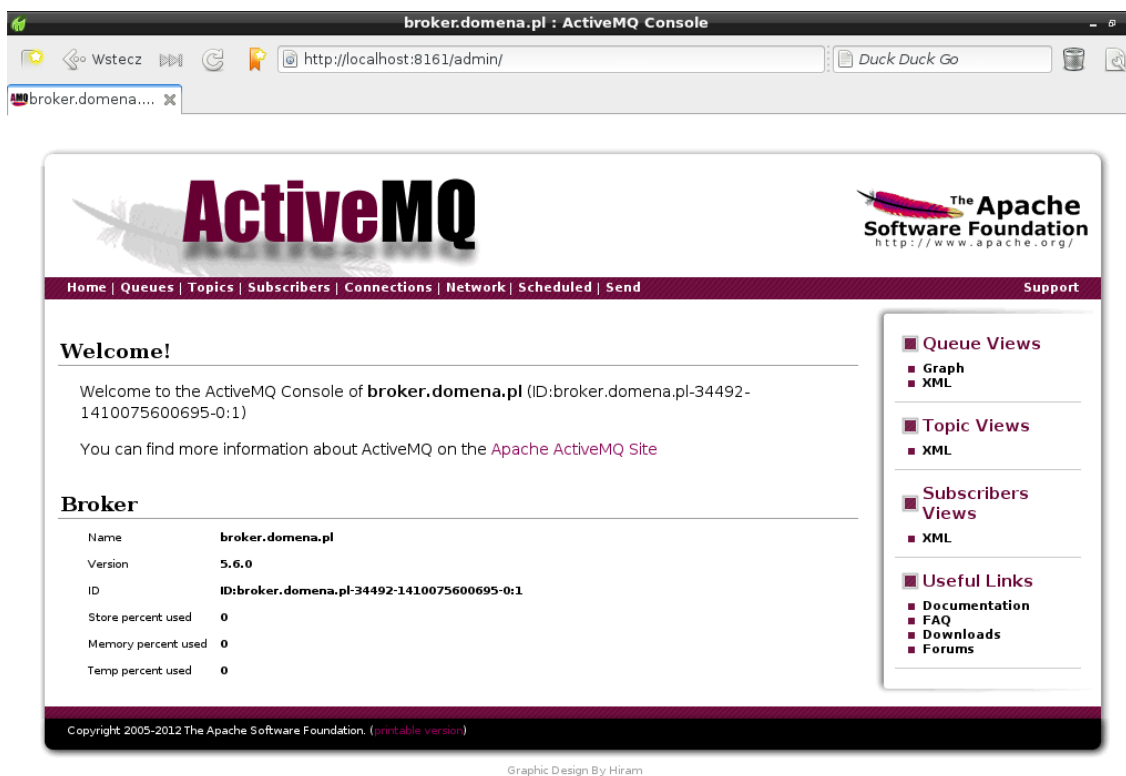
Po wykonaniu powyższych czynności należy skonfigurować system, aby uruchamiał ActiveMQ podczas startu, oraz uruchomić usługę. Trzeba również otworzyć odpowiedni port:

```
[#] lokkit --port=61613:tcp
[#] chkconfig activemq on
[#] service activemq start
```

Ostatnim krokiem jest skonfigurowanie /var/run. Należy w tym celu wydać następujące polecenie:

```
[#] cat <<EOF >/etc/tmpfiles.d/activemq.conf
d /var/run/activemq 0755 activemq activemq -
EOF
[#]
```

Po skonfigurowaniu ActiveMQ należy zweryfikować jego działanie. W tym celu należy wejść na <http://localhost:8161>. Po wpisaniu hasła skonfigurowanego w pliku /etc/activemq/jetty-realm.properties powinna pojawić się strona przedstawiona na Rysunku 22.



Rysunek 22. Weryfikacja konfiguracji ActiveMQ

5.2.2.4. Klient MCollective

W celu zapewnienia możliwości komunikacji z węzłami należy zainstalować klienta MCollective. Służy do tego następujące polecenie:

```
[#] yum install -y mcollective-client
```

Następnie należy uzupełnić plik konfiguracyjny, /etc/mcollective/client.cfg:

```
cat <<EOF > /etc/mcollective/client.cfg
topicprefix = /topic/
main_collective = mcollective
collectives = mcollective
libdir = /usr/libexec/mcollective
logfile = /var/log/openshift/broker/mcollective-client.log
loglevel = debug

# Plugins
securityprovider = psk
plugin.psk = unset

connector = activemq
plugin.activemq.pool.size = 1
plugin.activemq.pool.1.host = broker.domena.pl
plugin.activemq.pool.1.port = 61613
plugin.activemq.pool.1.user = mcollective
plugin.activemq.pool.1.password = marionette
EOF
```

5.2.2.5. Zarządca

Po zainstalowaniu technologii wykorzystywanych przez zarządcę, należy zainstalować samą aplikację służącą do zarządzania chmurą OpenShift Origin. W tym celu należy wydać następujące polecenie:

```
[#] yum install -y openshift-origin-broker openshift-origin-broker-util \
    rubygem-openshift-origin-auth-remote-user \
    rubygem-openshift-origin-auth-mongo \
    rubygem-openshift-origin-msg-broker-mcollective \
    rubygem-openshift-origin-dns-avahi \
    rubygem-openshift-origin-dns-nsupdate \
    rubygem-openshift-origin-dns-route53 \
    rubygem-passenger mod_passenger
```

Zarządca korzysta z kilku usług. Konieczne jest zapewnienie ich uruchomienia w czasie startu systemu:

```
[#] systemctl enable network.service
[#] systemctl enable sshd.service
[#] lokkit --service=ssh
[#] lokkit --service=https
[#] lokkit --service=http
```

Następnym krokiem jest wygenerowanie kluczy, za pomocą których pewne usługi będą komunikować się z zarządcą:

```
[#] openssl genrsa -out /etc/openshift/server_priv.pem 2048
[#] openssl rsa -in /etc/openshift/server_priv.pem -pubout > /etc/openshift/server_pub.pem
```

Należy również wygenerować parę kluczy, za pomocą której zarządca będzie komunikował się z węzłami:

```
[#] ssh-keygen -t rsa -b 2048 -f ~/.ssh/rsync_id_rsa
```

Frazę należy pozostawić pustą, ponieważ ta para kluczy będzie wykorzystywana do połączeń pomiędzy komputerami. Nowo utworzone klucze należy przekopiować do katalogu /etc/openshift

```
[#] cp ~/.ssh/rsync_id_rsa* /etc/openshift/
```

Następnym krokiem jest poprawne skonfigurowanie zmiennych SELinuxa. Konieczne do skonfigurowania zmienne oraz ich znaczenie zostały przedstawione w Tabeli 10.

Tabela 10. Zmienne SELinuxa niezbędne dla działania zarządcy

Zmienna SELinuxa	Znaczenie
httpd_unified	Pozwala zarządcy na zapisywanie plików posiadających kontekst http.
httpd_can_network_connect	Pozwala zarządcy na dostęp do sieci.
httpd_can_network_relay	Pozwala na dostęp do zarządcy rezydującego za serwerem Apache.
httpd_run_stickshift	Pozwala na zarządzanie uprawnieniami technologii Passenger.
named_write_master_zones	Pozwala zarządcy konfigurować DNS.
allow_ybind	Pozwala zarządcy na bezpośrednie komunikowanie się z serwerem nazw za pomocą ypbind.
httpd_verify_dns	Pozwala serwerowi Apache na zapytania dotyczące rekordów NS.
httpd_enable_homedirs	Pozwala serwerowi Apache na wykorzystanie jego katalogów domowych.
httpd_execmem	Pozwala demonowi httpd na wykonywanie programów wymagających dostępu do pamięci wykonywalnej oraz możliwej do zapisu.
httpd_read_user_content	Pozwala demonowi httpd na czytanie treści tworzonych przez użytkowników.

Następnie należy nadać odpowiedni kontekst SELinuxa pewnym plikom oraz katalogom. W celu skonfigurowania SELinuxa należy wydać następujące polecenia:


```
[#] setsebool -P httpd_unified=on httpd_can_network_connect=on httpd_can_network_relay=on \
    httpd_run_stickshift=on named_write_master_zones=on allow_yppbind=on \
    httpd_verify_dns=on httpd_enable_homedirs=on httpd_execmem=on \
    httpd_read_user_content=on

[#] (
echo fcontext -a -t httpd_var_run_t '/var/www/openshift/broker/httpd/run(/.*)?'
echo fcontext -a -t httpd_tmp_t '/var/www/openshift/broker/tmp(/.*)?'
echo fcontext -a -t httpd_log_t '/var/log/openshift/broker(/.*)?'
) | semanage -i -
[#] chcon -R -t httpd_log_t /var/log/openshift/broker
[#] chcon -R -t httpd_tmp_t /var/www/openshift/broker/httpd/run
[#] chcon -R -t httpd_var_run_t /var/www/openshift/broker/httpd/run
[#] fixfiles -R ruby193-rubygem-passenger restore
[#] fixfiles -R ruby193-mod_passenger restore
[#] fixfiles -R rubygem-passenger restore
[#] fixfiles -R mod_passenger restore
[#] restorecon -rv /var/run
[#] restorecon -rv /opt
[#] restorecon -rv /var/www/openshift/broker/tmp
[#] restorecon -v '/var/log/openshift/broker/user_action.log'
```

Po zainstalowaniu zarządcy należy odpowiednio zmodyfikować plik /etc/openshift/broker.conf, zawierający konfigurację zarządcy. Należy zdefiniować wielkości kontenerów dostępne dla przyszłych użytkowników chmury, domenę, zmienne niezbędne do poprawnego połączenia z serwerem MongoDB oraz ciągi bitów istotne dla bezpieczeństwa chmury. Ciągi te generuje się za pomocą poniższego polecenia:

```
[#] openssl rand -base64 64
```

Zmiany w pliku /etc/openshift/broker.conf przedstawione zostały poniżej (pominięto wpisy nieistotne dla omawianych zmian):

```
# Domain suffix to use for applications (Must match node config)
CLOUD_DOMAIN="domena.pl"
# Comma-separated list of valid gear sizes
# Eg: "small,medium,large"
VALID_GEAR_SIZES="small"

# For replica sets, use ',' delimiter for multiple servers
# Eg: MONGO_HOST_PORT="<host1:port1>,<host2:port2>..."
MONGO_HOST_PORT="localhost:27017"
MONGO_USER="openshift"
MONGO_PASSWORD="mo00"
MONGO_DB="openshift_broker_dev"
MONGO_TEST_DB="openshift_broker_test"
MONGO_SSL="false"
```

```
# Whenever this value is changed all gear encryption tokens must be recreated.
# It's recommended that you set this value using "openssl rand -base64 64". It
```

```
# must be the same across all Brokers.
AUTH_SALT="z+ad5Rp4wNuoyUc8Ngv1a/9kv7vgPs7c/oiodXkYmrf/z80LURqDqU6l097BirCs
8nKI4LGK6WvZDCqKw6DEuw=="
AUTH_PRIV_KEY_FILE="/etc/openshift/server_priv.pem"
AUTH_PRIV_KEY_PASS=""
AUTH_PUB_KEY_FILE="/etc/openshift/server_pub.pem"
AUTH_RSYNC_KEY_FILE="/etc/openshift/rsync_id_rsa"

# This session must be shared amongst all Brokers but otherwise secret. If
# this value is changed sessions will be dropped. This value is used for
# setting the rails secret_token (or the secret_key_base for Rails 4.0).
# "openssl rand -hex 64" can be use to generate a unique value.
#
SESSION_SECRET="dwo/68Z94vpF+
+XZJnkH3wwQKpVDgPHTERg8ynyqzMJHFIPsJV8o078mEOXHOQ9c
hXo7MKZBYXHSUhFxj58Ifg=="
```

5.2.2.6. Wtyczki wykorzystywane przez zarządcę

Zarządca komunikuje się z technologiami pomocniczymi za pomocą wtyczek. Pliki konfiguracyjne znajdują się w katalogu /etc/openshift/plugins.d. OpenShift Origin zapewnia szablony tych plików, które można wykorzystać do szybkiego skonfigurowania wtyczek. Pierwszym krokiem jest skopiowanie szablonów do właściwych plików konfiguracyjnych oraz konfiguracja wtyczki zapewniającej dostęp do klienta MCollective:

```
[#] cd /etc/openshift/plugins.d
[#] cp openshift-origin-auth-remote-user.conf.example openshift-origin-auth-remote-user.conf
[#] cp openshift-origin-msg-broker-mcollective.conf.example openshift-origin-msg-broker-mcollective.conf

[#] sed -i 's/MCOLLECTIVE_CONFIG.*MCOLLECTIVE_CONFIG=/etc/mcollective/client.cfg/g' /etc/openshift/plugins.d/openshift-origin-msg-broker-mcollective.conf
```

Następnie należy skonfigurować wtyczkę zapewniającą dostęp do serwera DNS. W tym celu należy wydać następujące polecenie:

```
[#] cat << EOF > openshift-origin-dns-nsupdate.conf
BIND_SERVER="127.0.0.1"
BIND_PORT=53
BIND_KEYNAME="${domain}"
BIND_KEYVALUE="${KEY}"
BIND_ZONE="${domain}"
EOF
```

Ostatnią wtyczką, którą należy skonfigurować, jest wtyczka służąca do autoryzacji użytkowników. W konfiguracji testowej wykorzystywany jest proces autoryzacji oparty na pliku /etc/openshift/htpasswd. W celu konfiguracji należy wydać następujące

polecenia:

```
[#] cp /var/www/openshift/broker/httpd/conf.d/openshift-origin-auth-remote-user-basic.conf.sample  
/var/www/openshift/broker/httpd/conf.d/openshift-origin-auth-remote-user.conf  
[#] htpasswd -c /etc/openshift/htpasswd mateusz
```

W celu zweryfikowania procesu tworzenia użytkownika, należy zapoznać się z zawartością pliku `/etc/openshift/htpasswd`. Poprawna zawartość pliku powinna wyglądać analogicznie do przedstawionej poniżej:

```
mateusz:$apr1$DBwAmC8f$6Jlv8Vgj/36hKEx/DQofu1
```

Zarządca, jako aplikacja stworzona za pomocą technologii Rails, wymaga pewnych bibliotek do poprawnego działania. Obecność wymaganych bibliotek weryfikuje się za pomocą następujących poleceń:

```
[#] cd /var/www/openshift/broker  
[#] bundle --local
```

Polecenie powinno zakończyć się informacją: „Your bundle is complete!”

Zarządca powinien uruchamiać się wraz z systemem. Zapewniają to następujące polecenia:

```
[#] systemctl enable openshift-broker.service  
[#] systemctl start openshift-broker.service
```

Po skonfigurowaniu zarządcy należy zweryfikować poprawność działania aplikacji. Służy do tego następujące polecenie:

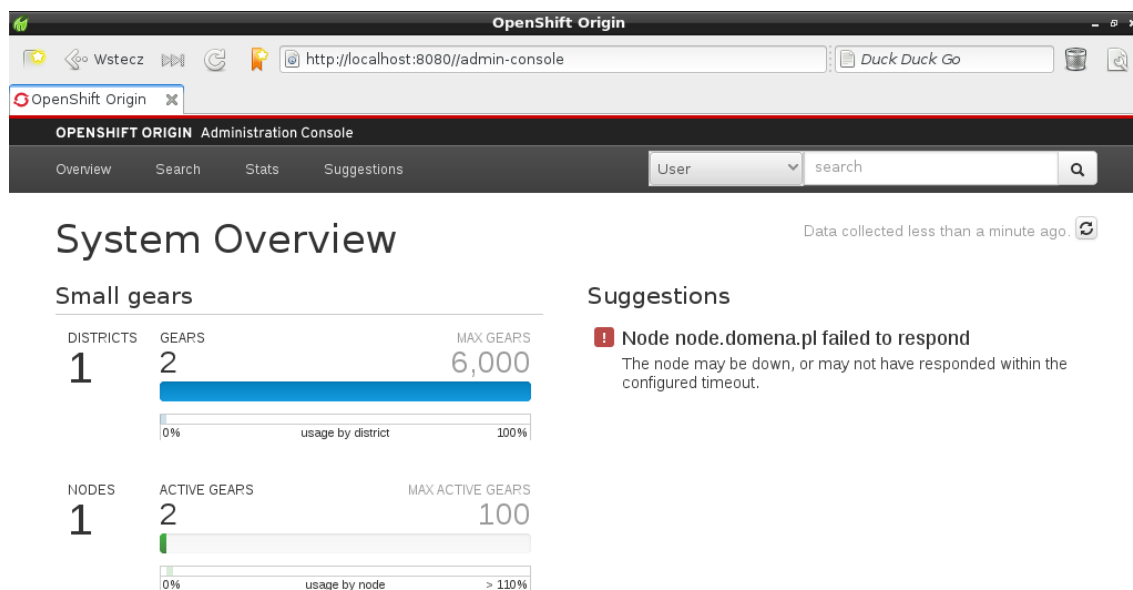
```
[#] curl -u mateusz:mateusz http://localhost:8080/broker/rest/api.json
```

5.2.2.7. Konsola administracyjna

OpenShift Origin udostępnia konsolę administracyjną, oferującą możliwość obserwacji stanu chmury. Do jej zainstalowania służy następujące polecenie:

```
[#] yum install -y rubygem-openshift-origin-admin-console
```

Do konfiguracji konsoli służy plik `/etc/openshift/plugins.d/openshift-origin-admin-console.conf`. Po zainstalowaniu konsoli należy zweryfikować poprawność instalacji, uruchamiając przeglądarkę i wpisując adres `http://localhost:8080/admin-console`. W wyniku tego powinna załadować się strona podobna do strony przedstawionej na Rysunku 23.



Rysunek 23. Konsola administracyjna

Informacje przedstawione na stronie mogą być inne niż te przedstawione na Rysunku 23 – zależy to od ilości węzłów w chmurze oraz ich stanu.

5.2.2.8. Konsola użytkownika

OpenShift Origin udostępnia również konsolę dla użytkowników, pozwalającą im na tworzenie oraz zarządzanie aplikacjami. Nie jest ona niezbędna do działania chmury, stanowi jednak duże ułatwienie dla użytkowników.

Do instalacji konsoli służy polecenie:

```
[#] yum install -y openshift-origin-console
```

Należy wygenerować ciąg bitów, służących jak sekret sesji. Do wygenerowania ciągu bitów należy wykorzystać polecenie:

```
[#] openssl rand -base64 64
```

Otrzymany ciąg należy umieścić w pliku `/etc/openshift/console.conf`:

```
SESSION_SECRET="bLC03djcvQNpktUfMx41v7k87tJmmWXYk1caYgQP8gaXzkO5kAt6o3Q7LDgF  
psrqJqJh/s5PZkCCp8Qg1Vq0Q=="
```

W przypadku wdrażanej konfiguracji konsola wykorzystuje ten sam plik z użytkownikami, co zarządca. W tym celu należy skopiować odpowiedni plik konfiguracyjny:

```
[#] cd /var/www/openshift/console/httpd/conf.d  
[#] cp openshift-origin-auth-remote-user-basic.conf.sample openshift-origin-auth-remote-user-  
basic.conf
```

Podobnie jak zarządca, konsola wymaga obecności pewnych bibliotek:

```
[#] cd /var/www/openshift/console  
[#] bundle --local
```

Polecenie powinno zakończyć się informacją: „Your bundle is complete!”.

Ostatnim krokiem jest ustawienie odpowiedniego kontekstu SELinuxa dla odpowiednich plików oraz katalogów:

```
[#] (  
echo fcontext -a -t httpd_log_t '/var/log/openshift/console(/.*)?'  
echo fcontext -a -t httpd_log_t '/var/log/openshift/console/httpd(/.*)?'  
echo fcontext -a -t httpd_var_run_t '/var/www/openshift/console/httpd/run(/.*)?'  
) | semanage -i -
```

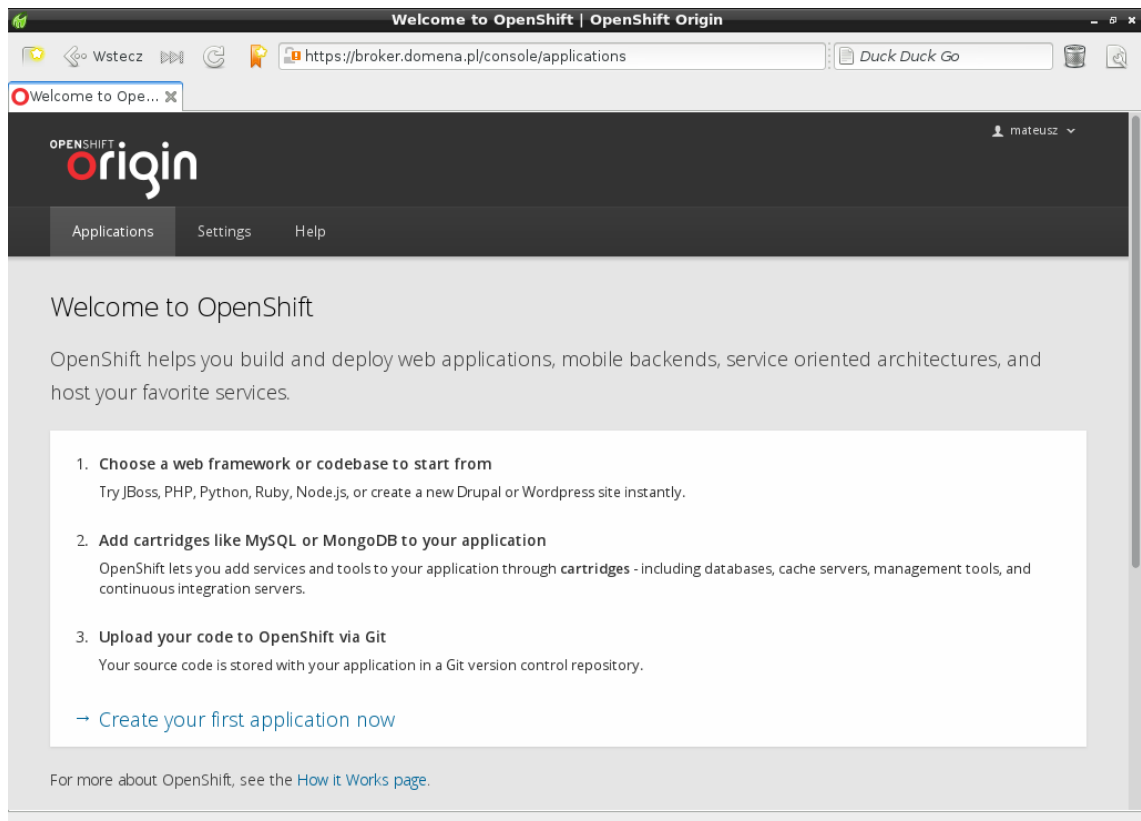
```
[#] fixfiles -R ruby193-rubygem-passenger restore  
[#] fixfiles -R ruby193-mod_passenger restore  
[#] fixfiles -R rubygem-passenger restore  
[#] fixfiles -R mod_passenger restore
```

```
[#] restorecon -rv /var/run  
[#] restorecon -rv /opt  
[#] restorecon -R /var/log/openshift/console  
[#] restorecon -R /var/www/openshift/console
```

Konsola powinna uruchamiać się wraz z systemem:

```
[#] systemctl enable openshift-console.service  
[#] systemctl start openshift-console.service
```

Po wykonaniu powyższych czynności należy uruchomić przeglądarkę i wpisać adres `broker.domena.pl/console`. Jako login i hasło należy wykorzystać dane z pliku `/etc/openshift/htpasswd`. Powinna pojawić strona przedstawiona na Rysunku 24.



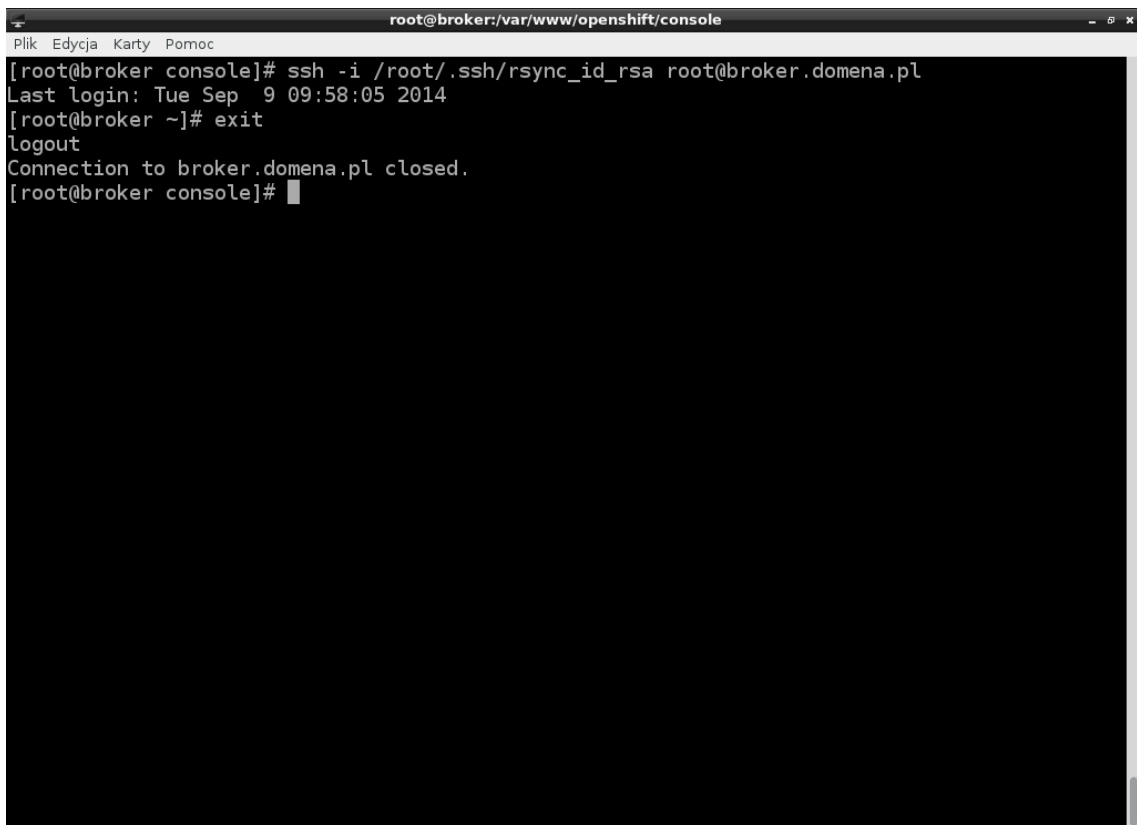
Rysunek 24. Konsola użytkownika

5.2.2.9. Konfiguracja kluczy SSH w systemie węzła

Należy skonfigurować węzeł tak, aby do łączności z zarządcą wykorzystywał klucze SSH, co umożliwi komunikację bez konieczności podawania hasła. W tym celu należy wydać następujące polecenia:

```
[#] cp -f /etc/openshift/rsync_id_rsa.pub /root/.ssh/  
[#] cat /root/.ssh/rsync_id_rsa.pub >> /root/.ssh/authorized_keys
```

Po wykonaniu powyższych poleceń należy zweryfikować poprawność konfiguracji. Należy w tym celu spróbować zalogować się do systemu za pomocą klucza. Poprawny przebieg procesu został przedstawiony na Rysunku 25.

A screenshot of a terminal window titled 'root@broker:/var/www/openshift/console'. The terminal shows a sequence of commands and their outputs: '[root@broker console]# ssh -i /root/.ssh/rsync_id_rsa root@broker.domena.pl', 'Last login: Tue Sep 9 09:58:05 2014', '[root@broker ~]# exit', 'logout', 'Connection to broker.domena.pl closed.', and '[root@broker console]#'. The terminal background is black with white text. The window has standard Linux window controls (minimize, maximize, close) in the top right corner.

Rysunek 25. Weryfikacja konfiguracji SSH

5.2.2.10. Serwer MCollective

W systemie węzła należy zainstalować serwer MCollective w celu zapewnienia łączności z zarządcą. Należy w tym celu wydać następujące polecenie:

```
[#] yum install -y openshift-origin-msg-node-mcollective
```

Po zainstalowaniu oprogramowania, należy zmodyfikować plik `/etc/mcollective/server.cfg`:

```
[#] cat <<EOF >/etc/mcollective/server.cfg
topicprefix = /topic/
main_collective = mcollective
collectives = mcollective
libdir = /usr/libexec/mcollective
logfile = /var/log/openshift/node/mcollective.log
loglevel = debug
daemonize = 0
direct_addressing = 1
registerinterval = 30

# Plugins
securityprovider = psk
```

```

plugin.psk = unset

connector = activemq
plugin.activemq.pool.size = 1
plugin.activemq.pool.1.host = broker.domena.pl
plugin.activemq.pool.1.port = 61613
plugin.activemq.pool.1.user = mcollective
plugin.activemq.pool.1.password = marionette

# Facts
factsources = yaml
plugin.yaml = /etc/mcollective/facts.yaml
EOF

```

Konieczna jest modyfikacja pliku usługi, żeby zapobiec wyłączeniu kontenerów w przypadku ponownego uruchomienia usługi MCollective. Po modyfikacji należy ponownie uruchomić usługę Systemd oraz zapewnić uruchomienie usługi MCollective podczas startu systemu:

```

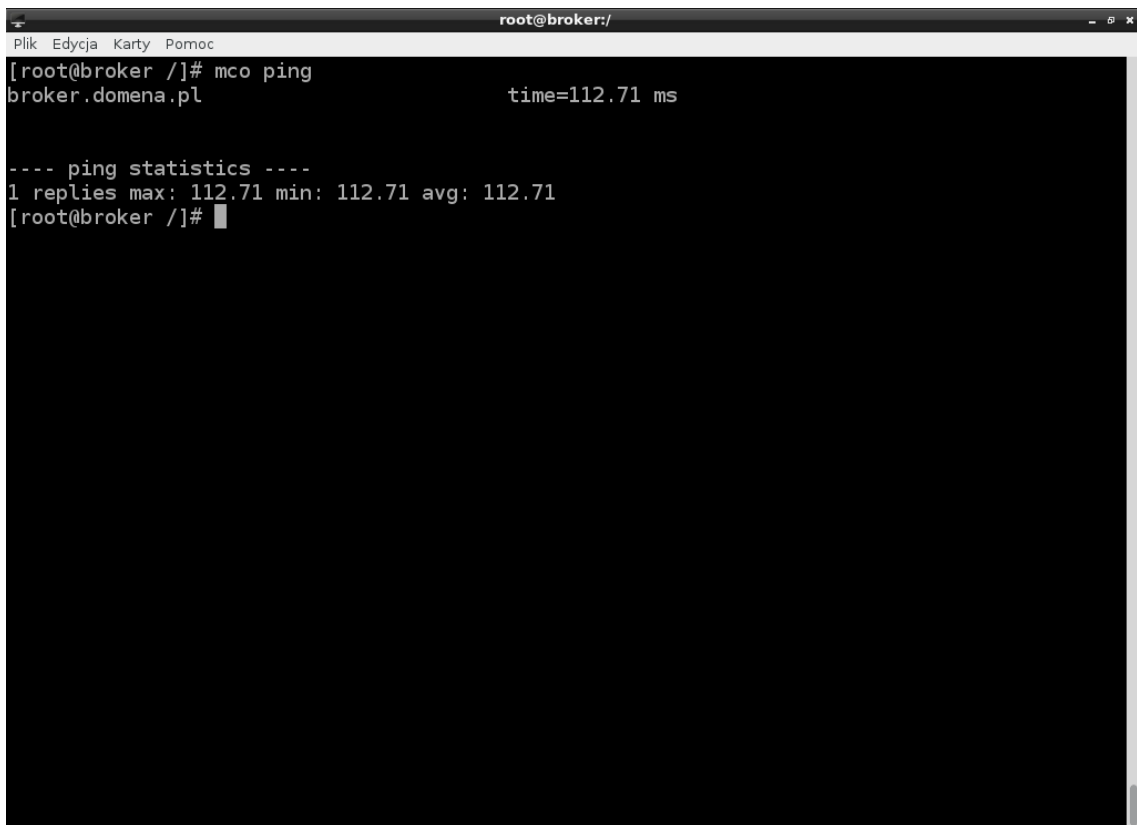
[#] cat <<EOF > /usr/lib/systemd/system/mcollective.service
[Unit]
Description=The Marionette Collective
After=network.target

[Service]
Type=simple
StandardOutput=syslog
StandardError=syslog
ExecStart=/usr/sbin/mcollectived --config=/etc/mcollective/server.cfg --pidfile=/var/run/mcollective.pid
ExecReload=/bin/kill -USR1 $MAINPID
PIDFile=/var/run/mcollective.pid
KillMode=process

[Install]
WantedBy=multi-user.target
EOF
[#] systemctl --system daemon-reload
[#] systemctl enable mcollective.service
[#] systemctl start mcollective.service

```

W tym momencie węzeł powinien móc skontaktować się z zarządcą za pośrednictwem MCollective. Należy to zweryfikować za pomocą polecenia mco ping. Poprawny efekt przedstawiony jest na Rysunku 26.



```
root@broker:/  
[root@broker /]# mco ping broker.domena.pl  
time=112.71 ms  
  
---- ping statistics ----  
1 replies max: 112.71 min: 112.71 avg: 112.71  
[root@broker /]#
```

Rysunek 26. Weryfikacja konfiguracji MCollective

5.2.2.11. Węzeł

Pakiety niezbędne do poprawnego działania węzła instaluje się za pomocą polecenia:

```
[#] yum install -y rubygem-openshift-origin-node \  
    rubygem-passenger-native \  
    openshift-origin-port-proxy \  
    openshift-origin-node-util \  
    rubygem-openshift-origin-container-selinux
```

Następnie nadać skonfigurować nazwę serwera Apache, umożliwić dostęp do informacji o kasetach oraz należy zainstalować wtyczki, zapewniające przekazywanie żądań do kontenerów umieszczonych na węźle:

```
[#] echo "ServerName broker.domena.pl" >  
/etc/httpd/conf.d/000001_openshift_origin_node_servername.conf  
[#] cat << EOF > /etc/httpd/conf.d/cartridge_files.conf  
<Directory /usr/libexec/openshift/cartridges/*/info/configuration >  
    Require all granted  
</Directory>  
EOF  
[#] yum install -y rubygem-openshift-origin-frontend-apache-mod-rewrite
```

```
[#] yum install -y openshift-origin-node-proxy rubygem-openshift-origin-frontend-nodejs-websocket
```

Koniecznym krokiem jest utworzenie odpowiednich reguł przekierowania:

```
[#] cat <<EOF > /tmp/nodes.broker_routes.txt
__default__ REDIRECT:/console
__default__ /console TOHTTPS:127.0.0.1:8118/console
__default__ /broker TOHTTPS:127.0.0.1:8080/broker
EOF
[#] mkdir -p /etc/httpd/conf.d/openshift
[#] cat /etc/httpd/conf.d/openshift/nodes.txt /tmp/nodes.broker_routes.txt >
/etc/httpd/conf.d/openshift/nodes.txt.new
[#] mv -f /etc/httpd/conf.d/openshift/nodes.txt.new /etc/httpd/conf.d/openshift/nodes.txt
[#] httpd2dbm -f DB -i /etc/httpd/conf.d/openshift/nodes.txt -o
/etc/httpd/conf.d/openshift/nodes.db.new
[#] chown root:apache /etc/httpd/conf.d/openshift/nodes.txt
/etc/httpd/conf.d/openshift/nodes.db.new
[#] chmod 750 /etc/httpd/conf.d/openshift/nodes.txt /etc/httpd/conf.d/openshift/nodes.db.new
mv -f /etc/httpd/conf.d/openshift/nodes.db.new /etc/httpd/conf.d/openshift/nodes.db
```

Należy zapewnić uruchomienie usług proxy wraz ze startem systemu oraz utworzyć niezbędne wpisy w konfiguracji zapory sieciowej:

```
[#] service openshift-node-web-proxy enable
[#] service openshift-node-web-proxy start
[#] iptables -N rhc-app-comm
[#] iptables -I INPUT 4 -m tcp -p tcp --dport 35531:65535 -m state --state NEW -j ACCEPT
[#] iptables -I INPUT 5 -j rhc-app-comm
[#] iptables -I OUTPUT 1 -j rhc-app-comm
[#] /usr/libexec/iptables/iptables.init save
```

Podczas dodawania wpisów do łańcuchów iptables mogą wystąpić błędy związane z podaniem błędnych indeksów. W takim przypadku, należy ponownie wydać polecenie utworzenia wpisów, modyfikując indeksy.

5.2.2.12. Instalacja kaset

Po zainstalowaniu aplikacji węzła należy zainstalować kasety zapewniające wsparcie dla wybranych technologii. W opisywanej konfiguracji zainstalowane zostaną kasety zapewniające wsparcie dla języka PHP, serwera baz danych MariaDB oraz HAProxy, niezbędnego do tworzenia aplikacji skalowalnych. Oprócz tego należy zainstalować kasety zapewniające wsparcie dla crona, wymaganą we wszystkich konfiguracjach chmury. W celu zainstalowania wyżej wymienionych kaset, należy wydać następujące polecenia:

```
[#] yum install -y openshift-origin-cartridge-cron openshift-origin-cartridge-haproxy openshift-origin-cartridge-php openshift-origin-cartridge-mariadb
[#] /usr/sbin/oo-admin-cartridge --recursive -a install -s /usr/libexec/openshift/cartridges/
```

Na koniec należy zapewnić odpowiednią konfigurację zapory sieciowej oraz uruchomienie odpowiednich usług podczas startu systemu:

```
[#] lokkit --service=ssh
[#] lokkit --service=https
[#] lokkit --service=http
[#] lokkit --port=8000:tcp
[#] lokkit --port=8443:tcp
[#] systemctl enable network.service
[#] systemctl enable sshd.service
[#] systemctl enable oddjobd.service
[#] systemctl enable openshift-node-web-proxy.service
```

5.2.2.13. Konfiguracja współdzielenia zasobów węzła

Należy odpowiednio skonfigurować system węzła, aby umożliwić współdzielenie wielu aplikacji. Do modyfikacji plików konfiguracyjnych warto wykorzystać program augeas, instalowany za pomocą polecenia:

```
[#] yum install -y augeas
```

OpenShift Origin dostarcza moduł PAM, zapewniający odpowiednią konfigurację procesu uwierzytelniania. Należy skonfigurować system węzła, aby wykorzystywał moduł PAM dostarczany przez OpenShift Origin:

```
[#] cat <<EOF | augtool
set /files/etc/pam.d/sshd/#comment[.='pam_selinux.so close should be the first session rule']
'pam_openshift.so close should be the first session rule'
ins 01 before /files/etc/pam.d/sshd/*[argument='close']
set /files/etc/pam.d/sshd/01/type session
set /files/etc/pam.d/sshd/01/control required
set /files/etc/pam.d/sshd/01/module pam_openshift.so
set /files/etc/pam.d/sshd/01/argument close
set /files/etc/pam.d/sshd/01/#comment 'Managed by openshift_origin'

set /files/etc/pam.d/sshd/#comment[.='pam_selinux.so open should only be followed by sessions to
be executed in the user context'] 'pam_openshift.so open should only be followed by sessions to be
executed in the user context'
ins 02 before /files/etc/pam.d/sshd/*[argument='open']
set /files/etc/pam.d/sshd/02/type session
set /files/etc/pam.d/sshd/02/control required
set /files/etc/pam.d/sshd/02/module pam_openshift.so
set /files/etc/pam.d/sshd/02/argument[1] open
set /files/etc/pam.d/sshd/02/argument[2] env_params
set /files/etc/pam.d/sshd/02/#comment 'Managed by openshift_origin'

rm /files/etc/pam.d/sshd/*[module='pam_selinux.so']

set /files/etc/pam.d/sshd/03/type session
set /files/etc/pam.d/sshd/03/control required
set /files/etc/pam.d/sshd/03/module pam_namespace.so
```

```

set /files/etc/pam.d/sshd/03/argument[1] no_unmount_on_close
set /files/etc/pam.d/sshd/03/#comment 'Managed by openshift_origin'

set /files/etc/pam.d/sshd/04/type session
set /files/etc/pam.d/sshd/04/control optional
set /files/etc/pam.d/sshd/04/module pam_cgroup.so
set /files/etc/pam.d/sshd/04/#comment 'Managed by openshift_origin'

set /files/etc/pam.d/runuser/01/type session
set /files/etc/pam.d/runuser/01/control required
set /files/etc/pam.d/runuser/01/module pam_namespace.so
set /files/etc/pam.d/runuser/01/argument[1] no_unmount_on_close
set /files/etc/pam.d/runuser/01/#comment 'Managed by openshift_origin'

set /files/etc/pam.d/runuser-l/01/type session
set /files/etc/pam.d/runuser-l/01/control required
set /files/etc/pam.d/runuser-l/01/module pam_namespace.so
set /files/etc/pam.d/runuser-l/01/argument[1] no_unmount_on_close
set /files/etc/pam.d/runuser-l/01/#comment 'Managed by openshift_origin'

set /files/etc/pam.d/su/01/type session
set /files/etc/pam.d/su/01/control required
set /files/etc/pam.d/su/01/module pam_namespace.so
set /files/etc/pam.d/su/01/argument[1] no_unmount_on_close
set /files/etc/pam.d/su/01/#comment 'Managed by openshift_origin'

set /files/etc/pam.d/system-auth-ac/01/type session
set /files/etc/pam.d/system-auth-ac/01/control required
set /files/etc/pam.d/system-auth-ac/01/module pam_namespace.so
set /files/etc/pam.d/system-auth-ac/01/argument[1] no_unmount_on_close
set /files/etc/pam.d/system-auth-ac/01/#comment 'Managed by openshift_origin'
save
EOF
[#] cat <<EOF > /etc/security/namespace.d/sandbox.conf
# /sandbox    \${HOME}/.sandbox/    user:iscript=/usr/sbin/oo-namespace-init    root,adm,apache
EOF
[#] cat <<EOF > /etc/security/namespace.d/tmp.conf
/tmp    \${HOME}/.tmp/    user:iscript=/usr/sbin/oo-namespace-init root,adm,apache
EOF
[#] cat <<EOF > /etc/security/namespace.d/vartmp.conf
/var/tmp    \${HOME}/.tmp/    user:iscript=/usr/sbin/oo-namespace-init root,adm,apache
EOF

```

Następnie należy skonfigurować uruchamianie usług odpowiedzialnych za zarządzanie grupami kontrolnymi:

```

[#] systemctl enable cgconfig.service
[#] systemctl enable cgred.service
[#] systemctl start cgconfig.service
[#] systemctl start cgred.service

```

W celu kontrolowania wykorzystywanej przez aplikacje użytkowników pamięci dyskowej należy dodać odpowiednią opcję do pliku /etc/fstab. Zmodyfikować należy wpis odnoszący się do punktu montowania partycji zawierającej katalog

/var/lib/openshift. W przypadku opisywanej konfiguracji jest to katalog /. Wpis w pliku /etc/fstab powinien wyglądać następująco:

```
/dev/mapper/fedora-root / ext4 defaults,usrquota 1 1
```

Następnie należy ponownie zamontować partycję oraz wygenerować informacje dotyczące ograniczeń:

```
[#] mount -o remount /  
[#] quotacheck -cmug /
```

Następnym krokiem jest odpowiednie skonfigurowanie zmiennych oraz kontekstu SELinuxa dla odpowiednich katalogów. Konieczne do skonfigurowania zmienne oraz ich znaczenie zostały przedstawione w Tabeli 11.

Tabela 11. Zmienne SELinuxa niezbędne dla działania węzła

Zmienna SELinuxa	Znaczenie
httpd_unified	Pozwala zarządcy na zapisywanie plików posiadających kontekst http.
httpd_can_network_connect	Pozwala zarządcy na dostęp do sieci.
httpd_can_network_relay	Pozwala na dostęp do zarządcy rezydującego za serwerem Apache.
httpd_run_stickshift	Pozwala na zarządzanie uprawnieniami technologii Passenger.
allow_polyinstantiation	Pozwala na wykorzystywanie wielu instancji tego samego katalogu.
httpd_enable_homedirs	Pozwala węzłowi na odczytanie danych aplikacji.
httpd_execmem	Pozwala demonowi httpd na wykonywanie programów wymagających dostępu do pamięci wykonywalnej oraz możliwej do zapisu.
httpd_read_user_content	Pozwala węzłowi na czytanie treści tworzonych przez użytkowników.

W celu skonfigurowania SELinuxa należy wydać następujące polecenia:

```
[#] setsebool -P httpd_unified=on httpd_can_network_connect=on httpd_can_network_relay=on \  
    httpd_read_user_content=on httpd_enable_homedirs=on httpd_run_stickshift=on \  
    allow_polyinstantiation=on httpd_execmem=on  
[#] restorecon -rv /var/run  
[#] restorecon -rv /var/lib/openshift /etc/openshift/node.conf /etc/httpd/conf.d/openshift
```

Kolejną czynnością jest zmodyfikowanie pliku `/etc/sysctl.conf` w celu umożliwienia istnienia wielu procesów demona `httpd`:

```
[# ]cat <<EOF | augtool
set /files/etc/sysctl.conf/kernel.sem "250 32000 32 4096"
set /files/etc/sysctl.conf/net.ipv4.ip_local_port_range "15000 35530"
set /files/etc/sysctl.conf/net.netfilter.nf_conntrack_max "1048576"
save
EOF
[#] sysctl -p /etc/sysctl.conf
```

Należy zmodyfikować konfigurację SSH. Konieczne jest, aby SSH akceptował zmienną środowiskową `GIT_SSH`. Trzeba również zwiększyć ilość dopuszczanych połączeń nawiązywanych z systemem węzła:

```
[#] cat <<EOF >> /etc/ssh/sshd_config
AcceptEnv GIT_SSH
EOF
[#] cat <<EOF | augtool
set /files/etc/ssh/sshd_config/MaxSessions 40
save
EOF
```

Ważne jest, aby zapewnić uruchomienie usługi kontrolującej połączenia nawiązywane z kontenerami:

```
[#] systemctl enable openshift-tc.service
```

Ponieważ na jednym węźle umieszczanych jest wiele kontenerów, należy zapewnić poprawne funkcjonowanie usługi zapewniającej przekazanie żądań HTTP do aplikacji, nasłuchujących na adresach pętli zwrotnej. Należy również upewnić się, że w przypadku ponownego uruchomienia węzła kontenery również zostaną uruchomione. Wyżej wymienione czynności wykonuje się za pomocą poniższych poleceń:

```
[#] lokkit --port=35531-65535:tcp
[#] systemctl enable openshift-port-proxy.service
[#] systemctl restart openshift-port-proxy.service
[#] systemctl enable openshift-gears.service
```

5.2.2.14. Konfiguracja aplikacji węzła

Należy skonfigurować nazwę domeny, nazwę systemu węzła, adres IP zarządcy, adres IP węzła oraz wykorzystywany interfejs sieciowy. Ustawienia te należy umieścić w pliku `/etc/openshift/node.conf`:

```
# These should not be left at default values, even for a demo.
# "PUBLIC" networking values are ones that end-users should be able to reach.
```

```

PUBLIC_HOSTNAME="broker.domena.pl"    # The node host's public hostname
PUBLIC_IP="192.168.1.125"            # The node host's public IP address
BROKER_HOST="broker.domena.pl"       # IP or DNS name of broker server for REST API

# Usually (unless in a demo) this should be changed to the domain for your installation:
CLOUD_DOMAIN="domena.pl"            # Domain suffix to use for applications (Must match broker
config)

# You may want these, depending on the complexity of your networking:
EXTERNAL_ETH_DEV='p2p1'              # Specify the internet facing public ethernet device
# INTERNAL_ETH_DEV='eth1'            # Specify the internal cluster facing ethernet device

```

Ostatnim krokiem jest zmodyfikowanie pliku `login.defs` oraz uaktualnienie bazy danych dla usługi MCollective:

```

[#] cat <<EOF | augtool
set /files/etc/login.defs/UID_MIN 500
set /files/etc/login.defs/GID_MIN 500
save
EOF
[#] /etc/cron.minutely/openshift-facts

```

5.2.3. Węzeł drugi

Proces konfiguracji węzła został opisany dokładnie w podpunkcie 5.2.2. Zdecydowaną większość czynności należy powtórzyć dla drugiej maszyny. Istnieje jednak kilka różnic wynikających z faktu, że na drugiej maszynie nie jest instalowana aplikacja zarządcy. Wobec tego podczas konfigurowania drugiego węzła nie definiuje się reguł przekierowywania z podpunktu 5.2.11. Oprócz tego trzeba wykonać kilka dodatkowych czynności.

5.2.3.1. Rejestracja systemu węzła w systemie DNS

Należy powiadomić serwer BIND o istnieniu węzła. W tym celu należy zalogować się na system zarządcy oraz wydać następujące polecenie:

```
[#] oo-register-dns -h node -d domena.pl -n 192.168.1.126 -k ${keyfile}
```

Należy również skonfigurować system węzła do korzystania z serwera DNS zainstalowanego na komputerze zarządcy. W tym celu do pliku `/etc/resolv.conf` węzła trzeba dodać wpis:

```
nameserver 192.168.1.125
```

Serwer DNS powinien rozwiązać nazwę `node.domena.pl`. Należy to zweryfikować za pomocą polecenia `dig`. Poprawny rezultat przedstawia Rysunek 27.

```
root@broker:~  
Plik Edycja Karty Pomoc  
[root@broker ~]# dig @127.0.0.1 node.domena.pl  
  
; <<> DiG 9.9.3-r1.13207.22-P2-RedHat-9.9.3-14.P2.fc19 <<> @127.0.0.1 node.domena.pl  
; (1 server found)  
;; global options: +cmd  
;; Got answer:  
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 63785  
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2  
  
;; OPT PSEUDOSECTION:  
; EDNS: version: 0, flags:; udp: 4096  
;; QUESTION SECTION:  
;node.domena.pl. IN A  
  
;; ANSWER SECTION:  
node.domena.pl. 180 IN A 192.168.1.126  
  
;; AUTHORITY SECTION:  
domena.pl. 1 IN NS ns1.domena.pl.  
  
;; ADDITIONAL SECTION:  
ns1.domena.pl. 1 IN A 127.0.0.1  
  
;; Query time: 75 msec  
;; SERVER: 127.0.0.1#53(127.0.0.1)  
;; WHEN: wto wrz 09 15:00:33 CEST 2014  
;; MSG SIZE rcvd: 93  
  
[root@broker ~]#
```

Rysunek 27. Weryfikacja rozwiązania nazwy węzła przez serwer DNS

Na koniec należy zmodyfikować nazwę systemu węzła:

```
[#] echo "node.domena.pl" > /etc/hostname  
[#] hostname node.domena.pl
```

5.2.3.2. Konfiguracja kluczy SSH w systemie węzła

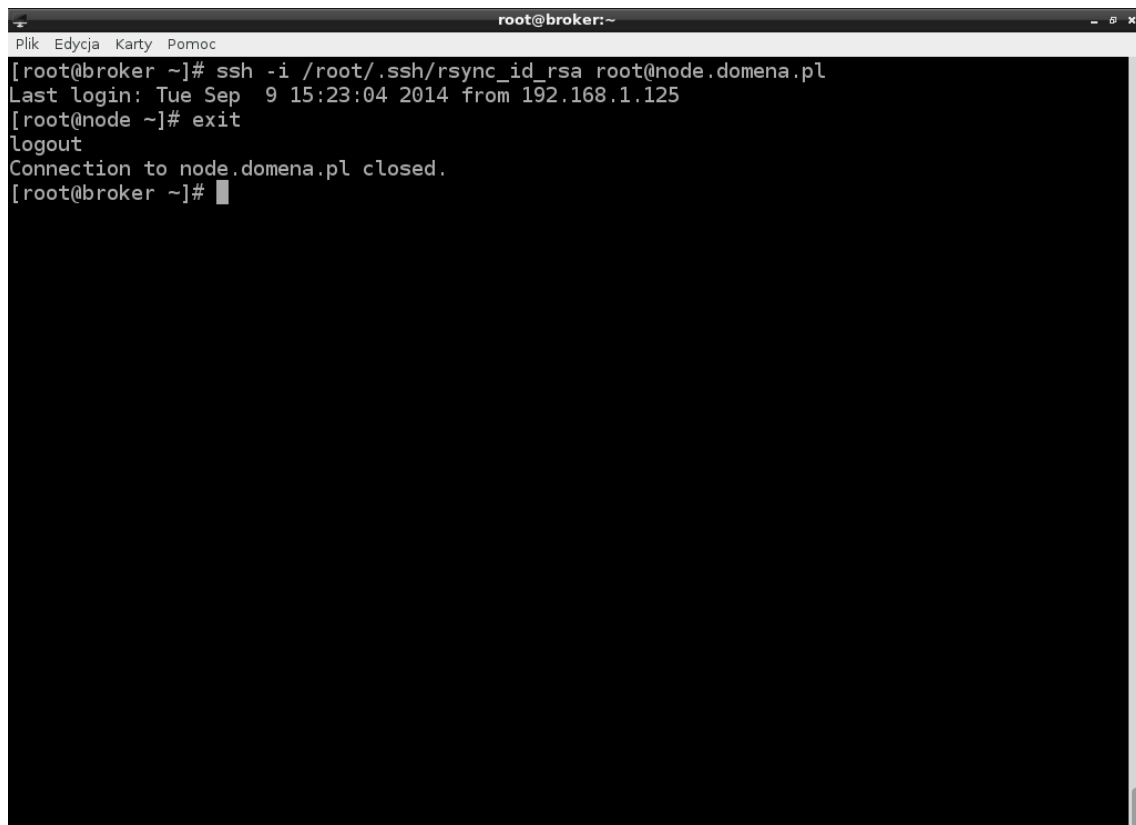
Należy skopiować klucz publiczny na system węzła. W tym celu na systemie zarządcy należy wydać następujące polecenie:

```
[#] scp /etc/openshift/rsync_id_rsa.pub root@node.domena.pl:/root/.ssh
```

Na systemie węzła należy skopiować klucz do pliku `authorized_keys`:

```
[#] cat /root/.ssh/rsync_id_rsa.pub >> /root/.ssh/authorized_keys
```

Należy zweryfikować konfigurację poprzez zalogowanie się za pomocą klucza. Poprawny przebieg operacji został przedstawiony na Rysunku 28.

A terminal window titled 'root@broker:~' with a menu bar containing 'Plik', 'Edycja', 'Karty', and 'Pomoc'. The terminal shows the following commands and output:

```
[root@broker ~]# ssh -i /root/.ssh/rsync_id_rsa root@node.domena.pl
Last login: Tue Sep  9 15:23:04 2014 from 192.168.1.125
[root@node ~]# exit
logout
Connection to node.domena.pl closed.
[root@broker ~]#
```

Rysunek 28. Weryfikacja konfiguracji SSH

5.2.3.3. Konfiguracja węzła

Należy wprowadzić odpowiednie informacje do pliku `/etc/openshift/node.conf` znajdującego się w systemie węzła:

```
# These should not be left at default values, even for a demo.
# "PUBLIC" networking values are ones that end-users should be able to reach.
PUBLIC_HOSTNAME="node.domena.pl"      # The node host's public hostname
PUBLIC_IP="192.168.1.126"             # The node host's public IP address
BROKER_HOST="broker.domena.pl"         # IP or DNS name of broker server for REST API

# Usually (unless in a demo) this should be changed to the domain for your installation:
CLOUD_DOMAIN="domena.pl"              # Domain suffix to use for applications (Must match broker config)

# You may want these, depending on the complexity of your networking:
EXTERNAL_ETH_DEV='p2p1'                # Specify the internet facing public ethernet device
INTERNAL_ETH_DEV='eth1'                 # Specify the internal cluster facing ethernet device
```

5.2.4. Konfiguracja rejonu

Po skonfigurowaniu obu maszyn należy dokonać skonfigurować rejon, do którego będą należały oba węzły. Stworzenie rejonu oraz dodanie węzłów wymaga wykonania kilku kroków. Po pierwsze, należy zmienić pewne parametry w pliku `/etc/openshift/plugins.d/openshift-origin-msg-broker-mcollective.conf`, co umożliwi korzystanie z rejonów oraz nie pozwoli tworzyć kontenerów na węzłach nie należących do żadnego rejonu:

```
# Some settings to configure how mcollective handles gear placement on nodes:
```

```
# Use districts when placing gears and moving them between hosts. Should be  
# true except for particular dev/test situations.  
DISTRICTS_ENABLED=true
```

```
# Require new gears to be placed in a district; when true, placement will fail  
# if there isn't a district with capacity and the right gear profile.  
DISTRICTS_REQUIRE_FOR_APP_CREATE=true
```

```
# Used as the default max gear capacity when creating a district.  
DISTRICTS_MAX_CAPACITY=6000
```

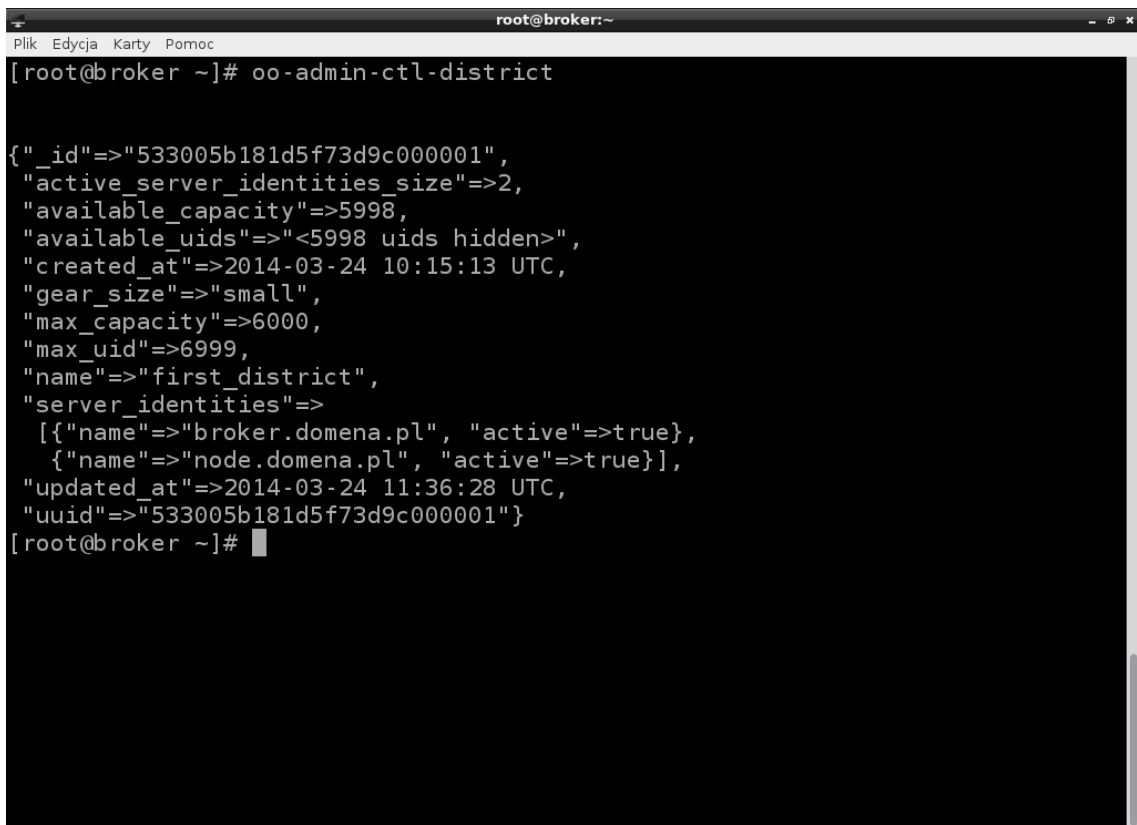
```
# It is unlikely these will need to be changed  
DISTRICTS_FIRST_UID=1000  
MCollective_Disctimeout=5  
MCollective_Timeout=180  
MCollective_Verbose=false  
MCollective_Progress_Bar=0  
MCollective_Config=/etc/mcollective/client.cfg  
MCollective_Fact_Timeout=10
```

```
# Place gears on nodes with the requested profile; should be true, as  
# a false value means gear profiles are ignored and gears are placed arbitrarily.  
Node_Profile_Enabled=true
```

Następnie należy stworzyć rejon oraz dodać do niego węzły. W tym celu należy zalogować się na system zarządcy oraz wydać następujące polecenia:

```
[#] oo-admin-ctl-district -c create -n first_district -p small  
[#] oo-admin-ctl-district -c add-node -n first_district -i broker.domena.pl  
[#] oo-admin-ctl-district -c add-node -n first_district -i node.domena.pl
```

Po wykonaniu każdego z powyższych poleceń powinny pojawić się informacje potwierdzające poprawne wykonanie operacji. Należy jednak raz jeszcze zweryfikować poprawność dodania węzłów do rejonu. Najprostszym sposobem jest wykorzystanie polecenia `oo-admin-ctl-district`. Poprawny wynik polecenia przedstawiony został na Rysunku 29.



```
root@broker:~  
Plik Edycja Karty Pomoc  
[root@broker ~]# oo-admin-ctl-district  
  
{  
  "_id"=>"533005b181d5f73d9c000001",  
  "active_server_identities_size"=>2,  
  "available_capacity"=>5998,  
  "available_uids"=>"<5998 uids hidden>",  
  "created_at"=>2014-03-24 10:15:13 UTC,  
  "gear_size"=>"small",  
  "max_capacity"=>6000,  
  "max_uid"=>6999,  
  "name"=>"first_district",  
  "server_identities"=>  
    [{  
      "name"=>"broker.domena.pl", "active"=>true},  
      {  
        "name"=>"node.domena.pl", "active"=>true}],  
  "updated_at"=>2014-03-24 11:36:28 UTC,  
  "uuid"=>"533005b181d5f73d9c000001"}  
[root@broker ~]#
```

Rysunek 29. Weryfikacja dodania węzłów do rejonu

5.3. Wdrożenie aplikacji testowych

Przetestowanie działania chmury wymaga stworzenia aplikacji. Do przetestowania chmury wykorzystane zostały dwie aplikacje, o nazwach `session` oraz `dbsession`. Obie aplikacje testowe realizują niezwykle prostą implementację koszyka z zakupami. Zostały one utworzone za pomocą języka PHP.

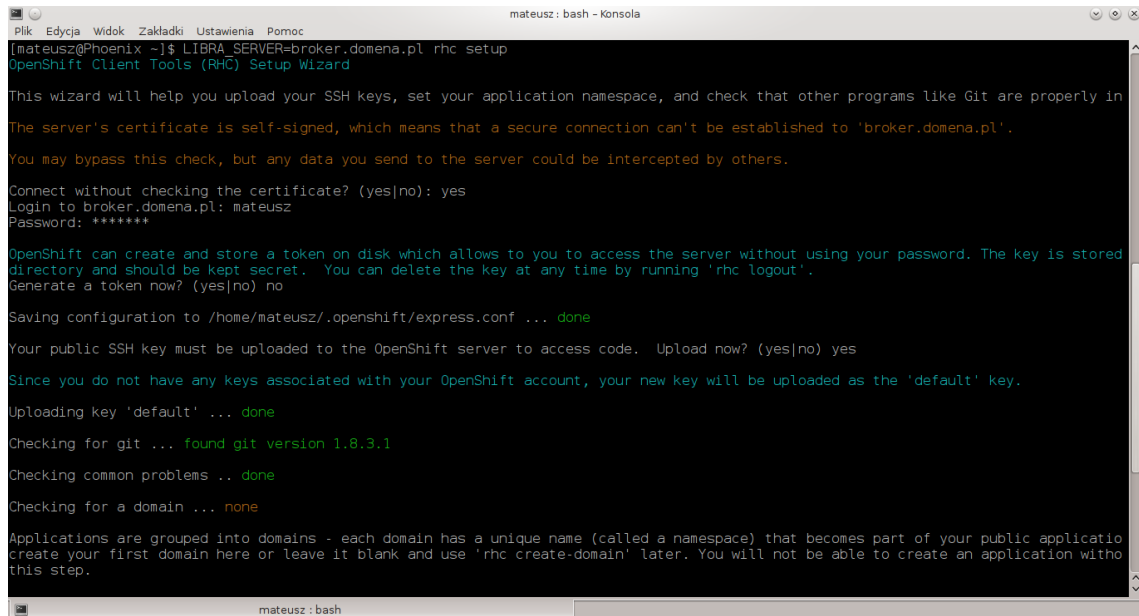
Narzędzie `rhc` służy do zarządzania aplikacjami umieszczonymi w chmurze OpenShift Origin z poziomu linii poleceń. Mimo początkowych trudności wynikających z nieznamomości poleceń, jest to najwygodniejsze narzędzie do zarządzania aplikacjami. Narzędzie instaluje się za pomocą polecenia:

```
[#] yum install rubygem-rhc
```

Następnie należy skonfigurować narzędzie do pracy. Domyślnie `rhc` łączy się serwerem uruchomionym przez firmę Red Hat. Należy jawnie wskazać serwer, z którym `rhc` ma się połączyć, za pomocą zmiennej `LIBRA_SERVER`:

[#] LIBRA_SERVER=broker.domena.pl rhc setup

Podczas konfiguracji narzędzia należy wgrać klucz SSH służący do bezpiecznego połączenia z kontem oraz podać nazwę domeny. Przebieg procesu przedstawiony został na Rysunkach 30 oraz 31.



```
mateusz@Phoenix ~]$ LIBRA_SERVER=broker.domena.pl rhc setup
OpenShift Client Tools (RHC) Setup Wizard

This wizard will help you upload your SSH keys, set your application namespace, and check that other programs like Git are properly in
The server's certificate is self-signed, which means that a secure connection can't be established to 'broker.domena.pl'.
You may bypass this check, but any data you send to the server could be intercepted by others.
Connect without checking the certificate? (yes|no): yes
Login to broker.domena.pl: mateusz
Password: *****

OpenShift can create and store a token on disk which allows to you to access the server without using your password. The key is stored
directory and should be kept secret. You can delete the key at any time by running 'rhc logout'.
Generate a token now? (yes|no) no

Saving configuration to /home/mateusz/.openshift/express.conf ... done

Your public SSH key must be uploaded to the OpenShift server to access code. Upload now? (yes|no) yes

Since you do not have any keys associated with your OpenShift account, your new key will be uploaded as the 'default' key.
Uploading key 'default' ... done

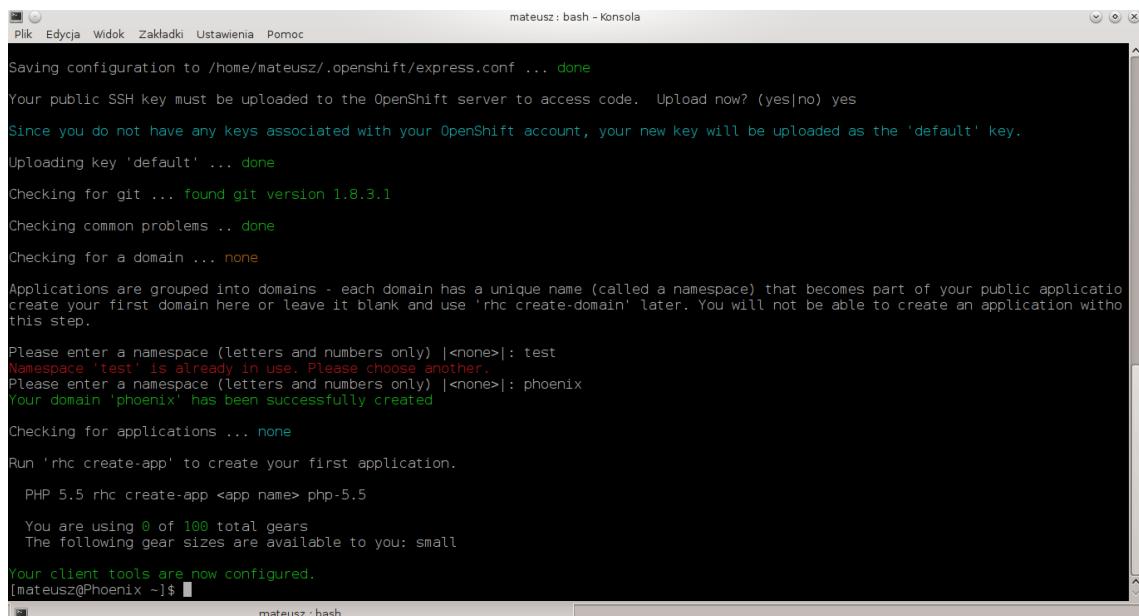
Checking for git ... found git version 1.8.3.1

Checking common problems .. done

Checking for a domain ... none

Applications are grouped into domains - each domain has a unique name (called a namespace) that becomes part of your public applicatio
create your first domain here or leave it blank and use 'rhc create-domain' later. You will not be able to create an application witho
this step.
```

Rysunek 30. Konfiguracja narzędzia rhc



```
mateusz@Phoenix ~]$ LIBRA_SERVER=broker.domena.pl rhc setup
OpenShift Client Tools (RHC) Setup Wizard

This wizard will help you upload your SSH keys, set your application namespace, and check that other programs like Git are properly in
The server's certificate is self-signed, which means that a secure connection can't be established to 'broker.domena.pl'.
You may bypass this check, but any data you send to the server could be intercepted by others.
Connect without checking the certificate? (yes|no): yes
Login to broker.domena.pl: mateusz
Password: *****

OpenShift can create and store a token on disk which allows to you to access the server without using your password. The key is stored
directory and should be kept secret. You can delete the key at any time by running 'rhc logout'.
Generate a token now? (yes|no) no

Saving configuration to /home/mateusz/.openshift/express.conf ... done

Your public SSH key must be uploaded to the OpenShift server to access code. Upload now? (yes|no) yes

Since you do not have any keys associated with your OpenShift account, your new key will be uploaded as the 'default' key.
Uploading key 'default' ... done

Checking for git ... found git version 1.8.3.1

Checking common problems .. done

Checking for a domain ... none

Applications are grouped into domains - each domain has a unique name (called a namespace) that becomes part of your public applicatio
create your first domain here or leave it blank and use 'rhc create-domain' later. You will not be able to create an application witho
this step.

Please enter a namespace (letters and numbers only) |<none>|: test
Namespace 'test' is already in use. Please choose another.
Please enter a namespace (letters and numbers only) |<none>|: phoenix
Your domain 'phoenix' has been successfully created

Checking for applications ... none

Run 'rhc create-app' to create your first application.

PHP 5.5 rhc create-app <app name> php-5.5

You are using 0 of 100 total gears
The following gear sizes are available to you: small

Your client tools are now configured.
[mateusz@Phoenix ~]$
```

Rysunek 31. Konfiguracja narzędzia rhc - dodanie domeny

Po tym kroku można przystąpić do tworzenia aplikacji. Do tworzenia aplikacji służy polecenie `rhc app-create`. Aby utworzyć aplikację o nazwie `session`, korzystając z kasety zapewniającej wsparcie dla PHP, należy wydać następujące polecenie:

```
[#] rhc app-create session php-5.5 -s
```

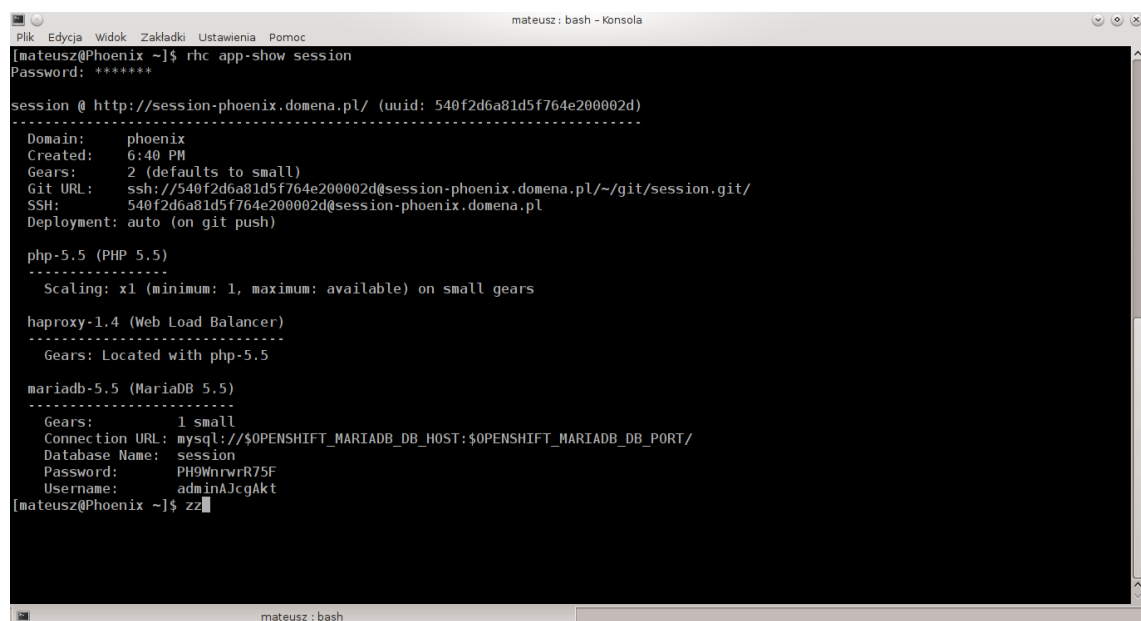
Następnym krokiem jest dodanie bazy danych. W tym celu należy dodać do aplikacji odpowiednią kasę. W celu dodania serwera baz danych MariaDB do aplikacji `session` należy wykorzystać następujące polecenie:

```
[#] rhc add-cartridge mariadb-5.5 --app session
```

Po obu powyższych poleceniach powinny pojawić się informacje o pozytywnym zakończeniu polecenia. Należy to zweryfikować za pomocą polecenia:

```
[#] rhc app-show session
```

Wynik polecenia powinien przypominać ten przedstawiony na Rysunku 32.



```
mateusz: bash - Konsola
Plik  Edycja  Widok  Zakładki  Ustawienia  Pomoc
[mateusz@Phoenix ~]$ rhc app-show session
Password: *****

session @ http://session-phoenix.domena.pl/ (uuid: 540f2d6a81d5f764e200002d)
-----
Domain:      phoenix
Created:     6:40 PM
Gears:       2 (defaults to small)
Git URL:     ssh://540f2d6a81d5f764e200002d@session-phoenix.domena.pl/~/.git/session.git/
SSH:         540f2d6a81d5f764e200002d@session-phoenix.domena.pl
Deployment:  auto (on git push)

php-5.5 (PHP 5.5)
-----
Scaling: x1 (minimum: 1, maximum: available) on small gears

haproxy-1.4 (Web Load Balancer)
-----
Gears: Located with php-5.5

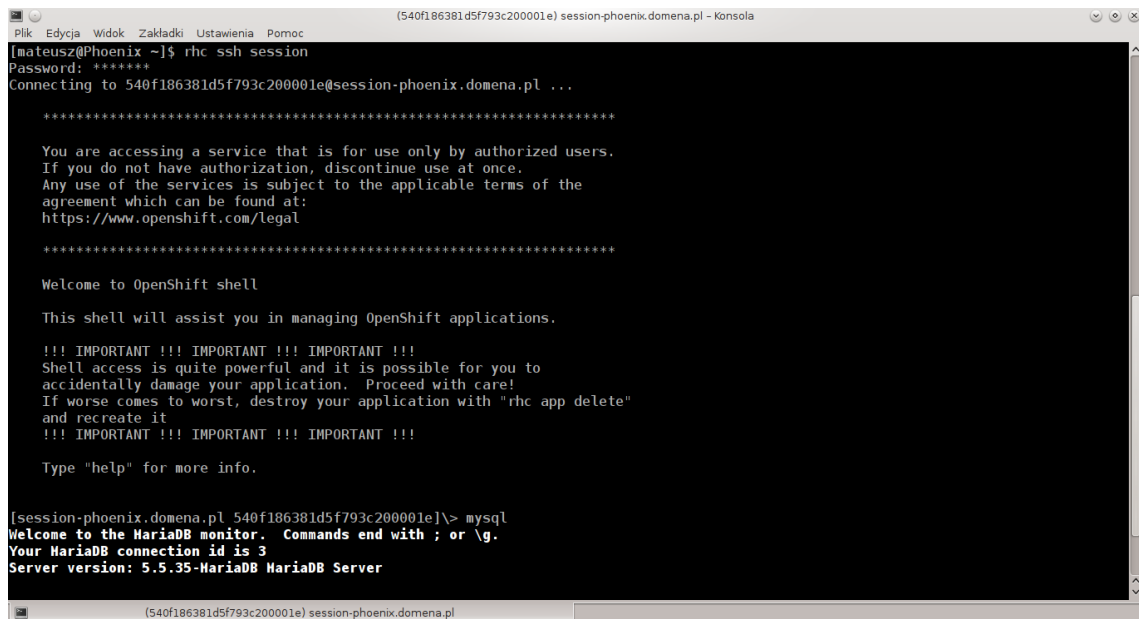
mariadb-5.5 (MariaDB 5.5)
-----
Gears:      1 small
Connection URL: mysql://$OPENSHIFT_MARIADB_DB_HOST:$OPENSHIFT_MARIADB_DB_PORT/
Database Name: session
Password:    PH9WnrwrR75F
Username:    adminA3cgAkt
[mateusz@Phoenix ~]$ zz
```

Rysunek 32. Informacje o utworzonej aplikacji

Następnym krokiem jest dodanie do bazy danych odpowiedniej tabeli. Baza danych została utworzona podczas dodawania kasety – nazwa bazy danych jest taka sama, jak nazwa aplikacji. W celu dodania tabeli do bazy danych należy zalogować się do kontenera za pomocą narzędzia `rhc`. Należy w tym celu wydać następujące polecenia:

```
[#] rhc ssh session
[#] mysql
```

Przebieg procesu został przedstawiony na Rysunku 33.



```
(540f186381d5f793c200001e) session-phenix.domena.pl - Konsola
Plik Edycja Widok Zakładki Ustawienia Pomoc
[mateusz@Phoenix ~]$ rhc ssh session
Password: *****
Connecting to 540f186381d5f793c200001e@session-phenix.domena.pl ...

*****

You are accessing a service that is for use only by authorized users.
If you do not have authorization, discontinue use at once.
Any use of the services is subject to the applicable terms of the
agreement which can be found at:
https://www.openshift.com/legal

*****

Welcome to OpenShift shell

This shell will assist you in managing OpenShift applications.

!!! IMPORTANT !!! IMPORTANT !!! IMPORTANT !!!
Shell access is quite powerful and it is possible for you to
accidentally damage your application. Proceed with care!
If worse comes to worst, destroy your application with "rhc app delete"
and recreate it
!!! IMPORTANT !!! IMPORTANT !!! IMPORTANT !!!

Type "help" for more info.

[session-phenix.domena.pl 540f186381d5f793c200001e]\> mysql
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 3
Server version: 5.5.35-MariaDB MariaDB Server
```

Rysunek 33. Logowanie się do kontenera za pomocą narzędzia rhc

Następnie należy przyłączyć się do bazy danych o nazwie session oraz dodać odpowiednią tabelę za pomocą poniższego polecenia:

```
[#] create table users(
    ID int not null auto_increment,
    name varchar(100) not null,
    surname varchar(100) not null,
    login varchar(100) not null unique,
    password varchar(100) not null,
    primary key (ID)
);
[#] insert into users(name, surname, login, password) values ("Mateusz", "Maciejewski",
"mateusz", "mateusz");
```

Ostatnim krokiem jest wdrożenie kodu aplikacji. Należy w tym celu umieścić kod aplikacji w katalogu php znajdującym się w katalogu session utworzonym podczas tworzenia aplikacji. Potem należy wykonać następujące polecenia:

```
[#] git add .
[#] git commit -m 'wdrozenie aplikacji'
[#] git push
```

Po wykonaniu powyższych poleceń aplikacja powinna być dostępna pod adresem session-phenix.domena.pl. Można to zweryfikować, wchodząc poprzez przeglądarkę

na podany adres. Powinna pojawić się strona przedstawiona na Rysunku 34.



Rysunek 34. Aplikacja testowa uruchomiona w chmurze

Wdrożenie drugiej aplikacji testowej przebiega w sposób analogiczny. Jedyną różnicą jest konieczność dodania do bazy danych drugiej tabeli, wykorzystywanej przez aplikację do przechowywania danych sesji. Należy ją stworzyć za pomocą następującego polecenia

```
[#] create table session_data(id varchar(32) primary key,  
    data text,  
    last_update timestamp  
);
```

5.3.1.1. Błąd związany z błędną konfiguracją zapory sieciowej

Podczas tworzenia aplikacji skalowalnych może pojawić się błąd przedstawiony na Rysunku 35. Jest on związany z niepoprawną konfiguracją zapory sieciowej. Aby mimo tego móc tworzyć aplikacje skalowalne, należy wykonać następujące kroki[32]:

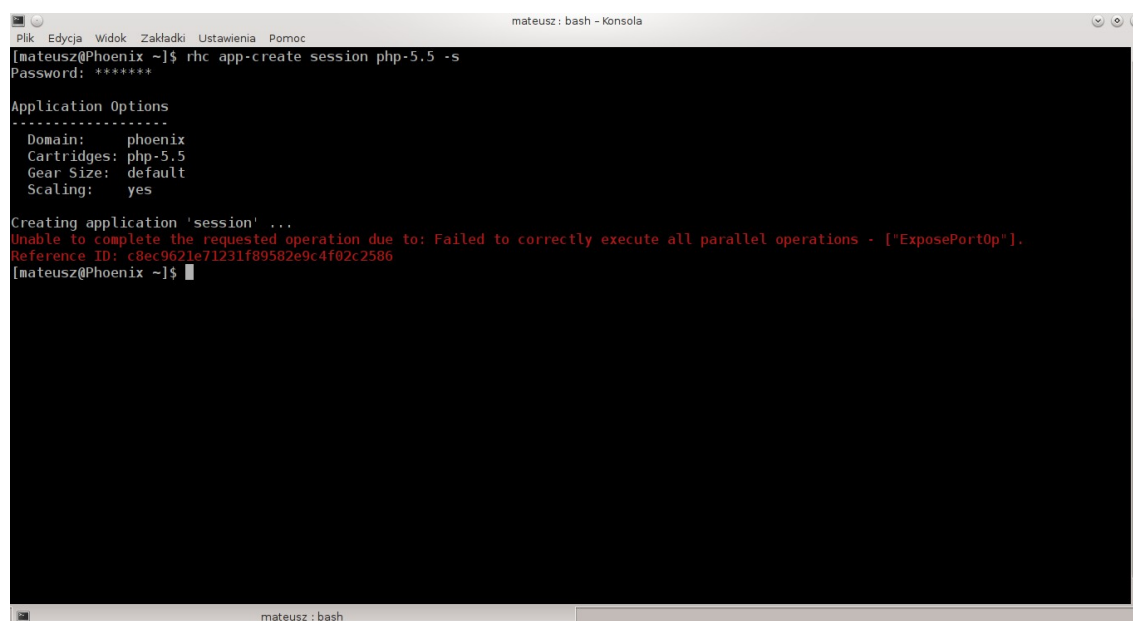
- do pliku `/etc/sysconfig/system-config-firewall` dodać następujące linijki:

```
--custom-rules=ipv4:filter:/etc/openshift/system-config-firewall  
--custom-rules=ipv4:filter:/etc/openshift/iptables.filter.rules  
--custom-rules=ipv4:nat:/etc/openshift/iptables.nat.rules
```

- wykonać poniższe polecenie

```
[#] lokkit --service=ssh
```

Powyższe kroki należy powtórzyć na każdym komputerze wchodzącym w skład chmury.



```
mateusz@Phoenix ~]$ rhc app-create session php-5.5 -s
Password: *****

Application Options
-----
Domain: phoenix
Cartridges: php-5.5
Gear Size: default
Scaling: yes

Creating application 'session' ...
Unable to complete the requested operation due to: Failed to correctly execute all parallel operations - ["ExposePortOp"].
Reference ID: c8ec9621e71231f89582e9c4f02c2586
[mateusz@Phoenix ~]$
```

Rysunek 35. Błąd podczas tworzenia aplikacji skalowalnej

5.4. Przenoszenie sesji HTTP pomiędzy instancjami aplikacji

Podczas pracy z wieloma instancjami aplikacji sieciowej bardzo często spotykanym problemem jest przenoszenie sesji HTTP pomiędzy instancjami. Sesja pozwala na rozwiązanie problemów wynikających z bezstanowej natury protokołu HTTP. Wszystkie informacje, które powinny zostać zachowane pomiędzy żądaniami (na przykład informacja o zalogowaniu użytkownika), zostają zapisane na serwerze. Na komputerze użytkownika zostaje umieszczony identyfikator sesji, pozwalający uzyskać dostęp do informacji zapisanych na serwerze. Identyfikator najczęściej zapisywany jest w niewielkim pliku, określanym mianem ciasteczka (ang. *cookie*)[33].

Podczas pracy z aplikacją umieszczoną na jednym serwerze obsługa sesji nie stanowi problemu. Sytuacja zmienia się, gdy żądania użytkowników obsługiwane są przez większą liczbę serwerów. Możliwa jest sytuacja, w której żądanie zostanie skierowane do serwera, który nie posiada informacji o sesji, którą została utworzona na innym serwerze obsługującym daną aplikację. Jednym z najłatwiej zauważalnych

objawów tego problemu jest sytuacja, w której uwierzytelniony użytkownik jest ponownie proszony o podanie danych identyfikacyjnych.

5.4.1. Dystrybucja żądań pomiędzy kontenerami

W OpenShift Origin za podział żądań pomiędzy kontenery zawierające aplikacje użytkownika odpowiada HAProxy. Poniżej znajduje się fragment pliku haproxy.cfg pochodzący z aplikacji testowej session

```
#-----
# common defaults that all the 'listen' and 'backend' sections will
# use if not designated in their block
#-----
defaults
    mode                http
    log                 global
    option              httplog
    option              dontlognull
    option http-server-close
    #option forwardfor   except 127.0.0.0/8
    option              redispatch
    retries              3
    timeout http-request 10s
    timeout queue        1m
    timeout connect      10s
    timeout client       1m
    timeout server       1m
    timeout http-keep-alive 10s
    timeout check        10s
    maxconn             128

listen stats 127.8.148.131:8080
    mode http
    stats enable
    stats uri /

listen express 127.8.148.130:8080

    cookie GEAR insert indirect nocache
    option httpchk GET /

    balance leastconn
    server local-gear 127.8.148.129:8080 check fall 2 rise 3 inter 2000 cookie local-540f2d6a81d5f764e200002d
```

Słowo kluczowe **balance** wskazuje algorytm, według którego HAProxy będzie rozdzielać żądania pomiędzy kontenery. W powyższym przypadku wybrany został algorytm nakazujący wybierać najmniej obciążony kontener. Słowo kluczowe **cookie** wraz opcjami zapewnia, że kolejne żądania pochodzące od tego samego klienta będą kierowane do tego kontenera, który został wybrany do obsłużenia pierwszego żądania

pochodzącego od klienta. Zapewnia to, że klient będzie miał dostęp do sesji, która została utworzona – ponieważ żądania kierowane są do tego samego kontenera, dane sesji są zawsze dostępne. Problem może się pojawić, jeżeli kontener zawierający dane sesji ulegnie wyłączeniu bądź awarii.

5.4.2. Przenoszenie sesji pomiędzy kontenerami

Do przetestowania procesu przenoszenia sesji pomiędzy kontenerami w chmurze OpenShift Origin wykorzystana zostanie aplikacja testowa o nazwie session. Wykorzystuje ona standardowy sposób przechowywania danych sesji, w którym dane sesji przechowywane są w lokalnej pamięci instancji. Pozwala to sprawdzić, czy serwer chmurowy zapewnia przenoszenie sesji pomiędzy kontenerami.

Przetestowanie przenoszenia sesji pomiędzy kontenerami wymaga pewnych przygotowań. Po pierwsze, należy zapewnić, że aplikacja wykorzystuje co najmniej dwa kontenery. OpenShift Origin zapewnia zwiększenie ilości kontenerów obsługujących aplikację skalowalną w chwili, gdy liczba żądań osiągnie zdefiniowany próg. Ilość kontenerów można również zwiększyć ręcznie za pomocą poniższych poleceń:

```
[#] rhc ssh session
[#] add-gear -a session -u 540f2d6a81d5f764e200002d -n phoenix
[#] exit
```

Następnie należy zgromadzić informacje o identyfikatorach kontenerów, węzłach na których zostały umieszczone oraz przydzielonych im adresach IP (ze względu na sposób działania serwera OpenShift Origin, będą to adresy zakresu 127.0.0.0/8). Dwie pierwsze informacje potrzebne są do testowego wyłączenia jednego z kontenerów. Adres IP potrzebny jest do rozpoznania, który kontener obsługuje żądanie. Informacje te można zdobyć za pomocą konsoli administracyjnej oraz poniższego polecenia:

```
[#] rhc app show session --gears
```

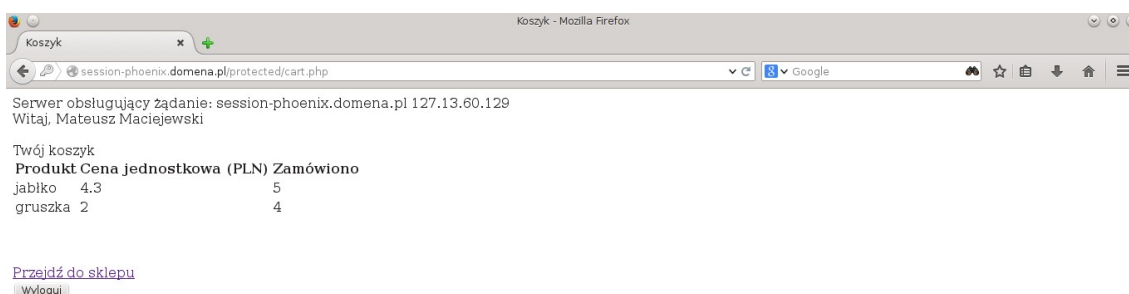
Potrzebne informacje zostały zebrane w Tabeli 12.

Tabela 12. Informacje o kontenerach zawierających aplikację session

Identyfikator kontenera	Węzeł	Adres IP
540f2d6a81d5f764e200002d	broker.domena.pl	127.8.148.129

5410947881d5f75320000002	node.domena.pl	127.13.60.129
--------------------------	----------------	---------------

Pierwszym krokiem testu jest nawiązanie połączenia z kontenerem, który został utworzony za pomocą polecenia add-gear. Trzeba w tym celu wykorzystać metodę prób i błędów, gdyż użytkownik nie ma wpływu, który kontener zostanie wyznaczony przez HAProxy do obsługi żądania. Po przekazaniu żądania do odpowiedniego kontenera należy zalogować się do aplikacji. Po zalogowaniu i „włożeniu” kilku produktów do „koszyka” aplikacja znajdzie się w stanie pozwalającym na przeprowadzenie testu. Stan ten przedstawiony jest Rysunku 36.



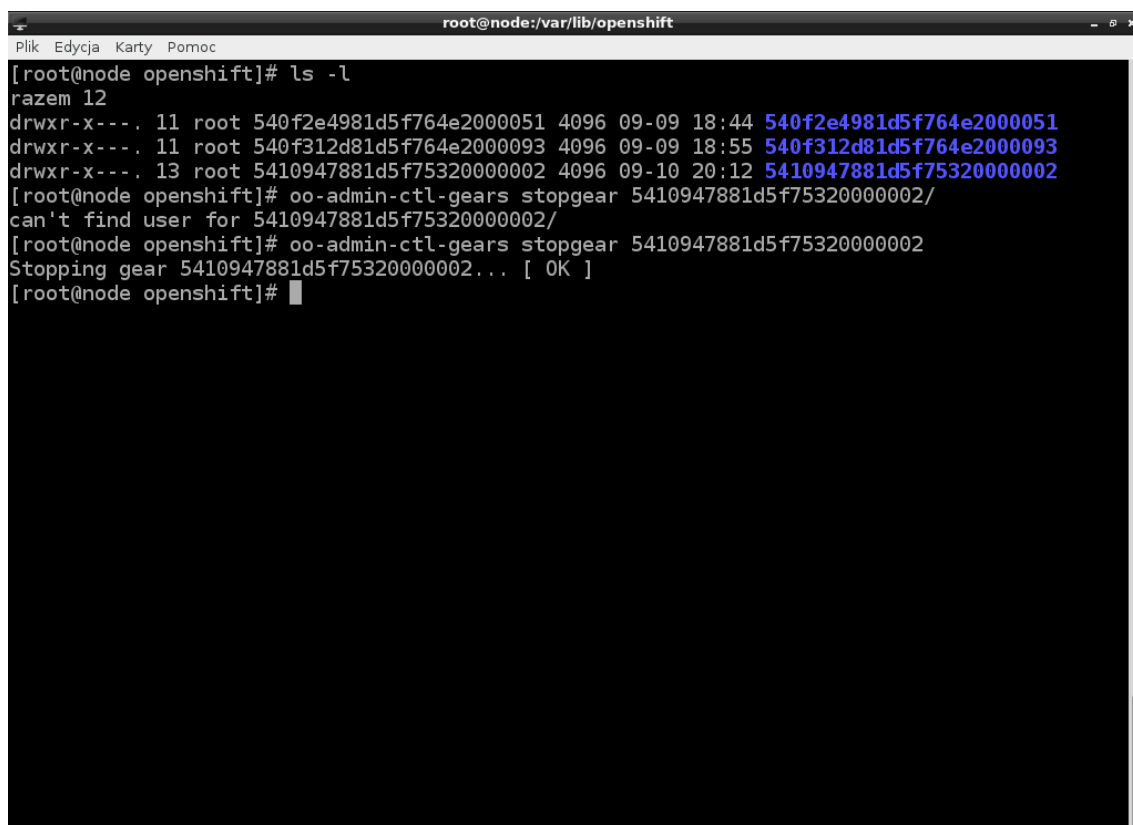
Rysunek 36. Stan aplikacji testowej session w chwili rozpoczęcia testu

Strona, do której prowadzi odnośnik „Przejdź do sklepu”, ukazany na powyższym rysunku, dostępna jest wyłącznie po uwierzytelnieniu. Jeżeli użytkownik nieuwierzytelniony spróbuje wejść na omawianą stronę, zostanie przekierowany na stronę informującą o konieczności zalogowania.

Następnie należy wyłączyć kontener, który przechowuje dane sesji. Należy w tym celu zalogować się na odpowiedni węzeł i wykonać następujące polecenie:

```
[#] oo-admin-ctl-admin 5410947881d5f75320000002
```

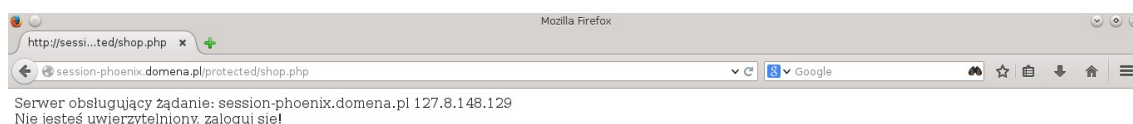
Poprawny przebieg operacji wyłączenia kontenera przedstawiony został na Rysunku 37.



```
root@node:/var/lib/openshift
Plik Edycja Karty Pomoc
[root@node openshift]# ls -l
razem 12
drwxr-x---. 11 root 540f2e4981d5f764e2000051 4096 09-09 18:44 540f2e4981d5f764e2000051
drwxr-x---. 11 root 540f312d81d5f764e2000093 4096 09-09 18:55 540f312d81d5f764e2000093
drwxr-x---. 13 root 5410947881d5f75320000002 4096 09-10 20:12 5410947881d5f75320000002
[root@node openshift]# oo-admin-ctl-gears stopgear 5410947881d5f75320000002/
can't find user for 5410947881d5f75320000002/
[root@node openshift]# oo-admin-ctl-gears stopgear 5410947881d5f75320000002
Stopping gear 5410947881d5f75320000002... [ OK ]
[root@node openshift]#
```

Rysunek 37. Wyłączenie kontenera aplikacji session

Po wyłączeniu kontenera należy wrócić do aplikacji oraz spróbować dostać się na stronę zarezerwowaną dla uwierzytelnionych użytkowników za pomocą odpowiedniego odnośnika. Wynik próby przedstawiony został na Rysunku 38.



Rysunek 38. Stan aplikacji session po wyłączeniu kontenera

Zmiana wyświetlanego adresu IP świadczy o tym, że żądanie zostało obsłużone przez drugi kontener. Informacja o konieczności zalogowania świadczy o tym, że sesja HTTP nie została przeniesiona. Po wyłączeniu kontenera przechowującego dane sesji, obsługę żądań użytkownika przejął drugi kontener, nie posiadający żadnych danych o dotychczasowych poczynaniach użytkownika.

5.4.3. Przenoszenie sesji HTTP za pośrednictwem bazy danych

Często spotykanym rozwiązaniem problemu przenoszenia sesji HTTP jest przechowywanie danych sesji w bazie danych, do której dostęp mają wszystkie instancje aplikacji. Zapewnia to, że każda instancja aplikacji będzie w stanie poprawnie obsłużyć żądanie.

Do przetestowania wyżej opisanego rozwiązania wykorzystana zostanie aplikacja testowa o nazwie dbsession. Ta aplikacja testowa wykorzystuje mechanizm niestandardowej obsługi sesji, oferowany programiście przez język PHP. Umożliwia to przechowywanie danych sesji, które dzięki tej operacji powinny być dostępne dla obu kontenerów zawierających aplikację testową.

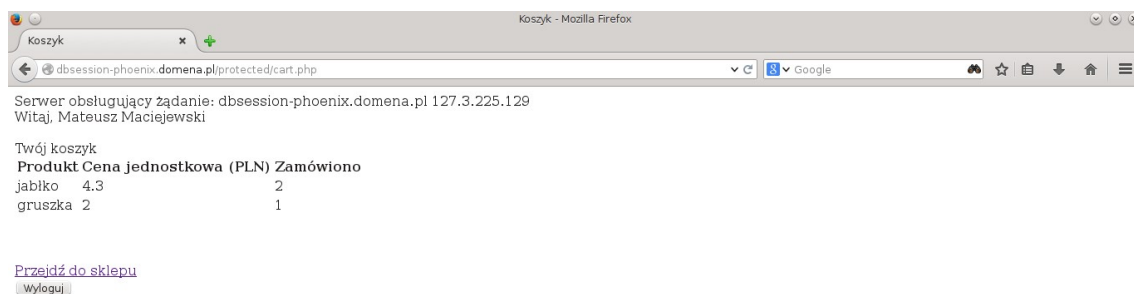
Do przeprowadzenia testu potrzebne są takie same informacje, jak te zgromadzone w celu przetestowania procesu przenoszenia sesji w aplikacji session. Informacje te można uzyskać w sposób analogiczny do opisanego wcześniej. Niezbędne informacje zostały przedstawione w Tabeli 13.

Tabela 13. Informacje o kontenerach zawierających aplikację dbsession

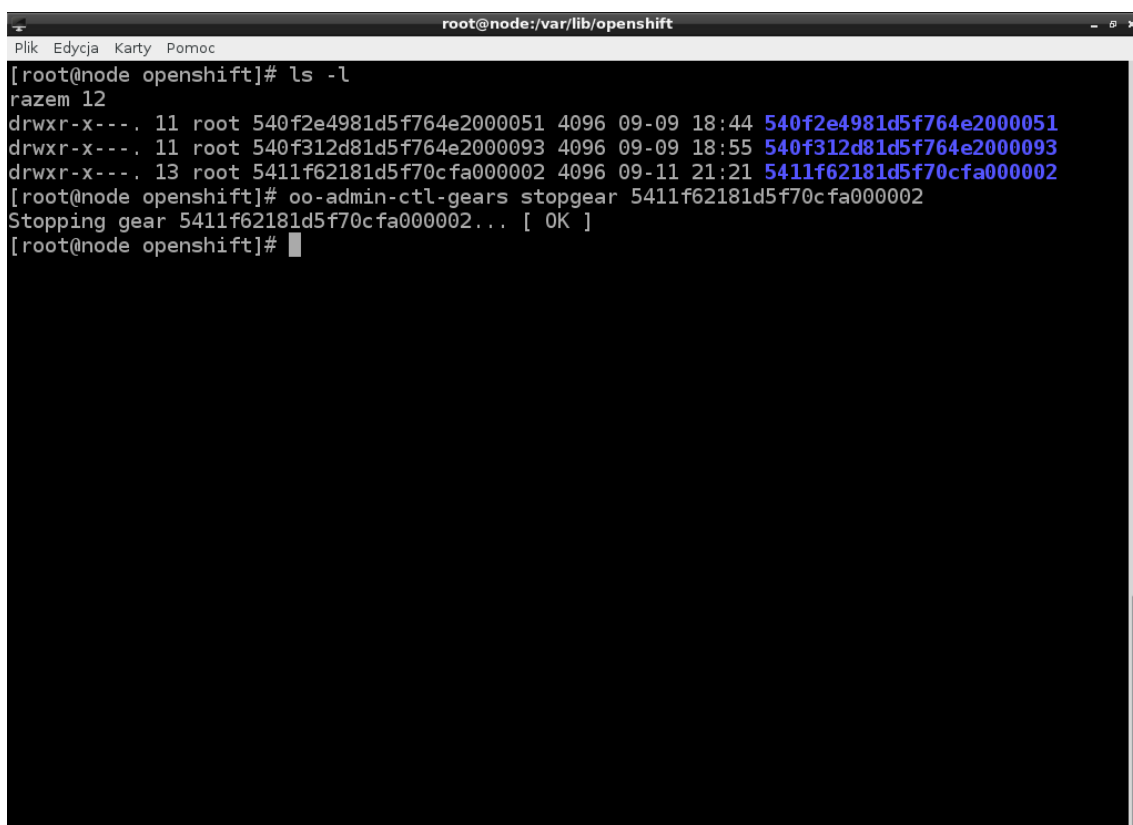
Identyfikator kontenera	Węzeł	Adres IP
540f30e481d5f764e200006f	broker.domena.pl	127.13.31.129
5411f62181d5f70cfa000002	node.domena.pl	127.3.225.129

Przebieg testu również nie różni się od wcześniej opisanej procedury. Stan aplikacji po nawiązaniu połączenia z odpowiednim kontenerem i wygenerowaniu stanu pozwalającego na przeprowadzenie testu został przedstawiony na Rysunku 39.

Następnie należy wyłączyć kontener obsługujący dotychczas żądania użytkownika. Proces ten został przedstawiony na Rysunku 40.

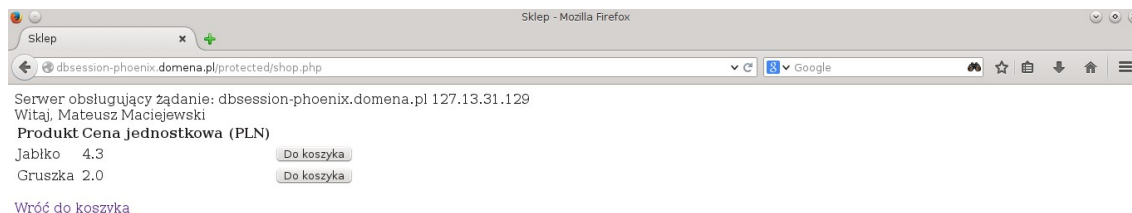


Rysunek 39. Stan aplikacji testowej dbsession w chwili rozpoczęcia testu



Rysunek 40. Wyłączenie kontenera aplikacji dbsession

Ostatnim krokiem jest próba przejścia na stronę zarezerwowaną dla uwierzytelnionego użytkownika. Wynik próby przedstawiony jest na Rysunku 41.



Rysunek 41. Stan aplikacji dbsession po wyłączeniu kontenera

Podobnie jak w przypadku aplikacji session, zmiana adresu świadczy o tym, że żądania obsługiwane są przez drugi kontener. Jak widać na Rysunku 41, próba przejścia na stronę zarezerwowaną dla uwierzytelnionego użytkownika zakończyła się powodzeniem. Informacja o uwierzytelnieniu została poprawnie zinterpretowana przez kontener, który wcześniej nie obsługiwał żądań użytkownika, w szczególności tych odpowiedzialnych za proces uwierzytelnienia.

6. Podsumowanie

Rozwiązania chmurowe stają się coraz popularniejsze. Firmy oraz osoby prywatne coraz częściej decydują się na wykorzystanie możliwości oferowanych przez chmury. Cechy chmur obliczeniowej, takie jak natychmiastowy oraz powszechny dostęp do zasobów, bądź współdzielenie dostępnych zasobów pomiędzy różnych użytkowników pozwalają na lepsze wykorzystanie mocy obliczeniowej komputerów wchodzących w skład chmury.

Na rynku można znaleźć kilka rozwiązań chmurowych należących do kategorii wolnego oprogramowania. Każda z nich wspiera inne języki programowania oraz inne technologie niezbędne do tworzenia nowoczesnych aplikacji, takie jak serwery baz danych. Zdecydowana większość rozwiązań pozwala na tworzenie rozszerzeń, zwiększających ich możliwości. Problemem podczas wykorzystywania chmury obliczeniowej może okazać się brak standardów. Każdy twórca rozwiązania chmurowego realizuje je w indywidualny sposób, co ogranicza w sposób znaczący możliwość współdziałania obecnych chmur obliczeniowych.

OpenShift Origin jest jedną z najlepszych obecnie dostępnych chmur typu Platform-as-a-Service. Wspiera dużą ilość języków oraz narzędzi programistycznych. Potrafi również dynamicznie zmieniać ilość zasobów przydzielonych aplikacji, w zależności od stopnia jej obciążenia. Omawiana chmura nie wspiera jednak przenoszenia sesji HTTP ani żadnych innych informacji pomiędzy instancjami aplikacji. To zagadnienie musi zostać rozwiązane przez twórcę aplikacji umieszczanej w chmurze.

Uruchomiona chmura, mimo ograniczeń związanych zarówno z funkcjonalnością, jak i ilością dostępnych zasobów fizycznych, w pełni realizuje założenia, które zostały przedstawione we wstępie pracy.

7. Bibliografia

- [1] Główny Urząd Statystyczny. Społeczeństwo informacyjne w Polsce, 2013.
- [2] United Nations Conference on Trade and Development. Information Economy Report, 2013.
- [3] Erl, Thomas. Cloud Computing: Concepts, Technology & Architecture. Prentice Hall, 2013.
- [4] Buyya, Rajkumar. Cloud Computing: Principles and Paradigms. John Wiley & Sons, Inc, 2011.
- [5] McGrath, Mike. Understanding PaaS. O'Reilly Media, 2012.
- [6] Strona projektu Cloud Foundry. [Online]
<http://www.cloudfoundry.org/about/index.html>.
- [7] Dokumentacja projektu Cloud Foundry. [Online]
<http://docs.cloudfoundry.org/buildpacks/>.
- [8] Blog projektu Paasmaker. [Online] <http://blog.paasmaker.org/>.
- [9] Dokumentacja projektu Paasmaker. [Online]
<http://docs.paasmaker.org/introduction.html>.
- [10] Strona projektu Paasmaker. [Online] <http://paasmaker.org/>.
- [11] Blog projektu Tsuru. [Online] <http://blog.tsuru.io/>.
- [12] Dokumentacja projektu Tsuru. [Online] <http://docs.tsuru.io/en/latest/overview.html>.
- [13] Dokumentacja projektu Tsuru. [Online] <http://docs.tsuru.io/en/latest/why.html>.
- [14] Strona firmy Red Hat. [Online]
<http://www.redhat.com/about/news/archive/2012/4/Announcing-OpenShift-Origin-Open-Source-Code-For-Platform-as-a-Service>.
- [15] Dokumentacja projektu OpenShift Origin. [Online]
https://github.com/openshift/origin-server/blob/openshift-origin-release-3/documentation/oo_cartridge_guide.adoc.
- [16] Miller, Adam. Implementing OpenShift. Packt Publishing, 2013.
- [17] Rhett, Jo. Learning MCollective. O'Reilly Media, 2014.
- [18] Dokumentacja projektu OpenShift Origin. [Online]
https://github.com/openshift/origin-server/blob/openshift-origin-release-3/documentation/oo_system_architecture_guide.adoc.
- [19] Strona z podręcznikami do poleceń systemu Linux. [Online]
<http://linux.die.net/man/1/curl>.
- [20] Dokumentacja projektu OpenShift Origin. [Online]
https://github.com/openshift/origin-server/blob/openshift-origin-release-3/documentation/oo_administration_guide.adoc.
- [21] Dokumentacja projektu OpenShift Origin. [Online]
https://github.com/openshift/origin-server/blob/openshift-origin-release-3/documentation/oo_user_guide.adoc.
- [22] Barrett, Daniel. SSH, The Secure Shell The Definitive Guide, 2nd Edition. O'Reilly Media, 2005.
- [23] Chacon, Scott. Pro Git. Apress, 2009.
- [24] Geisshirt, Kenneth. Pluggable Authentication Modules: The Definitive Guide to PAM for Linux SysAdmins and C Developers. Packt Publishing, 2007.
- [25] Vermeulen, Sven. SELinux System Administration. Packt Publishing, 2013.
- [26] Dokumentacja systemu Red Hat Enterprise Linux. [Online]

- https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Resource_Management_Guide/ch01.html.
- [27] Aitchison, Ron. Pro DNS and BIND 10. Apress, 2011.
- [28] Chodorow, Kristina. MongoDB: The Definitive Guide, 2nd Edition. O'Reilly Media, 2013.
- [29] Strona projektu HAProxy. [Online] <http://www.haproxy.org/>.
- [30] Negus, Christopher. Red Hat Linux 9 Biblia. Helion, 2003.
- [31] Dokumentacja projektu OpenShift Origin https://github.com/openshift/origin-server/blob/openshift-origin-release-3/documentation/oo_deployment_guide_comprehensive.adoc.
- [32] BugZilla. [Online] https://bugzilla.redhat.com/show_bug.cgi?id=1027122.
- [33] Coggeshall, John. PHP5. Księga eksperta. Helion, 2005.

8. Spis rysunków

Rysunek 1. Przegląd architektury OpenShift Origin.....	18
Rysunek 2. Komunikacja z zarządcą za pomocą narzędzia curl.....	20
Rysunek 3. Struktura katalogów w kontenerze.....	21
Rysunek 4. Powiązanie portu kontenera z portem węzła.....	22
Rysunek 5. Aplikacja w systemie OpenShift Origin.....	24
Rysunek 6. Tworzenie aplikacji.....	25
Rysunek 7. Schemat aplikacji skalowalnej.....	28
Rysunek 8. Architektura SSH.....	29
Rysunek 9. Rozproszony system kontroli wersji.....	31
Rysunek 10. Zapis wersji pliku w Gicie.....	31
Rysunek 11. Schemat działania PAM.....	33
Rysunek 12. Przypisanie podsystemów do hierarchii.....	34
Rysunek 13. Przypisanie podsystemu do hierarchii związanej z innym podsystemem.....	35
Rysunek 14. Przynależność zadania do grup.....	35
Rysunek 15. Przynależność zadania potomnego.....	36
Rysunek 16. Przykładowy kontekst SELinuxa.....	37
Rysunek 17. Hierarchia domen w systemie DNS.....	39
Rysunek 18. Model komunikacji wykorzystywany przez MCollective.....	42
Rysunek 19. Położenie HAProxy w architekturze.....	43
Rysunek 20. Weryfikacja konfiguracji serwera DNS.....	50
Rysunek 21. Weryfikacja konfiguracji MongoDB.....	52
Rysunek 22. Weryfikacja konfiguracji ActiveMQ.....	54
Rysunek 23. Konsola administracyjna.....	60

Rysunek 24. Konsola użytkownika.....	62
Rysunek 25. Weryfikacja konfiguracji SSH.....	63
Rysunek 26. Weryfikacja konfiguracji MCollective.....	65
Rysunek 27. Weryfikacja rozwiązania nazwy węzła przez serwer DNS.....	72
Rysunek 28. Weryfikacja konfiguracji SSH.....	73
Rysunek 29. Weryfikacja dodania węzłów do rejonu.....	75
Rysunek 30. Konfiguracja narzędzia rhc.....	76
Rysunek 31. Konfiguracja narzędzia rhc - dodanie domeny.....	76
Rysunek 32. Informacje o utworzonej aplikacji.....	77
Rysunek 33. Logowanie się do kontenera za pomocą narzędzia rhc.....	78
Rysunek 34. Aplikacja testowa uruchomiona w chmurze.....	79
Rysunek 35. Błąd podczas tworzenia aplikacji skalowalnej.....	80
Rysunek 36. Stan aplikacji testowej session w chwili rozpoczęcia testu.....	83
Rysunek 37. Wyłączenie kontenera aplikacji session.....	84
Rysunek 38. Stan aplikacji session po wyłączeniu kontenera.....	84
Rysunek 39. Stan aplikacji testowej dbsession w chwili rozpoczęcia testu.....	86
Rysunek 40. Wyłączenie kontenera aplikacji dbsession.....	86
Rysunek 41. Stan aplikacji dbsession po wyłączeniu kontenera.....	87

9. Spis tabel

Tabela 1. Polskie odpowiedniki sformułowań występujących w literaturze anglojęzycznej.....	6
Tabela 2. Lista paczek dostępnych w standardowej instancji serwera Cloud Foundry...	14
Tabela 3. Zawartość katalogów kontenera.....	21
Tabela 4. Struktura katalogów w repozytorium.....	26
Tabela 5. Podstawowe zmienne środowiskowe.....	26
Tabela 6. Najczęściej spotykane role SELinuxa.....	37
Tabela 7. Opis konfiguracji pierwszej maszyny wirtualnej.....	45
Tabela 8. Opis konfiguracji drugiej maszyny wirtualnej.....	45
Tabela 9. Szablony plików konfiguracyjnych dla ActiveMQ.....	53
Tabela 10. Zmienne SELinuxa niezbędne dla działania zarządcy.....	56
Tabela 11. Zmienne SELinuxa niezbędne dla działania węzła.....	69
Tabela 12. Informacje o kontenerach zawierających aplikację session.....	82
Tabela 13. Informacje o kontenerach zawierających aplikację dbsession.....	85

Łódź, dnia roku

.....
(imię i nazwisko Studenta)

.....
(adres)

.....
(numer albumu)

.....
(wydział)

.....
(kierunek studiów)

.....
(rodzaj i forma studiów)

Oświadczenie

Świadomy/a odpowiedzialności karnej za składanie fałszywych zeznań oświadczam, że przedstawiona praca licencjacka / inżynierska / magisterska ¹ na temat

.....
.....
.....

została napisana przeze mnie samodzielnie.

Jednocześnie oświadczam, że ww. praca

- nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i praw pokrewnych (j.t. Dz. U. z 2006 r. Nr 90, poz. 631, z późn. zm.) oraz dóbr osobistych chronionych prawem cywilnym, a także nie zawiera danych i informacji, które uzyskałem/am w sposób niedozwolony,
- nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadawaniem dyplomów wyższej uczelni lub tytułów zawodowych.

.....
(Czytelny podpis Studenta)

¹Niepotrzebne skreślić

Łódź, dnia roku

.....
(imię i nazwisko Studenta)

.....
(adres)

.....
(numer albumu)

.....
(wydział)

.....
(kierunek studiów)

.....
(rodzaj i forma studiów)

Oświadczenie

Świadomy/a odpowiedzialności karnej za składanie fałszywych zeznań oświadczam, że przedstawiona na nośniku CD/DVD praca licencjacka / inżynierska / magisterska ² na temat

.....
.....
.....

zawiera te same treści, co oceniany przez Opiekuna pracy i Recenzenta wydruk komputerowy.

.....
(Czytelny podpis Studenta)

²Niepotrzebne skreślić