Załącznik nr 1. do Zapytania ofertowego 1/10/2018

Dotyczy: Projekt "Opracowanie w ramach zaawansowanych prac rozwojowych innowacyjnej w skali świata Niezawodnej Efektywnej Macierzy Otwartej (NEMO) do bezpiecznego przechowywania i przetwarzania dużej ilości danych elektronicznych (Big Data)" nr POIR.01.01.02-00-0082/16 współfinansowany ze środków Europejskiego Funduszu Rozwoju Regionalnego.

Szczegółowy Opis Przedmiotu Zamówienia

I. Podstawowym celem projektu jest zakup wartości niematerialnych i prawnych - funkcjonalności oraz integracji z aplikacjami zewnętrznymi do rozproszonego systemu plików LizardFS.

II. Funkcjonalności zostały podzielone na następujące elementy:

A. GUI (Graphical User Interface)
Moduł musi być napisany w języku python 2.7 (aktualna minimalna wersja dostępna w dystrybucjach Centos 7, Redhat 6 i 7.

Odtworzenie funkcjonalności aktualnego cgi LizardFS.
Funkcjonalność logowania i zarządzania użytkownikami posiadającymi dostęp do GUI.

Rozszerzenie funkcjonalność GUI z dostępem po RESTful API o:

1. odczytywanie listy katalogów, plików
2. możliwość ustawienia goali dla plików i katalogów
3. odtworzenie wszystkich funkcji lizardfs-admin
4. ustawienia quot i wszystkich funkcji dostępnych w lizardfs-tools
   a) tools:
      (1) getgoal
      (2) setgoal
      (3) gettrashtime
      (4) settrashtime
      (5) geteattr
      (6) seteattr
      (7) deleattr
      (8) checkfile
      (9) fileinfo
      (10) appendchunks
      (11) dirinfo

(12) filerepair
(13) makesnapshot
(14) repquota
(15) setquota
(16) rremove
(17) help [tool name]

5. możliwość zmiany labeli dla poszczególnych chunkserwerów,
6. funkcjonalności meta (trash recovery / delete)
7. edycja plików konfiguracyjnych,
8. dodawanie lub usuwanie chunk serwerów
9. na żądanie generowanie raportu przesyłanego do producenta oprogramowania w celu identyfikacji przyczyn problemów

B. Integracja z HashiCorp Terraform
W wyniku przeprowadzonych prac po zdefiniowaniu elementów infrastruktury z poziomu Terraform bedzie możliwe automatyczne przydzielenie zasobów przestrzeni dyskowej z LizardFS.

Sterownik będzie umożliwiał następujące funkcjonalności:
- Create a share (stworzenie zasobu)
- Delete a share (wykasowanie zasobu)
- Allow share access (pozwól na dostęp do zasobu)
- Deny share access (odmów dostępu do zasobu)

C. Sterownik do współdzielonych systemów plików OpenStack
W wyniku przeprowadzonych prac po zdefiniowaniu elementów infrastruktury z poziomu OpenStack bedzie możliwe automatyczne przydzielenie zasobów przestrzeni dyskowej z LizardFS.

Sterownik będzie umożliwiał następujące funkcjonalności:
- Create a share (stworzenie zasobu)
- Delete a share (wykasowanie zasobu)
- Allow share access (pozwól na dostęp do zasobu)
- Deny share access (odmów dostępu do zasobu)

D. Sterownik do Kubernetes
Wynikiem prac będzie sterownik, który będzie umożliwiał alokację zasobów dyskowych klastra LizardFS do PODów Kubernetes. Zasoby dyskowe będą pełnić rolę Persistent Volumes. Polecenie PersistentVolumeClaim z poziomu Kubernetes będzie powodowało przydzielenie zasobów. Podłaczenie zasobów będzie możliwe zarówno w trybie tylko do oczytu jak i odczytu i

zapisu. Bedzie możliwe ustawienie takich parametrów jak QoS (Quality of Service) czy Quota. Jednym z rezultatów prac będzie również dodanie LizardFS do Kubernetes Storage Classes.
Status zasobów LizardFS będzie widoczny z poziomu Kubernetes (Pending, Running, Terminating and Finalizers. Działać będą wszystkie funkcjonalności takie jak Reclaiming, Retain, Delete, Recycle. Będzie możliwe rozszerzanie Persistent Volumes Claims również w locie. Status będzie widoczny po wpisaniu komendy kubectl describe pvc.
Typy dostępu:

- ReadWrtieOnce - zasób może byc podmontowany przez pojedynczy węzeł z prawem zapisu i odczytu
- ReadOnlyMany - zasób może być podmontowany w trybie odczytu przez wiele węzłów
- ReadWrtieMany - zasób może być podmontowany w trybie zapisu i odczytu przez wiele węzłów

Bedą wspierane opcje montowania (Mount Options). Administrator będzie mógł zdecydować o opcjach montowania także po zamontowaniu zasobu.
Z poziomu Kubernetes będzie również widoczna "faza" (Phase) - Available (zasób wolny), Bound (zasób zarezerwowany), Released (rezerwacja została odwołana, ale zasób nie został przejęty przez klaster), Failed (automatyczne przejęcie zasobu nie zadziałało).
Bedzie istaniała możliwość filtrowania zasobów za pomocą matchLabels i marchExpressions. Będą działać funkcje Volume Snapshot i Restore Volume.

E. Uwierzytelnianie chunk serwera
W celu wyeliminowania spoofingu chunkserverów w klastrze, lub przypadkowego podłączenia chunkservera z innego klastra dopisana zostanie metoda autoryzujaca dany chunkserver do pracy w przeznaczonym dla niego klastrze. Metoda wykorzysytwac będzie kryptograficzne metody szyfrowania symetrycznego, oraz asymetrycznego, między innymi w oparciu o klucze prywatne i publiczne certyfikatów.

F. Uwierzytelnianie klienta
Analogicznie do autoryzacji chunkserwerów klient klastra będzie za pomocą kluczy publicznych / prywatnych i/lub kryptograficznych symetrycznych metod szyfrowania weryfikowany pod kątem możliwości połączenia z danym klastrem. Będzie to podwyższało bezpieczeństwo klastra, w szczególności umożliwiając odłączenie po stronie mastera skompromitowanych klientów w sieci, bez potrzeby informacji o aktualnym adresie IP / hostname, lub innych informacjach o tym kliencie, jedyną operacją potrzebną w takiej sytuacji będzie blacklistowanie danego hasha certyfikatu.

Adam Adamski

Skytechnology sp. z o.o.
ul. Miłobędzka 35
02-634 Warszawa
NIP: 5213633370
KRS: 0000426381