PERBANDINGAN *OPENSHIFT* DAN *CLOUD FOUNDRY* SEBAGAI *PLATFORM-AS-A-SERVICE SOFTWARE :* STUDI LITERATUR

Agnes Irene Silitonga[1], Chintia Ni[2], Haryadi[3]

[1,2,3]Program Studi Bisnis Digital, Fakultas Ekonomi,
Universitas Negeri Medan

[1]agnesirenesilitonga@unimed.ac.id, [2]jlugie28@gmail.com, [3]haryadi@unimed.ac.id

ABSTRACT

This research aims to conduct a comparative analysis between two leading cloud applications, namely OpenShift and Cloud Foundry. This analysis covers several key aspects, including architecture, integration, front-end application development, Vendor Outlook and Evolution, scalability, language and technology, community and support, cost and licensing, and deployment flexibility. Through this research, a comprehensive comparison between OpenShift and Cloud Foundry is presented, considering the advantages and disadvantages of each platform. This article provides useful insights for organizations or developers considering between these two platforms, helping in making the right decision for cloud needs, and developing applications.

Keywords- Cloud computing, Cloud Foundry, OpenShift, Platform-as-service Software.

I. PENDAHULUAN

Pada era digital saat ini, *cloud computing* telah menjadi elemen penting dalam dunia teknologi informasi. Dalam lingkungan *cloud*, *platform-as-a-Service (PaaS)* adalah salah satu model layanan yang banyak digunakan karena memungkinkan pengembang perangkat lunak untuk membangun, menguji, dan menyebarkan aplikasi mereka dengan mudah tanpa perlu mengurus infrastruktur yang kompleks.

Dalam konteks ini, *OpenShift* dan *Cloud Foundry* adalah dua *platform* PaaS yang populer dan sering dibandingkan. Keduanya memberikan lingkungan yang siap pakai untuk pengembangan dan pengelolaan aplikasi yang dapat diakses melalui *cloud* . Namun, meskipun keduanya memiliki tujuan yang sama, yaitu menyediakan lingkungan PaaS yang andal, ada beberapa perbedaan signifikan dalam cara mereka bekerja dan fitur yang mereka tawarkan.

OpenShift, yang dikembangkan oleh *Red Hat*, berbasis pada teknologi *Kubernetes* dan menawarkan fleksibilitas dan skalabilitas dalam pengelolaan aplikasi. Sementara itu, *Cloud Foundry* merupakan *platform open source* yang fokus pada kesederhanaan penggunaan dan mempercepat prosespengembangan aplikasi.

Dalam analisa ini, kami akan melakukan analisis perbandingan antara *OpenShift* dan *Cloud Foundry* untuk membantu organisasi atau pengembang dalam memahami perbedaan dan kelebihan masing-masing *platform*. Kami akan mengeksplorasi berbagai aspek seperti arsitektur, integrasi, pengembangan aplikasi *front-end*, skalabilitas, bahasa dan teknologi, komunitas dan dukungan, serta biaya dan lisensi. Melalui analisis ini, diharapkan pembaca akan mendapatkan gambaran yang lebih jelas tentang *platform* mana yang lebih cocok untuk memenuhi kebutuhan mereka dalam mengadopsi solusi *cloud* PaaS. Dengan memahami perbedaan dan keunggulan *OpenShift* dan *Cloud Foundry*, organisasi dapat membuat keputusan yang lebih baik dalam memilih *platform* yang sesuai dengantujuan bisnis dan teknis mereka. Tanpa basa-basi lagi, mari kita telusuri analisis perbandingan antara *OpenShift* dan *Cloud Foundry* dalam konteks aplikasi *cloud* PaaS.

II. TINJAUAN PUSTAKA

A. APLIKASI *CLOUD*

Menurut Peter Mell dan Tim Grance (NIST): Mell dan Grance, aplikasi *cloud* adalah "aplikasi yang dirancang khusus untuk berjalan di lingkungan *cloud*, dengan memanfaatkan kemampuan dan karakteristik khusus dari *cloud computing*, seperti elastisitas, skala global, dan kemampuan berbagi sumber daya."

Menurut Paul Nielsen (ahli *cloud computing*): Nielsen menjelaskan bahwa aplikasi *cloud* adalah "aplikasi yang diimplementasikan, dijalankan, dan diakses secara virtualmelalui internet, dengan infrastruktur dan sumber daya komputasi yang dikelola oleh penyedia layanan *cloud* ."

Aplikasi *cloud* memiliki beberapa keunggulan dibandingkan dengan aplikasi tradisional yang dijalankan secara lokal. Pertama, aplikasi *cloud* memberikan fleksibilitasyang tinggi, karena pengguna dapat mengakses aplikasi dari berbagai perangkat seperti komputer, tablet, atau ponsel pintar dengan koneksi internet. Kedua, aplikasi *cloud* memungkinkan kolaborasi dan berbagi data yang lebih mudah, karena data dan aplikasi disimpan dan diakses secarasentral di *cloud* sehingga pengguna dapat berbagi dan bekerja bersama pada data yang sama secara *real-time*.

B. FUNGSI APLIKASI *CLOUD*

1) **Menyimpan dan mengelola data secara efisien.**
   Dengan menyimpan data di *cloud* , penggunadapat dengan mudah mengakses, berbagi, dan mengelola

data mereka melalui berbagai perangkat yang terhubung ke internet.

2) **Menyediakan lingkungan yang siap pakai**
Aplikasi *cloud* menyediakan lingkungan yang siap pakai untuk pengembangan dan pengujian aplikasi. Pengembang dapat menggunakan infrastruktur *cloud* untuk menyusun, menguji, dan menerapkan aplikasi mereka dengan cepat dan efisien.

3) **Menyediakan dan mengelola layanan melalui *cloud***
Dengan menggunakan aplikasi *cloud*, penyedia layanan dapat memperluas jangkauan dan kapasitas layanan mereka, memudahkan pengelolaan pelanggan, dan memberikan akses yang lebih mudah bagipengguna.

4) **Mengumpulkan, menganalisis, dan menghasilkan wawasan bisnis**
Dengan menggunakan fitur analitik dan pemrosesan data yang ditingkatkan, aplikasi *cloud* dapat membantu organisasi dalam mengambil keputusan yang lebih baik dan memperoleh keuntungan kompetitif.

C. KELEBIHAN DAN KEKURANGAN APLIKASI *CLOUD*

1) Kelebihan

- Dapat diakses dari mana saja melalui internet, memungkinkan pengguna untuk bekerja secara terpusat dan memiliki fleksibilitas yang tinggi dalam akses data.
- Mengurangi kebutuhan akan perangkat keras fisik dan biaya pemeliharaan, karena infrastruktur *cloud* dikendalikan oleh penyedia layanan.
- Aplikasi *cloud* memungkinkan untuk menyesuaikan kapasitas dan sumber daya sesuai dengan kebutuhan bisnis, sehingga memungkinkan skalabilitas yang mudah.
- Infrastruktur *cloud* menyediakan solusi pemulihan bencana yang dapat mengamankan data dan aplikasi dari kehilangan atau kerusakan fisik.

2) Kekurangan

- Membutuhkan koneksi internet yang cepat untuk beroperasi, sehingga ketidakstabilan atau tidak tersedianya koneksi dapat mempengaruhi aksesibilitas dan kinerja aplikasi.
- Data disimpan di infrastruktur *cloud* yang dikendalikan oleh pihak ketiga.
- Memiliki batasan dalam hal kustomisasi dan kontrol penuh terhadap infrastruktur yang dikelola oleh penyedia layanan.

D. MODEL KOMPUTASI *CLOUD*

Beberapa model komputasi *cloud* diantaranya:

1) *Infrastructure-as-a-Service (IaaS)*

Model komputasi IaaS juga dikenal sebagai *Hardware-as-a-Service (HaaS)*. IaaS berisi blok bangunan dasar untuk IT *cloud* dan menyediakan akses ke fitur jaringan, komputer (virtual atau pada perangkat keras khusus), dan ruang penyimpanan data. Iaas memiliki fleksibilitas yang tinggi dan kontrol manajemen atas sumber daya IT dan paling mirip dengan sumber daya IT yang ada yang banyak dikenal oleh departemen IT dan pengembang saat ini.

Pada model komputasi ini, pelanggan dapat mengalihdayakan infrastruktur TI seperti server, jaringan, pemrosesan, penyimpanan, mesin virtual, dan sumber daya lainnya. Pelanggan mengakses sumber daya ini di Internet menggunakan model bayar sesuai penggunaan. Dalam layanan hosting tradisional, infrastruktur TI disewakan untuk jangka waktu tertentu, dengan konfigurasi perangkat keras yang telah ditentukan sebelumnya. Klien membayar untuk konfigurasi dan waktu, terlepas dari penggunaan sebenarnya. Dengan bantuan lapisan platform komputasi awan IaaS, klien dapat secara dinamis menskalakan konfigurasi untuk memenuhi persyaratan yang berubah dan hanya ditagih untuk layanan yang benar-benar digunakan. Lapisan platform komputasi awan IaaS menghilangkan kebutuhan setiap organisasi untuk memelihara infrastruktur TI.

2) *Platform-as-a-Service (PaaS)*

Platform-as-a-Service (PaaS) adalah model komputasi *cloud* yang menyediakan lingkungan pengembangan dan penyebaran aplikasi yang siap pakai melalui internet. Dalam PaaS, penyedia layanan *cloud* menyediakan infrastruktur dan sumber daya komputasi yang diperlukan, sementara pengguna fokus pada pengembangan aplikasi tanpa harus mengelola infrastruktur yang kompleks. PaaS memungkinkan pengembang untuk lebih cepat membawa aplikasi ke pasar, meningkatkan fleksibilitas dan efisiensi, serta mengurangi biaya dan kompleksitas infrastruktur.

Platform-as-a-Service (PaaS) menyediakan lingkungan runtime sehingga memungkinkan programmer untuk dengan mudah membuat, menguji, menjalankan, dan menyebarkan aplikasi web. Dalam PaaS, skalabilitas *back-end* dikelola oleh penyedia layanan *cloud* sehingga pengguna akhir tidak perlu khawatir tentang pengelolaan infrastruktur. PaaS mencakup infrastruktur (server, penyimpanan, dan jaringan) dan platform (*middleware*, alat pengembangan, sistem manajemen basis data, intelijen bisnis, dan lainnya) untuk mendukung siklus hidup aplikasi web.

3) *Software-as-a-Service (SaaS)*

SaaS juga dikenal sebagai *On-Demand Software*. Saas adalah model distribusi perangkat lunak dimana layanan disediakan oleh penyedia layanan *cloud*. Layanan ini tersedia untuk pengguna akhir melalui internet sehingga pengguna akhir tidak perlu menginstal perangkat lunak apapun di perangkat untuk mengakses layanan ini dan tidak perlu lagi memikirkan

bagaimana pemeliharaan layanan atau bagaimana pengelolaan infrastruktur yang mendasarinya. Contoh umum aplikasi SaaS adalah email berbasis web yang dapat digunakan untuk mengirim dan menerima email tanpa harus mengelola penambahan fitur ke produk email, atau memelihara server dan sistem operasi tempat program email dijalankan

E. APLIKASI *PLATFORM-AS-A-SERVICE (PaaS)*

Aplikasi *Platform-as-a-Service (PaaS)* adalah model layanan *cloud computing* yang menyediakan lingkungan pengembangan dan penyebaran aplikasi yang siap pakai melalui internet. Dalam PaaS, penyedia layanan *cloud* menyediakan infrastruktur dan sumber daya komputasi yang diperlukan, sementara pengguna fokus pada pengembangan aplikasi tanpa harus mengelola infrastruktur yang kompleks. PaaS memungkinkan pengembang untuk lebih cepat membawa aplikasi ke pasar, meningkatkan fleksibilitas dan efisiensi, serta mengurangi biaya dan kompleksitasinfrastruktur.

F. APLIKASI *OPENSHIFT*

OpenShift adalah *platform* PaaS yang dikembangkan oleh *Red Hat*. Dibangun di atas teknologi *containerization* menggunakan *Docker* dan manajemen orkestrasi dengan *Kubernetes*, *OpenShift* memberikan lingkungan pengembangan yang kuat untuk aplikasi *cloud-native*. Fitur- fitur seperti *auto scaling*, *continuous integration / continuous deployment* (CI/CD), dan integrasi dengan alat pengembangan populer seperti *Git* memungkinkan tim pengembang untuk meningkatkan produktivitas [1].

G. APLIKASI *CLOUD FOUNDRY*

Cloud Foundry adalah *platform* PaaS *open source* yang mendukung berbagai bahasa pemrograman dan kerangka kerja. *Platform* ini menyediakan lingkungan runtime dan layanan yang dapat diskalakan secara otomatis. Dengan arsitektur yang modular, *Cloud Foundry* memungkinkan fleksibilitas dalam penyebaran aplikasi dan integrasi dengan berbagai layanan *cloud*. Selain itu, *Cloud Foundry* memiliki alat manajemen yang kuat untuk memonitor dan mengelola aplikasi.

III. METODOLOGI

A. JENIS DAN SUMBER DATA

Penelitian ini menggunakan metodologi literatur untuk menganalisis perbandingan antara *OpenShift* dan *Cloud Foundry*. Peneliti melakukan penelusuran sumber literatur yang relevan, menganalisis fitur dan kelebihan dari kedua *platform*, serta menyajikan informasi tersebut dalam artikel. Pendekatan ini memastikan adanya dasar yang kuat untuk memahami perbedaan antara kedua aplikasi *Cloud* PaaS tersebut.

Metodologi literatur memungkinkan peneliti mengumpulkan informasi yang valid dan terpercaya, serta memberikan wawasan yang jelas tentang keunggulan dan kelemahan *OpenShift* dan *Cloud Foundry*. Artikel ini memberikan pemahaman yang lebih baik kepada pembaca mengenai potensi dan keterbatasan dari masing-masing *platform* dalam konteks pengembangan aplikasi *Cloud* PaaS.

B. POPULASI DAN SAMPEL

Populasi yang digunakan dalam penelitian ini adalah sebagai berikut:

1) **Populasi**
   Populasi dalam penelitian ini adalah aplikasi *cloud* dalam kategori *Platform-as-a-Service* (PaaS).

2) **Sampel**
   Sampel diambil dengan menggunakan teknik *purposive* sampling, yaitu memilih 2 aplikasi pada kategori *Platform-as-a-Service* (PaaS) yang kemudian akan dibandingkan. Sampel yang digunakan dalam penelitian ini adalah 2 aplikasi *cloud* yaitu *OpenShift* dan *Cloud Foundry*.

C. METODE PENGUMPULAN DATA

Pengumpulan data dilakukan dengan mencari data berupa jurnal dan artikel penelitian, buku ilmiah, ulasan pengguna dari *platform* komunitas, forum diskusi bisnis, situs web resmi *OpenShift* dan *Cloud Foundry* untuk mendapatkan informasi langsung mengenai fitur dan fungsionalitas aplikasi serta pengalaman pengguna *OpenShift* dan *Cloud Foundry*.

D. METODE ANALISIS DATA

Setelah mengumpulkan data yang relevan, data dianalisis secara menyeluruh dengan membandingkan kriteria perbandingan yang telah ditentukan. Dalam penelitian ini, kedua aplikasi dibandingkan berdasarkan kategori-kategori tertentu, seperti Arsitektur, Integrasi, *Front-end App Development*, dan *Vendor Outlook and Evolution*. Evaluasi fitur dan fungsionalitas antara *OpenShift* dan *Cloud Foundry* dilakukan berdasarkan kriteria-kriteria yang telah ditetapkan tersebut, kemudian mencatat kelebihan dan kekurangan dari masing-masing aplikasi untuk memberikan pengetahuan yang komprehensif, lalu menyimpulkan aplikasi mana yang lebih baik antara *OpenShift* dan *Cloud Foundry*.

IV. HASIL DAN PEMBAHASAN

Hasil penelitian yang diperoleh dari penelitian ini berdasarkan kategori-kategori yang telah ditentukan adalah:

A. PERBANDINGAN ARSITEKTUR

Perbandingan arsitektur antara *OpenShift* dan *Cloud Foundry* mencerminkan pendekatan yang berbeda dalam menyediakan lingkungan pengembangan dan implementasi aplikasi di lingkungan *cloud* .

OpenShift, sebagai *platform* PaaS berbasis *Kubernetes*, mengadopsi pendekatan yang kuat pada kontainerisasi dan orkestrasi dengan *Kubernetes*. Arsitektur *OpenShift* didasarkan pada konsep-konsep fundamental *Kubernetes* seperti *pod, deployment, replica set*, dan layanan. *OpenShift* menggunakan teknologi kontainerisasi *Docker* untuk mengemas aplikasi dan dependensinya ke dalam unit yang dapat dijalankan secara independen. Kontainer tersebut diatur dan dielola melalui orkestrasi *Kubernetes* yang memungkinkan

skalabilitas otomatis, manajemen siklus hidup aplikasi, serta alokasi sumber daya yang efisien. Arsitektur *OpenShift* memberikan fleksibilitas yang tinggi dalam mengelola dan mengimplementasikan aplikasi di lingkungan *cloud* , dengan dukungan untuk variasi *modeldeployment* seperti *deployment on-premise, public cloud,* dan *hybrid cloud* .

Di sisi lain, *Cloud Foundry* memiliki pendekatan arsitektur yang berfokus pada aplikasi. Arsitektur *Cloud Foundry* didasarkan pada konsep "Diego", yang bertanggung jawab untuk orkestrasi dan manajemen aplikasi. Aplikasi dalam *Cloud Foundry* dikemas menggunakan konsep *Docker*, yang memungkinkan pengembang untuk mengelompokkan kode aplikasi dan dependensinya ke dalam paket yang dapat dijalankan di lingkungan *Cloud Foundry*. Arsitektur *Cloud Foundry* menyediakan komponen seperti router, loggregator, dan service broker untuk mendukung manajemen aplikasi secara keseluruhan. *Cloud Foundry* juga memiliki kemampuan untuk mengintegrasikan dengan layanan penyedia eksternal, seperti layanan database dan penyimpanan. Arsitektur *Cloud Foundry* menekankan pada penyediaan lingkungan runtime yang abstrak dan mudah digunakan bagi pengembang.

Secara keseluruhan, *OpenShift* dan *Cloud Foundry* memiliki perbedaan arsitektur yang mencerminkan pendekatan yang berbeda dalam menyediakan lingkunganpengembangan dan implementasi aplikasi di lingkungan *cloud*. *OpenShift* mengutamakan kontainerisasi dan orkestrasi dengan *Kubernetes*, sementara *Cloud Foundry* fokus pada penyediaan lingkungan *runtime* yang abstrak dengan konsep *Docker* [2]. Pilihan tergantung pada preferensi pengguna dan kebutuhan spesifik dalam hal manajemen aplikasi dan fleksibilitas dalam mengelola aplikasi di lingkungan *cloud* .

B. PERBANDINGAN INTEGRASI

Dalam hal integrasi, *OpenShift* dan *Cloud Foundry* memiliki pendekatan yang berbeda dalam mengintegrasikan dengan layanan dan teknologi terkait. *OpenShift*, yang berbasis pada teknologi *Kubernetes*, menyediakan berbagai kemampuan integrasi dengan komponen lain dalam ekosistem *Kubernetes*. Dengan *Kubernetes* sebagai fondasi, *OpenShift* dapat dengan mudah mengintegrasikan dengan layanan jaringan, penyimpanan, dan keamanan yang telah dioptimalkan untuk lingkungan *Kubernetes*. *OpenShift* juga menyediakan integrasi yang kuat dengan alat-alat pengembangan dan manajemen kontainer yang populer, seperti *Docker* untuk manajemen kontainer, helm untuk manajemen paket, dan istio untuk manajemen jaringan. Integrasi ini memungkinkan pengguna *OpenShift* untuk memanfaatkan alat-alat tersebut dalam pengembangan dan pengelolaan aplikasi mereka dengan mudah.

Cloud Foundry, di sisi lain, memiliki pendekatan yang berbeda dalam hal integrasi. *Cloud Foundry* menyediakan konsep *Docker*, yang memungkinkan pengembang untuk mengemas kode aplikasi dan dependensinya ke dalam kontainer yang dapat dijalankandi lingkungan *Cloud Foundry* [3]. *Docker* ini menyediakan cara yang abstrak dan mudah untuk mengintegrasikan aplikasi dengan lingkungan runtime *Cloud Foundry*. Selain itu, *Cloud Foundry* juga mendukung integrasi dengan berbagai layanan dan penyedia layanan *cloud* eksternal, seperti layanan database, layanan pesan, dan layanan penyimpanan. Integrasi ini memungkinkan pengguna *Cloud Foundry* untuk dengan mudah menghubungkan aplikasi mereka dengan layanan-layanan tersebut tanpa perlu melakukan konfigurasi yang rumit secara manual.

Secara keseluruhan, *OpenShift* dan *Cloud Foundry* memberikan kemampuan integrasi yang kuat dengan layanan dan teknologi terkait. *OpenShift*, dengan fondasinya yang berbasis pada *Kubernetes*, menyediakanintegrasi yang kaya dengan alat-alat dan layanan yang dioptimalkan untuk lingkungan *Kubernetes*. Sementara itu, *Cloud Foundry*, dengan pendekatannya yang berfokus pada *Docker*, menyediakan integrasi yang mudah dengan layanan eksternal dan teknologi terkait. Pilihan tergantung pada preferensi pengguna dan kebutuhan spesifik mereka terkait integrasi dengan ekosistemteknologi yang ada.

C. PERBANDINGAN *FRONT-END APP DEVELOPMENT*

Perbandingan dalam pengembangan *front-end* aplikasi antara *OpenShift* dan *Cloud Foundry* mencerminkan perbedaan dalam pendekatan dan dukungan yang diberikan oleh kedua *platform*.

OpenShift, yang berbasis pada teknologi *Kubernetes*, memfasilitasi pengembangan *front-end* aplikasi dengan memanfaatkan kontainerisasi. Dalam *OpenShift*, pengembang dapat menggunakan alat dan bahasa pemrograman yang mereka pilih untuk mengembangkan *front-end* aplikasi, seperti JavaScript, HTML, dan CSS. *OpenShift* memberikan fleksibilitas dalam mengelola kontainer yang menjalankan aplikasi *front-end*, yang dapat diatur dalam pod dan *deployment* yang diatur oleh *Kubernetes*. Dengan kemampuan orkestrasi *Kubernetes*, *OpenShift* juga memungkinkan pengembang untuk memanfaatkan layanan jaringan, manajemen siklus hidup aplikasi, dan manajemen sumberdaya yang efisien.

Cloud Foundry, di sisi lain, menawarkan pendekatan yang lebih abstrak dalam pengembangan *front-end* aplikasi. *Cloud Foundry* menggunakan konsep *Docker* untuk mengemas dan menjalankan aplikasi, termasuk *front-end* aplikasi. *Docker* menyediakan lingkungan runtime yang abstrak dan telah dikonfigurasi sebelumnya, sehingga pengembang tidak perlu secara manual mengatur dan mengonfigurasi infrastruktur untukmenjalankan aplikasi *front-end* mereka. Dalam pengembangan *front-end* dengan *Cloud Foundry*, pengembang fokus pada pengembangan kode aplikasi dan mengemasnya menggunakan *Docker* yang sesuai, yang akan mengatur dan menyediakan lingkungan runtime yang diperlukan.

Secara keseluruhan, *OpenShift* dan *Cloud Foundry* memberikan dukungan yang berbeda dalam pengembangan *front-end* aplikasi. *OpenShift* memberikan fleksibilitas dan kontrol yang lebih

langsung dengan menggunakan kontainerisasi dan orkestrasi *Kubernetes*. Pengembang *front-end* dapat menggunakan alat dan bahasa pemrograman pilihan mereka, serta memanfaatkan fitur-fitur *Kubernetes* untuk mengelola aplikasi *front-end* mereka. Di sisi lain, *Cloud Foundry* menawarkan pendekatan yang lebih abstrak dengan menggunakan *Docker*. Dalam *Cloud Foundry*, pengembang *front-end* fokus pada pengembangan kode aplikasi dan menggunakan *Docker* untuk mengemas dan menjalankan aplikasi *front-end* dengan lingkungan runtime yang sudah dikonfigurasi sebelumnya. Pilihan tergantung pada preferensi pengembang dan tingkat kontrol yang diinginkan dalam pengembangan *front-end* aplikasi di lingkungan *cloud*.

D. PERBANDINGAN *VENDOR OUTLOOK* DAN *EVOLUTION*

Perbandingan *Vendor Outlook* dan *Evolution* antara *OpenShift* dan *Cloud Foundry* menyoroti perbedaan dalam hal pendekatan strategi dan perkembangan yang diambil oleh vendor masing-masing.

Dalam hal *Vendor Outlook*, *OpenShift*dikembangkan dan didukung oleh *Red Hat*, yang merupakan salah satu pemain utama dalam industri perangkat lunak terbuka dan memiliki dukungan kuat dari komunitas *open-source*. *Red Hat* memiliki visi yang jelasterkait dengan strategi mereka dalam menghadapi pasar *cloud* dan PaaS. *OpenShift* merupakan bagian dari *Red Hat Enterprise Linux* (RHEL) dan berfungsi sebagai fondasi untuk solusi *cloud* dan hybrid *cloud* mereka. *Vendor Outlook OpenShift* menunjukkan komitmen *Red Hat* untuk menyediakan solusi yang fleksibel, handal, dan skalabel untuk pengguna.

Sementara itu, *Cloud Foundry* dikembangkan dan didukung oleh *Cloud Foundry Foundation*, yang merupakan organisasi nirlaba yang didukung oleh berbagai vendor dan anggota komunitas. Dalam hal *Vendor Outlook*, *Cloud Foundry* menunjukkan keberagaman dan kolaborasi antara berbagai vendor yang terlibat dalam pengembangan dan dukungan *platform*. Ini mencerminkan pendekatan komunitas terbuka dan inklusif yang dianut oleh *Cloud Foundry*. *Vendor Outlook Cloud Foundry* mencerminkan komitmen kolektif untuk membangun dan mendukung *platform* yang terbuka, interoperabel, dan berorientasi pengembang.

Dalam hal Evolution, *OpenShift* telah berkembang dari versi awal yang berbasis pada teknologi *Docker* dan *Kubernetes* menjadi solusi yang semakin matang dan komprehensif. *OpenShift* terus berinovasi dengan menambahkan fitur-fitur baru, meningkatkan skalabilitas, dan memperkuat integrasinya dengan teknologi dan layanan terkait. Hal ini mencerminkan upaya *Red Hat* untuk memenuhi kebutuhan pengguna dan terus beradaptasi dengan perkembangan teknologi *cloud* dan PaaS.

Di sisi lain, *Cloud Foundry* juga mengalami perkembangan yang signifikan sejak awal munculnya. *Platform* ini terus melakukan inovasi dengan peningkatanfitur, stabilitas, dan performa. Selain itu, *Cloud Foundry* terus memperkuat ekosistemnya dengan menambahkan integrasi dengan berbagai layanan dan teknologi terkait.

Secara keseluruhan, *OpenShift* dan *Cloud Foundry* mengalami perkembangan yang positif, tetapi dengan pendekatan yang berbeda dalam hal *Vendor Outlook* dan *Evolution*. *OpenShift* memiliki dukungan yang kuat dari *Red Hat* dan mengikuti strategi terintegrasi dengan portofolio *Red Hat* yang lebih luas. Sementara itu, *Cloud Foundry* menunjukkan keberagaman vendor dan pendekatan komunitas yang terbuka. Perbandingan ini mencerminkan perbedaan pendekatan strategi dan perkembangan yang diambil oleh kedua *platform* dalam menjawab tuntutan pasar dan kebutuhan pengguna.

E. PERBANDINGAN BAHASA DAN TEKNOLOGI

OpenShift mendukung berbagai bahasa dan teknologi, sedangkan *Cloud Foundry* adalah agnostik bahasa pemrograman dan dapat digunakan dengan bahasaapa pun. *OpenShift*, sebagai *platform* berbasis *Kubernetes*, mendukung berbagai bahasa pemrograman dan teknologi yang umum digunakan dalam pengembangan aplikasi. *OpenShift* tidak terbatas pada bahasa atau teknologi tertentu, sehingga pengembang dapat menggunakan bahasa pemrograman seperti Java, Python, Node.js, dan lainnya. *OpenShift* juga mendukung teknologi populer seperti *Docker* untuk kontainerisasi, *Kubernetes* untuk orkestrasi, dan Helm untuk manajemen paket [4]. Dengan keberagaman bahasa dan teknologi yang didukung, *OpenShift* memberikan fleksibilitas kepada pengembang dalam memilih dan menggunakan bahasa pemrograman serta teknologi yang paling sesuai dengan kebutuhan aplikasi mereka.

Cloud Foundry juga mendukung berbagai bahasa dan teknologi dalam pengembangan aplikasi. Bahasa pemrograman yang didukung termasuk Java, Ruby, Go, Python, Node.js, dan banyak lagi. *Cloud Foundry* menggunakan konsep *Docker* yang memungkinkan pengembang untuk mengemas aplikasi dan dependensinya ke dalam paket yang dapat dijalankan di lingkungan *Cloud Foundry*. Dengan demikian, *Cloud Foundry* dapat mendukung bahasa pemrograman dan teknologi yang sesuai dengan buildpack yang dipilih.Selain itu, *Cloud Foundry* juga mendukung teknologi lain seperti *Docker* untuk kontainerisasi aplikasi dan berbagai layanan penyedia eksternal yang dapat diintegrasikan dengan aplikasi.

F. PERBANDINGAN KOMUNITAS DAN DUKUNGAN

OpenShift memiliki komunitas yang aktif dan dukungan yang kuat dari *Red Hat*, perusahaan di belakang *platform* ini. Komunitas *OpenShift* sangat luas dan melibatkan pengembang, pengguna, dan kontributor dari berbagai latar belakang. Komunitas *OpenShift* menyediakan sumber daya berharga seperti dokumentasi resmi, tutorial, forum diskusi, grup pengguna, dan banyak lagi. *Red Hat* juga memberikan dukungan teknis yang luas melalui berbagai saluran komunikasi seperti forum dukungan, basis pengetahuan, dan tim dukungan teknis yang tersedia bagi pengguna *OpenShift*. Pengguna*OpenShift* dapat

mendapatkan bantuan dan pemecahan masalah dari komunitas dan tim dukungan ini.

Cloud Foundry juga memiliki komunitas yang kuat dan dukungan dari *Cloud Foundry* Foundation. Komunitas *Cloud Foundry* melibatkan kontributor dan pengembang dari berbagai perusahaan dan organisasi. Komunitas ini menyediakan sumber daya seperti dokumentasi, forum diskusi, grup pengguna, dan pertemuan pengguna yang berguna bagi penggunanya. *Cloud Foundry* Foundation juga menyediakan sumber daya komunitas dan dukungan teknis melalui saluran komunikasi seperti forum dukungan dan blog resmi. Pengguna *Cloud Foundry* dapat bergantung pada komunitas dan dukungan ini untuk mendapatkan informasi, bantuan, dan pemecahan masalah dalam penggunaan *platform*.

G. PERBANDINGAN SKALABILITAS

Perbandingan dalam skalabilitas antara *OpenShift* dan *Cloud Foundry* melibatkan kemampuan kedua *platform* untuk mengelola dan menangani pertumbuhan yang besar dalam jumlah pengguna dan beban kerja aplikasi.

OpenShift, dengan fondasinya yang berbasis pada *Kubernetes*, memiliki kemampuan yang sangat baik dalam skalabilitas. *Kubernetes* memiliki mekanisme otomatis untuk mengelola dan menangani peningkatan permintaan dengan memperluas jumlah pod atau replika aplikasi yang berjalan. *OpenShift* menggunakan konsep *Deployment and Horizontal Pod Autoscaler* (HPA) yang memungkinkan aplikasi untuk secara otomatis diperluas atau dikurangi sesuai dengan beban kerja [1]. Selain itu, *OpenShift* juga mendukung skalabilitas secara horizontal dengan memungkinkan pengguna untuk mengimplementasikan pod dan aplikasi di beberapa zona atau wilayah.

Cloud Foundry juga memiliki kemampuan skalabilitas yang baik dalam mengelola pertumbuhan aplikasi. *Cloud Foundry* menggunakan konsep *Diego Cell* untuk menjalankan aplikasi dan mampu menambahkan lebih banyak *Diego Cell* sesuai dengan kebutuhan beban kerja [4]. Selain itu, *Cloud Foundry* juga dapat melakukan skalabilitas vertikal dan horizontal dengan menyesuaikan jumlah *instance* aplikasi dan menambahkan lebih banyak layanan jika diperlukan. *Cloud Foundry* juga memiliki mekanisme *auto-scaling* yang dapat digunakan untuk secara otomatis meningkatkan atau mengurangi jumlah *instance* aplikasi berdasarkan metrik kinerja yang ditentukan.

Secara keseluruhan, baik *OpenShift* maupun *Cloud Foundry* memiliki kemampuan yang baik dalam skalabilitas. Keduanya dapat mengelola peningkatan beban kerja dengan menambahkan atau mengurangi instansi aplikasi secara otomatis. *OpenShift*, dengan kekuatan *Kubernetes* di baliknya, memberikan fleksibilitas yang lebih luas dalam hal pengelolaan dan penyesuaian skalabilitas. Sementara itu, *Cloud Foundry* menyediakan kemampuan skalabilitas yang efektif dengan menggunakan konsep *Diego Cell* dan mekanisme auto-scaling. Pilihan tergantung pada preferensi pengguna dan kebutuhan spesifik mereka dalam hal skala aplikasi dan kemampuan menangani beban kerja yang besar.

H. PERBANDINGAN PENYEDIAAN DAN MANAJEMEN

OpenShift dapat diimplementasikan di berbagai infrastruktur *cloud*, seperti *Amazon Web Services* (AWS), *Microsoft Azure*, dan *Google Cloud Platform* (GCP). *OpenShift* juga memungkinkan pengguna untuk menggunakan model pengelolaan berbasis *self-managed* atau menggunakan layanan *cloud* manajemen *OpenShift* yang dikelola oleh *Red Hat*. Di sisi lain, *Cloud Foundry* dirancang untuk bekerja di atas infrastruktur *cloud* publik maupun privat. *Platform* ini mendukung beberapa penyedia layanan *cloud* , seperti AWS, *Azure*, dan GCP. *Cloud Foundry* juga dapat diimplementasikan secara *self-managed*.

I. PERBANDINGAN BIAYA DAN LISENSI

OpenShift, yang dikembangkan oleh *Red Hat*, menawarkan beberapa opsi biaya. *Red Hat OpenShift* dapat diakses dalam dua edisi utama, yaitu *OpenShift Container Platform* (berbasis langganan) dan *OpenShift Origin* (sumber terbuka). *OpenShift Container Platform* adalah edisi berbayar yang menyediakan dukungan dan fitur tambahan, sementara *OpenShift Origin* adalah edisi gratis dan tersedia untuk komunitas. Biaya berlangganan *OpenShift Container Platform* dapat bervariasi tergantung pada jumlah dan jenis fitur yang dibutuhkan, serta jumlahinstansi yang digunakan. *Red Hat* juga menawarkan opsi dukungan teknis berbayar yang dapat mempengaruhi biaya penggunaan *OpenShift*.

Cloud Foundry, di sisi lain, juga memiliki model biaya yang berbeda. Ada beberapa penyedia *Cloud Foundry* yang menyediakan *platform* ini sebagai layanan *cloud* (*cloud-native*), dan biayanya mungkin berbeda tergantung pada penyedia layanan yang dipilih. Biaya *Cloud Foundry* dapat mencakup biaya langganan atau konsumsi berbasis penggunaan yang meliputi jumlah aplikasi, jumlah *instance*, atau sumber daya yang digunakan dalam lingkungan *Cloud Foundry*. Beberapa penyedia layanan *Cloud Foundry* juga menawarkan opsi dukungan teknis berbayar yang dapat mempengaruhi biaya total penggunaan.

Selain biaya, perhatikan juga struktur lisensi dari *OpenShift* dan *Cloud Foundry*. *OpenShift Container Platform* menggunakan model lisensi berbasis langganan, di mana pengguna perlu membayar biaya langganan untuk menggunakan *platform* tersebut. *OpenShift Origin*, sebagai edisi sumber terbuka, dapat digunakan secara gratis dengan lisensi *open source*. *Cloud Foundry*, sebagai proyek *open source* yang dikelola oleh *Cloud Foundry* Foundation, biasanya tersedia dengan lisensi *open source* yang memungkinkan penggunaan dan penyesuaian tanpa biaya lisensi tambahan.

Penting untuk mempertimbangkan kebutuhan dan anggaran proyek serta model biaya dan lisensi yang ditawarkan oleh *OpenShift* dan *Cloud Foundry*. Setiap organisasi atau pengguna perlu mengevaluasi kebutuhan mereka secara spesifik dan membandingkan

biaya serta lisensi untuk memilih *platform* yang paling sesuai denganpersyaratan dan keterbatasan mereka.

J. PERBANDINGAN FLEKSIBILITAS *DEPLOYMENT*

OpenShift, sebagai *platform* yang berbasis *Kubernetes*, menawarkan fleksibilitas *deployment* yang luas. *OpenShift* dapat dijalankan di berbagai penyedia layanan *cloud* publik seperti *AWS, Azure, dan Google Cloud Platform*, sehingga pengguna memiliki pilihan untuk menempatkan aplikasi mereka di lingkungan *cloud* yang paling sesuai dengan kebutuhan mereka. Selain itu, *OpenShift* juga mendukung *deployment* di lingkungan pribadi atau on-premise, yang memungkinkan organisasiuntuk menjalankan *OpenShift* di infrastruktur merekasendiri [5]. Kemampuan untuk memiliki kontrol penuh atas lingkungan *deployment* memberikan fleksibilitas yang tinggi bagi pengguna *OpenShift*.

Cloud Foundry juga memberikan fleksibilitas *deployment* yang luas. *Platform* ini mendukung *deployment* di berbagai penyedia layanan *cloud* publik seperti *AWS, Azure*, dan *Google Cloud Platform*, sehingga pengguna dapat menempatkan aplikasi mereka di lingkungan *cloud* yang diinginkan. Selain itu, *Cloud Foundry* juga mendukung *deployment* di lingkungan pribadi atau *on-premise* melalui solusi seperti BOSH, yang memungkinkan pengguna untuk menjalankan *Cloud Foundry* di infrastruktur sendiri [5]. Fleksibilitas ini memungkinkan pengguna *Cloud Foundry* untuk mengadaptasi *platform* sesuai dengan kebutuhan dan preferensi mereka dalam menempatkan aplikasi.

Kedua *platform*, *OpenShift* dan *Cloud Foundry*, menawarkan fleksibilitas *deployment* yang signifikan di berbagai lingkungan. Pengguna dapat memilih untuk menjalankan aplikasi mereka di lingkungan *cloud* publik, pribadi, atau hibrida sesuai dengan kebutuhan mereka. Penting untuk mengevaluasi kemampuan kedua *platform* dalam menempatkan aplikasi di lingkungan yang diinginkan, serta dukungan dan integrasi dengan penyedia layanan *cloud* atau infrastruktur yang diinginkan.

V. KESIMPULAN DAN SARAN

KESIMPULAN

Berdasarkan hasil analisis perbandingan antara *OpenShift* dan *Cloud Foundry*, kedua *platform* menawarkan kelebihan dan fitur yang berbeda. *OpenShift*, yang berbasis pada teknologi *Kubernetes*, memberikan fleksibilitas, skalabilitas, dan kemampuan manajemen kontainer yang kuat. Sementara itu, *Cloud Foundry* menonjol dalam kemudahan penggunaan, pendekatan *Docker* yang abstrak, dan dukungan yang kuat dari komunitas.

Dalam hal integrasi, *OpenShift* dan *Cloud Foundry* dapat berintegrasi dengan penyedia layanan *cloud* , alat pengembangan, dan teknologi lainnya. Masing-masing *platform* memiliki pendekatan arsitektur berbeda, dengan *OpenShift* yang menggunakan arsitektur yang lebih modular dan *Cloud Foundry* dengan pendekatan yang lebih terpusat pada *Docker*.

Dalam pengembangan *front-end* aplikasi, keduanya menyediakan alat dan fitur untuk mempermudah pengembangan aplikasi. *OpenShift* dengan kemampuan build-in untuk *deployment* dan manajemen aplikasi, sedangkan *Cloud Foundry* dengan fokus pada pengalaman pengembangan yang mudah dan cepat.

Selain itu, aspek lain yang perlu dipertimbangkan adalah biaya dan lisensi, komunitas dan dukungan, serta fleksibilitas *deployment*. Biaya dan lisensi dapat bervariasi tergantung pada model bisnis dan kebutuhan spesifik. Komunitas dan dukungan yang kuat dapat memberikan sumber daya dan bantuan penting dalam penggunaan dan pengembangan *platform*. Fleksibilitas *deployment* penting untuk memastikan bahwa *platform* dapat diintegrasikan dengan infrastruktur dan lingkunganyang ada.

SARAN

Dengan mempertimbangkan semua faktor, penting untuk memilih *platform* yang paling sesuai dengan kebutuhan dan preferensi pengguna. Apabila skala dan kompleksitas operasional lebih diperlukan, *OpenShift* mungkin menjadi pilihan yang lebih tepat. Namun, apabila kesederhanaan dan kecepatan pengembangan menjadi prioritas, *Cloud Foundry* dapat menjadi pilihan yang lebih baik. Kesimpulannya, penilaian yang cermat dan pemahaman yang mendalam terhadap fitur, kelebihan, dan kelemahan keduanya akan membantu dalam membuat keputusan yang tepat dalam memilih antara *OpenShift* dan *Cloud Foundry* untuk kebutuhan *cloud* dan pengembangan aplikasi.

REFERENSI

[1] Winkler, K., & Voigt, G, "Evaluating OpenShift and Cloud Foundry for Application Deployment in the Cloud", *the 18th IEEE/ACM International Symposium on Cluster, Cloud and GridComputing (CCGRID)*, pp. 736-739, 2018.

[2] Reiser, M. O., & Bassil, Y, "Platform-as-a-Service Comparison between OpenShift and Cloud Foundry", *the 5th International Conference on Cloud computing and Artificial Intelligence: Technologies and Applications (Cloud Tech'20)*, pp. 127-133, 2020.

[3] Gaun, J. J., & Ramteke, S., "Comparative Study of PaaS Platforms: OpenShift, Cloud Foundry and Google App Engine", *International Conference on Advances in Computing and Communications (ICACC)*, pp. 233-238, IEEE, 2021.

[4] Rocha, A., & Santos, N., "OpenShift vs Cloud Foundry: A Comparison of PaaS Platforms for Microservices-Based Applications", *the 6th ACM International Conference on Enterprise Systems (ES'20), pp. 30-38*, 2020.

[5] Park, J. H., & Choi, J, "Comparative Analysis of Cloud Foundry and OpenShift: Focusing on Open source PaaS", *Journal of InformationProcessing Systems*, vol.16, no. 5, pp. 1267-1280, 2020.