ALTOROS™

Architect's Guide

to Implementing the Cloud Foundry PaaS

From an architect's point of view, this ultimate guide describes the four stages of the platform's adoption within an enterprise.

Q2 **2017**

Roger Strukhoff Director of Research
Juan Pablo Genovese Field Cloud Foundry Architect
Alex Khizhniak Director of Tech Evangelism
Volha Kurylionak Technology Evangelist

Table of Contents

1. Executive Summary

Adopting a PaaS—such as Cloud Foundry—on the scale of an enterprise is complex and challenging. Achieving success goes far beyond getting the technology onboard. Unless there are deep changes in the processes and corporate culture, it is hard to get the best of it.

Generally, Cloud Foundry implementation involves four major stages:

1) evaluating Cloud Foundry
2) building a proof of concept (POC)
3) assessing the POC
4) rolling out the platform

This guide is designed to help Cloud Foundry adopters to successfully complete all of the required steps, from planning a POC to building a DevOps culture and integrating the platform throughout the organization. Here are just some of the things covered in this document:

- supported technologies and required skills
- infrastructure and TCO
- community and ecosystem
- POC goals and metrics
- security, scaling, and HA
- backups, upgrades, and monitoring
- processes, policies, and a DevOps culture

Although the guide is intended to be as complete as possible, new parts and updates may be added to it later. So, any feedback from the community is welcome.

2. Cloud Foundry Evaluation

2.1 Defining PaaS

PaaS stands for “Platform as a Service.” PaaS is generally thought of as part of a continuum within cloud computing, which also includes IaaS (Infrastructure as a Service) and SaaS (Software as a Service).

A PaaS system provides a way to develop, deploy, and manage apps without users having to manage the underlying infrastructure upon which the apps run. The platform lets users specify required computing resources, and then performs the detail needed to deploy and manage the apps.

If a company is migrating some or all of its resources toward a cloud computing architecture, a PaaS is the catalyst that provides the actual provisioning of apps.

A platform can be available in the form of standalone products and services, such as Cloud Foundry or OpenShift. A PaaS can also be a part of IaaS solutions in the form of Heroku from Salesforce, AWS Elastic Beanstalk, and within Microsoft Azure.

The definition of Platform as a Service has evolved over the past few years. The original PaaS tools, circa 2008, were accessed through their provider’s website. This followed the notion that all cloud computing services were to be delivered through a browser, from a service provider’s remote system

somewhere “in the cloud.”

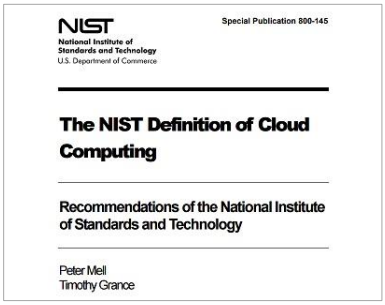
This notion, in turn, played a role in a definition of cloud computing developed by the U.S. National Institute of Standards and Technology (NIST) in 2011, in which cloud computing was “supported by the provider (and in which) the consumer does not manage or control the underlying cloud infrastructure.”

The NIST definition went on to specify “*public*” cloud computing as services provisioned by third-party providers, and “*private*” cloud computing as services “provisioned for exclusive use by a single organization, may be owned...by the organization, a third party, or some combination of them, and may exist on or off premises.”

The NIST definition is a guideline only, and not a law or regulation. Today, private cloud is understood to be owned and managed by an enterprise rather than a third-party provided. The computing resources are understood to be located on-site somewhere within the enterprise.

This evolution means that PaaS today is something that can be controlled by a third-party (when public cloud is being deployed) or by the enterprise itself (in the case of private cloud).

Notes:


NIST
National Institute of Standards and Technology
U.S. Department of Commerce

Special Publication 800-145

The NIST Definition of Cloud Computing

Recommendations of the National Institute of Standards and Technology

Peter Mell
Timothy Grance

1) The full 2011 NIST definition can be found here.

2) Here is a definition of PaaS from 2013. It differs from today’s conception.

3) Here is an excerpt from the Wikipedia definition of PaaS:

“**Platform as a service (PaaS)** is a category of cloud computing services that provides a platform allowing customers to develop, run and manage applications without the complexity of building and maintaining the infrastructure typically associated with developing and launching an app.

PaaS can be delivered in two ways: as a public cloud service from a provider, where the consumer controls software deployment and configuration settings, and the provider provides the networks, servers, storage and other services to host the consumer's application; or as software installed in private data centers or public infrastructure as a service and managed by internal IT departments.”

2.2 How Cloud Foundry fits in

The architectural concepts for what we know now as Cloud Foundry (CF) were developed at VMware. (The company is now a subsidiary of EMC.) CF was created to handle the challenges of deploying applications into pooled, virtualized cloud resources running on VMware’s products.

Cloud Foundry was spun out from VMware into a company called Pivotal Software in 2013. It then became the property of The Cloud Foundry Foundation, a non-profit organization that was founded in 2014. The Cloud Foundry Foundation includes Pivotal as a key, founding member, along with 60+ other members, including Altoros.

The platform is offered by the Cloud Foundry Foundation in its open-source form. (The Cloud Foundry Foundation is currently governed by The Linux Foundation.) Cloud Foundry has also been developed

into commercial, proprietary distributions by Pivotal, IBM (within its Bluemix services), CenturyLink, and others.

Cloud Foundry in all its forms competes directly with Red Hat's OpenShift, Jelastic, Apprenda, and WSO2 for PaaS customers. It also competes with several infrastructure providers that offer PaaS as part of their offering. This list includes Microsoft (Azure), Salesforce (Heroku), and Google (App Engine).

Market share numbers for each of these platforms are impossible to determine precisely. A general consensus among analysts indicates that perhaps US$20 billion will be spent on PaaS solutions in 2015, within a global budget of IT services of $300 billion and total global IT spend of more than $2 trillion.

An interesting roundup of market looks and projections can be found in this article by Louis Columbus in Forbes.

Again, specific market numbers are impossible to determine. Among the scant bits of available information, Pivotal has indicated that more than 100 enterprises are using its platform, and IBM is investing more than $1 billion to push its Bluemix strategy, which contains Cloud Foundry as a centerpiece.

More important than a facile market projection can happen when IT executives and managers analyze why their enterprises would consider Cloud Foundry. Such an analysis should focus on a few key points:

- the ability of Cloud Foundry to simplify and manage the complex task of provisioning cloud computing infrastructure
- the existing features and roadmap of Cloud Foundry
- the Cloud Foundry ecosystem

2.3 Cloud Foundry distributions

Cloud Foundry is open-source software and is offered as such by the non-profit Cloud Foundry Foundation. It is also offered in commercial, proprietary versions from Pivotal, IBM, HPE, CenturyLink, and others.

The open-source version is available for download through GitHub. There are several other, accompanying pieces of code available from this GitHub link. Documentation for the open-source version is available from the Cloud Foundry website.

Cloud Foundry works with applications that have been developed with Java, Ruby, Python, Go, PHP, or Node.js. A series of custom "buildpacks" (explained later) extend this ability into other languages.

Implementing Cloud Foundry requires moderate IT skills. Although it removes the complexity of managing underlying cloud infrastructure, the use of Cloud Foundry nonetheless requires skills in the areas of setup, integration, extension, and management.

For enterprise IT departments, there are three ways to address the challenges of implementing Cloud Foundry:

- dedicating full-time staff to learning, using, and managing it
- buying and implementing one of the commercial distributions (which still requires certain skills and knowledge to complete the integration work)
- working with a technology integrator, such as Altoros, to maximize Cloud Foundry's features and performance

The commercial distributions offer additional features—such as interfaces, operations management, role-based access and security, centralized logging and metrics, analytics—and commercial support. Pivotal and ActiveState (the original developer of the Stackato platform) cited all of these areas as reasons to adopt their respective commercial distributions.

Additionally, IBM, for example, has integrated Cloud Foundry into its overall Bluemix cloud app development platform. CenturyLink has integrated it into its CenturyLink Cloud. Both companies made major acquisitions of data center companies in recent years—SoftLayer for IBM and Tier 3 for CenturyLink—to provide IaaS for their PaaS customers.

Source	CF Distribution	Type	URL
Cloud Foundry Foundation	OSS Cloud Foundry	Open source	www.cloudfoundry.org
Pivotal	Pivotal Cloud Foundry (PCF)	Proprietary	www.pivotal.io
IBM	IBM Bluemix	Proprietary	http://www.ibm.com/cloud-computing/bluemix/
GE	Predix	Proprietary	https://www.predix.io/
HPE	Helion Stackato (acquired by SUSE)	Proprietary	https://www.hpe.com/us/en/solutions/cloud.html
CenturyLink	AppFog (integrated into CenturyLink Cloud)	Proprietary	https://www.ctl.io/appfog/
Huawei	FusionStage	Proprietary	http://e.huawei.com/us/solutions/technical/cloud-computing
SAP	HANA Cloud Platform	Proprietary	https://cloudplatform.sap.com/index.html
Swisscom	Application Cloud	Proprietary	https://www.swisscom.ch/en/business/enterprise/offer/cloud-data-center-services/paas/application-cloud.html
Atos	Atos Cloud Foundry	Proprietary	https://canopy-cloud.com/application-platforms/atos-cloud-foundry
Cisco	Cisco Containerized CF	Proprietary	http://container.cf/en/latest/
NTT Communications	Enterprise Cloud	Proprietary	http://www.ntt.com/en/services/cloud/enterprise-cloud.html

Enterprises that wish to avoid vendor lock-in can take the route of doing it on their own or turning to a specialized integrator.

2.4 The Cloud Foundry Foundation

The Cloud Foundry Foundation is a non-profit organization founded in December 2014. It is governed by The Linux Foundation from its headquarters in San Francisco.

Tiered membership is open to technology providers that wish to contribute to the ongoing development of Cloud Foundry. Platinum members have committed $500,000 annually for three years.

Platinum members—as of Q2 2017—include Cisco, Dell/EMC, IBM, Pivotal, SAP, SUSE, and VMware.

Gold members: Accenture, Allianz, Allstate, BNY Mellon, Capgemini, Cognizant, Ford, GE, Google, Huawei, NTT, Philips, SAS, Swisscom, and Testra

Silver members: Acetti, Altoros, Anynines, Armakuni, Atos, Biarca, Bloomberg, Bosch, CA, CloudSoft, Codenvy, Comcast, Docker, ECS Team, Engineer Better, Evoila, Fidelity, Fujitsu, Gemalto, Grape Up, Hazelcast, HPE, Hexad, Hitachi, Honeywell, Intel, Minio, JPMorgan, Mimacom, MoPaaS, NIA, Orange, Proximetry, QIQ, Redis Labs, Resilient Scale, RBC, Snyk, Stark & Wayne, Toshiba, and Volkswagen

The Foundation has a board comprised of nominated and elected representatives from among the Platinum and Gold members.

The Cloud Foundry Dojo

The Foundation offers a Dojo—based on the idea of Japanese training and “way of life” facilities—which provides a fast track for commit rights within Cloud Foundry. The challenge undertaken by the Cloud Foundry Foundation is to propel the technology forward through community contributions, but not to lose a strong technology core in the process.

Members have compared its approach with that of the OpenStack IaaS community and foundation, which offers its members a great deal of latitude in developing the technology for their own ends. The Cloud Foundry Foundation, in contrast, seems to focus on instituting more discipline and less variation on distributions as the technology develops.

2.5 The value of Cloud Foundry to the enterprise

Cloud computing is an important, growing aspect of enterprise IT. Although most current estimates indicate that cloud computing takes up between 8 and 10% of enterprise IT spend, it takes up the majority of budgets devoted to developing and deploying applications.

Cloud computing was originally cited by its proponents for simplifying IT architecture, reducing capital expenditure and brining new levels of agility, scalability, and flexibility to the enterprise. The reality is somewhat more nuanced and complex than that.

This is where PaaS in general and Cloud Foundry in particular enter the picture. Cloud Foundry can be said to bring benefits to an enterprise in these key areas:

- simplifying and reducing the time of deployment of apps to the cloud
- enabling an enterprise's reach into several key programming languages for app development
- extending that reach into cloud-centric practices, such as the use of containers and microservices
- improved infrastructure utilization
- guaranteeing a future-proof, cloud-based path to digital transformation
- migration toward a DevOps culture

CF Benefit	Metric	Result
Organizational agility	Time to validate new revenue stream	3x to 10x faster
DevOps productivity	Reduces cost of people and processes	• the number of manual steps reduced by 70% • time to push an app into production reduced from weeks to hours • blue-green deployments
Security and compliance	Simple commands keep time commitment to minimum	Dynamic provisioning
Infrastructure utilization	Increase percentage of server and storage usage	Increases from 10–15% to 60% and above

An informative chart is also provided by ActiveState, which offered the commercial Stackato version of Cloud Foundry. The slide refers to a Fortune 500 wireless component company working with ActiveState:

F500 Wireless Component Company

IT Domain	Before Stackato	After Stackato
Resource deployment	*Weeks*	*500% improvement*
Polyglot support	*Java only*	*10+ languages/stacks*
Application prototyping	*8+ weeks*	*1 day*
New application release	*Infrequent*	*Constant*
Partnering	*Disconnected*	*Custom applications*
API exposure	*None*	*Multiple*
IoT initiative	*Impossible*	*AWS-based*

2.6 How Cloud Foundry works

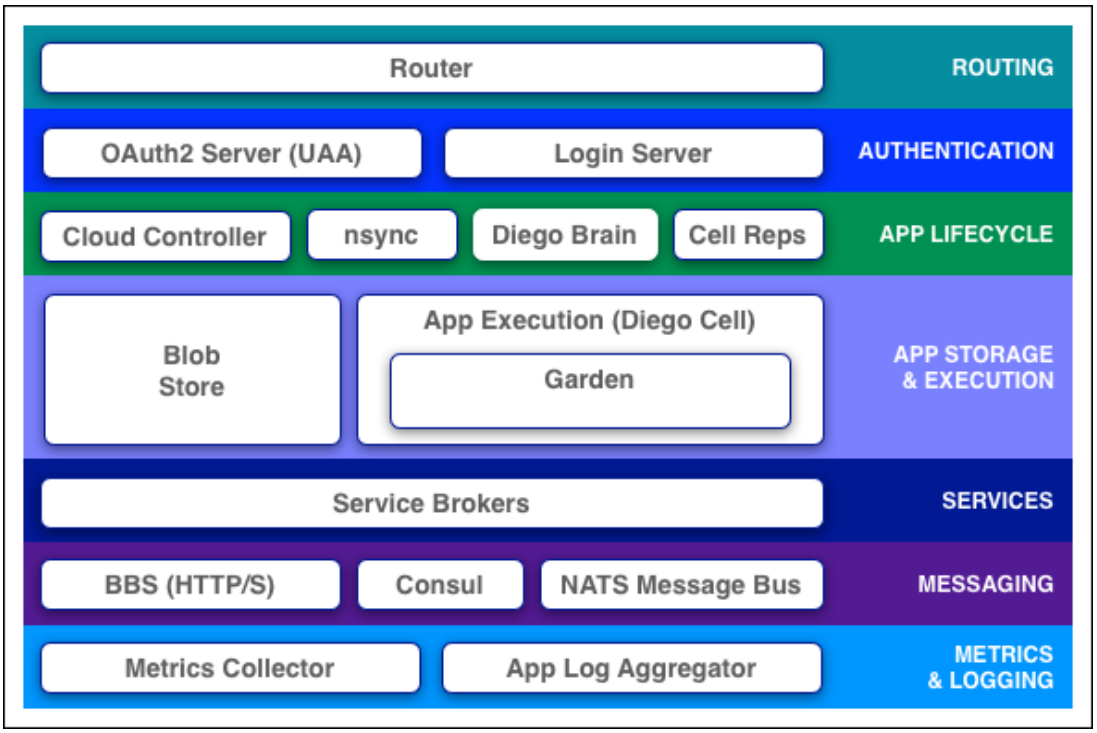
Cloud Foundry provides automation for different stages of an app development life cycle. Among its targets are infrastructure provisioning, deployment automation, hardware resilience, continuous integration, etc.

The platform employs Role-Based Access Control (RBAC) aka Role-Based Security (RBS) as its deployment strategy. Within that approach, it incorporates orgs, spaces, roles, and permissions to maintain order.

The *org* sits at the top of the hierarchy and is an account that can be used by one or multiple individuals or groups. Each application (or service) is mapped to a particular *space*. There is a lot of flexibility with spaces, as they can be created across the gamut of development and QA, deployment, maintenance, and updating.

Roles are similar to permissions, and apply to what's going on in a particular org. An individual can have multiple roles. There can be more than one org manager to perform tasks, such as adding and managing users, managing spaces, view various status readings and quotas, and add domains. *Permissions* are granted within each space and can be managed by one or more space managers.

Users of Cloud Foundry will find it has several key components. A general overview appears below.



Source: Pivotal

Buildpacks

The organization also provides *buildpack* support for several programming languages. The following table shows current buildpacks and GitHub resources.

Language / Technology	GitHub Repo
Go	Go source
Java (also supports Grails, Play, Spring, and other JVM-based languages and frameworks)	Java source
Node.js (also supports Node, JavaScript)	Node.js source
PHP	PHP source
Python	Python source
Ruby (also supports Rack, Rails, and Sinatra)	Ruby source

The Cloud Foundry CLI

Cloud Foundry has a command-line interface (CLI), which is installed with a simple point-and-click process. The latest version of the CLI is written in the Go language. The update performs better than previous versions written in Ruby; it now has native installers for all major operating systems.

The key command within the CLI is the simple `cf push` command, which deploys the application. Apps can access external resources via Cloud Foundry services. CLI v6 has new commands for creating and updating user-provided service instances. Once created, user-provided service instances can be bound to an application with `cf bind-service`, unbound with `cf unbind-service`, renamed with `cf rename-service`, and deleted with `cf delete-service`.

Management consoles are provided by proprietary vendors for accessing and integrating services. Open-source Cloud Foundry users do so by writing scripts in BOSH.

BOSH

The BOSH tool chain (short for the "BOSH Outer Shell") orchestrates the deployment process of distributed systems. It can be used to provision and deploy apps and services over several—even hundreds—VMs. BOSH can perform scaling, health monitoring, and recovery of failed instances, as well as provide updates with zero or near-zero downtime. It was originally designed to deploy Cloud Foundry, but can be used to package, install, and manage life cycle of any software.

BOSH can work with public and private infrastructures, as well as containers (Warden/Garden, Docker). Out-of-the-box support is implemented for VMware vSphere/vCloud, Amazon Web Services, Microsoft Azure, CloudStack, and OpenStack. It is also possible to enable BOSH on any other IaaS, using a cloud provider interface (CPI).

BOSH is a Cloud Foundry project; its GitHub repository can be found here.

Docker

There is a service broker for use with Docker containers, albeit characterized so far as an “experimental project.” The Containers Service Broker for Cloud Foundry can be found on GitHub.

Developed to be a generic containers service broker, it enables Cloud Foundry operators to expose and provision/unprovision services offerings that run inside a Docker container, and bind/unbind applications to the service.

Docs

The most complete and current documentation for Cloud Foundry can be found here.

2.7 The Cloud Foundry ecosystem

The Cloud Foundry ecosystem can best be viewed through the lens of the 60+ members of the Cloud Foundry Foundation, the non-profit organization formed in 2014 to further the collective development of the platform.

Key concepts

The ecosystem is a key concept for technology providers who cite a lack of vendor lock-in, but wish to be taken seriously enough to do significant enterprise IT business. The reality is that any company that offers a proprietary technology is working on an expectation of vendor lock-in. Architects should examine the Cloud Foundry ecosystem for companies who can provide their services and solutions without locking others out.

Future-proofing is the corollary to vendor lock-in; architects need to know they're not going with something that will be obsolete in a few years or enmesh them into continuing with technologies that are not at the leading edge.

In its mission statement, the Cloud Foundry Foundation says that its members are working to create "the global industry standard open source PaaS technology with a thriving ecosystem; to deliver continuous quality, value and innovation to users, operators and providers of Cloud Foundry technology; and, to provide a vibrant agile experience for the community's contributors that delivers the highest quality cloud-native applications and software, at high velocity with global scale."

To what degree all members can and will be held accountable to this credo is a matter that each enterprise should debate internally. Beyond that, it does seem valid to say that as increasingly data-driven, mobile cloud architectures emerge along with the nascent Internet of Things culture, that Cloud Foundry in general will continue to be at the leading edge.

2.8 Required organizational skills

Cloud Foundry is not a programming language, nor does it require the use of one. Commands in Cloud Foundry are simple and relatively intuitive. They leverage the functionality within Cloud Foundry to deploy and manage the complexity of the underlying infrastructure.

A classic Cloud Foundry command known to students of the platform launches a theoretical Java web app called "spring-music" as follows:

```
cf push spring-music -i 2 -m 512M -n spring-music-v1 -p build/libs/spring-music.war
```

A review of the latest Cloud Foundry commands and usages is published by the Cloud Foundry Foundation.

Enterprise IT managers should assume that people who will use Cloud Foundry have some familiarity with programming, so that they can fully understand what is occurring during app deployment.

Should the Cloud Foundry users wish to become contributors to Cloud Foundry, the Foundation has set up a "dojo" that cuts the typical ramp-up time for a committer from about a year to just a few weeks.

The most important organizational skill needed by Cloud Foundry adopters is awareness in the company's C-suite and other management positions that the platform represents a fundamental change from centralized, stovepiped applications to highly distributed services across virtual infrastructures.

The use of Cloud Foundry also leads enterprises down the path to faster app development and deployment, a continuous delivery model, and even to the emerging DevOps culture.

2.9 Implementation costs

As with all enterprise-grade technology, the total cost of ownership should be considered over specific price points for specific services.

For instance, Pivotal offers a simple calculator that shows straight line pricing from one instance of 128 MB memory (at $2.70 per month) to 16 instances of 2 GB memory (at $691.20 per month). This sort of thing has been put in use by Amazon Web Services for years as a way to pre-price its infrastructure services. See the latest Amazon Web Services pricing.

Such pricing models have shown a steadily declining cost-per-unit over the years, although a typical build-vs.-buy analysis will show that companies with capital investment funding will save money on a year-to-year basis.

The advantage of instant pricing is the ability to scale up and test applications on a short-term basis without having to commit to a capital expense that will become sunk costs. But it does not take into account the longer term total cost of ownership, and does not consider the longer term implications of adopting PaaS in an organization.

Downloading the open-source version of Cloud Foundry is free, as one would expect. Enterprises who opt for this approach then would plan for how many people will be engaged with Cloud Foundry and making it work for the organization—and at what cost—and how much should be budgeted (if any) for outside integration services, as well.

For implementing the platform, it is essential to select a Cloud Foundry distribution and an IaaS. Depending on this combination, the implementation cost could differ significantly. The most popular combinations are PCF upon AWS or OSS Cloud Foundry on top of OpenStack.

The PCF+AWS solution will require, at least, 13 x t2.micro, 15 x t2.small, 2 x m3.medium, 6 x m3.xlarge, 3 x m3.2xlarge and 1 x db.m3.xlarge, worth more than $3,200 per month. In addition, you will need a Pivotal CF license.

Alternatively, you can use OpenStack as an IaaS, paying only for hardware, and install the open-source distribution of Cloud Foundry. As a result, you will get all the required software almost free of charge.

2.10 Cloud Foundry in use

There are use cases for Cloud Foundry across a variety of industries. The technology is best used in larger enterprises, those with hundreds of developers and hundreds to thousands of apps to deploy. Not surprisingly, major technology companies can intuitively understand the benefits of Cloud Foundry, and are among the earliest and savviest CF adopters. A few major examples:

Intel operates 64 data centers worldwide, with 80% of its servers now virtualized. It has standardized on open-source Cloud Foundry as its PaaS solution, after an evaluation process that began in 2012. The company's employees use 147,000+ devices, including 43,000+ handheld devices, with 57 mobile apps having been developed. Intel's cloud journey embraces a private-cloud approach in which the company aims to benefit from cloud computing for its high internal-only workloads.

General Electric. To GE Software, a thing within the Internet of Things might be a locomotive, an aircraft engine, or a CT scanner. With Cloud Foundry, the company can deliver the functionality it needs, with big cost savings. Cloud Foundry also enables creation of Industrial Data Lakes in aviation, transportation, and healthcare. Now GE is developing Predix—an IoT platform based on CF.

Cisco uses Cloud Foundry to deploy much of the functional code of the Cisco Collaboration Cloud. This new platform has a differentiated hybrid architecture based on a service-oriented architecture (SOA), and was designed to be an engine for innovation for Cisco and its partner ecosystem. The Cisco Collaboration Cloud was featured recently on The Jimmy Kimmel Show, during a broadcast that followed Oscars 2015, the 87th Academy Awards. The cloud was used to create the "Wall of America," a mass collaboration effort that will be featured on the show on a regular basis. Viewers simply sign up with their e-mail, login during the show's taping hours, and a random selection of them is featured on the program's wall.

Philips: The Netherlands-based electronics giant has undertaken a strategy to make IT central to its vision for services and experiences involves the use of Cloud Foundry. An example is provided by the HealthSuite Digital Platform, which faced limitations imposed by complex legacy systems. In some geographies, Philips supported small clinics with services deployed from a small server footprint. On the East and West coasts in the US, the company had large sets of clinics that would add a terabyte of data every month for numerous patients. In China, clinics supported almost a million of patients or potential patients. Accessing public data or private, HIPAA-compliant data sources required developers to write up different types of code in each instance in order to access and then analyze data.

AT&T: Cloud Foundry and IBM Bluemix play a role in the telco giant's M2X project. From trucks and turbines to vending machines and freight containers, M2X enables devices that power businesses to connect and share data. It is a cloud-based fully managed time-series data storage service for network connected machine-to-machine (M2M) devices and the Internet of Things (IoT). AT&T is also positioned for IoT deployments through a visual development editor called Flow Designer, which integrates IBM's node-red IoT "wiring" tool.

There are several other prominent Cloud Foundry use cases, in fields as diverse as media, education, and analytics:

Axel Springer, the global publishing company headquartered in Berlin, is in the midst of a major digital transformation of its business from the print world to online. It uses Cloud Foundry as a key component in the transformation. With Cloud Foundry, the company is realizing vision of an innovative, fully automated service delivery platform, accelerating the time and quantity-to-market rate, and reducing IT costs.

Warner Music Group must serve its own developers, other employees, and customers in all regions of the world. It comprises an array of businesses aimed at helping artists to achieve long-term creative and financial success—while providing consumers with the highest-quality music content available. It has extended Cloud Foundry into its multi-datacenter environment and also created a unique cloud console for its developers and non-developers alike. Part of its approach lies with using the Cloud Foundry User Account and Authentication (UAA) server to provide an interface to apps running on top of CF. It also runs a second UAA as a Cloud Foundry application itself, managed by BOSH, where it can design and deliver its modifications. It has integrated UAA with Microsoft Active Directory, as the company's IT originally came from a traditional Microsoft-centric enterprise.

CoreLogic is the market-leading provider of comprehensive property information, real estate transaction services, and analytics for mortgage lenders and servicers, capital market investors, and real estate sales professionals. After years of growth through mergers and acquisitions, the company's IT leaders realized they were supporting a complex enterprise infrastructure with siloed, outdated technology stacks and disparate datasets. The systems contained redundant data and were expensive and time consuming to maintain. To modernize, reign in excessive costs, and improve application development agility, CoreLogic established the Innovation Development Center spearheaded by Leurig. One goal was to consolidate 700+ apps down to approximately 300 to reduce maintenance costs and decrease data redundancy. A second goal was to establish an ecosystem that improves data management to facilitate innovation and drive faster product delivery.

Anchora is a leading cloud platform and services provider in China, participating in all layers of the stack. It has built the Shanghai Jiao Tong University (SJTU) PaaS, a community cloud PaaS, using Cloud Foundry and OpenStack. This community PaaS serves more than 10,000 professors, instructors, and researchers, and more than 40,000 undergraduate and graduate students. It's been designed to provide an agile cloud application platform for R&D and teaching.

Channel IQ operates from its headquarters in Chicago. The company is the leading provider of online retail intelligence solutions for manufacturers, distributors, and retailers. It works with CenturyLink's Cloud Foundry distribution to create a flexible cloud platform. Transforming its traditional software offering into Software-as-a-Service (SaaS) met Channel IQ's mission to provide the easiest access to the best data. The challenge was to scale to its customer base without a costly re-architecting of its application, and without having to manage the complexities of the cloud platform.

Monsanto's vision for sustainable agriculture sees farming in the future as increasingly information-driven. The company has leveraged Cloud Foundry in its strategy to use PaaS as an internal cloud and increase the organization's adoption of agile DevOps and Continuous Deployment. It set out to adopt a technology stack that is open and which allows portability among a variety of IaaS providers. Within this PaaS strategy, Monsanto also has goals to increase developer agility and productivity by enabling automated self-service application management, and to keep the cost of entry low.

LDS Church's IT department wished to address several business problems: slow provisioning time with hundreds of apps, an infrastructure that was increasingly difficult to manage, and a bias toward fault tolerance over scale for most of its apps. It now runs Cloud Foundry on vSphere and deploys more than 200 applications. CF has become the platform of choice for deploying all of the organization's custom applications, using a small team of three developers, three operations personnel, and an intern. Cloud Foundry's self-service capability and customizability feature

prominently here, as numerous custom service brokers facilitate integration with existing systems. The self-service model has also been extended to other teams within the organization, managing Oracle databases, for example.

Exterro is a recognized leader for legal governance, risk, and compliance management software solutions. It is also a pioneer in workflow management technology for electronic discovery, an industry that is expanding at a rate of 40–50% annually. It needed a secure, enterprise-grade platform that could integrate with the firm's existing processes and service levels, yet would be scalable to flex and grow. With CF, Exterro now runs all of its operations in a secure, enterprise-grade cloud that allows it to pass on enterprise-grade service level agreements (SLAs) to its own customers, while also allowing it to spin up quickly and scale effortlessly as their customers' litigation needs change. The company has also significantly lessened its QA workload, while enhancing its ability to serve customers in new ways and reach out to new markets.

Healthcare provider: Cloud Foundry can be deployed as part of a solution to solve complex, distributed computing problems and to save expense and drive new revenue. It can also be part of solutions to save lives. This is the top-level takeaway from a Cloud Foundry project undertaken by Altoros for a major healthcare systems provider. The project involves the automation of medical devices provisioning for 300,000 systems spread across 1,000 healthcare facilities. The solution processes and manages 7 terabytes of data per day, incorporating two regional datacenters. The Ubuntu operating system and OpenStack reside at the bottom of the stack to provide core infrastructure. The Cloud Foundry app engine works in concert with RabbitMQ, Cassandra, and MariaDB to deploy the solution to the drug-delivery apps.

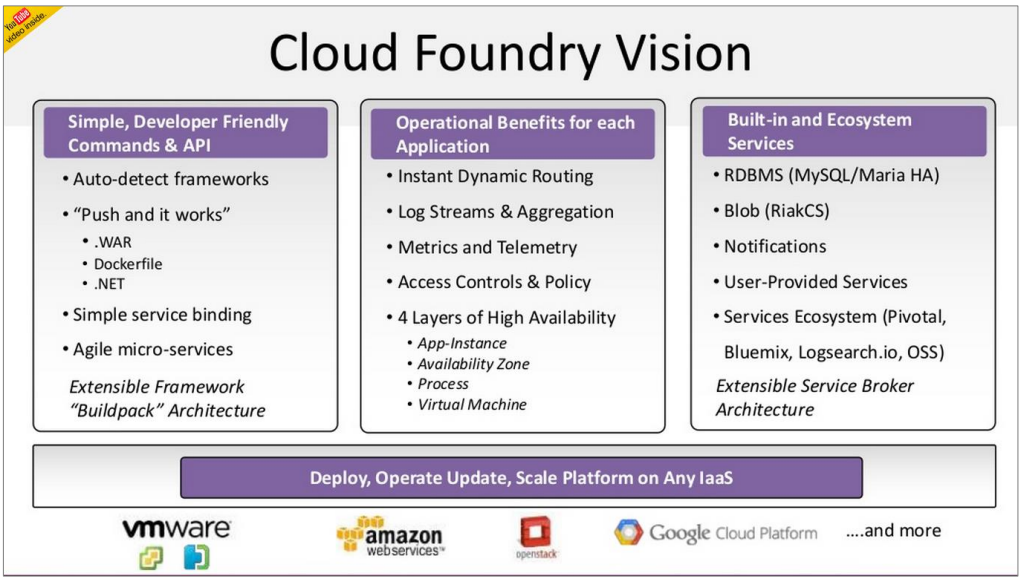
2.11 The Cloud Foundry roadmap

A fairly detailed look at the immediate Cloud Foundry roadmap is found in a slide presentation from Pivotal, presented at the Cloud Foundry Summit in 2014.

There are six key bullet points contained in this vision:

- global industry standard and scale
- open source PaaS
- thriving ecosystem
- continuous delivery
- agile experience for contributors
- cloud-native apps and software

Visually, it looks like this:



Additionally, what's known as Diego is a key thrust at the moment. In essence, Diego will consolidate and simplify many features and much functionality within Cloud Foundry. A technical blog from 2014 provides details of its first phase. Starting 2017, the Diego runtime is required for Cloud Foundry Certification.

The use of BOSH is another major topic of discussion, present and future. The tool chain can be difficult to master, and there is an alternate way to accomplish its tasks using Canonical's orchestration solution called Juju Charms. Here is a presentation from Altoros on the topic and a blog post about the use of BOSH.

BOSH 2.0 significantly simplifies generation of deployment manifests, offering such capabilities as:

- dynamic IP management
- global cloud config
- first-class support for multi-AZ job striping
- manifest enhancements

What does this all mean from the high-level viewpoint?

The key point is that the Cloud Foundry community is committed to ensuring that the platform runs on most major IaaS systems. Applications and services must be portable. This is the open-source vision of the Cloud Foundry Foundation, to be sure, even as it seems to run counter to the idea of the commercial, proprietary distributions being offered by Pivotal and others in the market.

2.12 Cloud Foundry and the Internet of Things

The Internet of Things (IoT) promises to connect billions of devices and generate millions of petabytes of data within a few years. It represents a fundamental shift in how the Internet and World Wide Web are being used, in that data is created on the edges of an IoT deployment, by digital cameras and sensors.

The IoT will be creating vast new stores of data—think of "data lakes" of information generated in small packages by sensors at hospitals, along roads, and throughout manufacturing. New generations of analytics software will plumb the depths of these lakes for reports and actionable insights.

Microservices and containers seem to be destined to play a major role in deploying the discrete parts of modern and future IoT deployments, and the role of PaaS will remain at the center of all this.

Some Cloud Foundry distributions have already been adopted for IoT needs. These include GE's Predix, IBM Bluemix, and SAP HANA.

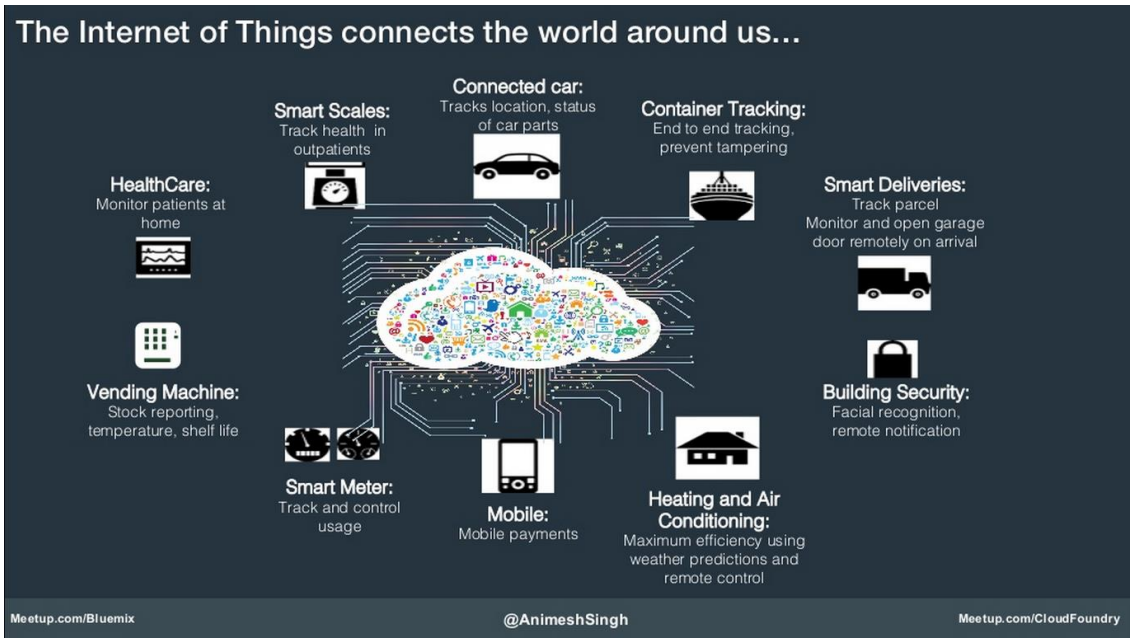
Predix is claimed to enable industrial-scale analytics for asset performance management (APM) and to optimize operations by providing a standard way to connect machines, data, and people.

The SAP Cloud Platform is designed to facilitate and support the implementation of IoT applications. It provides tools for connecting devices and recording their specific data types, enabling secure transmission of device data to a database, storing the data, and providing easy access to it.

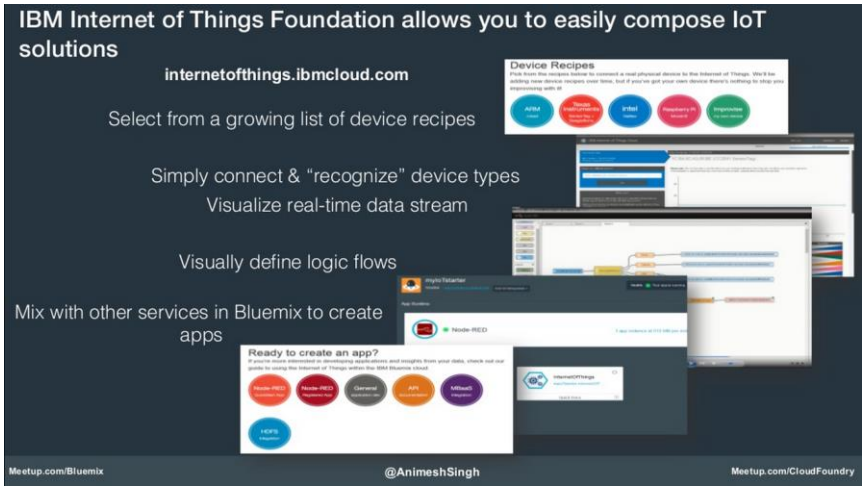
IBM has already equipped Cloud Foundry for the IoT. Its Bluemix strategy, which includes Cloud Foundry as the PaaS, has been extended to the IBM IoT Foundation, which offers several features for IoT deployments:

- device registration
- device and application connectivity
- securely receiving data and sending commands to devices
- storage and access to historic data

A high-level view of how the IoT will connect to several types of "things" is provided in this slide from IBM's architect Animesh Singh:

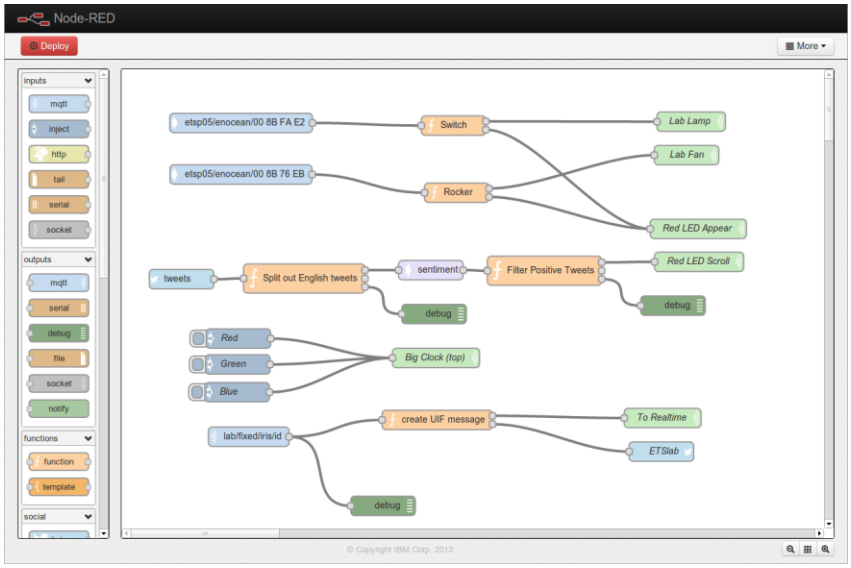


IoT Foundation "provides simple, but powerful application access to IoT devices and data," according to the company, designed to "rapidly compose analytics applications, visualization dashboards, and mobile IoT apps." An overview of the IoT Foundation's core capabilities is shown below:

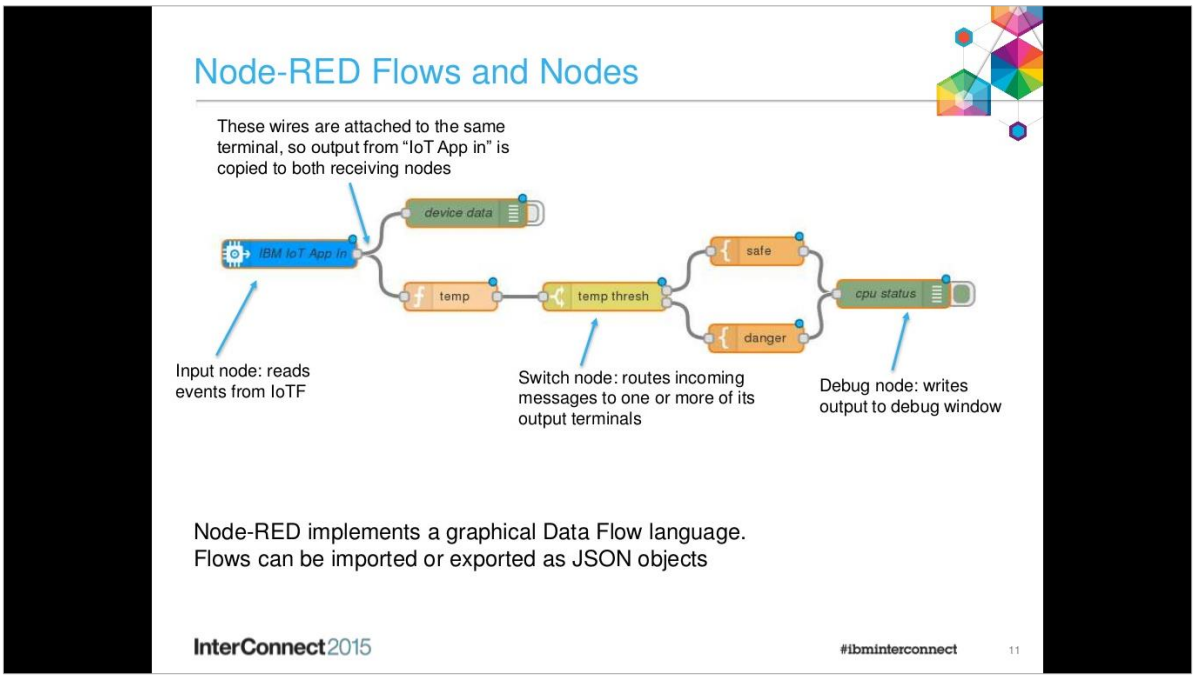


The entire presentation from Animesh Singh on Cloud Foundry and the IoT can be found here. Information about the IoT Foundation is available online.

IBM is also behind a technology called Node-RED, a tool designed for ease in wiring together hardware devices, APIs, and online services. A node-red screenshot appears below:



A SlideShare presentation from node-red is found here. A self-explanatory screenshot from that presentation:



3. Cloud Foundry POC

3.1 Ideas for your Cloud Foundry POC

A Cloud Foundry POC has the best chance to be successful with cloud-native workloads and staying away from monolithic apps on a single server. Applications that will find the most success will meet the following criteria:

- Do not write to the local file system
- Short-lived/ephemeral
- Instances of the same app do not share a local file system
- Do not persist or replicate HTTP sessions
- Take into account HTTP and HTTPS port limitations

In general, any 12-factor app will work without modification. Here's a quick review of 12-factor apps.

Other key concerns for applications that will run on Cloud Foundry:

Concern	Description
Dependency management	Environments differ, so one should never rely on implicit existence of system-wide packages, libraries, etc. There are great tools to manage dependencies, such as Bundler and Maven, but nothing as effective as building and packaging an app to contain the very same dependencies no matter what environment it is running in. It is also recommended to build the workflow on tools that help to create reproducible environments. Consider using Docker for these purposes.

Stateless sessions	Applications keep user and app data in HTTP session variables. If there is more than one app instance running behind a load balancer then a 'Sticky Session' pattern has to be implemented. Generally, it is better to build applications that do not store anything in sessions. If that is impossible, 'Sticky Sessions' is a good alternative.
Local disk storage	This concerns applications that need to share files through a local file system. In a PaaS, instances are ephemeral. They can crash and new ones can be automatically launched. You can scale down, destroying the instances that are no longer necessary, etc.
Configuration variables with environment variables	Store `config` in environment variables. Hardcoded `config` goes against maintaining multiple environments.
Available ports	PaaS systems do not allow for having multiple ports opened. Only ports 80, 443, and 4443 (Web sockets) are allowed. Do not rely on port numbers at all, because an application can be started in a container using a port provided with the PaaS. For instance, see the `$PORT` variable in Cloud Foundry env variables.
Runtimes and frameworks	If the application server, framework, etc. is customized, the corresponding buildpack will also need to be customized. Try using standard frameworks and libraries to avoid problems with PaaS integration.
Background applications (with no Web interface)	If the application runs as a background job, custom deployment parameters need to be provided, so that the PaaS does not consider it to be down / crashed. In case of Cloud Foundry, the --no-route option will need to be added to push a command.
Stateless processes	In Cloud Foundry, processes can be recreated in a separate container environment in order to scale application. In this case, new processes (application instances) will know nothing about the local state of other instances.
External services	The connection to external services must be used wisely as it can take time to bring traffic to them. Security questions must not be forgotten. It is strictly recommended to use SSL. An even better option is to use services that are embedded in Cloud Foundry.
Separation of concerns	Decouple work between processes and increase separations of concerns. Processes can be started, and errors localized, faster.
Hardware architecture dependencies	Avoid using precompiled components. Do not use database data files and other components that may rely on a specific architecture.
CI tools	Running tests on the PaaS side will increase reliability of deployments.
Ignored files	Not everything needs to be pushed. Avoid sending temp files and configuration files, for example, with credentials or individual setup to Cloud Foundry.
System libraries	Use framework/language libraries to perform system work. For instance, it is better to use a `Tempfile` object rather than create something in the `/tmp/` directory.

3.2 How Cloud Foundry works with existing infrastructure

Cloud Foundry is built to provide portability between different types of infrastructures. The table below summarizes the types of infrastructures that can be used for a POC.

Infrastructure for a POC	Description
IaaS or virtualized infrastructure	Cloud Foundry can be deployed to an existing IaaS that provides virtualization with on-demand services or a virtualized platform, such as vSphere. Currently, there are BOSH CPIs for AWS, OpenStack, Google Cloud Platform, Azure, Photon, vSphere, vCloud Air, and vCloud Director. It is also possible to create a custom BOSH script to deploy Cloud Foundry to any virtual platform.
A local machine	Cloud Foundry can be installed locally, using Vagrant and Warden BOSH CPI. At this moment, bosh-lite supports VMware Fusion, Virtualbox, and AWS. A local deployment is good for trying out some CF features. But such a deployment will not be distributed and will not provide shared resources. It may thus not be suitable for a POC.

Hardware requirements and cluster sizing

The hardware requirements for Cloud Foundry clusters vary depending on the architecture, number and type of services, type of infrastructure, etc. For example, typical requirements for a Pivotal Cloud Foundry deployment on vSphere are as follows:

- 80 vCPU, 28 GHz
- 120 GB of RAM
- 2 TB of hard disk space

The recommended requirements suitable for a POC:

- eight physical CPU cores
- 128 GB of RAM
- 1 TB of disk space
- two NICs

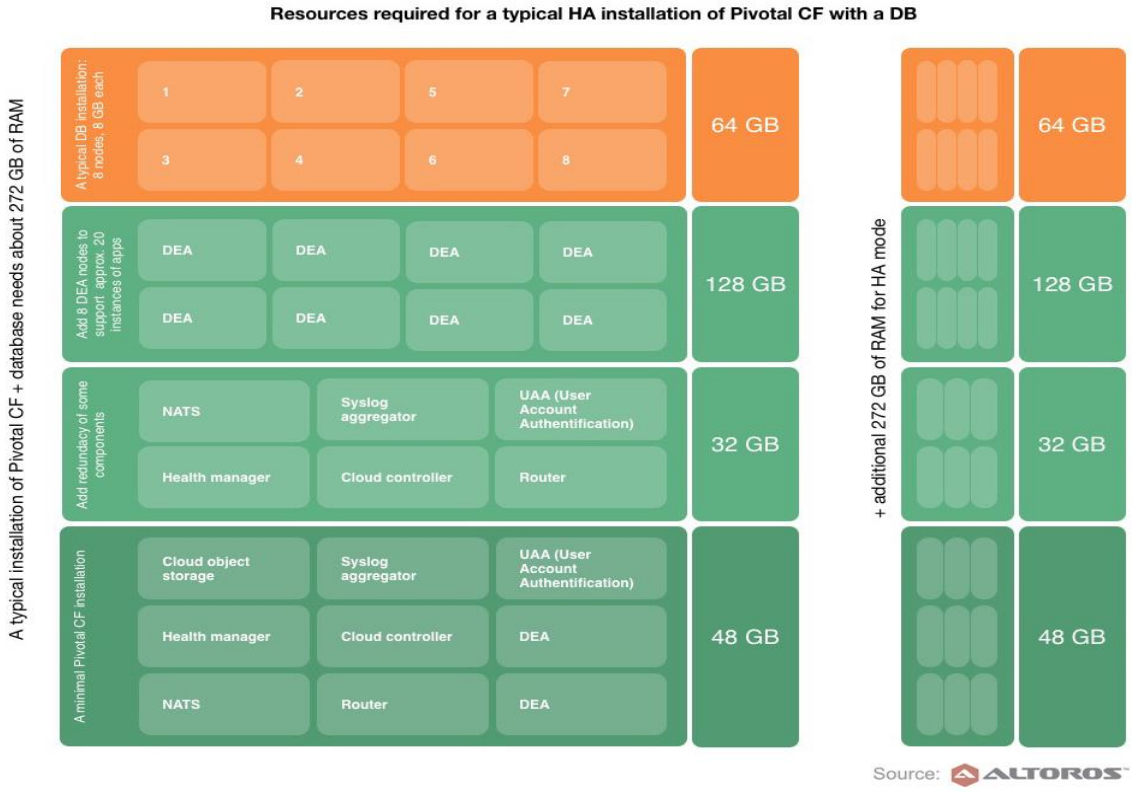
For OpenStack, the requirements are as follows:

- 56 vCPU
- 112 GB of RAM
- 1 TB storage

The Cloud Foundry documentation has more detail.

A minimum Pivotal CF deployment on OpenStack is described in the Pivotal documentation. This deployment consists of 26 instances and requires 42 GB of RAM, 27 CPU cores, 165 GB for ephemeral disk, and 123 GB of persistent disk. These figures will have to be doubled to provide high availability.

The diagram below also provides sizing guidelines for a typical Cloud Foundry POC cluster.



3.3 Choosing an IaaS for Cloud Foundry

Cloud Foundry is considered to be IaaS-agnostic. The key to this feature lies in BOSH's Cloud Provider Interface (CPI), which will allow Cloud Foundry to communicate with an IaaS. This decouples the platform from the infrastructure, thus preventing vendor lock-in. (A list of offerings is provided in the table below.)

Cloud Foundry users can choose from many cloud IaaS offerings that are supported directly or can create a CPI to interact with an IaaS that’s not on the supported list.

There is similarity among the available IaaS offerings in that they all provide server virtualization, storage, managed services, and other core functionality. So, how to make a choice?

Here are some considerations:

- **SLA**. This can be taken as a starting point. IaaS cloud SLAs vary widely, so a legal team needs to clarify sections as required for an enterprise’s specific use case. It is helpful to find customer references and their experience with a specific SLA.
- **Risk management**. Look for risk management qualifications and standards, such as HIPAA and PCI compliance.
- **Location, location, location**. It is important to take in consideration where customers are coming from, as there can be significant legal implications in keeping data from customers from abroad. Server location can also affect the speed perceived by customers.

- **Pricing**. It can be difficult--impossible--to produce a complete apples-to-apples cost comparison. It's important to beyond basic prices for given resources, and create a TCO picture that takes into account management, longer term cost, and build-vs-buy considerations.
- **Customer reference**. References can be of invaluable assistance in learning about support response, downtime, and overall satisfaction.
- **Benchmarks**. It can be quite useful to conduct your own benchmarks if there is uncertainty about specific IaaS performance.

IaaS offerings for Cloud Foundry

IaaS	Support / Docs	BOSH CPI	BOSH CPI Docs
Amazon AWS	Officially supported by Cloud Foundry. Very good documentation and examples.	https://github.com/cloudfoundry/bosh/tree/master/bosh_aws_cpi	https://bosh.io/docs/aws-cpi.html
OpenStack	Officially supported by Cloud Foundry. Very good documentation and examples.	https://github.com/cloudfoundry/bosh/tree/master/bosh_openstack_cpi	http://bosh.io/docs/openstack-cpi.html
vCloud	Officially supported by Cloud Foundry. Very good documentation.	https://github.com/cloudfoundry/bosh_vcloud_cpi	http://bosh.io/docs/vcloud-cpi.html
vSphere	Officially supported by Cloud Foundry. Good documentation.	https://github.com/cloudfoundry/bosh/tree/master/bosh_vsphere_cpi	http://bosh.io/docs/vsphere-cpi.html
Microsoft Azure	Community maintained. Good documentation.	https://github.com/nterry/bosh_azure_cpi	https://github.com/nterry/bosh_azure_cpi/blob/master/README.md
CloudStack	Community maintained. Average documentation.	https://github.com/cloudfoundry-community/bosh-cloudstack-cpi	
Google Compute Engine (GCE)	Community maintained. Documentation is not so good.	https://github.com/cf-platform-eng/bosh-google-cpi/tree/google-cpi	https://github.com/cf-platform-eng/bosh-google-cpi/blob/google-cpi/bosh_google_cpi/INSTALL.md
Photon	VMware maintained. Documentation is not so good.	https://github.com/cloudfoundry-incubator/bosh-photon-cpi-release	https://github.com/cloudfoundry-incubator/bosh-photon-cpi-release/blob/master/PHOTON_CPI.md

3.4 How to backup and restore Cloud Foundry

Cloud Foundry's (and BOSH) backup and recovery capabilities are currently somewhat limited, with procedures relying on the IaaS of choice. The procedures involve creating snapshots of the created disk volumes to back up the disk contents, and using database tools to backup and restore data content.

Some points to consider:

- **Colocation**. Deploy each component in a separate physical instance. In this way only a catastrophic failure will force a deployment of all the Cloud Foundry components.
- **Configuration**. Keep all *yml* and any other configuration files backed up and up-to-date. With those ones in hand, recovering Cloud Foundry is fairly easy.
- **Data**. Cloud Foundry will not perform data backup. This must be done through the IaaS or with separate backup and restoration tools.
- **Disaster recovery guide**. There is a GitHub guide that is the closest thing currently to official documentation on how to backup and restore Cloud Foundry. BOSH also has a disaster recovery guide.

3.5 Integrating other technologies

Cloud Foundry's nature as a catalyst for deploying applications and services means that it integrates with other open-source and commercial technologies.

Cloud Foundry was originally designed for private cloud, so it can be deployed on a virtualized platform, such as VMware's vSphere, or on a local machine.

Since Cloud Foundry's initial release, BOSH scripts have been developed to integrate Cloud Foundry with public cloud services (e.g., AWS, GCP, Azure, vSphere, or vCloud Air) and private clouds, such as OpenStack and Photon. There are some good sources for use of Cloud Foundry with these platforms.

OpenStack

From the ActiveState blog: "If you are looking at full stack dynamic cloud architecture to deploy applications, then OpenStack plus Cloud Foundry is the way to go."

From the OpenStack Summit 2013: https://www.openstack.org/summit/portland-2013/session-videos/presentation/cloud-foundry-your-paas-on-openstack

From the Cloud Foundry Foundation docs: http://docs.cloudfoundry.org/deploying/openstack/

AWS

From the Cloud Foundry Foundation docs: http://docs.cloudfoundry.org/deploying/ec2/

GCE

http://googlecloudplatform.blogspot.com/2014/05/see-what-cloud-foundry-is-doing-with-google-compute-engine.html

vCloud Air

http://vcloud.vmware.com/

Other integrations

Beyond the obvious issue of integration with infrastructure, the use of Cloud Foundry involves technology in several other areas. Some of the more popular areas and technologies are listed in the table below.

Category	Technology	URLs
Configuration Management	Bcfg2 (bee-config)	http://bcfg2.org
	CFEngine	www.cfengine.com
	Chef	https://www.chef.io/chef/ http://www.infoq.com/presentations/cloud-foundry-bosh
	Puppet	http://www.infoq.com/presentations/cloud-foundry-bosh
Containers	Docker	www.docker.com
Continuous Integration	CloudBees (based on Jenkins CI)	www.cloudbees.com
	Jenkins CI	https://cloudfoundry.cloudbees.com/index.html
Databases	Cassandra	www.cassandra.apache.org http://www.datastax.com/what-we-offer/products-services/datastax-enterprise/apache-cassandra
	Couchbase	www.couchbase.com
	MongoDB	www.mongodb.org
	Redis	www.redis.io http://blog.pivotal.io/cloud-foundry-pivotal/products/redis-in-action-with-cloud-foundry
Frameworks	Hadoop	https://hadoop.apache.org
	Grails	https://grails.org
	Play	www.playframework.com

	Ruby on Rails	http://rubyonrails.org
	Django	www.djangoproject.com
IDE	Eclipse	https://eclipse.org/ https://marketplace.eclipse.org/content/cloud-foundry-integration-eclipse http://docs.run.pivotal.io/buildpacks/java/sts.html
Log Management	Splunk	www.splunk.com
	LogLogic (Tibco)	www.loglogic.com
	Loggly	www.loggly.com
	Sumo Logic	www.sumologic.com

These are the most popular service brokers for Cloud Foundry:

- Cassandra
- Memcache
- MySQL
- RabbitMQ
- Redis

3.6 High availability requirements

High availability (HA) refers to the ability of a system to stay online continuously for a specified period of time. The task becomes significantly more difficult and expensive as expectations rise, and is often specified in a service-level agreement (SLA).

As an example, a “four-nine” or 99.99% availability equates to almost a full hour of downtime per year, an amount which can be more than needed, perfectly acceptable, or clearly unacceptable, depending on the application or service.

Cloud Foundry provides a series of tools designed to satisfy a customer’s availability requirements. The platform’s HA abilities can be divided into four layers:

1) Ability **to deploy all Cloud Foundry components** to different availability zones and keep the traffic flowing to the remaining functioning availability zones if one or more are compromised.

2) Ability **to restore and restart failed Diego Cells**, which means the required amount of application instances will be up and running.

3) Ability **to restore and restart a failed elastic runtime process**. If any of the elastic runtime processes fails, Cloud Foundry’s Health Monitor will restart that process and send e-mails, pages, or any necessary alert to a designated destination.

4) Ability **to restore and/or restart a failed VM**. As with the elastic runtime process, if a whole virtual machine goes down, Health Monitor can spin up a new VM with the appropriate machine image.

It is critical to note that Cloud Foundry does not provide HA for databases. Customers must plan accordingly to provide availability for databases outside the Cloud Foundry deployment.

The following are the minimum numbers of scalable processes to achieve HA with Cloud Foundry (source):

Component / Process	Amount
Diego Cell	≥ 3
Diego Brain	≥ 2
Diego BBS	≥ 3
Consul	≥ 3
PostgreSQL Server	1 or 0
MySQL Proxy	≥ 2
NATS Server	≥ 2
Cloud Controller API	≥ 2
Cloud Controller Worker	≥ 2
Router	≥ 2
HAProxy	0 or 1
UAA	≥ 2
Doppler Server	≥ 2
Loggregator TC	≥ 2
etcd	≥ 3

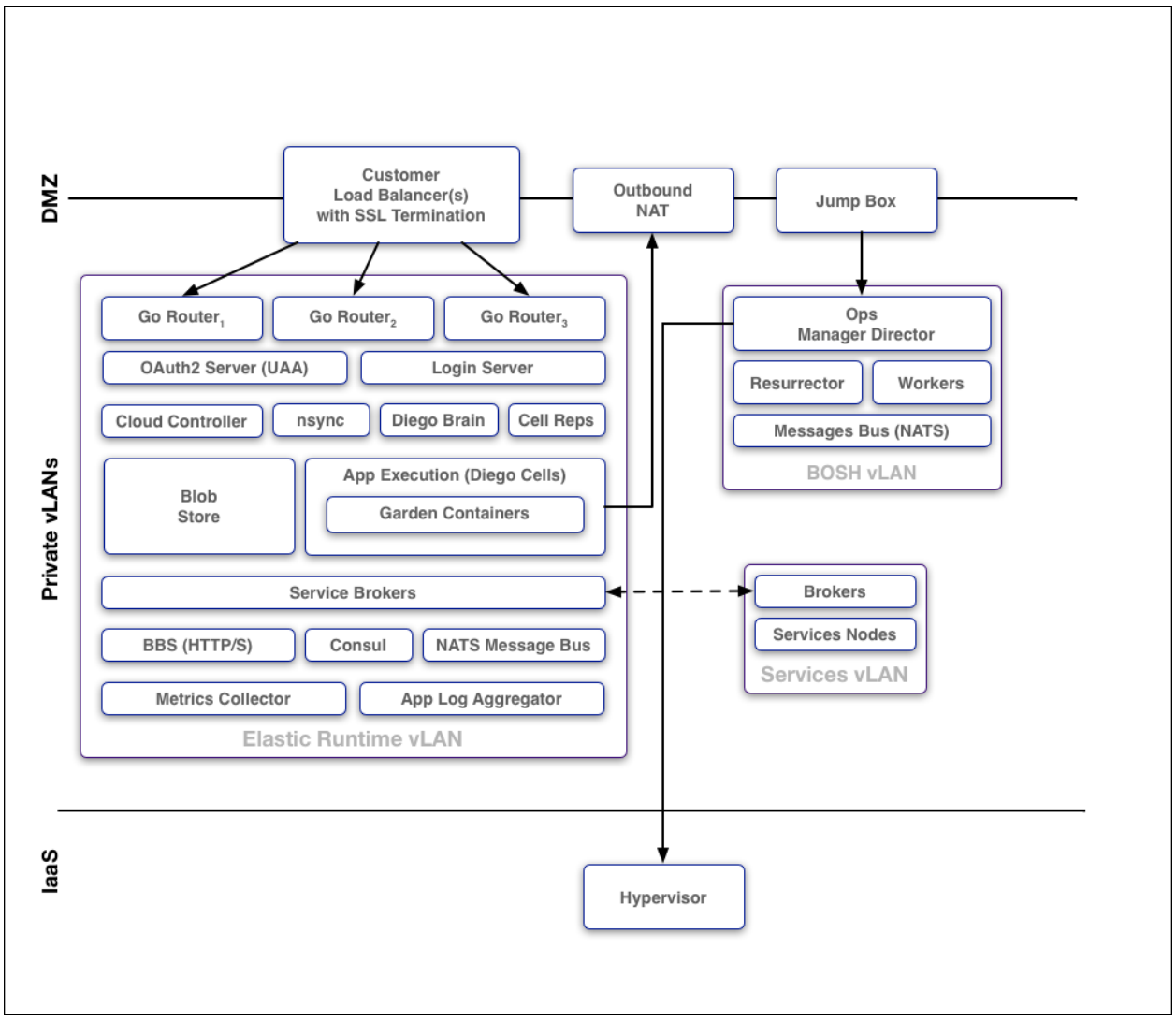
3.7 Security requirements

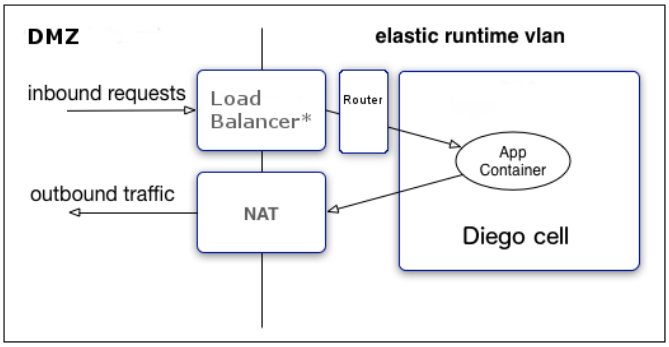
In all applications, some security requirements are needed in order to protect the data, and therefore, the business. The infrastructure that contains the necessary artifacts to run the application also needs to be thoroughly protected, not only from external attacks but also from internal mismanagement and accidental actions.

Cloud Foundry was designed to have a tight security built in:

- RBAC (Role-Based Access Control) is used to allow Cloud Foundry users to affect only the space to which they were granted access.
- In the case of multi-tenant environments, application security is ensured.
- DoS (Denial of Service) attacks are prevented by resource blocking and starvation.

A couple of diagrams from Pivotal illustrate the process in a high-level way.





External requests: The sole point of contact with the outside world is thru a load balancer that you need to provide. This load balancer will be passing requests to Cloud Foundry's Router (one or more), so the Diego Cells are not exposed.

Internal security: All virtual machines are protected by using Linux *iptables*. This ensures that only requests from known IPs inside a Cloud Foundry deployment will be accepted.

3.8 Authorization process

For authentication and authorization, Cloud Foundry uses the User Account and Authentication (UAA) service. UAA provides centralized identity management. It acts as an OAuth2 Authorization Server—i.e., issues digitally signed JSON Web Tokens to grant access to platform resources for applications acting on behalf of Cloud Foundry users or themselves.

It authenticates users with their Cloud Foundry credentials (it can also provide SSO using those or other credentials), and can store user passwords in the UAA database using *bcrypt* (if configured by operators), or connect to external user stores through LDAP and SAML (e.g., Microsoft Active Directory). In this case, users can access Cloud Foundry with their corporate credentials without creating a separate account.

The UAA server is implemented as a Spring MVC Web app. It supports a number of APIs, including:

1) the `/authorize` and `/token` OAuth2 endpoints
2) a `/login_info` endpoint
3) a `/check_token` endpoint
4) an SCIM (System Cross-domain Identity Management) endpoint
5) OpenID connect endpoints

More information on APIs supported by the User Account and Authentication Service can be found in this repository. Cloud Foundry documentation also has a detailed overview of UAA.

3.9 Monitoring Cloud Foundry

Cloud Foundry provides all the necessary metrics for monitoring apps, services, users, and its own components. These can be collected by various external monitoring services, such as Zabbix or Splunk. Adding them is a trivial task that can be easily accomplished. Out-of-the-box, open-source Cloud Foundry has basic monitoring capabilities only. For instance, BOSH CLI can show parameters for Cloud Foundry jobs—such as CPU load or RAM usage. CF CLI can get application parameters—such as disk usage, CPU load, memory usage, or basic logs.

Monitoring the platform

There are plenty of ways and tools you can use to monitor Cloud Foundry components.

Tool / Approach	Description
OpsConsole	Available in Pivotal CF (PCF)
AdminUI	Available for Pivotal CF and OSS Cloud Foundry
Third-party tools	Including custom solutions
Cloud Foundry CLI	Only provides basic stats—such as CPU usage, number of instances, etc.

Let's review the tools in detail.

Criteria	Altoros Heartbeat	Pivotal Ops Manager	AdminUI	CF CLI	App Dynamics	NewRelic
Platform Metrics	Yes	No	No	No	No	No
App Metrics	Yes	Yes (with Apps Manager)	Yes	Yes	Yes	Yes
App Logs	Yes (with ELK integration)	Yes (with Apps Manager)	Yes	Yes (limited)	No	No
Service metrics	Yes	Yes	Yes	No	Yes	Yes

For these tools, data is provided by:

1) **Collector**, which aggregates metrics. Collector can be connected to Datahog, Graphite, etc. to display the metrics on the Web.
2) **Loggregator**, which aggregates logs. The logs can then be dumped to Logstash or Zabbix.

Some of the data requires configuring a buildpack, while other may need installing an agent.

Monitoring Cloud Foundry services

It is possible to monitor Cloud Foundry services with a tool like Zabbix, but you may not be able to see statistics, such as who is consuming the service and how much, how access is provided, etc. How far you can go with monitoring depends on the service broker.

Cloud Foundry also has an API that can be used for monitoring service usage. Finally, the Cloud Foundry CLI client provides some data for service monitoring.

Monitoring Cloud Foundry apps

Cloud Foundry can be easily integrated with a variety of third-party tools for monitoring apps. Other options include:

- monitoring apps using the logs collected by Loggregator (in this case, make sure the output of your logs goes out via STDOUT)
- using Pivotal CF Console (if you are running PCF)
- using the Cloud Foundry CLI
- using ELK

Monitoring Cloud Foundry users

Users can be monitored through the CLI or through Pivotal CF Console (available in Pivotal CF).

The monitoring process

If everything works well, all you need to do is check the logs from time to time to see that everything is up and running. The person monitoring the cluster needs to have a good understanding of CF components and how these components must work. If there is some kind of error, they will need to have a look at configuration files for the particular component that failed. In most cases, fixing the issue involves adjusting/restarting something or re-deploying a component (in more severe cases).

More on the topic:

- Cloud Foundry Monitoring with Admin UI: Technical Overview
- Monitoring Pivotal Cloud Foundry Applications with New Relic

3.10 Scaling Cloud Foundry applications

Scaling refers to increasing the amount of resources available for an application or service. A core appeal of cloud computing is the ability to scale the infrastructure almost immediately to meet new and/or evolving demands.

Scaling is required as the number of users increases, user activity increases, and when an application or service becomes more sophisticated and demands more resources.

Scaling can be vertical or horizontal. With vertical scaling, a server simply gets more resources—more memory, more storage, or a faster processor. The infrastructure scales *up.* With horizontal scaling, additional servers are added. The infrastructure scales *out.*

Horizontal scaling is the more common, and powerful, approach with cloud computing. Applications and services thus become *distributed* across multiple servers when scaled beyond a single server, and can become increasingly distributed across x-number of servers as needs increase.

In public cloud, the *instances* typically have a horizontal scaling effect—moving across the range of AWS instances, for example, can increase the number of CPUs available from 1 to 32. With private clouds, the story remains the same, with local administrators deploying an increased number of servers to the task.

Scaling applications in Cloud Foundry

Cloud Foundry users can scale their application horizontally and vertically: increasing the quantity of application instances or adding more memory and hard drives, accordingly.

Cloud Foundry also supports auto-scaling if apps based on different criteria, such as CPU load, memory, etc. Auto-scaling is disabled by default. To use it, the operator needs to indicate what parameters will trigger the system to scale. Automatic scaling is a very useful feature that makes it possible to increase capacity in anticipation of traffic spikes or cut infrastructure expenses by decreasing the number of instances when the workload is low.

Cloud Foundry users can employ the `cf scale` command with corresponding arguments to scale apps up or down to meet changes in traffic or demand. In this respect, this is similar to `cf push`, in

which the simplest of commands is used to accomplish several very sophisticated processes going on underneath within the infrastructure.

Cloud Foundry documentation provides more details on scaling apps.

Scaling Cloud Foundry's own components

Cloud Foundry's components may also need scaling (e.g., to accommodate for a greater number of apps). Similarly to applications, they can be scaled manually and automatically. Currently, the default method is manual scaling. The operator monitors the system and uses BOSH to add or remove instances of Cloud Foundry components. Auto-scaling can be added using the BOSH AutoScaler plug-in. Here is a great slide deck that explains how to automatically scale Cloud Foundry with BOSH.

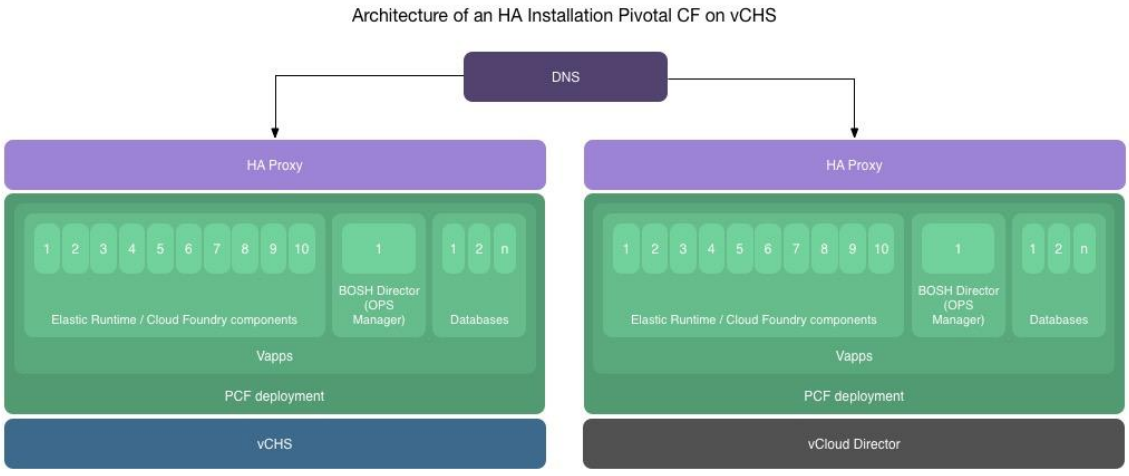
Scale vs. infrastructure expenses

Pricing can be one way to gauge how much scaling is needed, and the price of instances from a public-cloud provider should be compared to the total cost of accomplishing the same infrastructure with in-house resources. With the flexibility of cloud computing and the ability of Cloud Foundry to instantly scale up or down, short- to medium-term capacity planning can be adjusted as the true computing needs of an application or service emerge.

3.11 Availability and stability

Cloud Foundry is built around providing high availability for applications, but it is also important to ensure high availability of its own components. This can be done in several ways, e.g. by putting copies of CF components in different availability zones.

The following diagram shows the architecture for a highly available Pivotal CF deployment on vCHS. Here, we created redundancy and used load balancers with health checks.



If you have a properly set up cluster, Cloud Foundry is generally stable. App-related failures are rare and usually get fixed soon after they are detected. Problems with stability may arise if you try to change something (e.g., re-deploy releases or change your infrastructure). Most issues that can compromise stability of Cloud Foundry are due to errors in the manifest file.

So, the stability of the system also depends on how well your engineers understand Cloud Foundry (again, get training). The level of expertise for developers who install/support/administer the platform is highly important. The better they know how the platform works and how to implement redundancy, load balancing, etc., the more stable it is.

In general, there is a large community behind the Cloud Foundry project, so most things have already been fixed and containerization works well.

3.12 Setting and meeting POC goals and metrics

Measuring the success of a Cloud Foundry POC should be similar to measuring the success of any other technical project. To be considered viable/feasible, the POC will need to meet technical requirements in auto-scaling, rolling updates, automated recovery of failed instances, etc., while also creating value for the business.

Since criteria for assessment are closely related to POC requirements, the most obvious strategy is to base metrics on project goals. Consequently, the goal-setting stage is the best time to decide on what will be measured.

Metrics should pinpoint the areas where Cloud Foundry can help to cut costs or increase productivity. These will be a function, of course, of things such as the type of business, processes in use at the company, and how working hours and productivity are tracked and measured.

It can be quite valuable to evaluate overall system performance as well as the performance of certain tools running on Cloud Foundry, the availability of desired services, and how well the current technology stack is supported. System maintainability, how Cloud Foundry will evolve in the future, and how well developers respond to using Cloud Foundry can also be part of an overall evaluation.

The table below provides examples of possible Cloud Foundry POC goals and metrics.

Cloud Foundry POC Goals	Metrics
Speed up time-to-production	• time to provision infrastructure • time to release • time savings for developers • cost of lost benefits • cost of infrastructure provisioning • cost of release
Reduce total cost of ownership (TCO)	• cost of maintenance • development savings • utilization costs • savings per month
Reduce the cost of security and compliance	• time to implement the required security features • incident probability on Cloud Foundry vs. existing platform
Get automated HA and low latency	• customer satisfaction • losses due to outage

The people on the team testing the POC should be from among future platform users. This will include developers, operations, and IT engineers, all of whose workflow will be affected in a significant way.

Finally, provision should be made to provide test-site user training.

- For developers, an introduction to Cloud Foundry CLI commands that they will be using to deploy apps should be sufficient.
- For DevOps engineers who will be maintaining and monitoring the system, the training should include both theory (Cloud Foundry components, properties specified in manifest files, etc.) and practice (how to deploy and troubleshoot Cloud Foundry).

4. POC Assessment

4.1 Adjusting goals and metrics

As Tolstoy famously wrote, “All happy families are alike, each unhappy is unhappy in its own way.” If initial results of a POC are disappointing, the question arises as whether to re-attempt the POC and try to improve the results.

Common reasons for initial results to be poor include:

- Insufficient training
- Choosing the wrong apps to deploy to Cloud Foundry
- Using the wrong processes
- Resistance to cultural change within the POC team

The following table contains some questions you may want to ask yourself to understand what caused the failure:

Issues	Questions to Ask
People-related	Do architects, DevOps, and/or developers need more training? Are there enough people to implement the POC?
Processes-related	Were the right processes for the job being used? Do the people implementing the POC need to embrace some cultural changes?
Technology-related	Is the IaaS being used for the POC fully compatible with Cloud Foundry? If not, did this cause a problem? Did you select the right type of application to run on CF?

It can also be worthwhile to be sure Cloud Foundry will be optimal for the organization. Cloud Foundry has strengths, or “sweet spots,” in several areas:

1) **Company size.** An average Cloud Foundry user is an organization with 1,000+ employees and at least 100 developers who work on hundreds or even thousands of apps. For a business with fewer than 10 apps and 25 developers, a full-blown private Cloud Foundry deployment will probably be overkill. If that is the case, consider a public Cloud Foundry installation, such as Pivotal CF, IBM Bluemix, etc. to get all the benefits without the hassle.

2) **Existing technology stack.** Cloud Foundry aims to cover a maximum number of technologies, but there are some limitations. For example, the support for .NET is still limited. If the POC results are not optimal, it can be worthwhile to see if a technical limitation of Cloud Foundry is affecting things, and if it can be overcome.

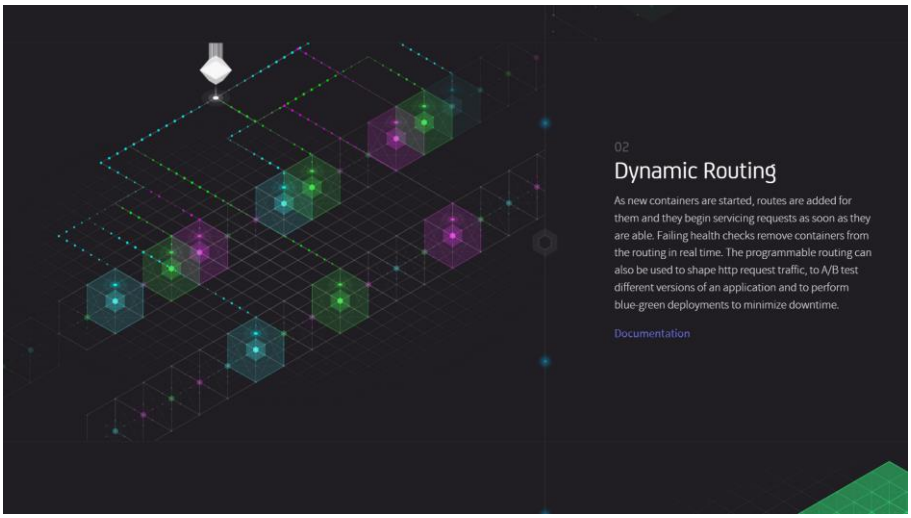
3) **Current needs.** Cloud Foundry is less advantageous to businesses whose applications do not get a lot of traffic and therefore do not need significant scaling, or if their target release cycles are three months or longer. It becomes more advantageous, of course, if the company is in this situation currently yet planning for rapid growth.

If the initial POC results are disappointing in some area, it may be useful to ask if the original expectations were too high—or if the expected developer and operations productivity was done without a commitment to changes in processes and culture.

After carefully studying the initial results, determining what went wrong, making some necessary adjustments, and composing a new set of goals and metrics, the next POC attempt should be more successful.

Lattice

Pivotal is addressing the opportunity for smaller organizations to use Cloud Foundry with its Lattice project. Lattice provides the new Diego container runtime, the Cloud Foundry Router, and its logs and metrics aggregator (formerly called Loggregator and now called Doppler).



Lattice is for single-tenant deployments, and has been designed to be simple yet comprehensive, and extensible. It omits several traditional Cloud Foundry components, such as BOSH-based deployment functionality. The intention is to create a less steep learning curve and smaller installation footprint for Cloud Foundry. Pivotal does say that the goal nevertheless remains for Lattice to be able to handle scaling to manage resilient native cloud apps on any infrastructure.

More information regarding Lattice:

- from Pivotal
- a separate, independent report from ECS Team
- GitHub

4.2 Integrating results with enterprise IT

Once the POC results are satisfactory, the next step is to introduce the platform within the organization. Here, the rule of thumb is to start with a small project, gradually increasing the scale. The steps that usually follow a POC are:

1) A pilot project
2) Adoption on a group level
3) Enterprise-wide adoption

The table below shows common challenges and lessons learned during each stage from enterprises working with Cloud Foundry.

Stage	Timeline	Challenges	Lessons Learned (from Altoros's customer experience)
1. POC	1–4 weeks	• Setting up the deployment on the infrastructure • Steep learning curve for BOSH / Cloud Foundry	• Most first-time issues originate from manifest file errors • Need to be familiar with Cloud Foundry components before starting the POC
2. Pilot project	2–3 mon	• Security • Getting first apps to CF • Third-party dependencies/ blockers	• Use a proven framework for Cloud Foundry adoption • Get training for architects/ DevOps engineers
3. Group-level adoption	4–6 mon	• Changing definitions of "production-grade" • Dev groups learn slower than expected • App portability issues arise at the last moment	• Find app samples and how-tos on getting started
4. Enterprise-wide adoption	12–36 mon	• Too many fronts to fight (dev / test, staging, production) for a small team • Solving complexity of multi-DC deployments	• Solving developer pushback requires some "hacks" • Retaining architects and DevOps engineers is critical

Adoption

From the technical standpoint, the adoption process has four parts:

1) Assessing current applications and technology stack
2) Creating a CF cluster
3) Modifying and deploying apps to the CF cluster
4) Removing the legacy architecture

CF Adoption Steps	Who is Responsible?	What Does This Step Include?
1. Analyze the applications	Architects / developers	Define which apps are cloud-ready, which ones can be moved with some refactoring, and which ones cannot work in the cloud.
2. Build a production CF cluster	DevOps and Ops engineers	Deploy and prepare a production-ready Cloud Foundry cluster, deploy the necessary services, customize buildpacks, etc.
3. Modify/move applications to Cloud Foundry	Developers	Modify/refactor existing applications to run in the cloud and on CF, using the approaches for building cloud-native software. Deploy the applications to the production Cloud Foundry cluster.
4. Remove excessive infrastructure	Developers	After applications have been moved to Cloud Foundry, the infrastructure they used can be removed or decreased in size.

Before migrating applications to the PaaS, it must be ensured that they are compatible with it. Architects and/or developers will need to analyze and sort all existing applications into three major categories:

- **Cloud-ready apps** that can run on Cloud Foundry without any modifications. These are usually built using 12-factor app best practices, and/or have an architecture based on microservices.
- **Apps that need modifications** to work in the cloud. From our experience, most well-written applications can be modified to run on Cloud Foundry fairly easily.
- **Apps that do not run on CF.** Some applications are simply not built for the cloud (e.g., they use the file system for storage). This does not mean they will never work. There are a number of options, such as refactoring or integrating them with the PaaS.

After that, a Cloud Foundry deployment needs to be built by the IT staff. This step also includes deploying custom services that the apps may need, modifying buildpacks, etc.

A small cluster can be created within a week, provided everything works as expected. A fairly large deployment may take a month and a team of three to four DevOps engineers. The actual amount of time and required team size can only be determined based on the size of the cluster, number and size of apps, and other factors.

During the third stage, the development team needs to modify (if necessary) and deploy the applications to Cloud Foundry. In our opinion, the best strategy is to start with cloud-ready apps.

Another option is to begin with new projects that are still in development, avoiding the use of the legacy architecture. A deployment can be created on Cloud Foundry while also using the old architecture, which will be removed after the migration process is complete.

From our experience, most well-written applications only need minor updates to run on Cloud Foundry. Applications will most likely work as is if they:

- do not write to the local file system
- do not persist or replicate HTTP sessions
- consider HTTP and HTTPS port limitations
- are built based on 12-factor app best practices

If an app is poorly written (e.g., it uses the file system a lot), there is a big chance of something breaking during refactoring. Modifying such apps may take as much time and effort as creating new ones from scratch.

Another important factor is the size of the application. The larger it is, the harder it may be to deploy / modify.

Once applications are running on Cloud Foundry, the legacy infrastructure can be removed. The entire process of migrating to the platform may take as little as seven months. It may also take considerably longer, depending on how many apps there are and how complex they are. It is again critical to note that there also needs to be cultural change within the organization to take advantage of the benefits of using Cloud Foundry.

Here is a slide deck that describes the steps for migrating to Cloud Foundry in more detail.

4.3 How results affect developers

Cloud Foundry boosts developer productivity. It helps dev teams free up a considerable amount of time for many reasons:

- A high level of abstraction from infrastructure
- Fast and secure deployment methods
- The same workflow of deployment and implementation for testing and production
- Support for a wide range of tech stacks from the base installation (Ruby, Node.js, Java, Go, Python, and PHP)
- Dynamic binding of services
- Fault-tolerant apps
- Built-in scalability

All this means that developers can focus on writing code. With self-service access to deploy apps as many times a day as necessary, developers can get new features/apps out in weeks instead of months—the same number of people can develop more software faster.

Two perspectives

From the cost-cutting perspective, a PaaS helps to economize resources and get more value out of development teams. From the growth perspective, it enables companies to produce more business value, test/implement ideas faster, and generally get more bang for the buck when hiring developers.

After Cloud Foundry has been adopted—i.e., once there is a fully operational production cluster—then properly setup buildpacks and services provide configuration and setup. It is not necessary for developers to understand configuration and setup, so they can be less experienced and still produce great code.

Adopting a PaaS in general inevitably brings changes to the technical workflow. Developers working as part of a Cloud Foundry team are normally responsible for helping other development teams, as well as DevOps and architects, to design, develop, and onboard applications for Cloud Foundry. They must also be responsible for troubleshooting apps within Cloud Foundry.

Apps, of course, need to be cloud-native to run smoothly. This does not rule out the monolithic approach—but using the 12-factor app model, microservices-based architectures, and Agile methodologies for new software will make the development and deployment process far more efficient.

A paper on microservices from Sergei Sverchkov, an architect at Altoros, can be found here.

4.4 How results affect operations

The ratio of servers to staff in IT departments increases considerably as things scale out with Cloud Foundry. This is due not to Cloud Foundry *per se,* but to the realization of virtualization—enterprise IT managers and executives can now truly think of server "instances" rather than racks of discrete machines.

For instance, Altoros has a customer who is able to support 2,000 instances with only six engineers. So thus comes an opportunity to economize and achieve new efficiencies as an enterprise IT infrastructure enabled by Cloud Foundry scales up.

Resource management in this new world actually goes beyond the abstracted concept of instances. Instead of managing individual deployments, IT engineers working with Cloud Foundry deal with centrally managed *apps* and *services*.

Within that context, there are several features that make the life of DevOps engineers easier:

1) Stable, predictable, and self-healing release engineering
2) Portability among different IaaS environments
3) Fault tolerance in all components, with the exception of DBs (which can be enabled)
4) Ability to address and fix security issues quickly
5) A large, open-source community that provides components, including cf-release, BOSH, and stemcells
6) Great support from this community, especially from mailing lists, such as vcap-dev, bosh-dev, and bosh-users

The nature of cloud computing means that some functionality, such as monitoring and fault tolerance, which is traditionally provided by IT, becomes part of the application. This is where operations professionals can step up, embrace the DevOps culture, and collaborate with developers—few developers have the skills to span IT development and operations.

The responsibilities of DevOps engineers on a Cloud Foundry team generally include:

- Deploying Cloud Foundry and BOSH
- Integrating monitoring tools with Cloud Foundry
- Integration testing

- Integration and maintenance of buildpacks, service brokers, Docker images, and BOSH releases

The picture below provides an example of metrics that an operator sees when monitoring a Cloud Foundry cluster.

An IT shop that is incorporating DevOps culture will be more efficient, but its people will need to be skilled in all things PaaS.

In particular, operations engineers working with Cloud Foundry need considerable expertise in deploying and troubleshooting it. They must understand the inner architecture of Cloud Foundry, be able to look through the source code of apps, understand what the buildpacks are doing, etc. For a person who will be managing and monitoring Cloud Foundry, the required skills include:

- BOSH internals
- Understanding all Cloud Foundry components, their properties in the manifest file, and what they do
- xNIX systems
- Practical experience with deploying Cloud Foundry, log monitoring, and problem-solving
- Reading source code, understanding the messages apps pass between each other
- Familiarity with the Cloud Foundry ecosystem

4.5 How results can drive DevOps

The practice of developers and operations engineers using the same techniques and working together throughout the entire application life cycle forms the core of DevOps culture. Closely related to Agile development, DevOps removes the border that separates IT infrastructure and development.

Within this context, Platform-as-a-Service blurs the line between developers and operators. In the table below, the responsibilities of developers and operations engineers are separated. In reality, developers may do the job of system administrators and vice versa. They need similar critical skills, such as familiarity with UNIX-based systems, Cloud Foundry internals, and the Cloud Foundry ecosystem.

Consider a typical five-person Cloud Foundry team, with a summary of responsibilities and required skills:

Role	FTE	Critical Skills	Responsibilities	Area
DevOps (Cloud Foundry Operator)	2	• BOSH internals • Cloud Foundry internals • xNIX systems • Cloud Foundry and the ecosystem	• Deployment of CF, BOSH • Integration (monitoring / metrics) • Integration testing • Integration and maintenance of buildpacks, service brokers, Docker images, BOSH releases	DevOps, infrastructure
Full-Stack Cloud Foundry Engineer	2	• xNIX systems • Cloud Foundry internals • Cloud Foundry and the ecosystem • Services and service brokers	• Assist development / DevOps teams and architects with app design, development, and onboarding • Application troubleshooting	Development, DevOps
Team Lead	1	• CF architecture • Communication and cultural change	• Deployment of BOSH and Cloud Foundry • Integration (monitoring / metrics)	Enterprise application architecture

The notion of DevOps is new and can seem rather vague in how it functions operationally.

It is helpful to consider a typical situation in traditional enterprise IT, in which IT cannot deploy a critical app because it will not run on the existing infrastructure—the architecture does not match the network/storage/deployment/security model, etc. In a DevOps culture, developers and operations engineers are put together from the start, so that they cooperate on creating applications.

Thus, Cloud Foundry capabilities, such as continuous delivery—deploying code as often as necessary—are inseparable from the DevOps approach. This is the latest iteration on Agile development and similar fast methodologies, such as Kanban and Scrum, and must be adopted throughout the entire organization, including development, IT, QA, etc.

5. Cloud Foundry Rollout

5.1 How results can affect corporate culture

The DevOps approach is a combination of techniques that help to improve efficiency with collaboration and the right technology. Taking advantage of the capabilities provided by Cloud

Foundry is possible only if the entire organization embraces the DevOps culture and the software development methodologies associated with it.

If developers already use Agile, Kanban, or Scrum, but the IT department is still working with the waterfall method, for example, the organization is unlikely to benefit from adopting a PaaS. To make real progress, operations must be run in a way similar to that of the Agile development teams.

Several steps

Creating a DevOps culture to support the adoption of PaaS in a company may involve some or all of the following steps:

1) Defining where the organization is and setting goals
2) Getting buy-in and continued support from top management
3) Choosing the right project and test group in terms of size, complexity, and goals
4) Adjusting the approaches to evaluating success (e.g., going from measuring how fast a project is completed to measuring how fast code becomes production ready)
5) Using the right DevOps tools and technologies
6) Continuously improving products, finding and eliminating bottlenecks, etc.

Generally speaking, the table below summarizes DevOps principles and offers some ideas on how they can be implemented.

DevOps Principles	Ways to Implement the Principles
Culture first	Cultural change must come before technology. Unless developers and IT communicate and work together, there will be little use in automation, continuous delivery, and the benefits they accrue.
Automation	Use tools for release management, provisioning, configuration management, systems integration, monitoring and control, and orchestration. CF provides many of these features out-of-the-box.
Process	Application life cycle includes multiple domains; processes should also be linked across all of them.
Measurement	Only that which is measured can be improved. Use monitoring and quantify results whenever possible.
Sharing	Create an environment where people can share ideas and problems. This will help to get valuable feedback. Sharing may also include using open-source technologies. Invite operations engineers to developer meetings and vice versa to spot issues and get them fixed earlier.
Continuous improvement	Use continuous delivery and test-driven development to keep up with the changing customer requirements and technologies.
Simplicity	Strive to create simple, repeatable, and re-usable solutions.

Software factories

Companies that deliver products (rather than completing internal projects) and employ DevOps on

cloud platforms like Cloud Foundry are sometimes referred to as “software factories.” In this environment, developers are continuously polishing products for customers, the IT department is working on maintaining and improving the platform, which is a product for developers, etc. This video describes a software factory built by Warner Music.

Source: WMG

5.2 New training requirements

Achieving success with Cloud Foundry on enterprise scale is about three things: people, people, and people. An organization that wants to use a PaaS will need to secure support from top management as well as those who will be using it, change the culture and processes, and of course train its staff.

As with any complex system, Cloud Foundry has a learning curve. Getting training before attempting to use Cloud Foundry will not only decrease the possibility of failure, but also help to speed up the process of migration. From our experience at Altoros, a trained team can get up to twice as many apps to Cloud Foundry than with traditional methods.

Overview

Because Cloud Foundry is a cloud-based system, developers and operators using it need to have a good understanding of basic cloud concepts. They should know how a cloud IaaS is built, and how to use object storage, message queues, services, message buses, etc.

Another important aspect is understanding the functionality of clouds and how to employ it when building apps. For instance, those who work with AWS need to understand what services are available and how they can be used in applications.

Then come the actual Cloud Foundry concepts. There are several key training topics that should be covered for developers and operators who will be building and using apps.

Developers	Operators
• Why PaaS	• Why DevOps
• Cloud Foundry architecture	• Cloud Foundry architecture
• Application life-cycle management	• Application life-cycle management
• Best practices for developers	• Learning the deployment playbook
• Runtimes	• BOSH
• Services	• Administrative console
• Zero-downtime deployments	• User management and security
• Troubleshooting	• Troubleshooting
• Scalability and high availability (HA)	• Scalability and high availability (HA)
• CLI and the Eclipse plug-in	• Logging, monitoring, and alerts
• Database migrations	• Customizations

Hands-on exercises are also critical for deployers and operators. Working with Cloud Foundry requires a deep understanding of its inner architecture, as well as practical experience with deploying it. An operator should be able to look through the source code and understand how buildpacks work, for example, to troubleshoot infrastructure issues.

Courses and certifications

There are many ways to get Cloud Foundry training, in the form of online courses, webinars, and in-person classes are also available from a number of sources:

Organization	Brief Description	URL
Altoros	Online and on-site Cloud Foundry trainings, including cloud-native development, microservices, etc.	https://cf-training.altoros.com https://www.altoros.com/training.html
Anynines	Consulting and training in all aspects of Cloud Foundry operations	https://www.anynines.com/training
Cloud Foundry Foundation	Cloud Foundry documentation is a comprehensive source of information. A free online training is also offered.	http://docs.cloudfoundry.org https://www.cloudfoundry.org/training
Pivotal	Lab-based courses and certifications. Role-based certification exams offered through Pearson VUE.	http://pivotal.io/training
PluralSight	An intro course offered, as well as courses on deploying and managing applications, and advanced topics.	https://www.pluralsight.com/courses/cloud-foundry-developers

5.3 Integrating the platform throughout an organization

Switching to a new way of delivering applications is more complex than getting the technology onboard. People are the key to success and securing support from your team can mean the difference between success and failure. The general strategies for converting PaaS sceptics into advocates are as follows:

Strategy	How It Helps to Prevent / Overcome Pushback
Provide hard data	Measure and quantify the improvements that Cloud Foundry can bring to your organization to prove its value to users, management, and other stakeholders.
Show what CF can do	Use demos or other means to demonstrate Cloud Foundry's core features and how they can help to improve your organization.
Get PaaS training for your staff	Having someone who understands how to use a PaaS will help to promote it within the organization.
Build apps for the cloud	The advantages of having a PaaS are most visible when building cloud-native distributed apps.

There can be some differences in explaining why Cloud Foundry is necessary to developers and DevOps engineers.

For developers

There is no need to clarify the benefits of PaaS to those who have tried building HA solutions from scratch, managed applications on two different servers, worked with synchronization, and/or tried to support such apps manually. These people should understand how much effort they can save if these features will be available out-of-the-box.

Those who have always built apps hosted on a single server may be less willing to use Cloud Foundry. To be honest, a PaaS is probably not really necessary for such use cases. Still, while many of them do not care about the features it provides right now, they are likely to need them in the coming years.

Companies that build cloud-based apps and use a PaaS can have auto-scaling, high availability, auto-healing, and things like that by default. This is a significant competitive advantage over those who try to implement them from scratch. Inevitably, more and more customers will want such features in all their apps. As more workloads will be moved to the cloud, businesses and developers will have to look for ways to enable them.

Here is a list with some of the benefits enjoyed by developers who work with Cloud Foundry:

- No need to do sys admin work (e.g., installing database/application servers, installing messaging software, etc.)
- All the infrastructure details, including the IP address and port of the machine, specifics of the network ranges and storage volumes, etc., are abstracted away, so developers can focus on writing code.
- Simple assembly, deployment, and management of Web apps
- Self-service provisioning of resources, no need to ask operations for resources
- Automated configuration and setup of applications

- Continuous integration

For DevOps engineers

As we said, an ever growing number of companies are moving their workloads to the cloud and many of them will probably start using Cloud Foundry or other PaaS systems to automate manual tasks. So, for operations engineers, a switch from managing traditional infrastructures to managing a PaaS will be a natural evolution.

Some think that Cloud Foundry is built solely around developers and their needs. In truth, it does a lot for operations engineers, too. For example:

- Fast self-service infrastructure provisioning (no more processing tickets)
- Automated configuration of VMs, installing things on VMs, networking VMs, etc.
- Time to push an app to production is reduced from weeks to hours or even minutes.
- Centralized management of services and apps (one dashboard)
- Four levels of high availability built into the PaaS
- IaaS-independent, apps can run on different clouds.
- Fewer problems when deploying apps to production, since there is no discrepancy between production and staging environments.
- Loggregator gathers all logs from app instances and the router—no need to SSH into separate machines.
- Automatic health monitoring and recovery of failed instances
- A high level of security thanks to application boundaries, containers, etc.
- Containerization that brings improved server utilization, portability, consistency, and speed
- Zero-downtime blue-green deployments and rolling updates

5.4 How Cloud Foundry delivers new capabilities

The term “PaaS” is like most terms in the IT industry in that it sometimes gets distorted and abused by vendors who are trying to force-fit products and services into a particular marketing position. It is also sometimes conflated with IaaS by vendors who seem primarily interested in locking their customers into their offerings.

Cloud Foundry is a PaaS; there is no mistake about this. It works with a number of IaaS offerings, and can be used with an enterprise’s own custom IaaS. It is offered in an open-source version, thus completely obviating lock-in risk. It is also offered in commercial versions, with sophisticated features (and complementary training) that reduce the steepness of Cloud Foundry’s learning curve and, at least, still avoid lock-in when it comes to the underlying infrastructure.

Two key points that are raised often in this architect’s guide are the promises of continuous development, deployment, and management that comes with Cloud Foundry, and the necessity to develop a DevOps culture to fulfill those promises.

The promises and obligations stem from Cloud Foundry’s reality as a true enterprise PaaS. This is not an idle phrase, with no less an authority than Pivotal’s James Watters—the company’s VP and General Manager of the Cloud Group—who has written eloquently about what Cloud Foundry brings to the table in 2013.

The table below summarizes the points James makes:

Main Point	Commentary
1. App- and services-centric life-cycle API	• 10x+ productivity increase from not dealing with infrastructure and middleware • 90% decrease in developer time by a large Internet company
2. High-performance dynamic routing	• The heart of rapid application deployment (RAD) and horizontal scaling
3. Buildpack support	• Allows simple use of push command across languages and frameworks
4. Data and Web services brokers	• Single, consistent developer experience
5. Linux container management	• Rapid, high-density access to capacity • Economical provisioning of robust environments
6. Role-based access and teams	• Can immediately form ad-hoc teams and start working on applications out of the box
7. Active app health management	• Solves this difficult problem while eliminating overhead
8. Standards-based user authentication and authorization	• User Account and Authorization (UAA) meets numerous enterprise sign-in requirements
9. Integrated Real-Time Logging API	• Loggregator views and tails unified application log streams and works with popular logging and analytics tools.
10. Multi-provider ecosystem	• Dev teams can work in multiple clouds, from more than one provider

5.5 Cloud Foundry and ROI

Measuring return on investment (ROI) is a Holy Grail for the financial departments of any enterprise that is buying technology. It is especially prized for considerations of new technology.

It is very helpful for architects and IT departments if they can control what ROI means, how it is defined, and its implications for their enterprise. This guide mentions several times that culture change is critical for a successful rollout of Cloud Foundry, so any simple measure of ROI will be detached from that, and perhaps even irrelevant.

How to define it

Reaching back in history a bit, we find an article dating from 2002 in *InformationWeek* that outlines the issue, evoking the image of chasing a butterfly.

As another article from the Center for Technology in Government (at the University of Albany, SUNY), notes, “If an ROI analysis were just one simple thing, then there would be one simple way to measure

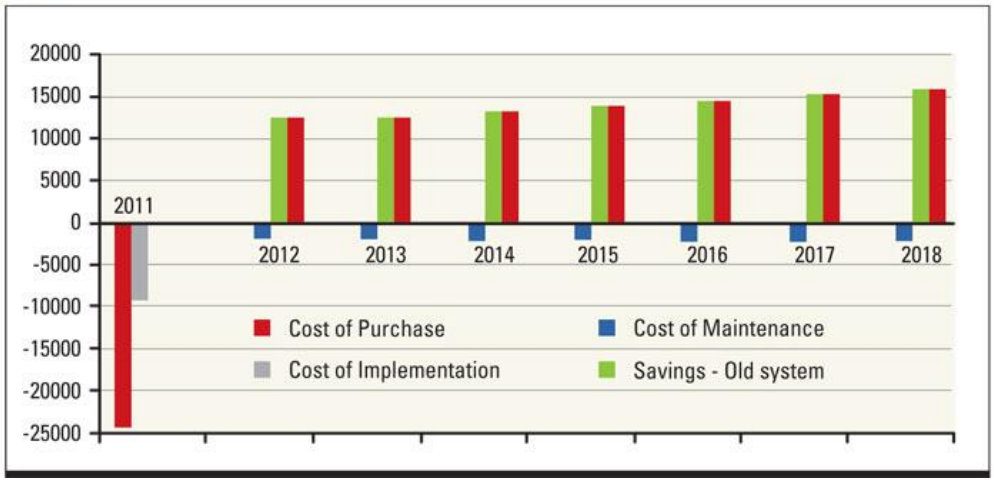
the costs, returns, and benefits. In practice, however, there can be many different questions asked of an ROI analysis requiring different measurement approaches to fit those questions."

This sentiment applies to the business world, as well. Before trying to formulate a relevant ROI for technology in general and Cloud Foundry in particular, many fundamental questions need to be addressed:

- What are the overarching goals of the organization?
  - To produce the existing number of applications with fewer people?
  - To produce more with the same number of people?
  - To use new apps and services to gain new customers?
  - To gain more revenue from existing customers?
  - To gain market share?
  - To enter new markets?
- Or is Cloud Foundry part of an overall initiative to transform the enterprise?
- If so, what are the goals in this case?
  - Revenue growth?
  - Profitability?
  - To increase brand equity as well as revenue?
  - Is this a public company or privately held?
  - How do the goals differ?

In this spirit, another article takes a look at ROI in the healthcare industry (see the picture below). Issues include "hard-to-quantify benefits like patient satisfaction, outcomes improvement and market positioning enabled by the investment."

FIG.1 - SAMPLE DATA FOR IRR COMPUTATION / IT PROJECT +/- 17% IRR



ROI calculators

There are a couple of simple "ROI calculators" available online. Apprenda, for example, has produced this one.

In the interests of objectivity, we have also found a Red Hat's OpenShift ROI calculator. It was created by Vizuri, an IT consulting firm. This calculator has more apparent sophistication than the one from Apprenda, although clearly one should always be cautious about relying on any sort of quick, online analysis.

Both calculators actually show estimated financial savings, not ROI. The numbers that these calculators spit out may look impressive, and may enable IT management to make an initial foray into building a case for the use of Cloud Foundry.

But ROI also implies a direct correlation, or even cause-and-effect, between the costs of a new technology and how long it takes for the company to make its money back. Yet if an organization is using OSS Cloud Foundry in a DIY manner, for example, then there is no direct product cost; the costs would be tied to the number of people involved in the effort.

Even with the use of a commercial, paid version of Cloud Foundry, the staffing costs and implications of organizational change are something that really cannot be captured by a simple ROI analysis. So, we include this section in the Architect's Guide—and conclude the guide with it—with the realization that the question of ROI will likely emerge within enterprises adopting Cloud Foundry.
We welcome any specific use cases that include an ROI analysis.

6. About the Authors

Roger Strukhoff is Director of Research at Altoros and Executive Director of the Tau Institute for Global ICT Research, with offices in Illinois and Manila. He is Conference Chair of CloudExpo and ThingsExpo, and Editor of SYS-CON Media's Cloud Computing, Big Data, and IoT journals. He holds a BA from Knox College and conducted MBA studies at CSU-East Bay.

Juan Pablo Genovese is Field Cloud Foundry Engineer at Altoros. He has been developing software for 17 years and, as a Jack of All Trades, has also been into DevOps work. Juan Pablo is focused on training new CF engineers at Altoros and identifying the needs of the community. His professional interests include high performance / HA solutions with cloud technologies, as well as designing architectures that meet customer expectations.

Volha Kurylionak is Technology Evangelist at Altoros. With four years of experience in technology evangelism and blogging, Volha is a big fan of open source and Ubuntu.

Altoros *brings Cloud Foundry-based "software factories" and NoSQL-driven "data lakes" into organizations through training, deployment, and integration. With 250+ employees across 9 countries in Europe and Americas, Altoros is the company behind some of the world's largest Cloud Foundry and NoSQL deployments. For more, please visit www.altoros.com.*

To download more Cloud Foundry guides and tutorials:

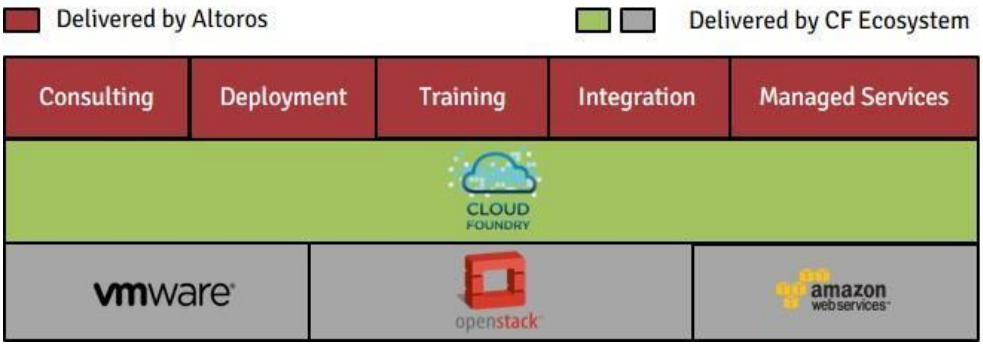
- *check out our resources page,*
- *subscribe to the blog on all things Cloud Foundry,*
- *or follow @altoros for daily updates.*

Featured: A Production-grade Cloud Foundry Deployment

Seen the Demo?
Check out the Pricing!

- By integrating solutions offered by the Cloud Foundry ecosystem, Altoros continuously perfects delivery and operation of “software assembly lines” for its customers.
- We understand the difference between success and failure in implementing Cloud Foundry at large enterprises, including those in highly regulated industries.
- As a result, Altoros’s customers discover and monetize application-driven competitive advantages sooner than competition. They get turn-key, managed “software factories” and “data lakes” using best-of-breed solutions developed by members of the Cloud Foundry ecosystem.
- Altoros is proud to be behind some of the largest Cloud Foundry implementations in the world.

CLOUD FOUNDRY SILVER MEMBER



Customers: Joyent, RIGHTSCALE, SONY make.believe, Consafe Logistics

Partners: Pivotal, vmware, amazon web services Partner Network CONSULTING PARTNER, Microsoft Partner Gold

“I recommend Altoros training to any of our partners or potentials customers who want to set up Cloud Foundry using BOSH. Hands-on nature of the training was absolutely essential for learning BOSH.”

Wes Gruver,
Consulting Instructor
Pivotal

“Canonical partnered with Altoros for automation of federated Cloud Foundry & DBaaS deployments using Juju Charms platform.”

Maarten Ectors,
Senior Cloud Strategist
CANONICAL