THE UNIVERSITY OF BUEA
P.O Box 63,
Buea,South West Region
Cameroon
Tel: (237) 674354327
Fax: (237) 3332 22 72

REPUBLIC OF CAMEROON

PEACE-WORK-FATHERLAND

FACULTY OF ENGINEERING AND TECHNOLOGY

Department of Computer Engineering

Specialty: Network Engineering

CEF450 : CLOUD COMPUTING AND SOA

Platform as a Service Case Study: Cloud Foundry

Presented by

GROUP-3

Dr. Eng. Ines Djouela

Course Supervisor

2rd Semester

2023/2024 Academic Year

No	NAMES	MATRICULE
1	KONGNYUY RAYMOND AFONI	FE21A219
2	SIRRI THERESIA	FE21A306
3	TABOT JOEL EBANGHA	FE21A308
4	TAKEM JIM-RAWLINGS E	FE21A309
5	TAKO DIEUDONNE OJONG	FE21A310
6	TANWIE BRUNO ADEY	FE21A316
7	TEGUE NONO MIKEL MODEIRO	FE21A321
8	TEWOM FERDINAND TENYI	FE21A323
9	THERESE BLESSED	FE21A324
10	TIANI PEKINS EBIKA	FE21A325
11	TSAPZE ZAMBOU ROSELINE CINTHIA	FE21A328
12	VTALAH DJOUMG L DE GERAUD	FE21A329
13	ZELEFACK MARIE NOELLE	FE21A330
14	EKUNDIME GLEAN MAKOGE	FE21A433
15	MATHO SONKWA HESTIE MAYELLE	FE21A438
16	NANGLEFACK LEODIA FIETSOP	FE21A244
17	NGI KEVIN AYUK	FE20A076
18	AJAMAH GODLOVE NKEMAJAH	FE21A129
19	AKENGNI KEANLI EMMANUEL	FE21A132
20	AZEFACK JUNIOR	FE21A146
21	BOBGALA HARRISON DIMA	FE21A151
22	DJEUNOU DJEUNOU MARIEKE JETTIE	FE21A168
23	DJOUMESSI NGEUFACK IVAN	FE21A174
24	EBOMESUMBE NGOLE DANDY BRADLEY	FE21A177
25	EFUETANZOH ASONG RODERIC	FE21A179
26	EYONG OSCAR ENOWNYUO	FE21A187
27	FRU BELTMOND CHINJE	FE21A198
28	FUKA NEVILLE TENYI	FE21A199
29	GATCHUINNE M0MO MURIELLE CYNTHIA	FE21A200
30	KANKO KAMEDJEU DUPLEX	FE21A210
31	KENEDY MALLEY ITUKA	FE21A212
32	SIAHA TOUKO AUBIN	FE21A304
33	MBELI NGO NOEL	FE21A229
34	MBI AYAMBA DIANNA	FE21A230
35	MBUA SEDRICK GOBINA KOFI	FE21A233
36	METIEGE OTILI-FAVOUR MECHANE	FE21A238
37	NDEH TAMINANG	FE21A246
38	NGOUNG TASSA ALAIN SHIRESS	FE21A262
39	NIBA VERINE KAJOCK	FE21A267
40	NICCI NSE NCHAMI	FE21A268
41	NJOYONG GODWILL	FE21A273
42	NKAFU VANIC ASONG	FE21A274
43	SAMUEL OSOH	FE21A303
44	TAKOH CLOVERT NFUA	FE21A311
45	TETUH WIBINAH ENGONWEI	FE21A322
46	TAMBONG KERSTEN MELENGFE	FE21A440
47	MUYANG ROSHELLA MBAMUZANG	FE21A243
48	TCHUIDJAN JORDAN BRYANT	FE21A320
49	MOHAMAN BELLO	FE18A040

Abstract

This report provides a comprehensive overview of Cloud Foundry, an open-source platform as a service (PaaS) for building, deploying, and managing applications. It outlines the benefits of PaaS in general and highlights the key features of Cloud Foundry, including its multi-cloud support, open-source nature, and flexible deployment options.

The report delves into the technical details of Cloud Foundry's architecture, explaining its components like Cloud Controller, Diego, and UAA. It covers security measures implemented by Cloud Foundry, such as containerization, access control, and encrypted connections. Additionally, the report details mechanisms for achieving high availability through features like availability zones and external load balancers.

The final sections provide instructions for installing the Cloud Foundry Command Line Interface (cf CLI) and deploying applications on the platform. It also covers how Cloud Foundry handles app crashes and shutdowns.

Contents

1. Platform as a Service: A Brief Overview

1.1 Introduction

Platform as a Service (PaaS) is a complete cloud environment that includes everything developers need to build, run, and manage applications—from servers and operating systems to all the networking, storage, middleware, tools, and more.[1]

PaaS, is a type of cloud computing service model that offers a flexible, scalable cloud platform to develop, deploy, run, and manage apps. PaaS provides everything developers need for application development without the headaches of updating the operating system and development tools or maintaining hardware. Instead, the entire PaaS environment—or platform—is delivered by a third-party service provider via the cloud. different PaaS include:

SAP Cloud, Microsoft Azure, Salesforce Lightning, AWS Lambda, Google App Engine, Pivotal Cloud Foundry, AWS Elastic Beanstalk, IBM Cloud Foundry, Red Hat OpenShift, Oracle Cloud Platform[2].

PaaS helps businesses avoid the hassle and cost of installing hardware or software to develop or host new custom applications. Development teams simply purchase pay-as-you-go access to everything they need to build custom apps, including infrastructure, development tools, operating systems, and more.

The result is simpler, faster, and secure app development that gives developers the freedom to focus on their application code.

Figure 1 Best practices to implement PaaS

1.1.2 How does PaaS work? And Benefits of PaaS

1. How does PaaS work?

Unlike IaaS or SaaS service models, PaaS solutions are specific to application and software development and typically include:

- **Cloud infrastructure**: Data centers, storage, network equipment, and servers
- **Middleware software**: Operating systems, frameworks, development kits (SDK), libraries, and more
- **User interface**: A graphical user interface (GUI), a command line interface (CLI), an API interface, and in some cases, all three

PaaS is typically delivered as a secure online platform that developers can access over the internet, allowing them to work on projects from anywhere and collaborate freely with other members of their team. Applications are built directly on the PaaS system and can be immediately deployed once they are completed.

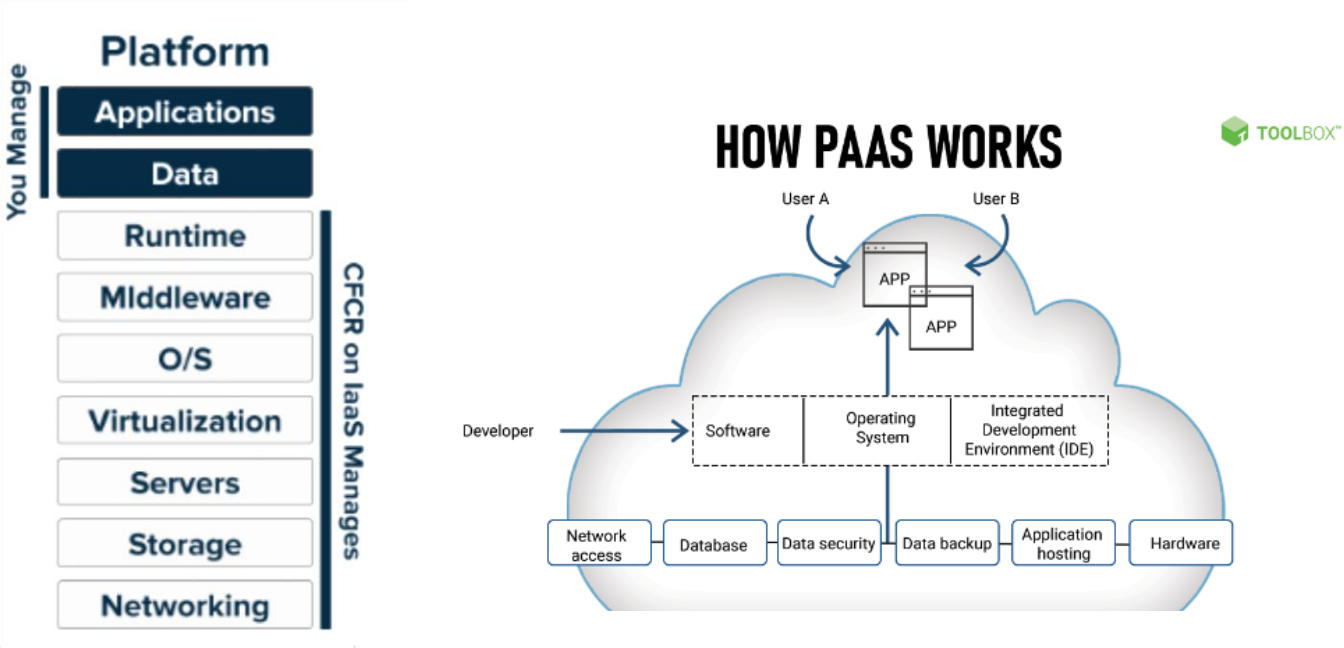


Figure 2 How PaaS works

2. Benefits of PaaS.

The most common benefits of PaaS compared to running and maintaining your own environment include:

- Faster time to market
- Low maintenance
- Cost-effective pricing
- Easy scalability
- Flexible access
- Shared security

2. Case Study: Cloud Foundry

Cloud Foundry is an open-source cloud app platform, providing a choice of clouds, developer frameworks, and app services. Cloud Foundry makes it faster and easier to build, test, deploy, and scale apps. It is an open-source project and is available through a variety of private cloud distributions and public cloud instances

History of Cloud Foundry

In 2009, VMware engineers Mark Lucovsky, Derek Collison and Vadim Spivak led the team that designed and architected the Cloud Foundry platform, which was launched in April 2011. VMware called the platform the industry's first open source PaaS.

In 2012, VMware and its parent company EMC announced plans to spin off parts of their cloud and software business -- including Cloud Foundry -- into a new business called Pivotal Software.

In 2016, Dell Technologies acquired EMC, including its VMware and Pivotal businesses.

However, in 2019, VMware acquired Pivotal with Dell Technologies remaining a majority stockholder of VMware. That move was followed by the completed spinoff of Dell's equity ownership in VMware in November 2021 and the 2022 announcement of VMware's acquisition by chipmaker Broadcom.

.

2.1 About Cloud Foundry

Cloud Foundry as industry-standard cloud platform.Not all cloud platforms are created equal. Some have limited language and framework support, lack key app services, or restrict deployment to a single cloud.As an industry-standard cloud platform, Cloud Foundry offers the following:

- **Open source code**: The platform’s openness and extensibility prevent its users from being locked into a single framework, set of app services, or cloud.
- **Deployment automation**: Developers can deploy their apps to Cloud Foundry using their existing tools and with zero modification to their code.
- **Flexible infrastructure**: You can deploy Cloud Foundry to run your apps on your own computing infrastructure
- **Commercial options**: You can also use a PaaS deployed by a commercial Cloud Foundry cloud provider.

- **Community support**: A broad community contributes to and supports Cloud Foundry. Cloud Foundry is ideal for anyone interested in removing the cost and complexity of configuring infrastructure for their apps.

2.2 How Cloud Foundry Works

Cloud platforms let anyone deploy network apps or services and make them available to the world in a few minutes. When an app becomes popular, the cloud scales it to handle more traffic, replacing build-out and migration efforts that once took months with a few keystrokes. Cloud platforms enable you to focus exclusively on your apps and data without worrying about underlying infrastructure.

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**.

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.

Cloud Foundry Layer	Challenge to Maintain	Contains	Description	Roles
Foundations	Hardest	Orgs	For shared components: domains, service tiles, and the physical infrastructure	Admin, Admin Read-Only, Global Auditor
Orgs	Average	aSpaces	A group of users who share a resource quota plan, apps, services availability, and custom domains	Org Manager, Org Auditor, Org Billing Manager
Spaces	Easiest	Apps	A shared location for app development, deployment, and maintenance	Space Manager, Space Developer, Space Auditor

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

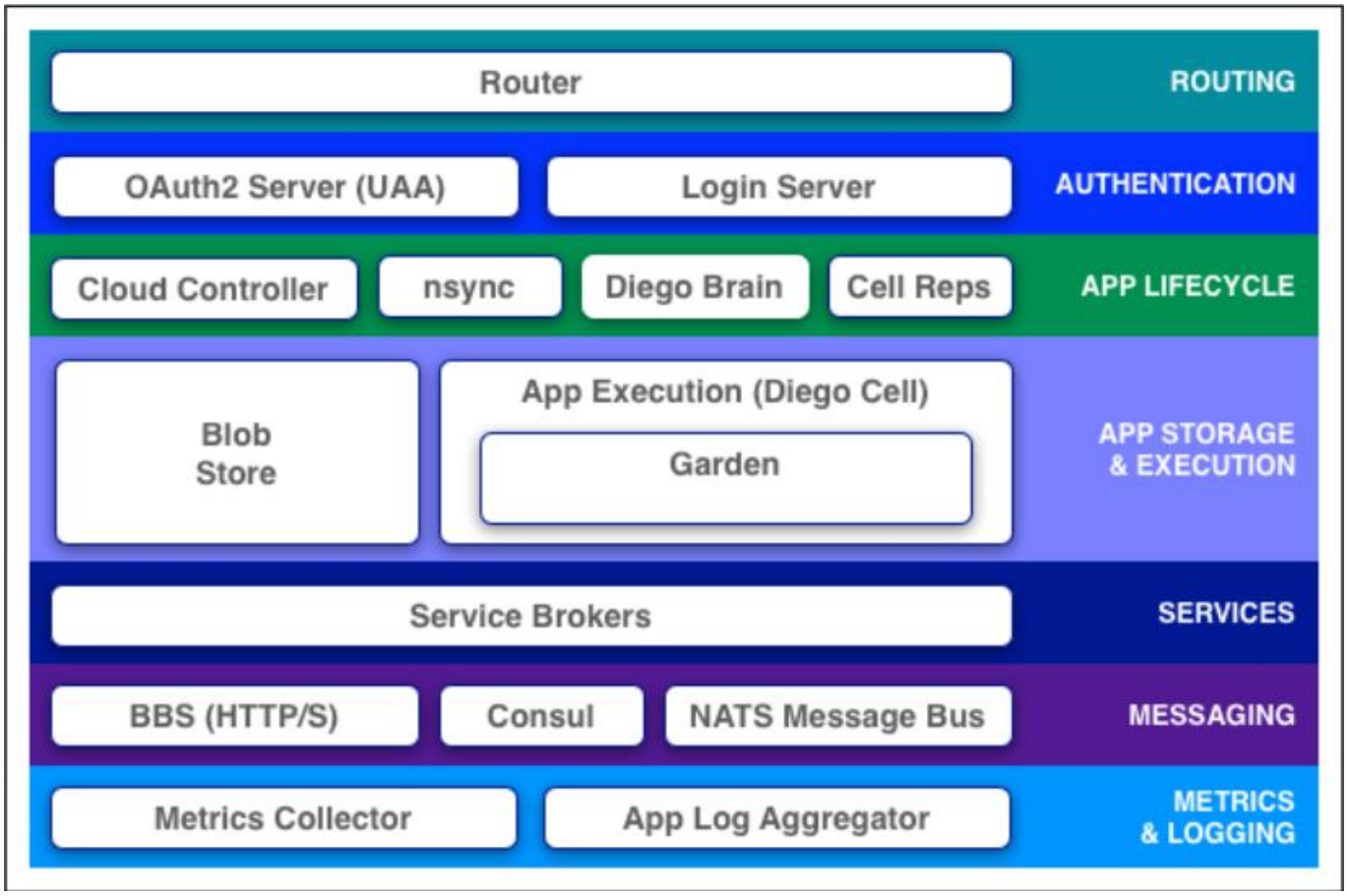


Figure 3 Cloud Foundry key components

2.2.1 Components

Clouds balance their processing loads over multiple machines, optimizing for efficiency and resilience against point failure. A Cloud Foundry installation accomplishes this using the following components:

- **BOSH** creates and deploys VMs on top of a physical computing infrastructure, and deploys and runs Cloud Foundry on top of this cloud.
- **Cloud Controller** runs the apps and other processes on the cloud's VMs, balancing demand and managing app lifecycles.
- **The Gorouter** routes incoming traffic from the world to the VMs that are running the apps that the traffic demands, usually working with a customer-provided load balancer.

2.2.2 Running apps

how the platform packages your apps to run on VMs.Cloud Foundry designates the following types of VMs:

- **Component VMs** make up the platform's infrastructure.
- **Host VMs** host your apps for the outside world.

Within Cloud Foundry, the Diego system distributes the hosted app load over all of the host VMs, and keeps it running and balanced through demand surges, outages, or other changes. Diego accomplishes this through an auction algorithm.

2.2.3 Diego: The Containerization System

Diego is the containerization system responsible for managing the lifecycle of applications deployed to Cloud Foundry. It utilizes containers to isolate applications from each other and the underlying operating system. Diego also employs self-healing mechanisms to automatically restart failed application containers, ensuring application uptime

2.2.4 Organizing users and workspaces

Cloud Foundry manages user accounts through two User Account and Authentication (UAA) servers, which support access control as OAuth2 services and can store user information internally, or connect to external user stores through LDAP or SAML.

User Account and Authentication (UAA) Server. The following table describes what the two UAA servers do:

Server	Purpose
First UAA server	Grants access to BOSH Holds accounts for operators who deploy runtimes, services, and other software onto the BOSH layer directly
Second UAA server	Controls access to the Cloud Controller Defines user roles, such as admin, developer, or auditor, and grants them different sets of privileges to run Cloud Foundry commands Scopes the roles to separate, compartmentalized orgs and spaces within an installation to manage and track use

2.2.5 Storing resources

The following table describes where Cloud Foundry stores resources:

Resource	Storage Location
Source code Buildpacks Documentation Custom configurations Other platform resources	GitHub
Large binary files Droplets	Internal or external blobstore
Internal component states Other temporary information	MySQL

2.2.6 Monitoring and Analyzing

Cloud Foundry generates system logs from Cloud Foundry components and app logs from hosted apps. As Cloud Foundry runs, its component and host VMs generate logs and metrics. Cloud Foundry apps also typically generate logs.

Log Type	Destination
Cloud Foundry component logs	Rsyslog agents
Cloud Foundry component metrics	Loggregator
App logs	Loggregator

Component logs stream from rsyslog agents, and the cloud operator can configure them to stream out to a syslog drain.The Loggregator system aggregates the component metrics and app logs into a structured, usable form, the Firehose. You can use all of the output of the Firehose, or direct the output to specific uses, such as monitoring system internals, triggering alerts, or analyzing user behavior, by applying nozzles..

3. Cloud Foundry security

Cloud Foundry implements the following measures to mitigate against security threats:

1. Minimizes network surface area.
2. Isolates customer apps and data in containers.
3. Encrypts connections.
4. Uses role-based access controls, applying and enforcing roles and permissions to ensure that users can only view and affect the spaces for which they have been granted access.
5. Ensures security of app bits in a multi-tenant environment.
6. Prevents possible denial of service attacks through resource starvation.
7. Protocols.

3.1 Authentication and authorization

User Account and Authentication (UAA) is the central identity management service for Cloud Foundry and its various components.

UAA acts as an OAuth2 Authorization Server and issues access tokens for apps that request platform resources. The tokens are based on the JSON Web Token and are digitally signed by UAA.

Operators can configure the identity store in UAA. If users register an account with the Cloud Foundry platform, UAA acts as the user store and stores user passwords in the UAA database using bcrypt.

3.2 Security event logging and auditing

For operators, Cloud Foundry provides an audit trail through the bosh tasks command. This command shows all actions that an operator has taken with the platform. Additionally, operators can redirect Cloud Foundry component logs to a standard syslog server using the syslog_daemon_config property in the metron_agent job of cf-release.

3.2 Container security

Cloud Foundry secures containers through the following measures:.

- **Hardening containers.** Only allowing outbound connections to public addresses from app containers. This is the original default. Admins limiting function and access rights .
- **Running app instances in unprivileged containers by default:** Unprivileged and privileged.To enable privileged containers for buildpack-

based apps, set the capi.nsync.diego_privileged_containers property to true in the Diego manifest.To enable privileged containers for staging tasks, set the capi.stager.diego_privileged_containers property to true in the Diego manifest.

4. High Availability in Cloud Foundry

You can use availability zones, external load balancers, and external blob storage to ensure high availability for your deployment.

4.1 Availability zones

Availability Zones (AZs) are locations where public cloud services offer data centers. Cloud Foundry supports deploying apps instances across multiple AZs.You can assign and scale components in multiple AZs to help maintain high availability through redundancy. To configure sufficient redundancy, deploy Cloud Foundry across three or more AZs and assign multiple component instances to different AZs.

4.2 External load balancers

External load balancers distribute traffic coming from the internet to your internal network

4.3 External blob storage

Blobs are large binary files, such as PDFs or images. To store blobs for high availability, use external storage such as Amazon S3 or an S3-compatible service.

Scaling platform capacity.

Vertical scaling: Add memory and disk to each VM.

Horizontal scaling: Add more VMs that run instances of Cloud Foundry components.

Disadvantages of Cloud Foundry

While Cloud Foundry offers a compelling platform for cloud development, there are some drawbacks to consider:

- **Limited stateful application support**: Cloud Foundry primarily focuses on stateless applications. While workarounds exist, it's not ideal for applications that require persistent storage of data.

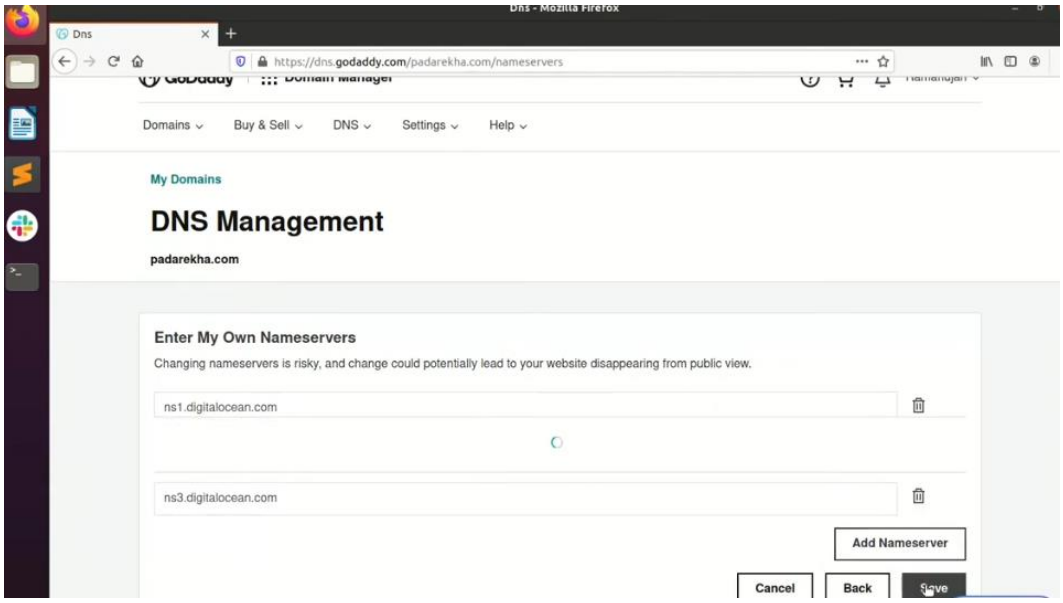
- **Learning curve**: While advertised as developer-friendly, Cloud Foundry has its own complexities. Managing the platform itself can add to the initial learning curve for developers.
- **Limited marketplace**: Compared to some other cloud platforms, Cloud Foundry has a smaller selection of services available in its marketplace. This might limit your options for finding specific functionalities you need.
- **Documentation and support**: Some users report that the documentation and support available for Cloud Foundry, particularly for less popular buildpacks, can be lacking. This can make it challenging to troubleshoot issues or find examples for specific use cases.
- **Scalability limitations**: Though Cloud Foundry allows for application-level scaling, scaling the platform itself can be more complex compared to some public cloud offerings with virtually infinite resources.

6. Installing app on Cloud Foundry

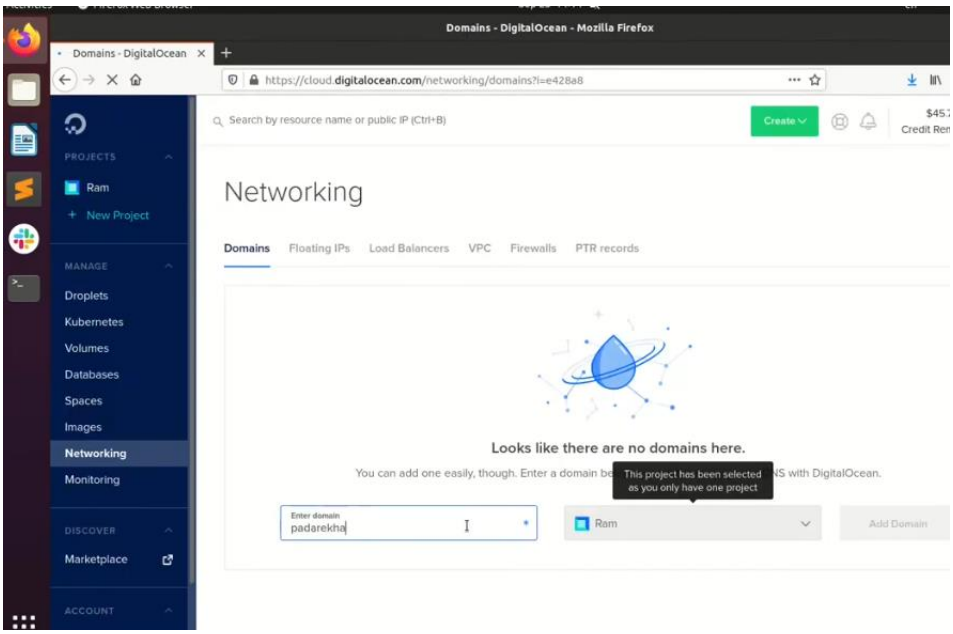
6.1 Installing Cloud Foundry On A Digital Ocean kubernetes Cluster Linux installation

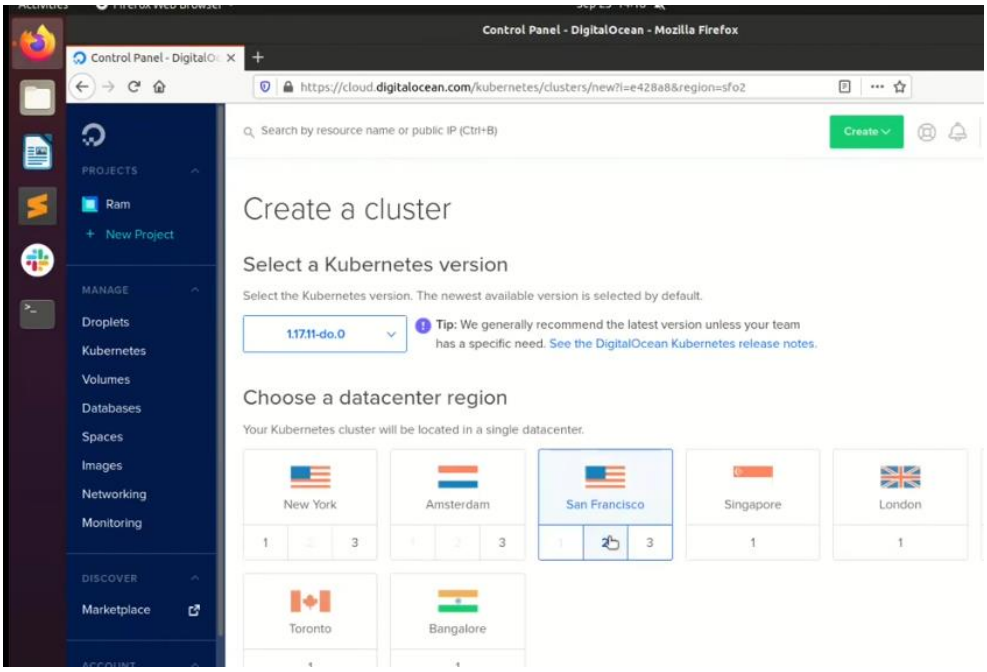
The below steps are used to install cloud foundry for kubernetes:

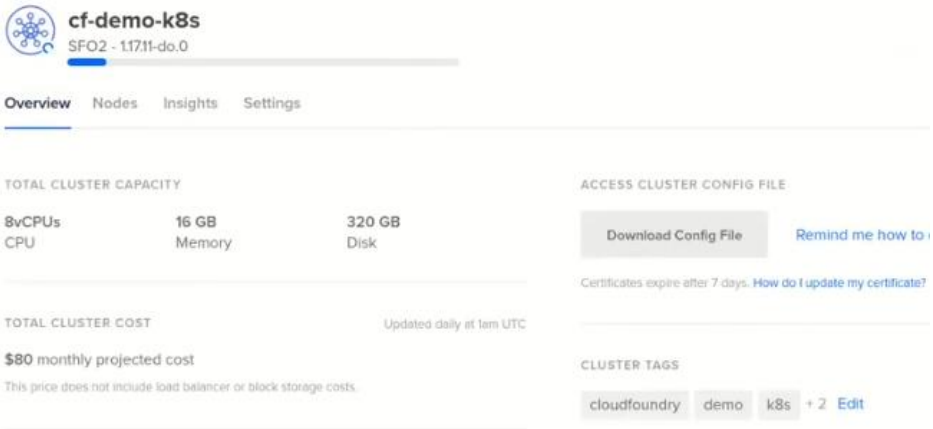
- Add digital ocean name severs to the DNS provider of a domain that you own



- Navigate to the digital ocean concole and add a domain under the network you have and create a kubernetes cluster


Domains - DigitalOcean - Mozilla Firefox
Domains - DigitalOcean
https://cloud.digitalocean.com/networking/domains?i=e428a8
Search by resource name or public IP (Ctrl+B)
Create
PROJECTS
Ram
New Project
MANAGE
Droplets
Kubernetes
Volumes
Databases
Spaces
Images
Networking
Monitoring
DISCOVER
Marketplace
ACCOUNT
Networking
Domains
Floating IPs
Load Balancers
VPC
Firewalls
PTR records
Looks like there are no domains here.
You can add one easily, though. Enter a domain be
This project has been selected as you only have one project
NS with DigitalOcean.
Enter domain
padarekha
Ram
Add Domain


Control Panel - DigitalOcean - Mozilla Firefox
Control Panel - DigitalO
https://cloud.digitalocean.com/kubernetes/clusters/new?i=e428a8&region=sfo2
Search by resource name or public IP (Ctrl+B)
Create
PROJECTS
Ram
New Project
MANAGE
Droplets
Kubernetes
Volumes
Databases
Spaces
Images
Networking
Monitoring
DISCOVER
Marketplace
Create a cluster
Select a Kubernetes version
Select the Kubernetes version. The newest available version is selected by default.
1.17.11-do.0
Tip: We generally recommend the latest version unless your team has a specific need. See the DigitalOcean Kubernetes release notes.
Choose a datacenter region
Your Kubernetes cluster will be located in a single datacenter.
New York
Amsterdam
San Francisco
Singapore
London
Toronto
Bangalore


cf-demo-k8s
SFO2 - 1.17.11-do.0
Overview
Nodes
Insights
Settings
TOTAL CLUSTER CAPACITY
8vCPUs
CPU
16 GB
Memory
320 GB
Disk
ACCESS CLUSTER CONFIG FILE
Download Config File
Remind me how to
Certificates expire after 7 days. How do I update my certificate?
TOTAL CLUSTER COST
Updated daily at 1am UTC
$80 monthly projected cost
This price does not include load balancer or block storage costs.
CLUSTER TAGS
cloudfoundry
demo
k8s
+ 2
Edit

Tools needed next(installed and checking them)

1. Digital Ocean cloud console (doctl)

```
ram@padarekha:~$ doctl version
doctl version 1.46.0-release
Git commit hash: d088bfae
```

2. kubernetes cloud console (kubectl)

```
ram@padarekha:~$ kubectl version
Client Version: version.Info{Major:"1", Minor:"18", G
tVersion:"v1.18.8", GitCommit:"9f2892aab98fe339f3bd70
3c470144299398ace", GitTreeState:"clean", BuildDate:"
020-08-26T20:32:49Z", GoVersion:"go1.13.15", Compiler
"gc", Platform:"linux/amd64"}
Unable to connect to the server: error executing acce
s token command "/snap/google-cloud-sdk/149/bin/gclou
 config config-helper --format=json": err=fork/exec /
nap/google-cloud-sdk/149/bin/gcloud: no such file or
irectory output= stderr=
ram@padarekha:~$
```

3. ytt

```
ram@padarekha:~$ ytt version
ytt version 0.30.0
```

4. kapp

```
ram@padarekha:~$ kapp version
kapp version 0.33.0

Succeeded
```

5. cloud foundry cli

```
ram@padarekha:~$ cf -v
cf version 6.51.0+2acd15650.2020-04-07
```

6. BOSH cli

```
ram@padarekha:~$ bosh -v
version 6.3.1-c44e8a1d-2020-07-09T21:08:12Z

Succeeded
```

- **Adding the cluster to current context in Kubeconfig**

```
ram@padarekha:~$ doctl kubernetes cluster kubeconfig s
ave cf-demo-k8s
Notice: Adding cluster credentials to kubeconfig file
found in "/home/ram/.kube/config"
Notice: Setting current-context to do-sfo2-cf-demo-k8s
ram@padarekha:~$
```

- **Clone the github repository for Cloud Foundary for k8s**

```
ram@padarekha:~$ git clone https://github.com/cloudfou
ndry/cf-for-k8s.git
Cloning into 'cf-for-k8s'...
remote: Total 9184 (delta 47), reused 49 (delta 14), p
ack-reused 9062
Receiving objects: 100% (9184/9184), 2.24 MiB | 1.45 M
Receiving objects: 100% (9184/9184), 2.39 MiB | 1.43 M
iB/s, done.
Resolving deltas: 100% (5554/5554), done.
```

- **Enter the below director for k8s**

```
ram@padarekha:~$ cd cf-for-k8s
```

- **Create a temporal swap space in directory**

```
ram@padarekha:~/cf-for-k8s$ mkdir ~/tempdir
```

- **Using the script we auto generate values for the deployment of app**

```
ram@padarekha:~/cf-for-k8s$ ./hack/generate-values.sh
-d padarekha.com > ~/tempdir/cf-values.yml
WARNING: The hack scripts are intended for development
 of cf-for-k8s.
  They are not officially supported product bits.  The
ir interface and behavior
  may change at any time without notice.
```

- **Parameters for successful deployment**

```
ram@padarekha:~/cf-for-k8s$ echo "add_metrics_server_c
omponents: true" >> ~/tempdir/cf-values.yml
```

- **Add credentials to container**

```
ram@padarekha:~/cf-for-k8s$ vi ~/tempdir/cf-values.yml
```

```
#@data/values
---
system_domain: "padarekha.com"
app_domains:
#@overlay/append
- "apps.padarekha.com"
cf_admin_password: 2t503nnqnlagme4xrrg0

blobstore:
  secret_access_key: f32y8ywj29ab6p8rf8gt

cf_db:
  admin_password: 0ubmpadrq8in8ewh9r6g

capi:
@
"~/tempdir/cf-values.yml" 161 lines, 31225 characters
```

- **Use ytt file to generate the final template we are going to use for deployment**

```
ram@padarekha:~/cf-for-k8s$ kapp deploy -a cf -f ~/tem
pdir/cf-values-rendered.yml
Target cluster 'https://7c95854b-7ee3-42c2-ae48-25e6b2
40e940.k8s.ondigitalocean.com' (nodes: pool-2hqx6na5j-
3tblh, 2+)
```

```
3:30:23PM: ok: reconcile deployment/log-cache (apps/v1
) namespace: cf-system
3:30:23PM: ---- applying complete [6/6 done] ----
3:30:23PM: ---- waiting complete [6/6 done] ----

Succeeded
```

- **Back in Digital ocean console, add an error code map to your load balancer**
- **Then check if installation is complete**

```
ram@padarekha:~$ cf api --skip-ssl-validation api.padarekha.com
Setting api endpoint to api.padarekha.com...
OK

api endpoint:   https://api.padarekha.com
api version:    2.154.0
ram@padarekha:~$
```

```
ram@padarekha:~$ cf login
API endpoint: https://api.padarekha.com
```

Now we can start to deploy apps

7. Deploying a sample app Using Cloud Foundry to K8s

App deployment involves uploading, staging, and starting the app in a container. Your app must complete each of these phases within a certain time limit. The default time limits for the phases are:

Upload: 15 minutes

Stage: 15 minutes

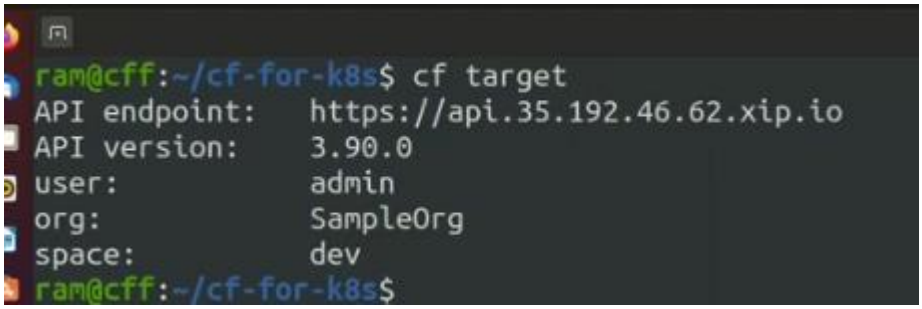
Start: 60 seconds

Your administrator can change these default settings. Check with your administrator for the actual time limits set for app deployment.

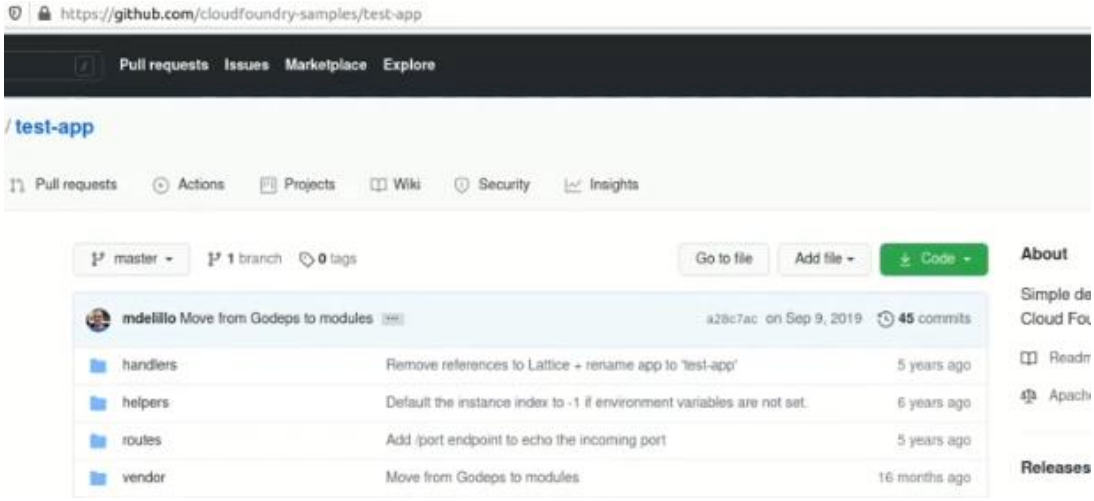
Developers can change the time limit for starting apps through an app manifest or on the command line.as not exited after ten seconds, Cloud Foundry sends a SIGKILL.

7.1 Steps

- **First our target is kubanetes cluster**



- **Deploying a sample app stored on github, cloud foundry test app**



- **Clone the app from its github repo and view**

```
ram@cff:~/cf-for-k8s$ cd
ram@cff:~$ git clone https://github.com/cloudfoundry-samples/test-app.git
Cloning into 'test-app'...
remote: Enumerating objects: 305, done.
remote: Total 305 (delta 0), reused 0 (delta 0), pack-reused 305
Receiving objects: 100% (305/305), 85.88 KiB | 240.00 KiB/s, done.
Resolving deltas: 100% (116/116), done.
ram@cff:~$ cd test-app
ram@cff:~/test-app$ ls
build.sh    go.mod  handlers  LICENSE  manifest.yml  README.md  vendor
Dockerfile  go.sum  helpers   main.go  Procfile      routes
```

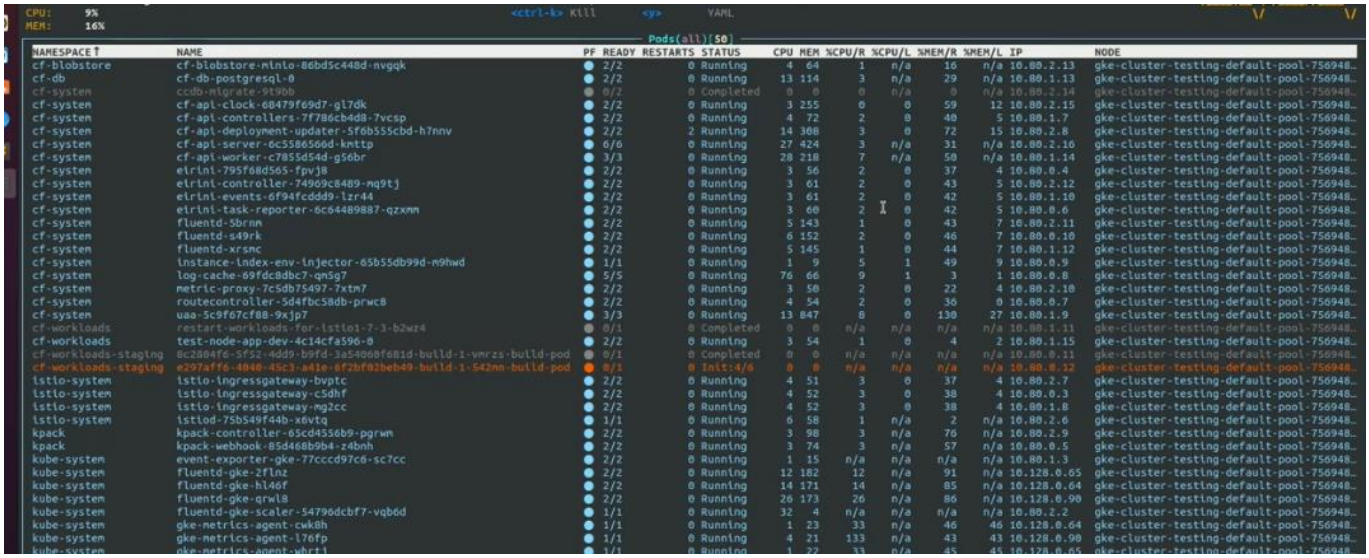
- **Check manifest file**
- **It contains the name of the app and gives other info like the number of instances to be depoyed**

```
ram@cff:~/test-app$ more manifest.yml
---
applications:
  - name: test-app
    memory: 256M
    instances: 1
    random-route: true
ram@cff:~/test-app$
```

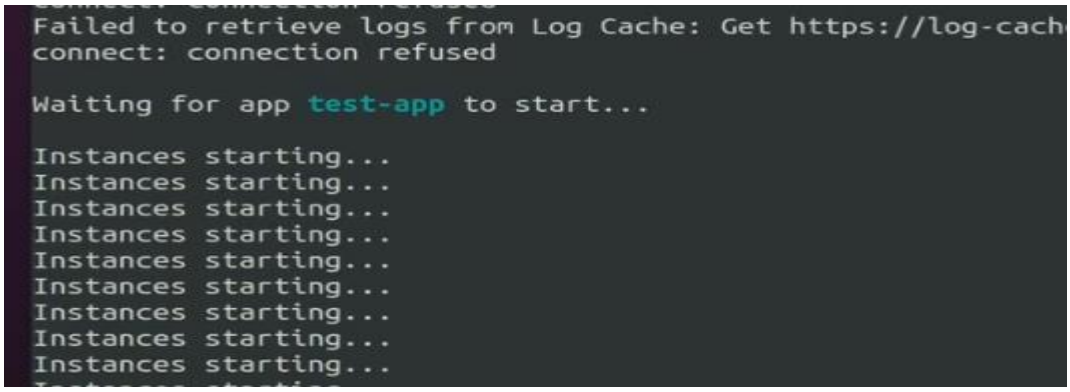
- **Deploying using cf to kubanetes using cf push since we already hard the app loction known in the test app directory**

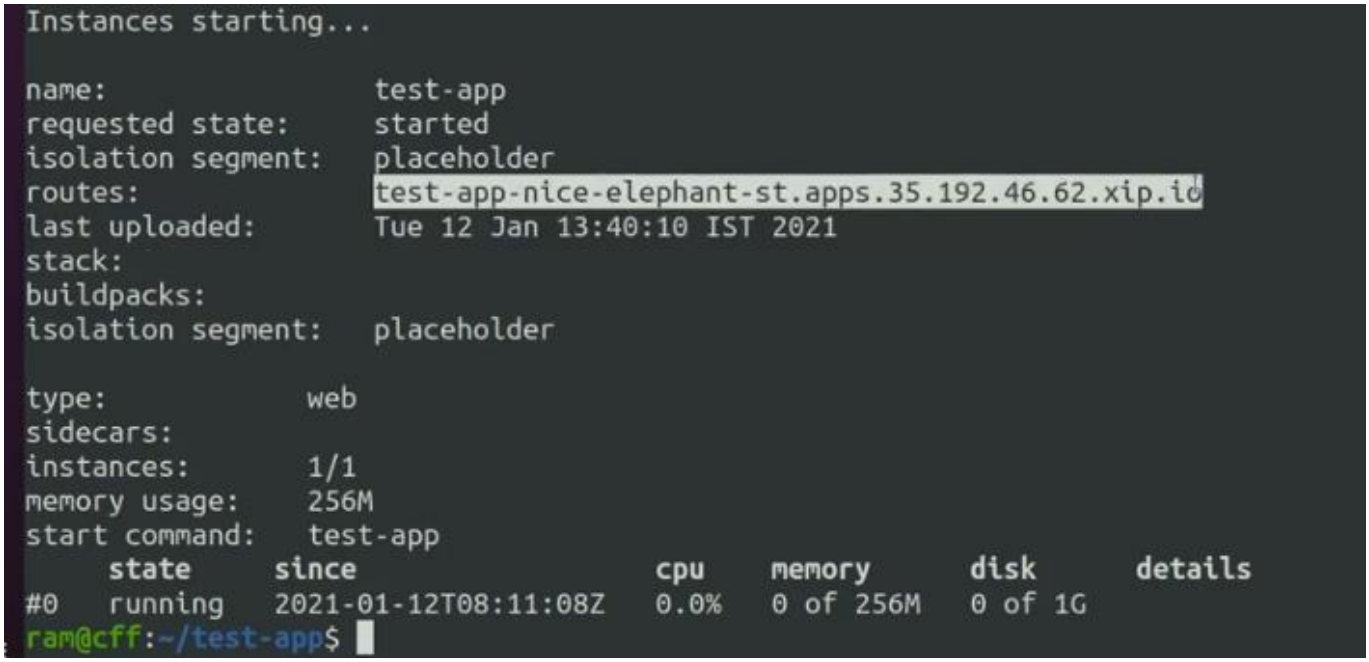
```
ram@cff:~/test-app$ cf push
Pushing app test-app to org SampleOrg / space dev as admin...
Applying manifest file /home/ram/test-app/manifest.yml...
Manifest applied
Packaging files to upload...
Uploading files...
 36.87 KiB / 36.87 KiB [==========================================================================================] 100.00% 1s

Waiting for API to complete processing files...

Staging app and tracing logs...
Failed to retrieve logs from Log Cache: Get https://log-cache.35.192.46.62.xip.io/api/v1/info: dial tcp 78.47.226.171:443:
connect: connection refused
Failed to retrieve logs from Log Cache: Get https://log-cache.35.192.46.62.xip.io/api/v1/info: dial tcp 78.47.226.171:443:
connect: connection refused
Failed to retrieve logs from Log Cache: Get https://log-cache.35.192.46.62.xip.io/api/v1/info: dial tcp 78.47.226.171:443:
connect: connection refused
Failed to retrieve logs from Log Cache: Get https://log-cache.35.192.46.62.xip.io/api/v1/info: dial tcp 78.47.226.171:443:
connect: connection refused
Failed to retrieve logs from Log Cache: Get https://log-cache.35.192.46.62.xip.io/api/v1/info: dial tcp 78.47.226.171:443:
connect: connection refused
```

- **ksa console**

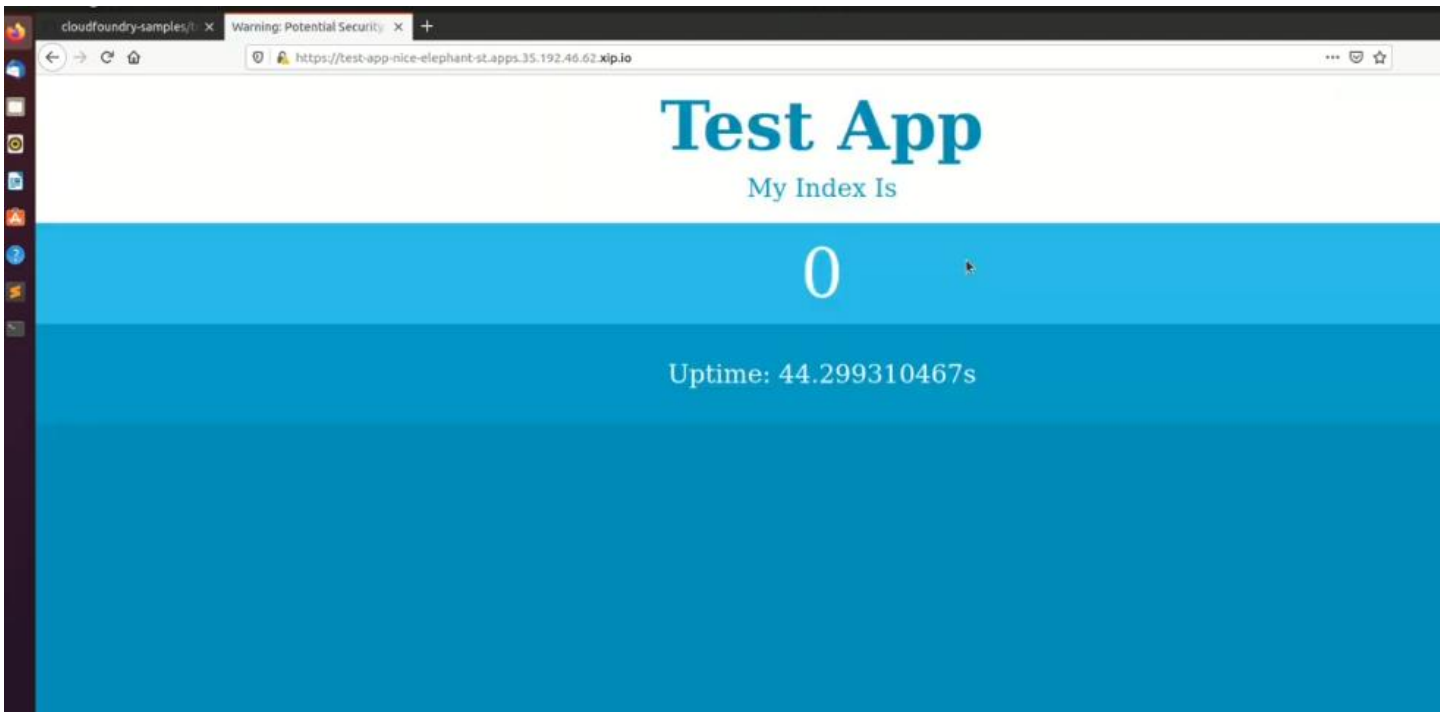


- **Takes at least 10 minutes for app to start up.**

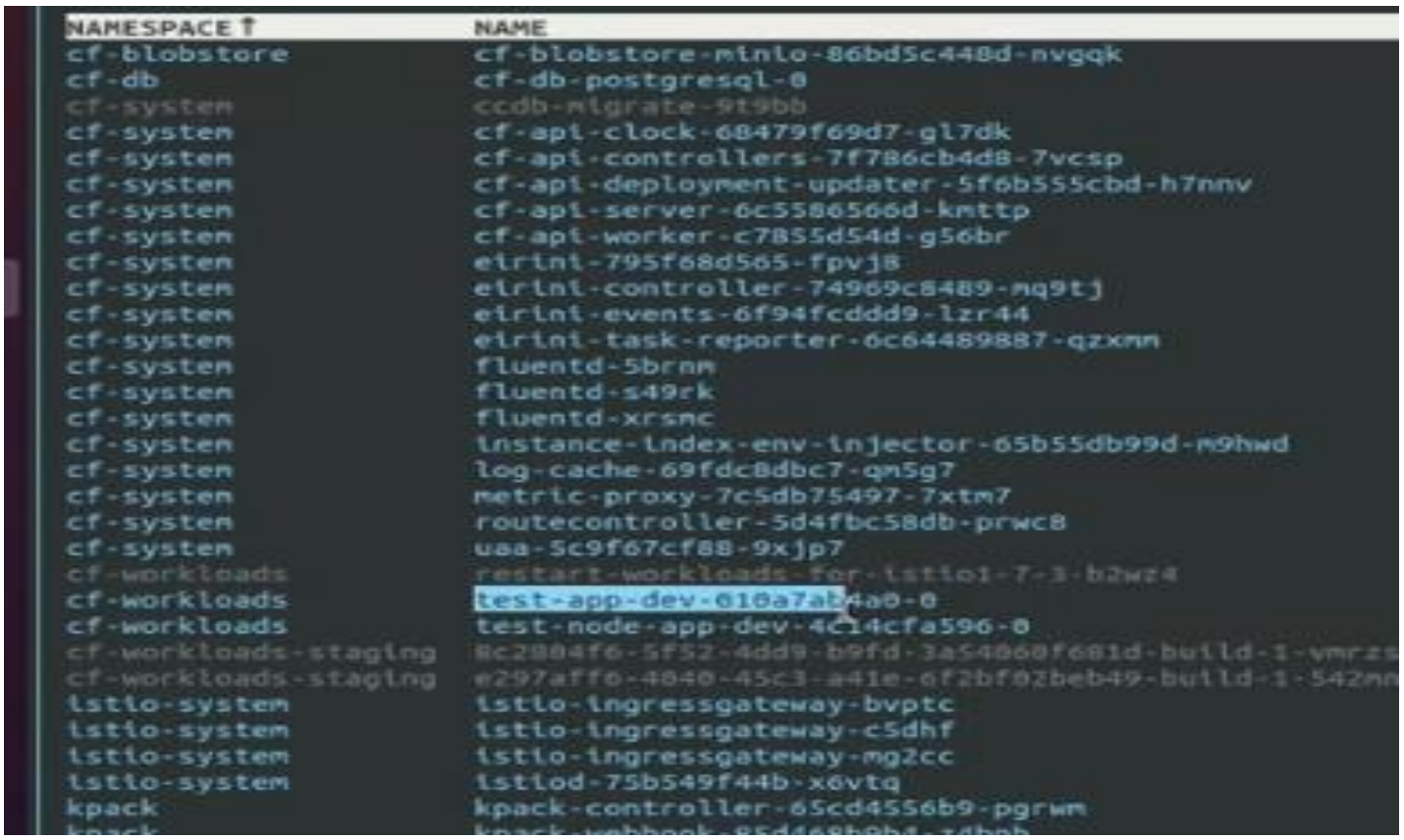




- **Copy the highlighted route after app has started and paste on browser to access app**



- **Opening ksa console again, we see app is running**



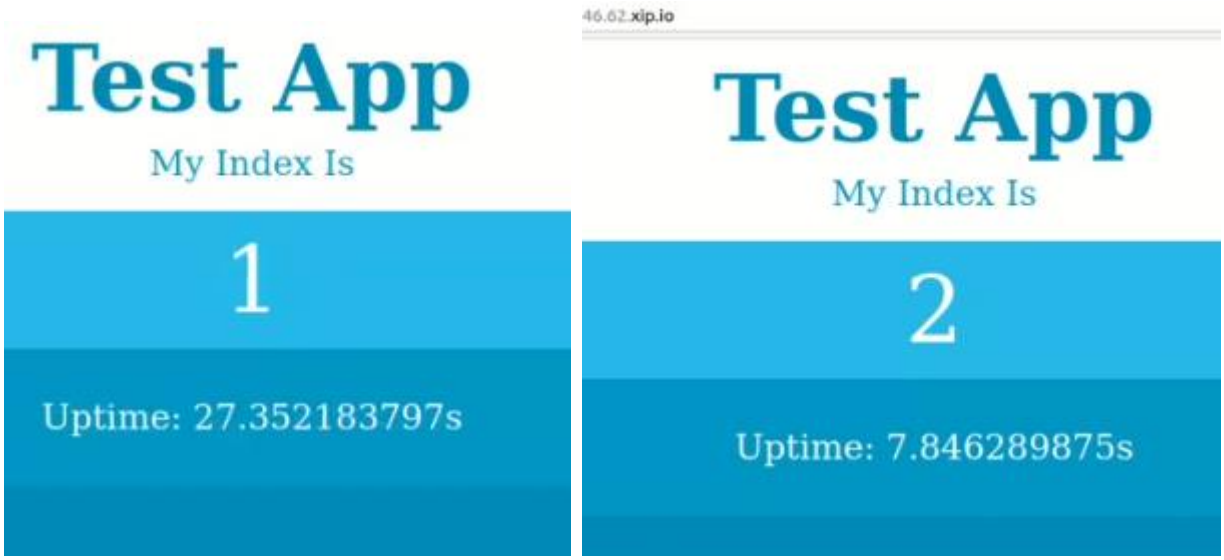
- **Scaling application when need be(providing 3 instances of the test app)**

```
ram@cff:~/test-app$ cf scale test-app -i 3
Scaling app test-app in org SampleOrg / space dev as admin...

Instances starting...

Showing current scale of app test-app in org SampleOrg / space dev as admin...

name:                test-app
requested state:     started
isolation segment:   placeholder
routes:              test-app-nice-elephant-st.apps.35.192.46.62.xip.io
last uploaded:       Tue 12 Jan 13:40:10 IST 2021
stack:
buildpacks:
isolation segment:   placeholder

type:           web
sidecars:
instances:      1/3
memory usage:   256M
     state      since                  cpu    memory          disk      details
#0   running    2021-01-12T08:11:07Z   0.0%   5.2M of 256M    0 of 1G
#1   starting   2021-01-12T08:26:43Z   0.0%   0 of 256M       0 of 1G
#2   starting   2021-01-12T08:26:43Z   0.0%   0 of 256M       0 of 1G
ram@cff:~/test-app$
```

- **Check instances created on k8s**

```
cf-system              uaa-5c9f67cf88-9xjp7
cf-workloads           restart-workloads-for-istio1-7-3-b2wz4
cf-workloads           test-app-dev-010a7ab4a0-0
cf-workloads           test-app-dev-010a7ab4a0-1
cf-workloads           test-app-dev-010a7ab4a0-2
cf-workloads           test-node-app-dev-4c14cfa596-0
cf-workloads-staging   8c2804f6-5f52-4dd9-b9fd-3a54060f681d-build-1-
cf-workloads-staging   e297aff6-4048-45c3-a41e-6f2bf82beb49-build-1-
istio-system           istio-ingressgateway-bvptc
```

- **On browser three instances of index 0, 1, 2 load balance at time interval**



7.2 Crash events

If an app instance fails, Cloud Foundry restarts it by rescheduling the instance on another container three times. After three failed restarts, Cloud Foundry waits 30 seconds before attempting another restart. The wait time doubles each restart until the ninth restart, and remains at that duration until the 200th restart. After the 200th restart, Cloud Foundry stops trying to restart the app instance.

7.3 Shutdown

Cloud Foundry requests a shutdown of your app instance when:

- A user runs `cf scale`, `cf stop`, `cf push`, `cf delete`, or `cf restart-app-instance`.
- A system event occurs, such as the replacement procedure during Diego Cell evacuation or when an app instance stops because of a failed health check probe.

To stop the app, Cloud Foundry sends the app process in the container a SIGTERM. By default, the process has ten seconds to shut down gracefully. If the process h

Conclusion

In conclusion, Cloud Foundry offers a powerful open-source platform for application development. This report explored its benefits as a PaaS solution, its core functionalities, and how to deploy applications on it. Cloud Foundry's emphasis on open-source development, security, and flexibility makes it a compelling choice for developers seeking a robust environment to build and manage their apps.

References

[1] https://cloud.google.com/learn/what-is-paas#:~:text=Platform%20as%20a%20Service%20(PaaS)%20is%20a%20complete%20cloud%20environment,middleware%2C%20tools%2C%20and%20more.

[2] https://www.spiceworks.com/tech/cloud/articles/what-is-platform-as-a-service/

https://www.techtarget.com/searchcloudcomputing/definition/Cloud-Foundry

https://docs.cloudfoundry.org/concepts/

https://www.suse.com/suse-defines/definition/cloud-foundry

https://cloudacademy.com/blog/cloud-foundry-benefits/