PCF

PCF is a cloud env also similar to AWS

Ref links

|  |  |
| --- | --- |
| <https://github.com/in28minutes/pcf-crash-course-with-spring-boot> |  |

From PCF CLI from local itself u can push the jar and deploy

In PCF space means environment

Routes should be globally unique

Best way to deploy is 1) prepare a manifest.yml file and 2) execute a command called “cf push” as we are using the default file name “manifest.yml” file pcf will take inputs from

That file and it will create instances

Commands

|  |  |
| --- | --- |
| cf login ex:- cf8 login  and then it will ask email , password  in wells fargo It will ask for uid- u849670 not email id |  |
| cf target  u can see the target server name | **Getting all env**  Cf spaces (space is nothing but environment)  Cf orgs  Cf apps 🡪 to see what are all the apps running  Cf services 🡪 to see all app available services  Cf routes --> to see all the url’s mapped to each appln  Cf logs <app name>🡪 u can see all the logs  Cf env <app name> 🡪 to see all the env related infor – like all properties |
| **To deploy**  cf push <application-name>  cf push <application-name> 🡨 --random-route>  if same route already exists it will create a new route | **pushing the jar**  cf push <app/component name> -p <path of the jar file>  where p refers to path of the jar file  ex:- cf push <CA9999> -p <target/codi-db-producer.jar>  even if u input as a jar, droplet is the one that is going to be deployed finally |
| Cf apps  To see the list of applications and how much memory its consuming | Cf routes |
| |  |  | | --- | --- | | To access an application we can configure many url’s like  Normal url, MTLS url  Lly we can configure many url’s/route’s (ways) |  |   And u can see all the url’s / routes using command “cf routes” |  |
| **To stop an app**  Cf stop <app name> | **To start/re-start an app**  Cf start <app-name> / cf restart <app-name>  Cf start hello-world-rest-api |
| **Manifest file creation –**  All the details like –   1. path of where is the jar file 2. ram space like java build pack/.net build pack, 3. num of instances we want 4. how many routes(url’s) we want for same app to access, 5. build pack information (like java build pack/ .net build pack) 6. u can tell which pcf services u need to bind to 7. all the details we can keep in yaml file and u can pass that yaml file directly to pcf, so that it will create all app with those details   Now if u do cf push by default it will take manifest.yml file as an input with all those details it will install the appln | Default name is manifest.yml  Here If u don’t mention build pack information all languages build pack info will be downloaded |
|  |  |
| Environment variables:  Here if u see all the environment variables are set to a new variable called  “VCAP\_APPLICATION” , “VCAP\_SERVICES”  So VCAP\_SERVICES is an environment variable name which will have all the properties listed (this is similar to hash map which will have all the details) | “cf env <app name>” to see the list of environment variables |

Package = jar + manifest.yml

Droplet = package + build pack = encapsulated source code (process of creating a source code is called staging)

This jar will be deployed into a server/cell

Services

Services – databases, queues, JNDI registry etc. which depends on ur application

|  |  |
| --- | --- |
| These are all the services that are available in PCF which u can use  Ex:- these SQL db’s are also will be installed in the cloud  Just creating the service is not enough we have **to bind that service** to our app    Now u can see    This new service is binded to our app, once we binded our application will be automatically connected to pcf database service |  |