Tutorial

User Story (main task):

develop micro service which will consume list of field which describe customer, will produce account #

Request: have to include: First Name,
Last Name,
DOB,
SSN,
Email ID,
Mobile Number,
Home Address,
Mailing Address,
Account Type [Savings, Checking, CD],
Minimum Balance [If Savings - $200, Checking - $300, CD-$2000],

Response:
account #

Tasks:

task 0: pre requirement
task 1: develop microservices
task 2: Code have to be pushed to GitHub
task 3: deploy onto PCF
task 4: install Jenkins, make automatization deployment with Jenkins

Requirement:

task 1: have to use:
- Spring boot 2+
  - RestControllers
  - Services
  - Repositories (CRUDRepository)
- Java 8 (Streams)
- lombok
  - for all beans (POJOs) have to used lombok
- Logger
- CrudRepository
- MySQL

task 3:
- Source code have to be on the GitHub

task 2:
- Microservice have to be deployed onto PCF
- DB have to be installed on the PCF
- Microservice have to save data into DB on the PCF
- account # have to be primary key in the DB table
- * Microservice have to provide result thru Postman
- ** DB have to provide connection thru WorkBench

task 4:
- Jenkins have to be installed
- 3 Jobs have to be added to Jenkins: Download code from GitHub, build fat jar, deploy onto PCF

Expected result:

task 1: developed microservices, on the local in the STS

task 2: Code have to be pushed to GitHub, can check on the GitHub + pull

task 3: deploy onto PCF, can check on the Postman

task 4: install Jenkins, make automatization deployment with Jenkins, run Jenkins jobs, application deployed/run on the PCF

Implementation:

Task 0: installation

1.install JDK 8, last build

Java SE Development Kit 8 Downloads

* here i pointed to the Java 8 because if you will use another JDK, then in your project will be need another dependencies

```
sergeys-mbp:~ Sergey$ java -version
java version "1.8.0_191"
Java(TM) SE Runtime Environment (build 1.8.0_191-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.191-b12, mixed mode)
sergeys-mbp:~ Sergey$
```

2.install git

https://git-scm.com/download/win

after installation you have to check on the console (terminal)

command: git --version

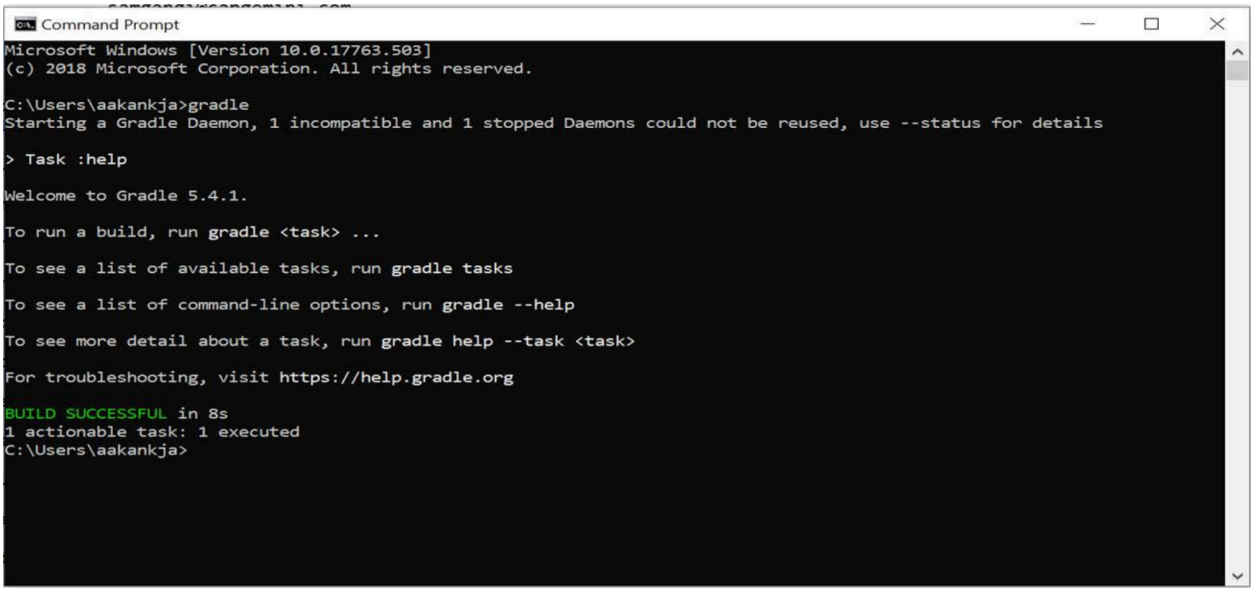
if you can see version, then installation correct

```
sergeys-mbp:~ Sergey$ git --version
git version 2.20.1 (Apple Git-117)
sergeys-mbp:~ Sergey$
```

3.install gradle

https://gradle.org/install/

https://gradle.org/guides/



if you see this result, then it means so installation success.

4. install STS

https://spring.io/tools

STS is Eclipse for Spring

i hope you are familiar with Eclipse

5. Install MySQL

https://www.mysql.com/

You will need MySQL server and WorkBench

please download community edition

if web page ask you to provide oracle password, please create account on the oracle.com

I highly recommend to use 5.7 version, because on the 8 version you will need to install too much extra soft (visual studio)

https://dev.mysql.com/downloads/

Here was described few easy installation things, i was not describe those in deep because i hope you can do it.

Task 1: develop microservices

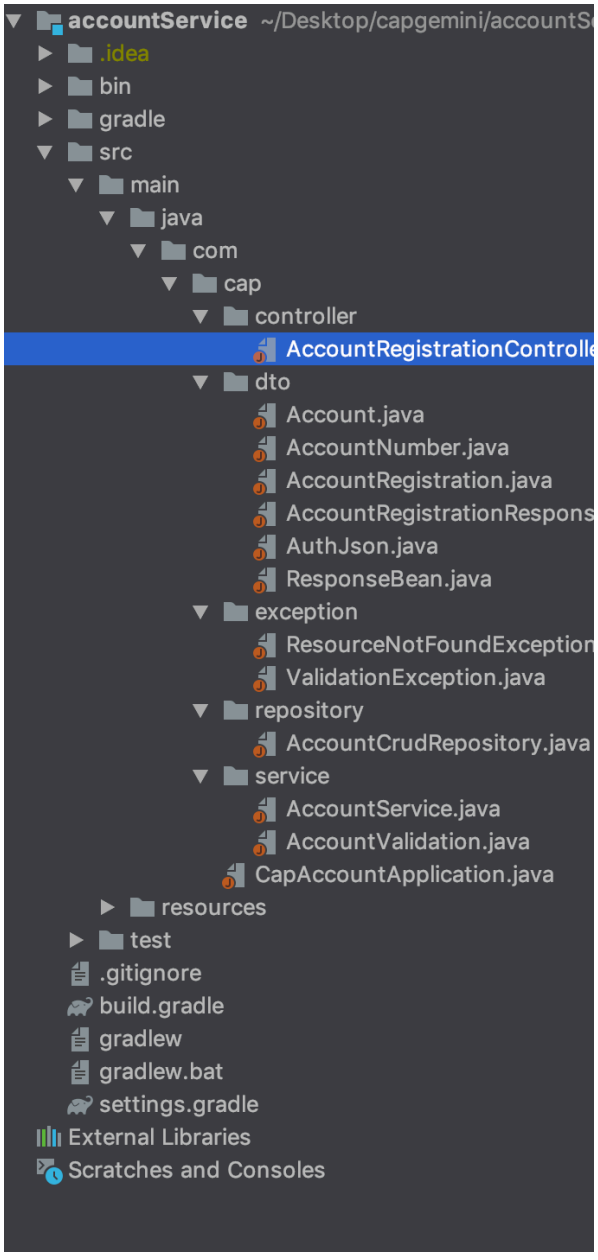
For the developing micro services we have to use:

- Spring boot 2+
  - RestControllers
  - Services
  - Repositories (CRUDRepository)
- Java 8 (Streams)
- lombok
  - for all beans (POJOs) have to used lombok
- Logger
- Spring JPA
- MySQL

Application have to include 3 layers: Representation, Logic, Repositories.
As connection to the DB we have to use Spring JPA, CrudRepository.
We have to add Unit tests. Tests have to include mockMVC and Reflection.
We have to create own Exception.
Microservice always have to return correct response, newer return exception.
On the POJO's we have to use Lombok.

Please take a look to the project example:
https://github.com/capgeminiAccountProject/accountService.git

project structure example:



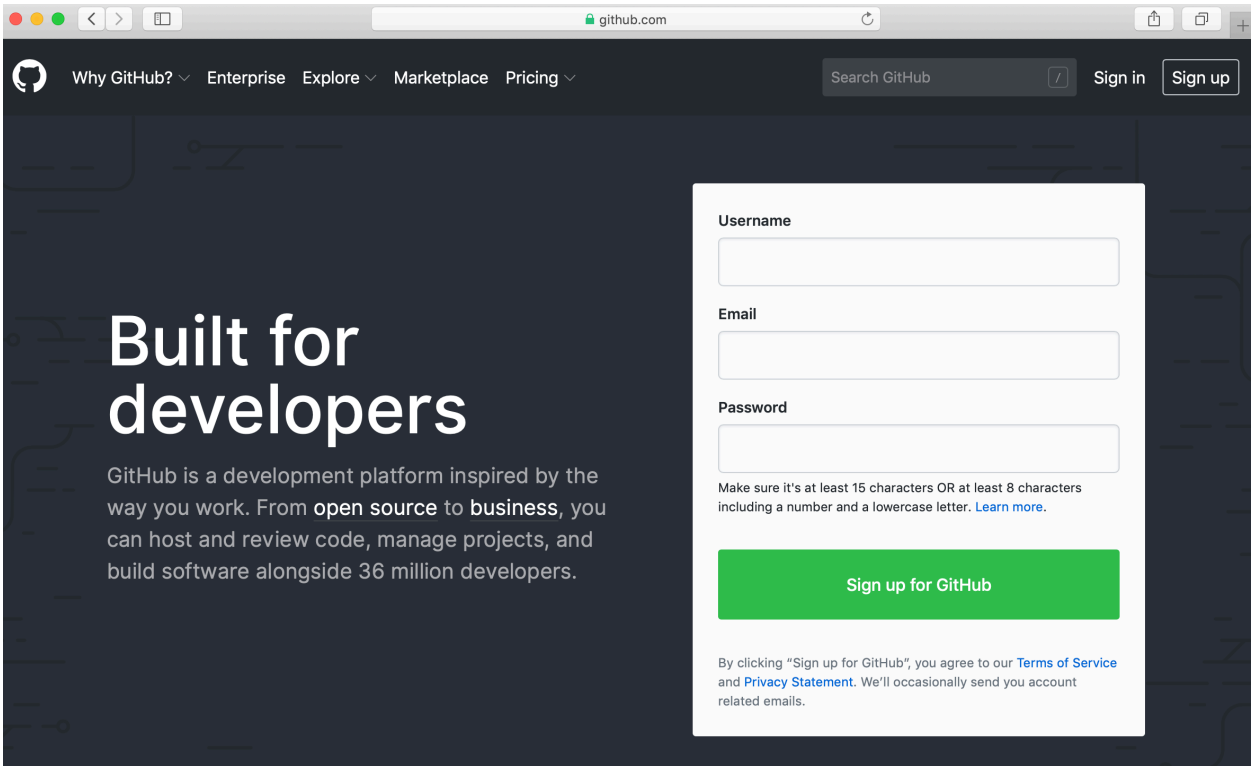
Task 2: Code have to be pushed to GitHub

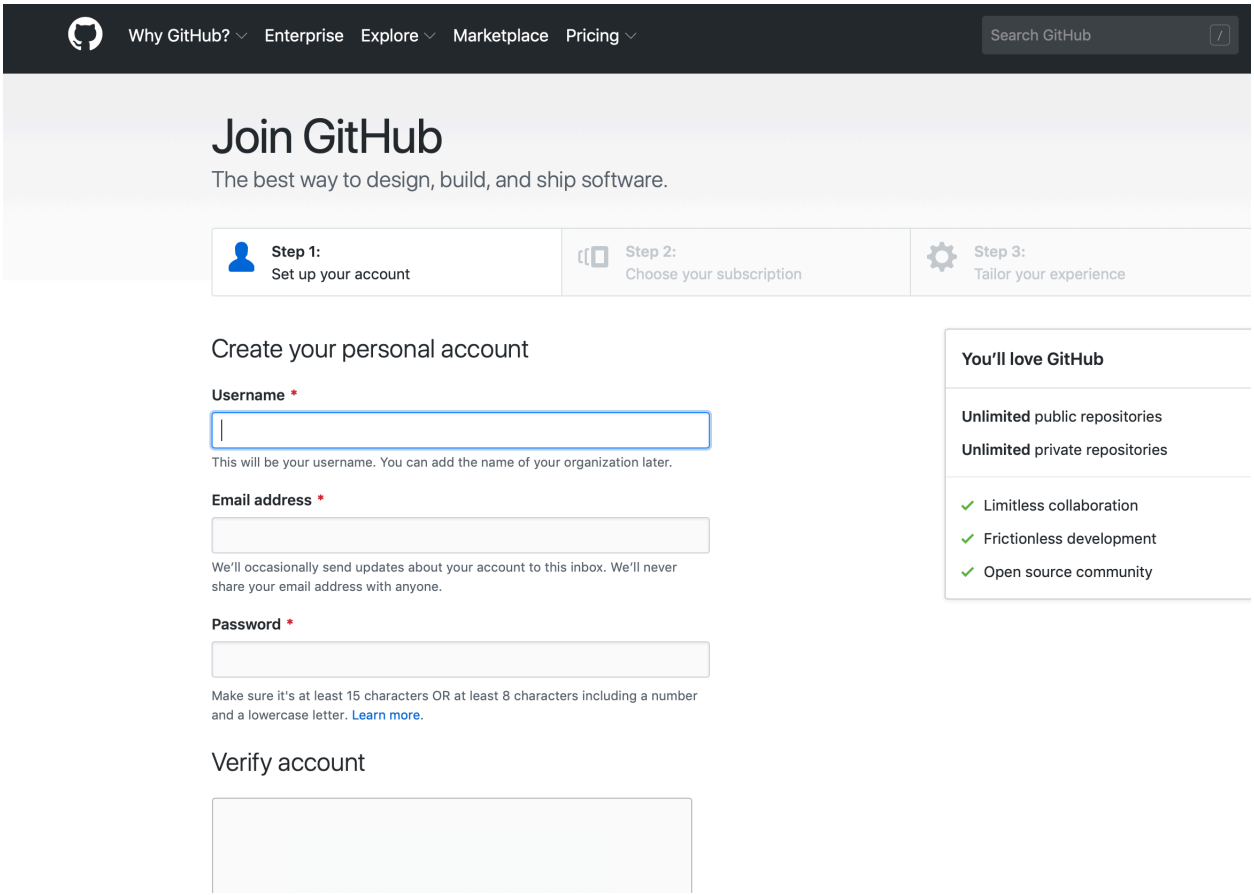
0. How GitHub is working (please read carefully, this is very important)

those steps will be described in depth

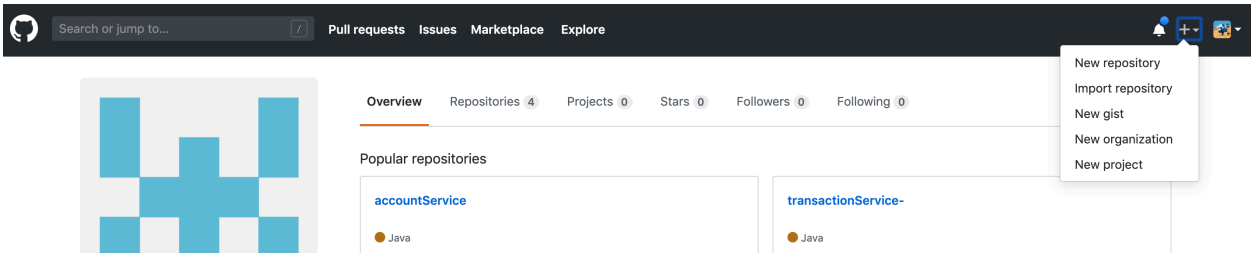
1.Create repository (this is like bucket on the GitHub). When you will creating please chose “Read me” check box.
2.Clone repository to the local (it will be empty folder, with only read me file)
3.Add your project (files from Eclipse workspace) to that folder
4.add those file to the git (only files with code and properties, do NOT add binary files)
5. commit files to the git
6. push files to the GitHub

0.Please create account on the GitHub:
(only if you have NO created before, if you have own one, then best idea to use it.
https://github.com/

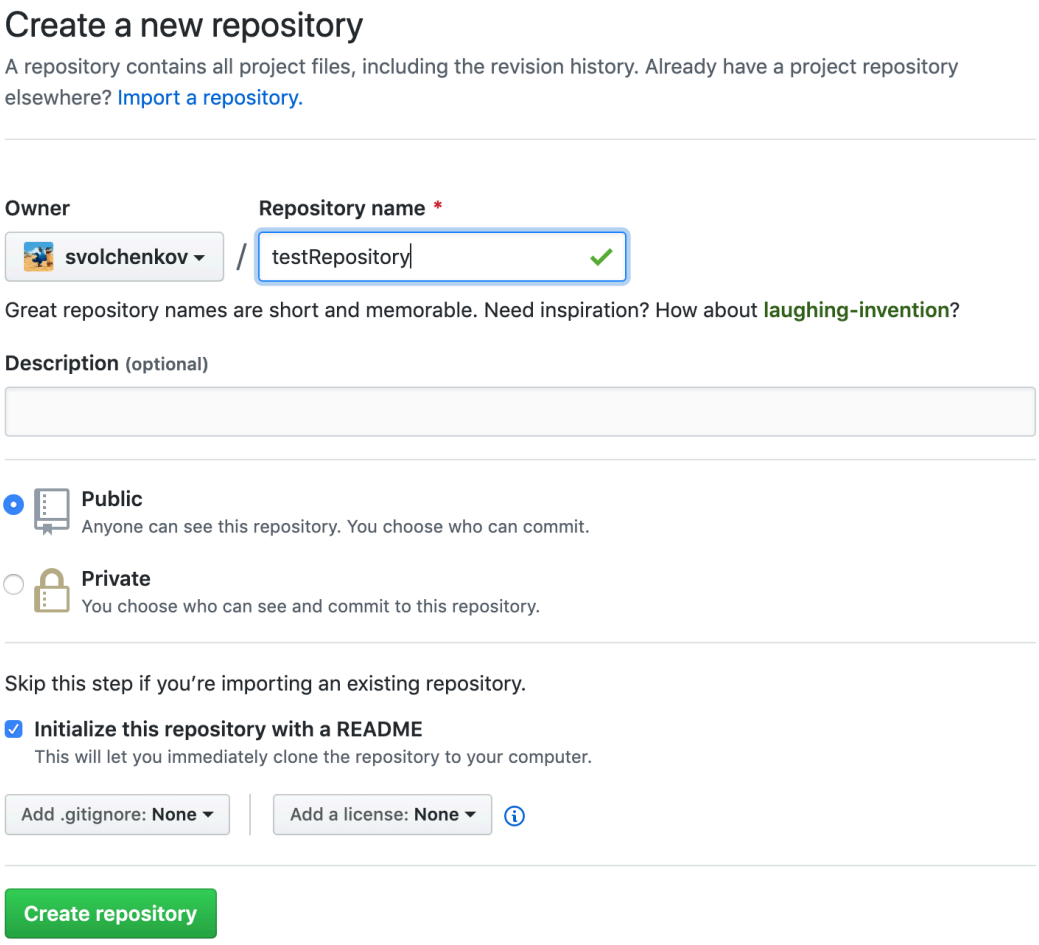




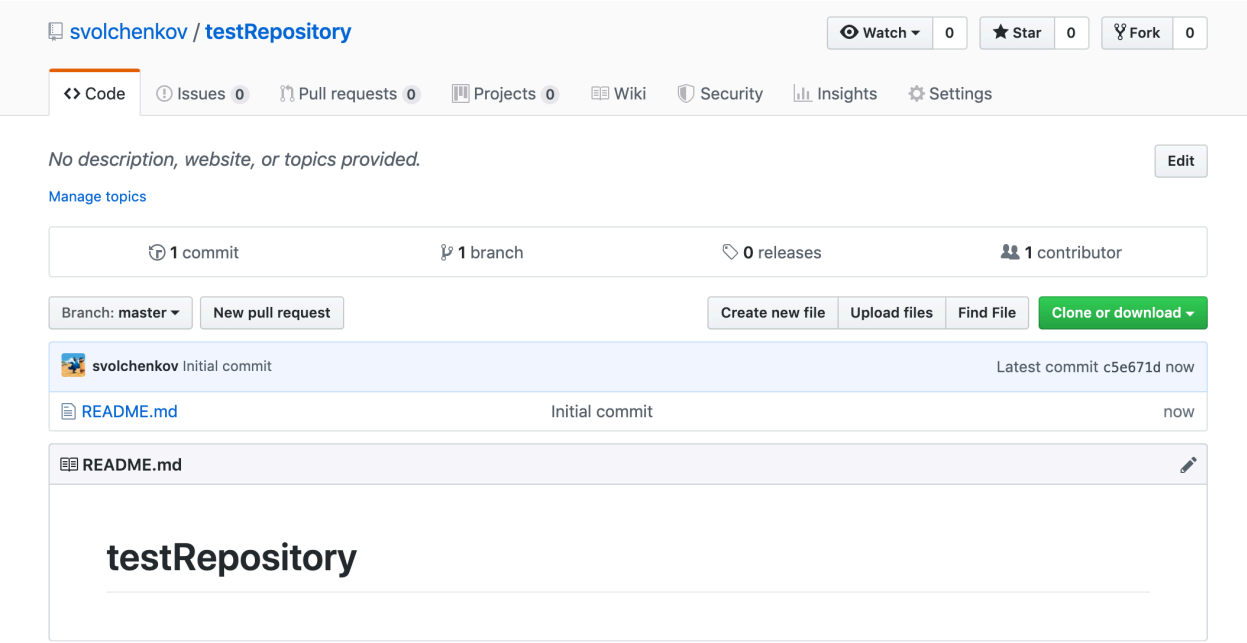
1.Create repository
step1



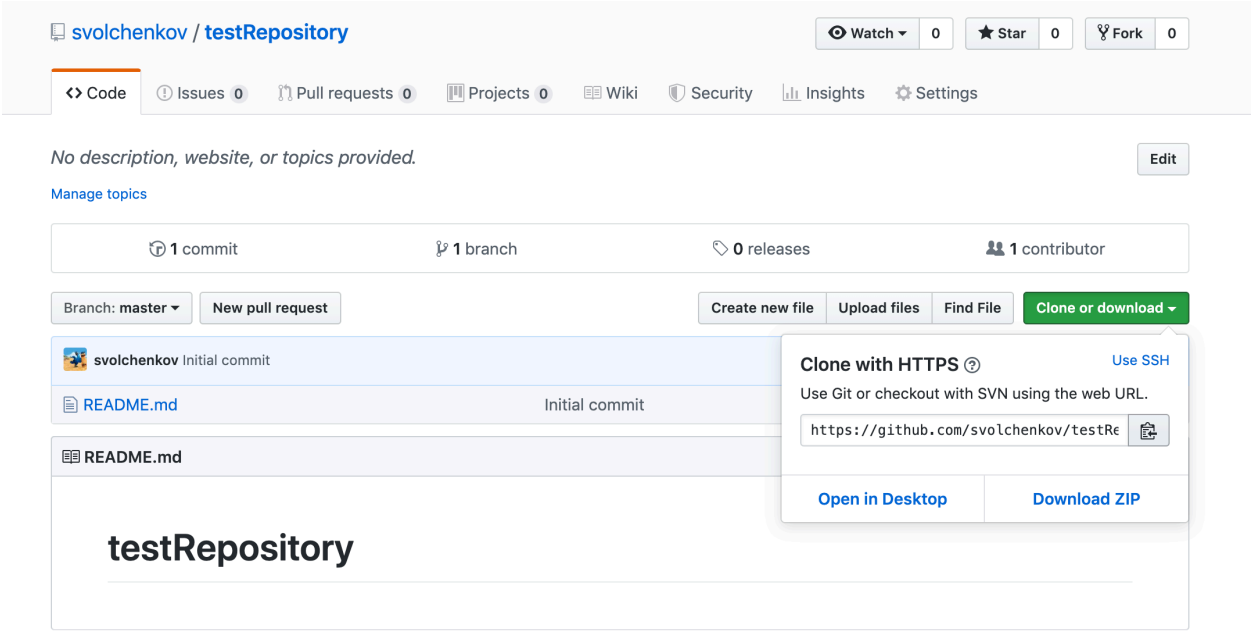
step2


Create a new repository
A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.
Owner
Repository name *
svolchenkov
/
testRepository
Great repository names are short and memorable. Need inspiration? How about laughing-invention?
Description (optional)
Public
Anyone can see this repository. You choose who can commit.
Private
You choose who can see and commit to this repository.
Skip this step if you're importing an existing repository.
Initialize this repository with a README
This will let you immediately clone the repository to your computer.
Add .gitignore: None
Add a license: None
Create repository

step3


svolchenkov / testRepository
Watch 0
Star 0
Fork 0
Code
Issues 0
Pull requests 0
Projects 0
Wiki
Security
Insights
Settings
No description, website, or topics provided.
Edit
Manage topics
1 commit
1 branch
0 releases
1 contributor
Branch: master
New pull request
Create new file
Upload files
Find File
Clone or download
svolchenkov Initial commit
Latest commit c5e671d now
README.md
Initial commit
now
README.md
testRepository

2.Clone repository
step1: for cloning repository you have to press clone or download button

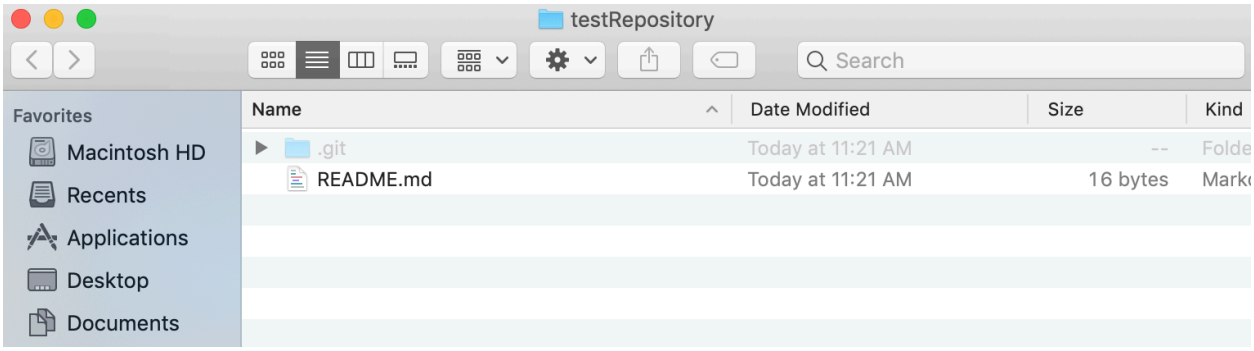


step2: and copy link

step3:
run cmd (or console or terminal),
change directory to yours (Eclipse's) workspace,
and run command
git clone https://github.com/svolchenkov/testRepository.git
here have to copied link

```
sergeys-mbp:Desktop Sergey$ git clone https://github.com/svolchenkov/testRepository.git
Cloning into 'testRepository'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
sergeys-mbp:Desktop Sergey$
```

step4:
as result you will have a folder with cloned project:



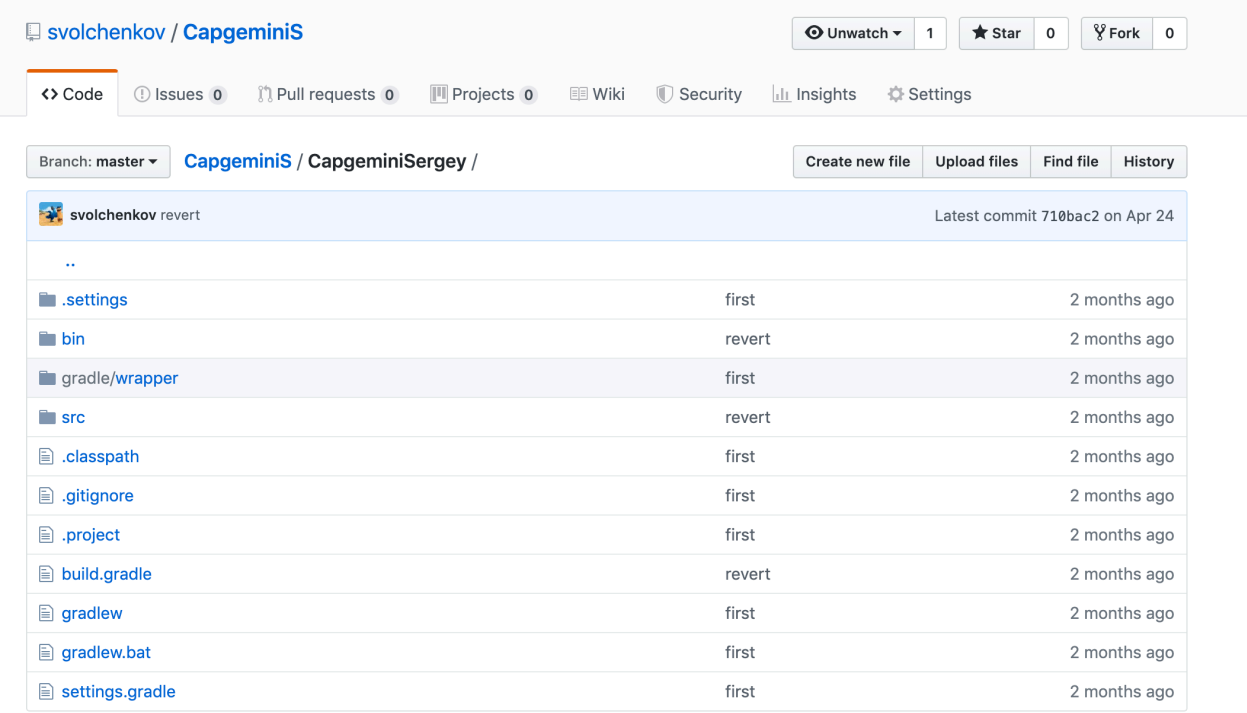
here we can see readme file, in this file you can add recommendation for project (like installation recommendation, or another)
.git folder, that folder will include git log for project (when, what was added, changed)

3.Add your project (files from Eclipse workspace) to that folder
You can copy your project to the cloned folder (and pointed your Eclipse to the new, cloned, folder)

4. Add those file to the git (only files with code and properties, do NOT add binary files)
Then you have to add files to the git (really, here datas about files will be added to the .git folder, and git will know about files and changes in those files).

5. commit files to the git
commit files means, so you will create a save point in the local git. This say point will be saved in the local .git folder. Each commit always has own hash number, and message. Usually message required.

6. push files to the GitHub
Next step push files to the GitHub. On this command git will push last commit to the GitHub. After this step you will see your files on the GitHub.



We have done with first commit.

Then we have to study PullRequests.
When you are working with real project, you will work thru next steps:
1.Clone project to the local
2.Create own branch from cloned branch
3.Make changes
4.Push your own branch to the GitHub
5.create pull request from your own branch to the branch, which you cloned
6.Your team leader will check your code
7.She will add comments to your code
8.You will fix her notes
9. You will push one more time
10. Your team lead will check your code
11. If all is well, she will merge your code to the main branch

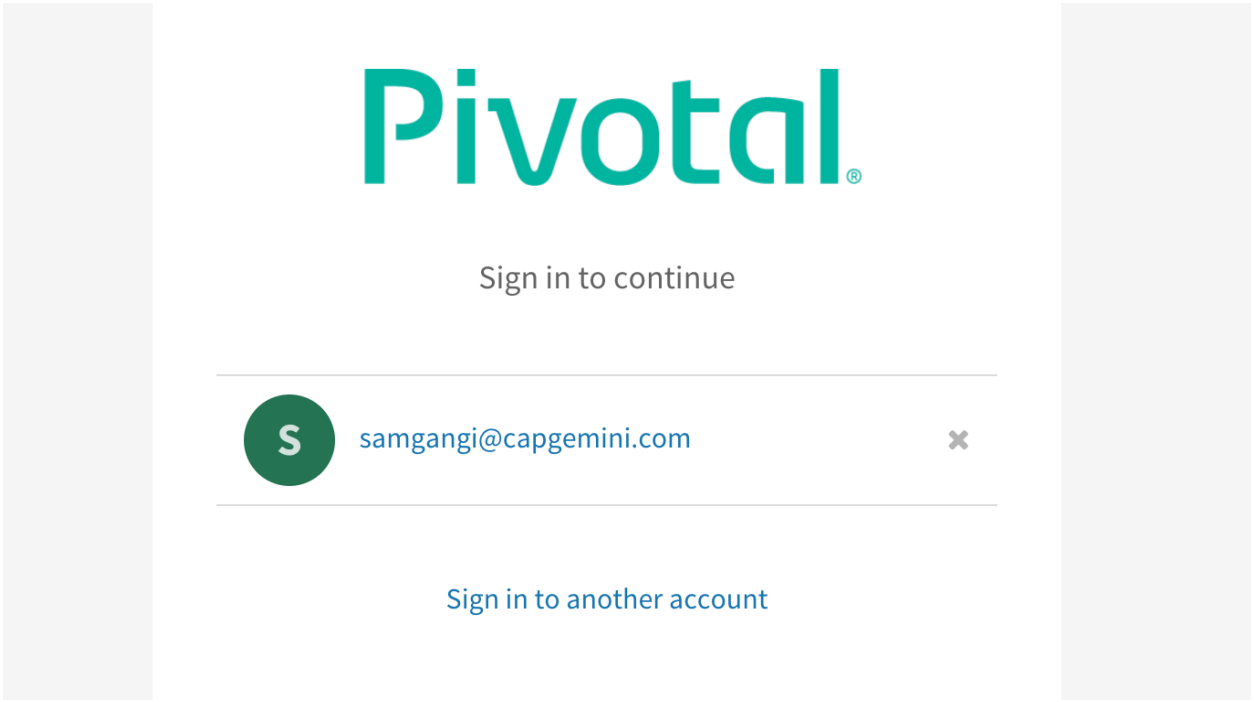
Task 3: deploy onto PCF

for this step better to use in depend WiFi connection, own hot spot or XS4Mobile from capgemini, because those network has no NAT (Network address translator)

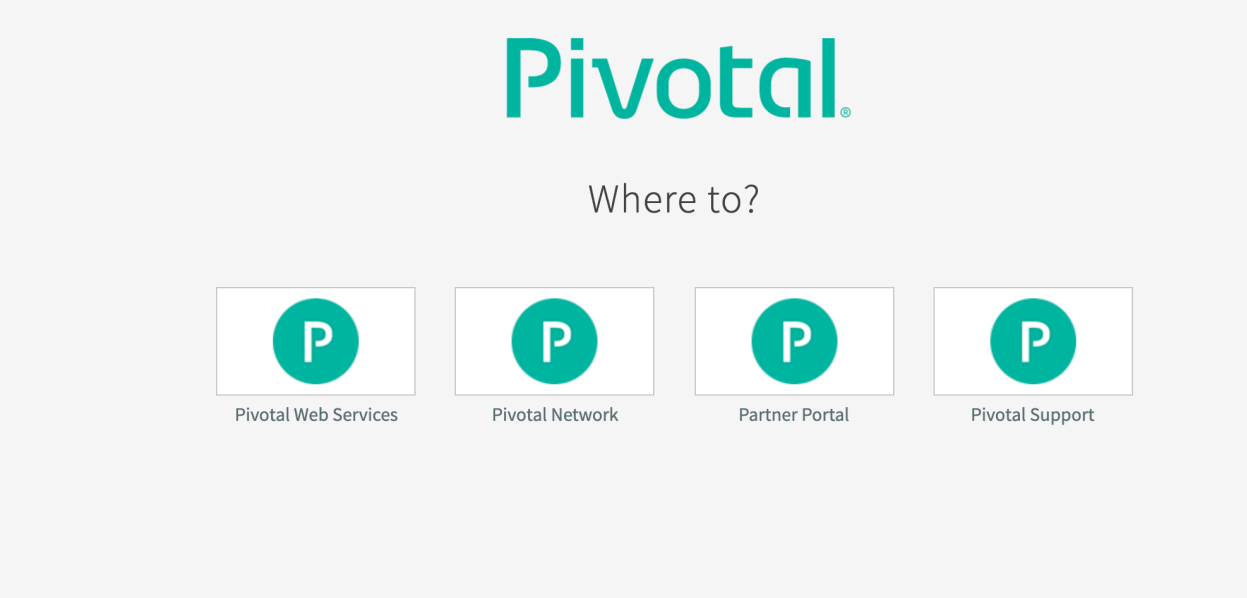
1.Crete account on the PCF
2.Create organization and workspace
3.Create DB on the PCF
4.Receive DB connection string
5.Add properties to the project
6.Rebuild project for PCF environment
7.Deploy project onto PCF

step1: Crete account on the PCF

goto: https://pivotal.io/
goto create account
fill out form
! Your account will be bind to the your phone #
! if you remove organization, then your account can start to ask same payments, - don't remove org
after your account was created, please log in:

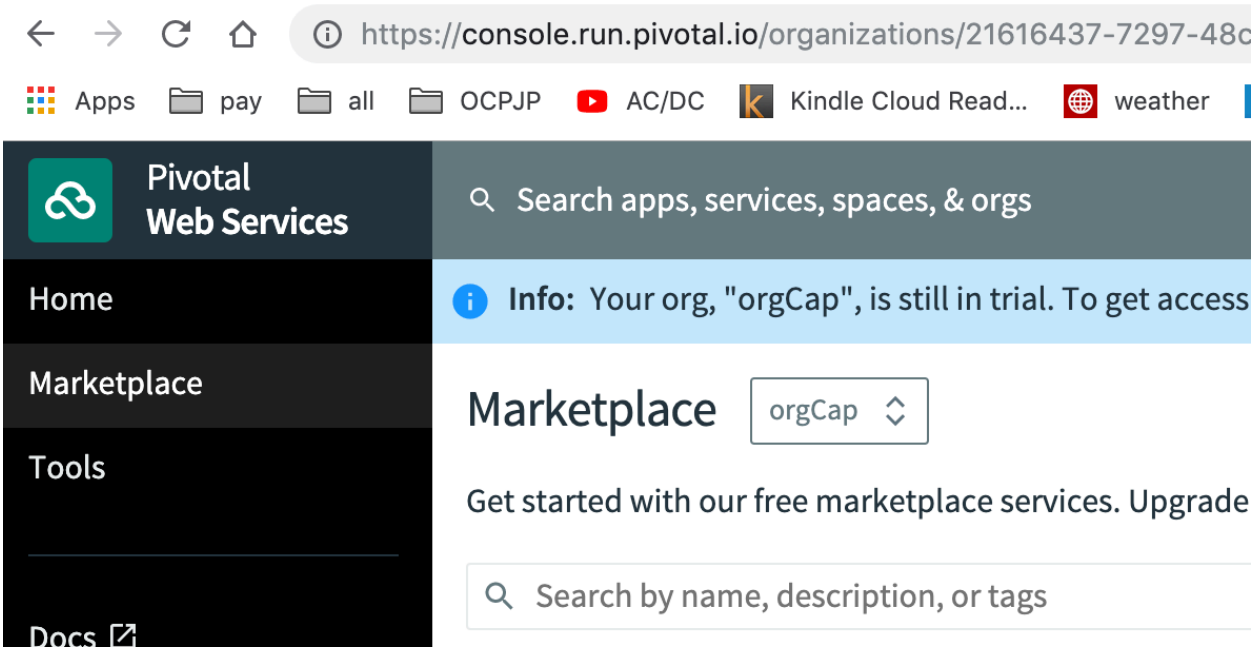


goto pivotal web services

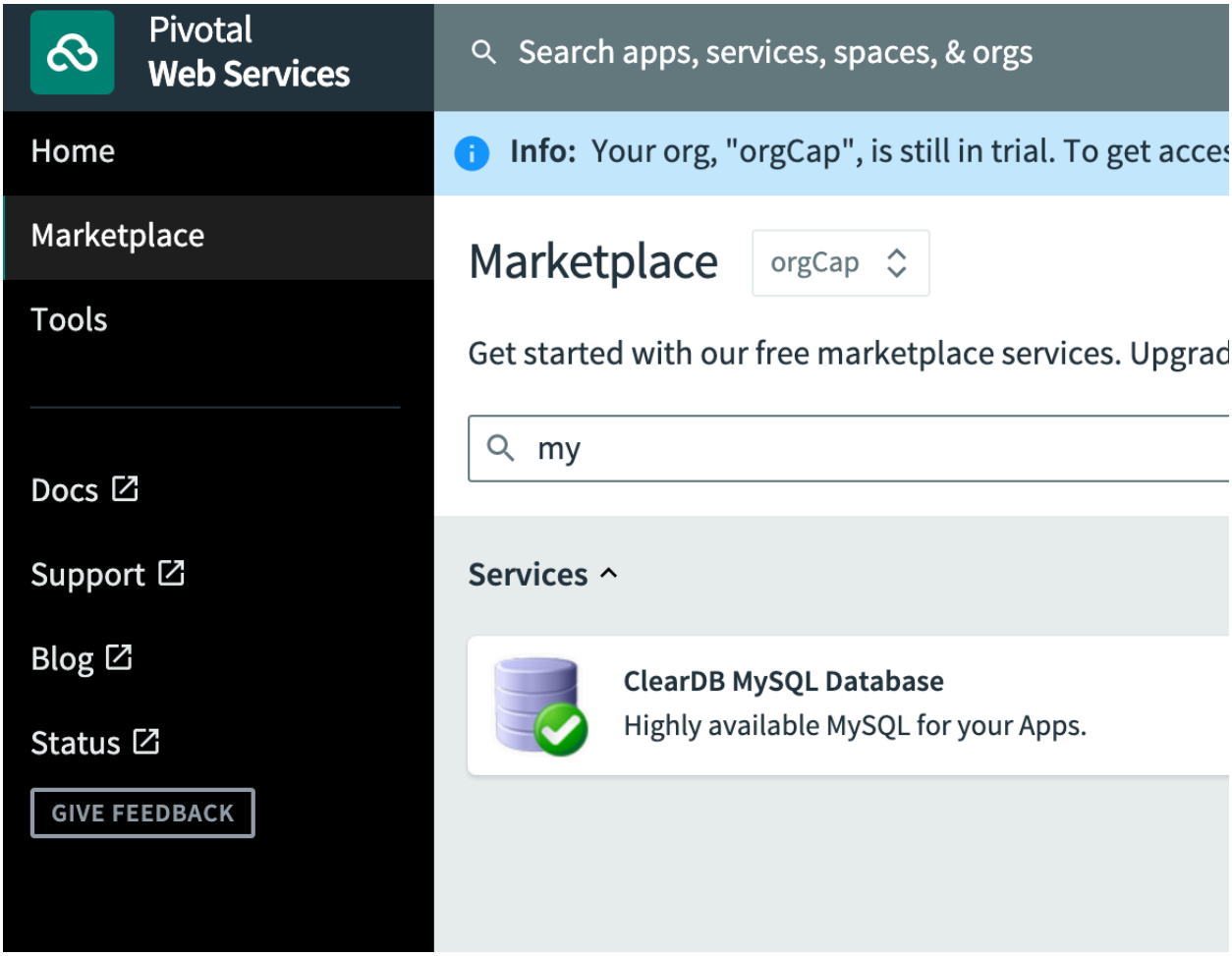


2.Create organization and workspace

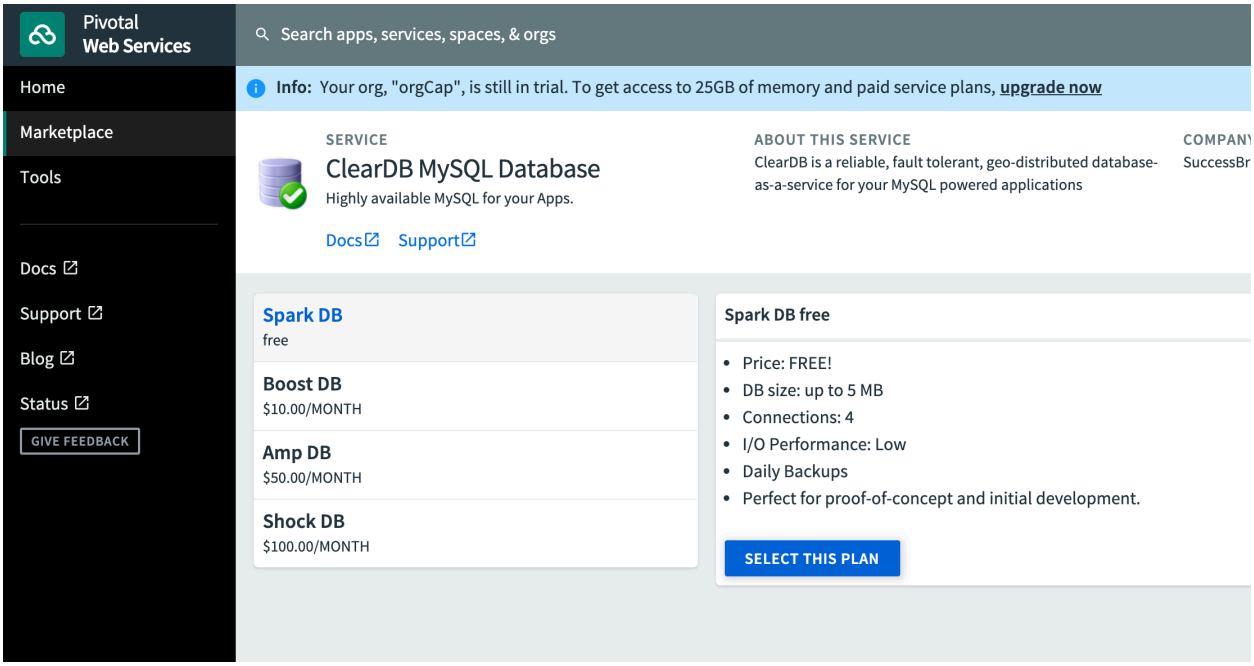
3.Create DB on the PCF
Here very interesting and difficult step, please be attention:
goto marketplace:



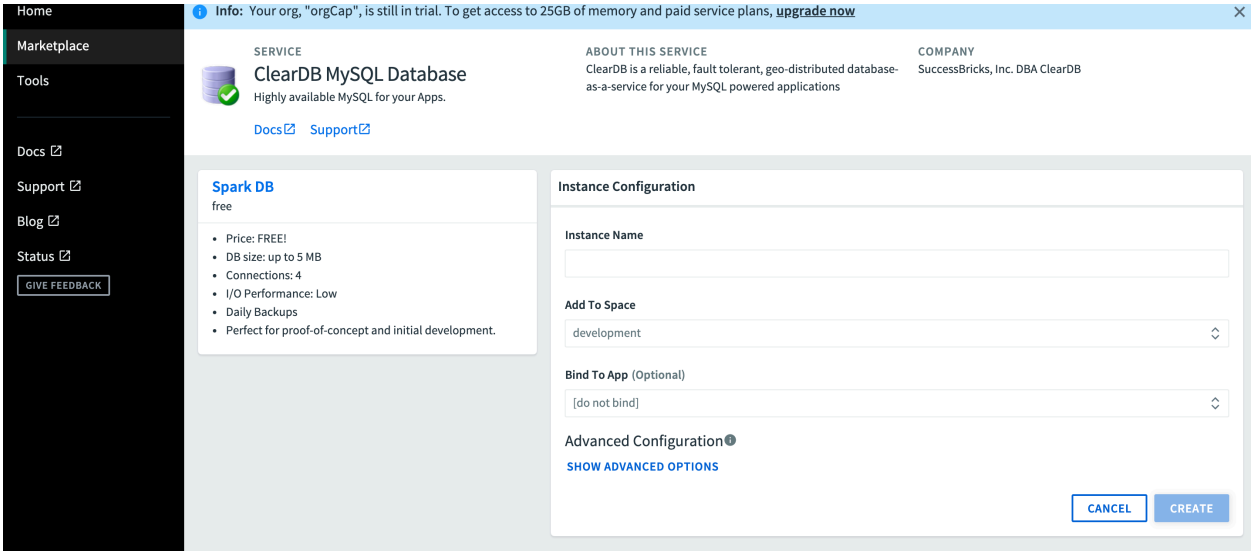
search mysql



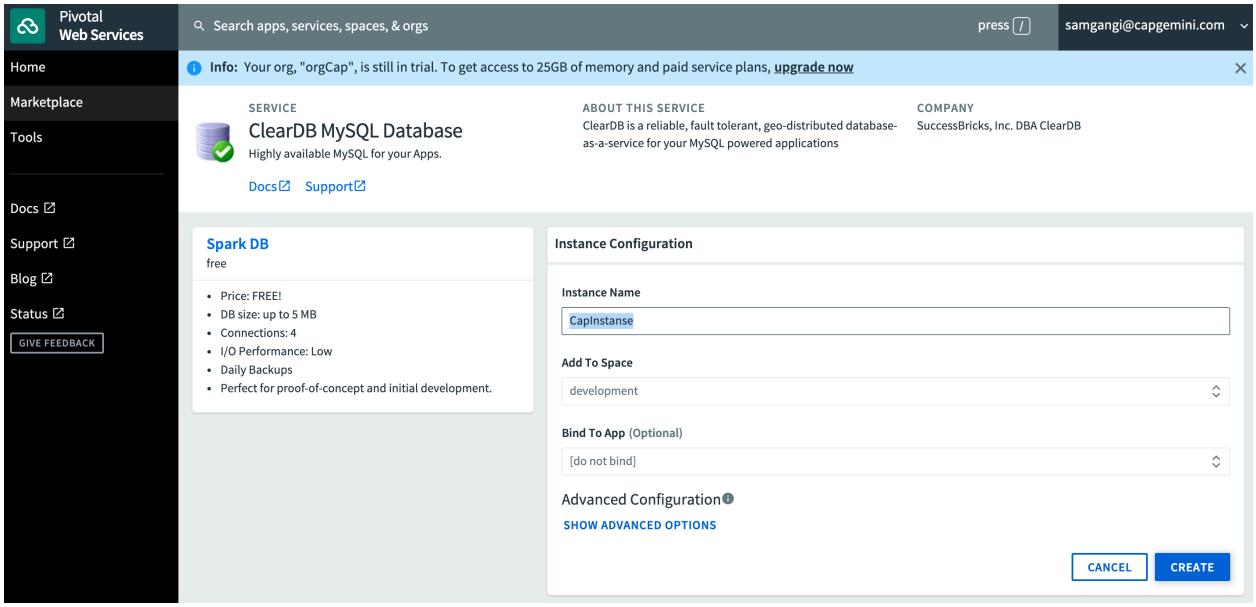
chose that DB



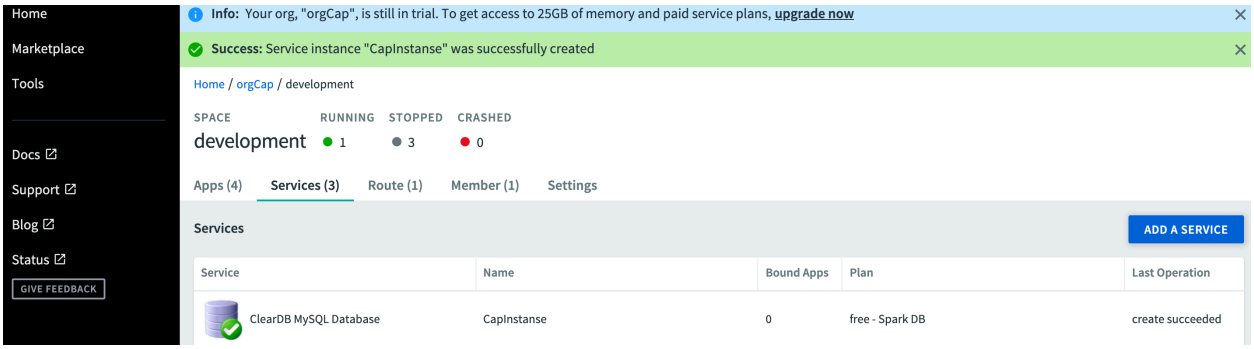
press select this plan (free)



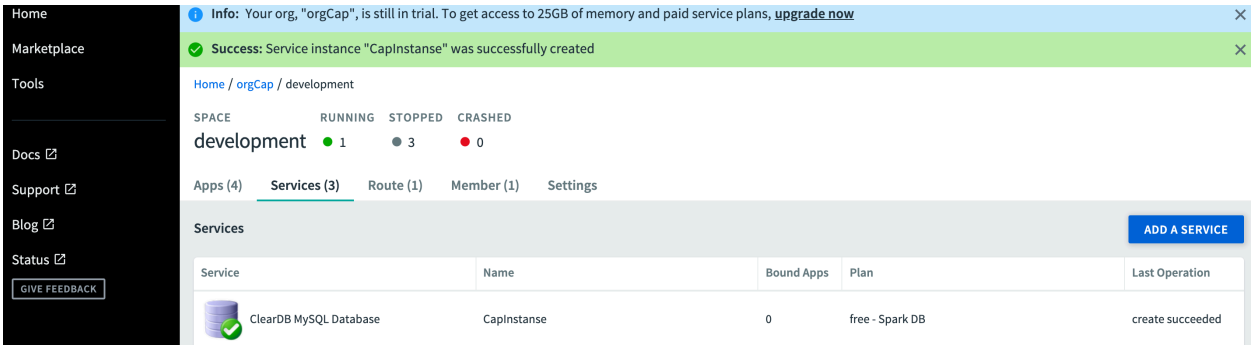
fill in one field
press create



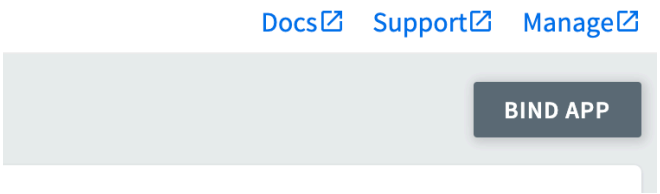
then you will create successful



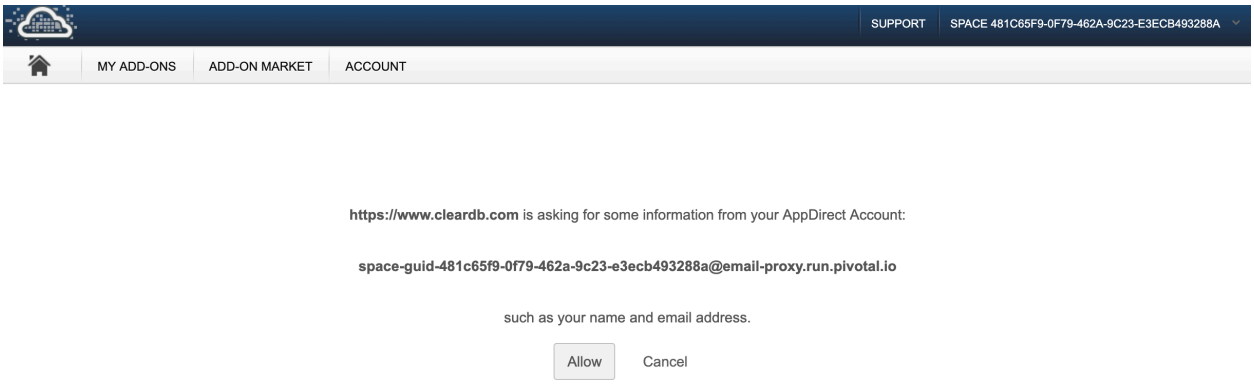
press link close the DB



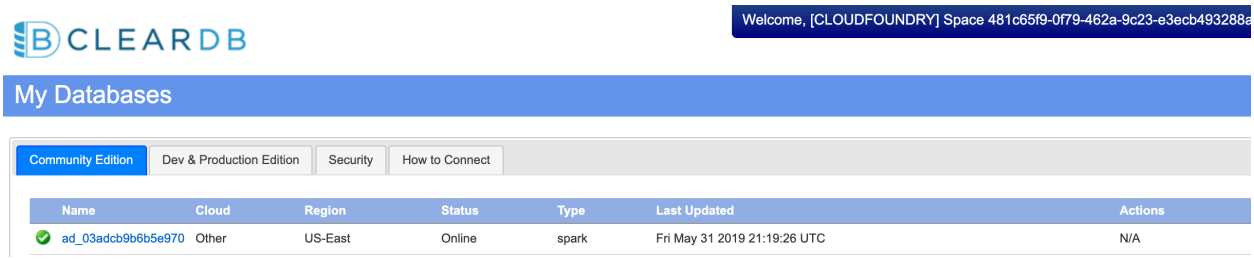
this window will be opened
goto manage in the right



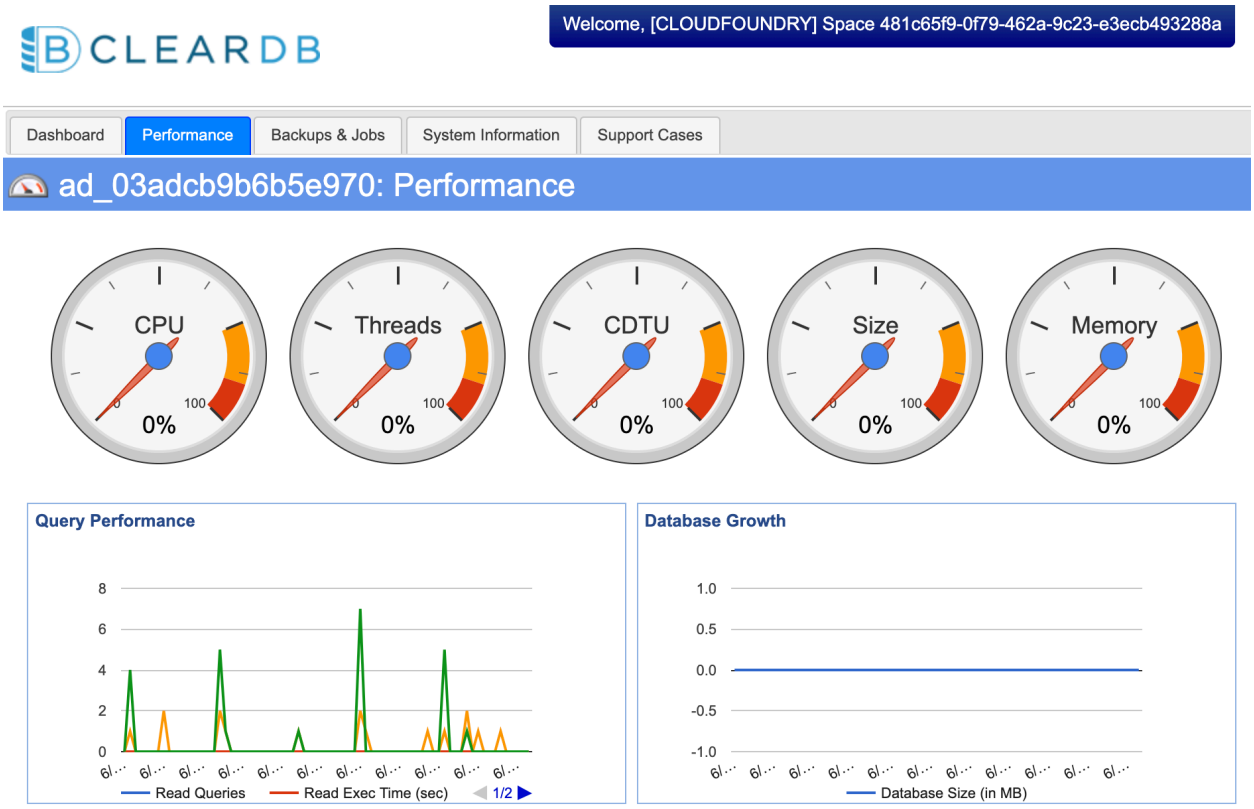
press allow



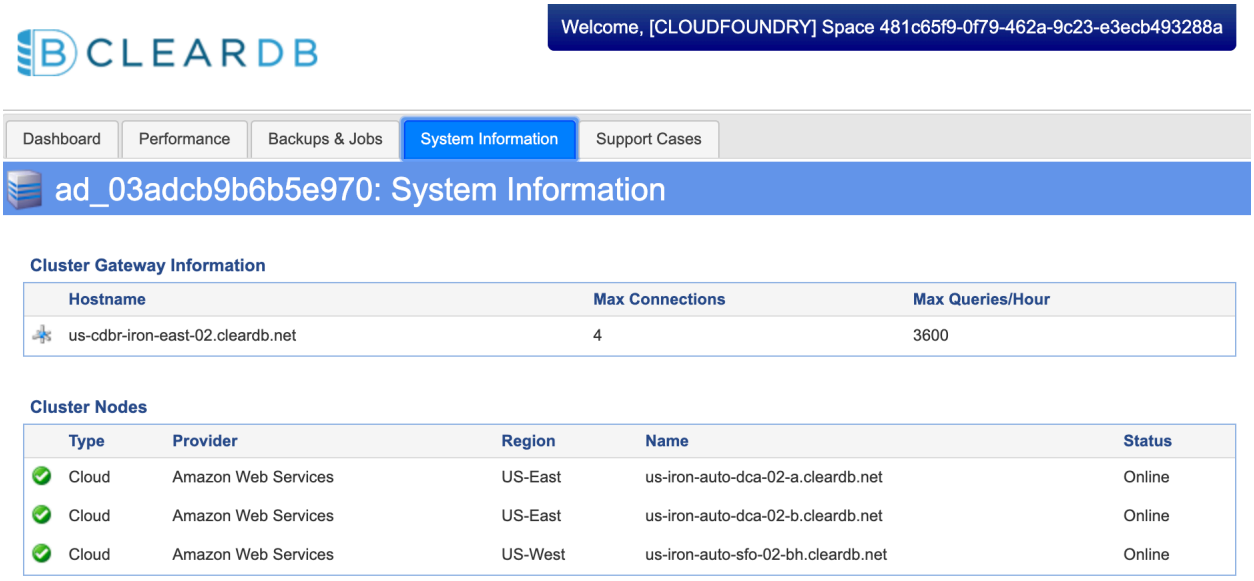
press link under the name



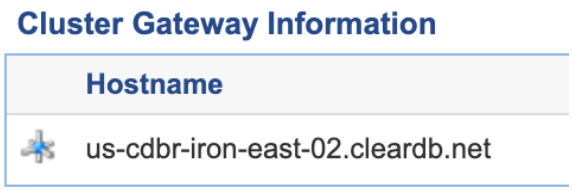
you will see this one

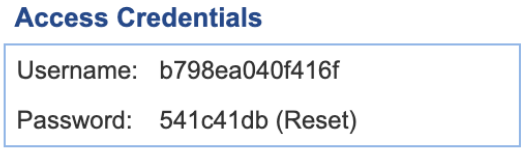


goto System information tab



and, bingo, we see connection parameters





4.Receive DB connection string
On the previous step we can see connection parameters for DB

5.Add properties to the project
!working with many environments: like qa, dev, prod, prepared
for the adding special environment to the project we have to:

in the spring application create application-<env>.properties

application-test.properties

add JVM parameter

```
-Dspring.profiles.active=test
```

6.Rebuild project for PCF environment

goto folder with project
run ./gradlew build

```
sergeys-mbp:token Sergey$ ./gradlew build
Starting a Gradle Daemon, 13 busy and 5 incompatible Daemons could not be reused, use --status for details

BUILD SUCCESSFUL in 9s
5 actionable tasks: 5 executed
sergeys-mbp:token Sergey$
```

than goto build/libs folder


```
sergeys-mbp:token Sergey$ ./gradlew build
Starting a Gradle Daemon, 13 busy and 5 incompatible Daemons could not be reused, use --status for details

BUILD SUCCESSFUL in 9s
5 actionable tasks: 5 executed
sergeys-mbp:token Sergey$
```


there you will see fat jar, which ready to the deploy to the PCF
also you have to put manifest file to same folder with jar
manifest file have to include:

```
---
applications:
- name: capgemini-0.0.1
  memory: 768M
  instances: 1
  random-route: true
  path: capgemini-0.0.1.jar
  buildpacks:
  - https://github.com/cloudfoundry/java-buildpack
```

7.Deploy project onto PCF

always you can see tutorial
https://pivotal.io/platform/pcf-tutorials/getting-started-with-pivotal-cloud-foundry/deploy-the-sample-app

for the deployment to the PCF we need 2 files: jar and manifest

goto folder with those to files:

cd …

```
sergeys-mbp:deploy Sergey$ cf login -a https://api.run.pivotal.io
API endpoint: https://api.run.pivotal.io

Email> samgangi@capgemini.com

Password>
Authenticating...
OK

Targeted org orgCap

Targeted space development


API endpoint:   https://api.run.pivotal.io (API version: 2.136.0)
User:           samgangi@capgemini.com
Org:            orgCap
Space:          development
sergeys-mbp:deploy Sergey$ push
-bash: push: command not found
sergeys-mbp:deploy Sergey$ cf push
```

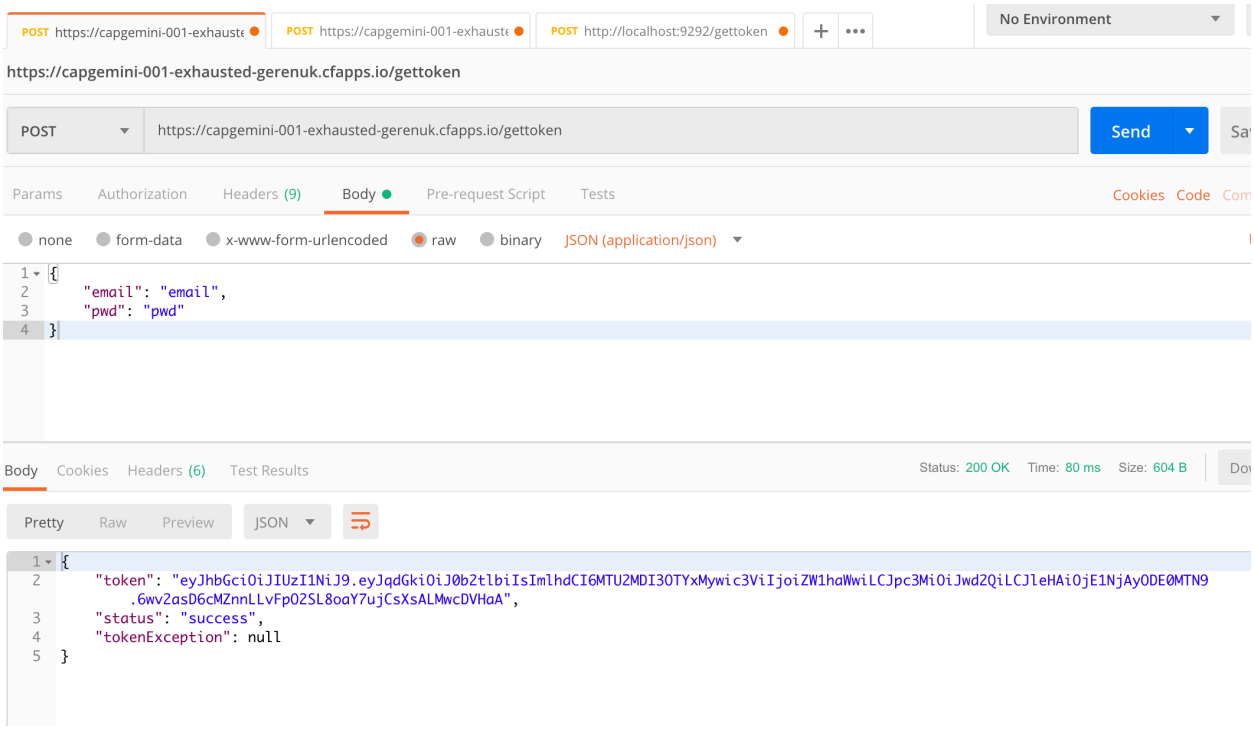
run commands from tutorial

```
sergeys-mbp:deploy Sergey$ cf login -a https://api.run.pivotal.io
Email> samgangi@capgemini.com
Password>
cf push
```

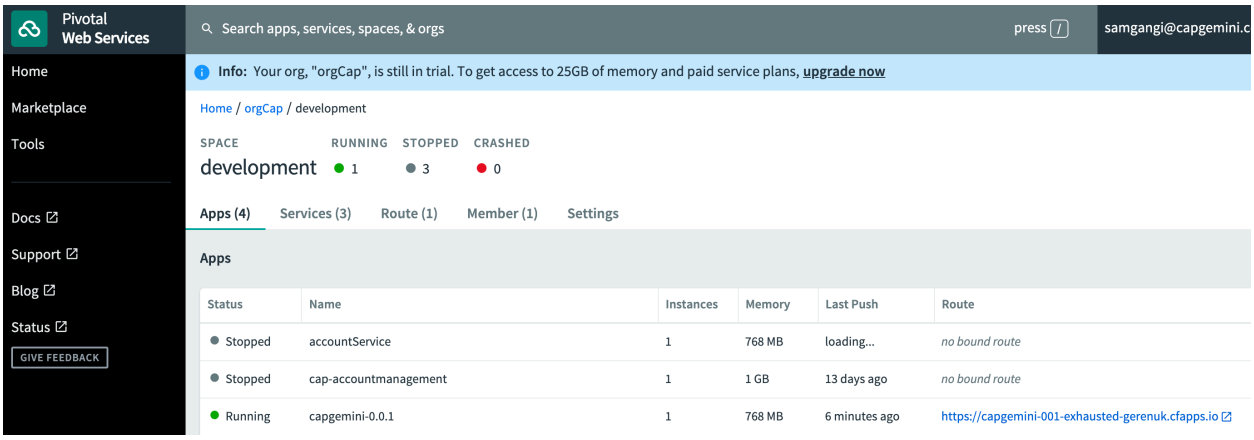
after successful deployment you will see:

```
  routes:
    capgemini-001-exhausted-gerenuk.cfapps.io

Updating app capgemini-0.0.1...
Mapping routes...
Comparing local files to remote cache...
Packaging files to upload...
Uploading files...
 314.40 KiB / 314.40 KiB [==========================================================================================

Waiting for API to complete processing files...

Stopping app...

Staging app and tracing logs...
   Cell 8875efb3-ebd9-42dc-adfa-3978c9e35594 creating container for instance 00c6db97-b7c2-49bf-a2fc-79ef24172bd0
   Cell 8875efb3-ebd9-42dc-adfa-3978c9e35594 successfully created container for instance 00c6db97-b7c2-49bf-a2fc-79
   Downloading app package...
   Downloading build artifacts cache...
   Downloaded app package (14.5M)
   Downloading build artifacts cache failed
   -----> Java Buildpack 542d66c | https://github.com/cloudfoundry/java-buildpack#542d66c
   -----> Downloading Jvmkill Agent 1.16.0_RELEASE from https://java-buildpack.cloudfoundry.org/jvmkill/bionic/x86_
   -----> Downloading Open Jdk JRE 1.8.0_212 from https://java-buildpack.cloudfoundry.org/openjdk/bionic/x86_64/oper
          Expanding Open Jdk JRE to .java-buildpack/open_jdk_jre (1.0s)
          JVM DNS caching disabled in lieu of BOSH DNS caching
   -----> Downloading Open JDK Like Memory Calculator 3.13.0_RELEASE from https://java-buildpack.cloudfoundry.org/m
          Loaded Classes: 11801, Threads: 250
   -----> Downloading Client Certificate Mapper 1.8.0_RELEASE from https://java-buildpack.cloudfoundry.org/client-c
   -----> Downloading Container Security Provider 1.16.0_RELEASE from https://java-buildpack.cloudfoundry.org/conta
   -----> Downloading Spring Auto Reconfiguration 2.7.0_RELEASE from https://java-buildpack.cloudfoundry.org/auto-r
   Exit status 0
   Uploading droplet, build artifacts cache...
   Uploading droplet...
   Uploading build artifacts cache...
   Uploaded build artifacts cache (43.3M)
   Uploaded droplet (58M)
   Uploading complete
   Cell 8875efb3-ebd9-42dc-adfa-3978c9e35594 stopping instance 00c6db97-b7c2-49bf-a2fc-79ef24172bd0
   Cell 8875efb3-ebd9-42dc-adfa-3978c9e35594 destroying container for instance 00c6db97-b7c2-49bf-a2fc-79ef24172bd0

Waiting for app to start...

name:              capgemini-0.0.1
requested state:   started
routes:            capgemini-001-exhausted-gerenuk.cfapps.io
last uploaded:     Tue 11 Jun 13:56:29 CDT 2019
stack:             cflinuxfs3
buildpacks:        https://github.com/cloudfoundry/java-buildpack

type:            web
instances:       1/1
memory usage:    768M
start command:   JAVA_OPTS="-agentpath:$PWD/.java-buildpack/open_jdk_jre/bin/jvmkill-1.16.0_RELEASE=printHeapHistog
                 -Djava.ext.dirs=$PWD/.java-buildpack/container_security_provider:$PWD/.java-buildpack/open_jdk_jre/
                 $JAVA_OPTS" && CALCULATED_MEMORY=$($PWD/.java-buildpack/open_jdk_jre/bin/java-buildpack-memory-calc
                 -stackThreads=250 -vmOptions="$JAVA_OPTS") && echo JVM Memory Configuration: $CALCULATED_MEMORY &&
                 exec $PWD/.java-buildpack/open_jdk_jre/bin/java $JAVA_OPTS -cp $PWD/. org.springframework.boot.loa
     state     since                  cpu    memory           disk           details
#0   running   2019-06-11T18:56:47Z   0.4%   145.3M of 768M   126.5M of 1G

sergeys-mbp:deploy Sergey$
```

after deploy we can check our application in the postman:



and PCF


Pivotal
Web Services
Search apps, services, spaces, & orgs
press /
samgangi@capgemini.c
Home
Marketplace
Tools
Docs
Support
Blog
Status
GIVE FEEDBACK
Info: Your org, "orgCap", is still in trial. To get access to 25GB of memory and paid service plans, upgrade now
Home / orgCap / development
SPACE
development
RUNNING
1
STOPPED
3
CRASHED
0
Apps (4)
Services (3)
Route (1)
Member (1)
Settings
Apps
Status
Name
Instances
Memory
Last Push
Route
Stopped
accountService
1
768 MB
loading...
no bound route
Stopped
cap-accountmanagement
1
1 GB
13 days ago
no bound route
Running
capgemini-0.0.1
1
768 MB
6 minutes ago
https://capgemini-001-exhausted-gerenuk.cfapps.io

Task 4: install Jenkins, make automatization deployment with Jenkins

1. Install Jenkins
2. Run Jenkins
3. Add tasks: clone from GitHub, change privileges, copy files to another place, deploy to the PCF

Download Jenkins war:
https://jenkins.io/download/

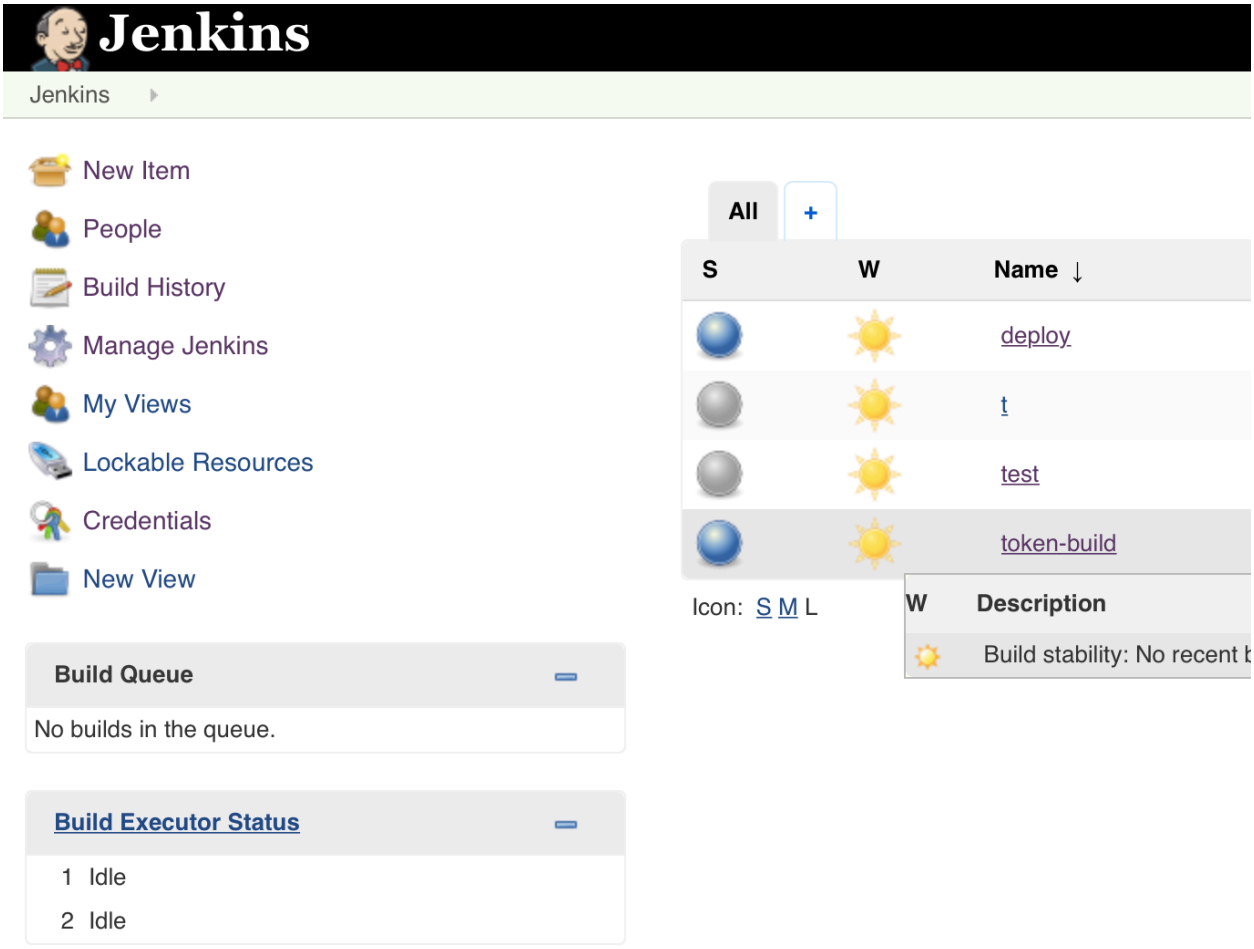
run Jenkins with
```
java -jar jenkins.war
```

goto:
http://localhost:8080/

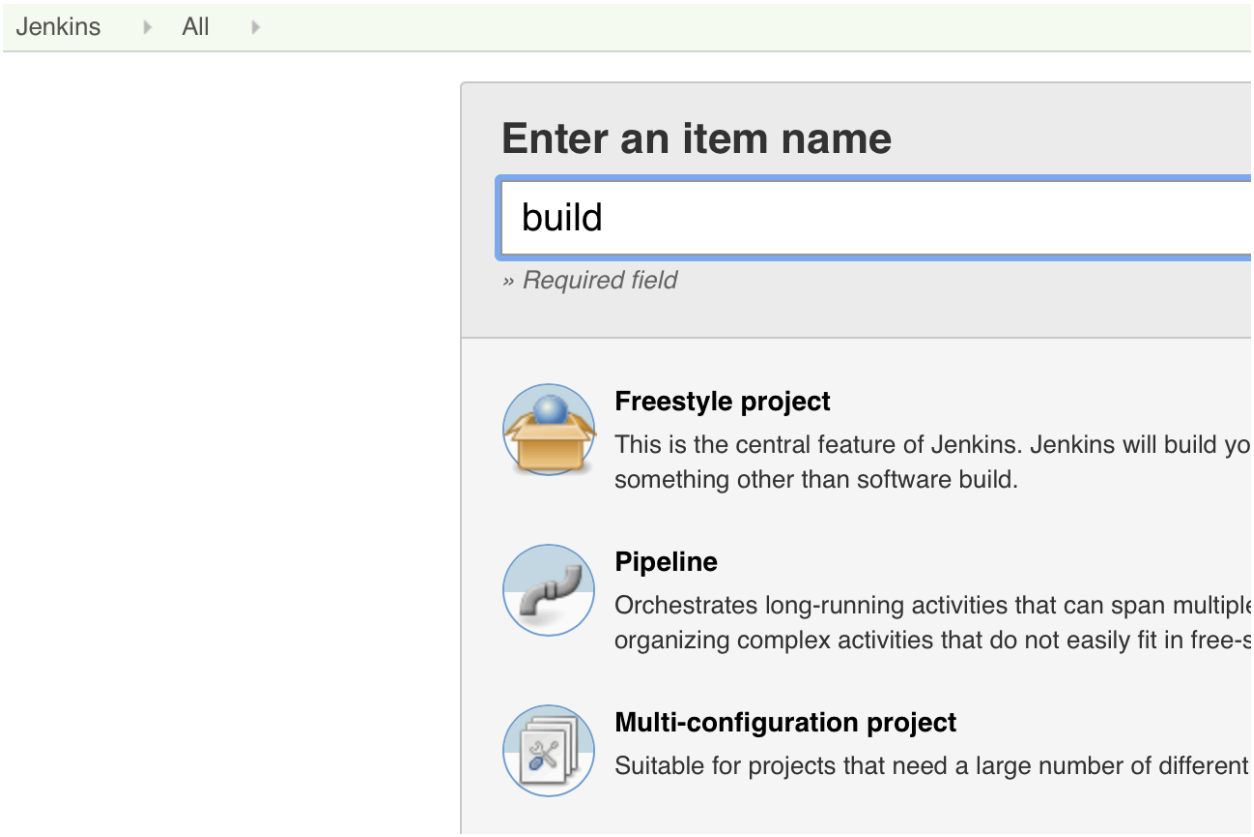
To unlock Jenkins, copy the password from the file at **C:\Program Files (x86)\Jenkins\secrets\initialAdminPassword** and paste it in the **Administratorpassword** field.

then you have to add our tasks to the Jenkins:
task1: clone from GitHub

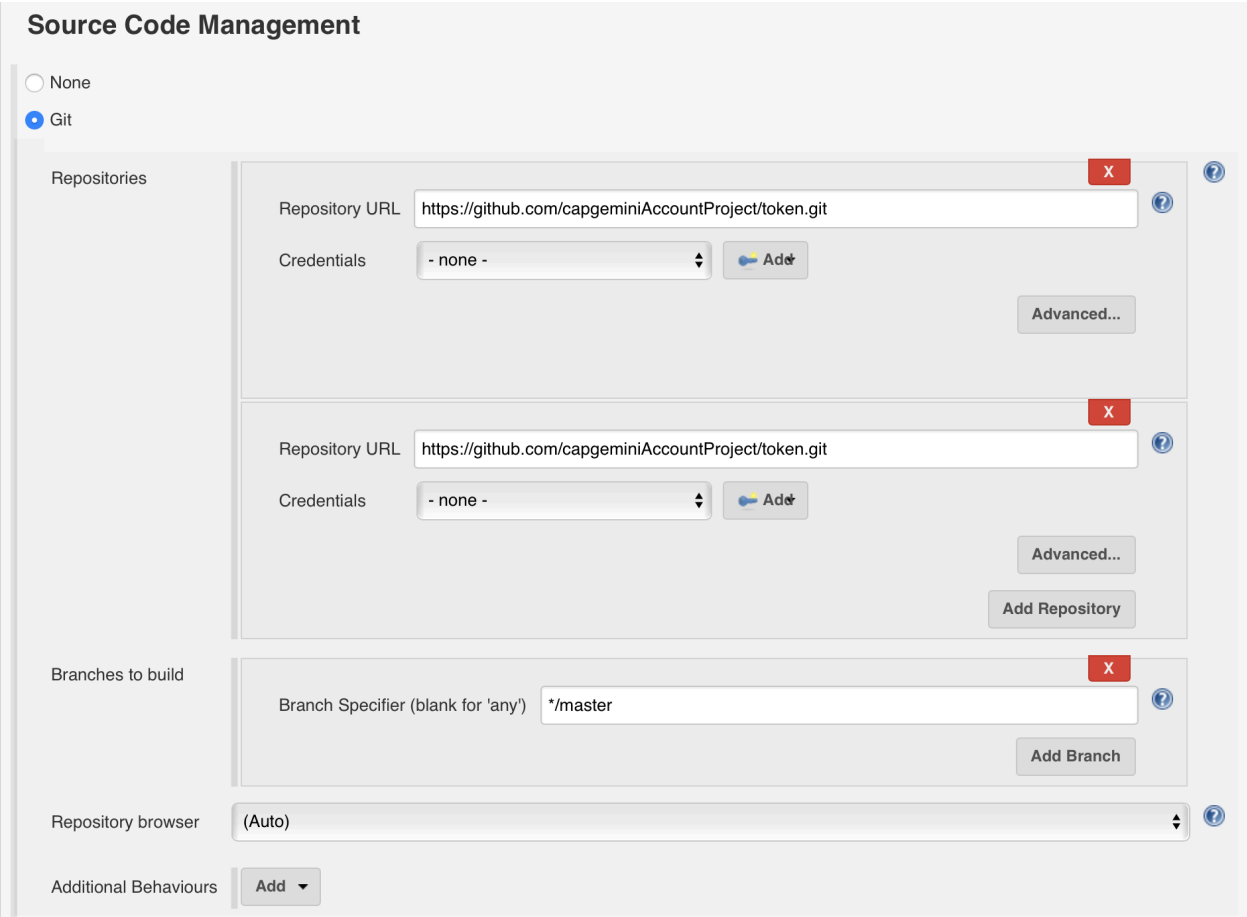
please press new item:



create freestyle project with name: build

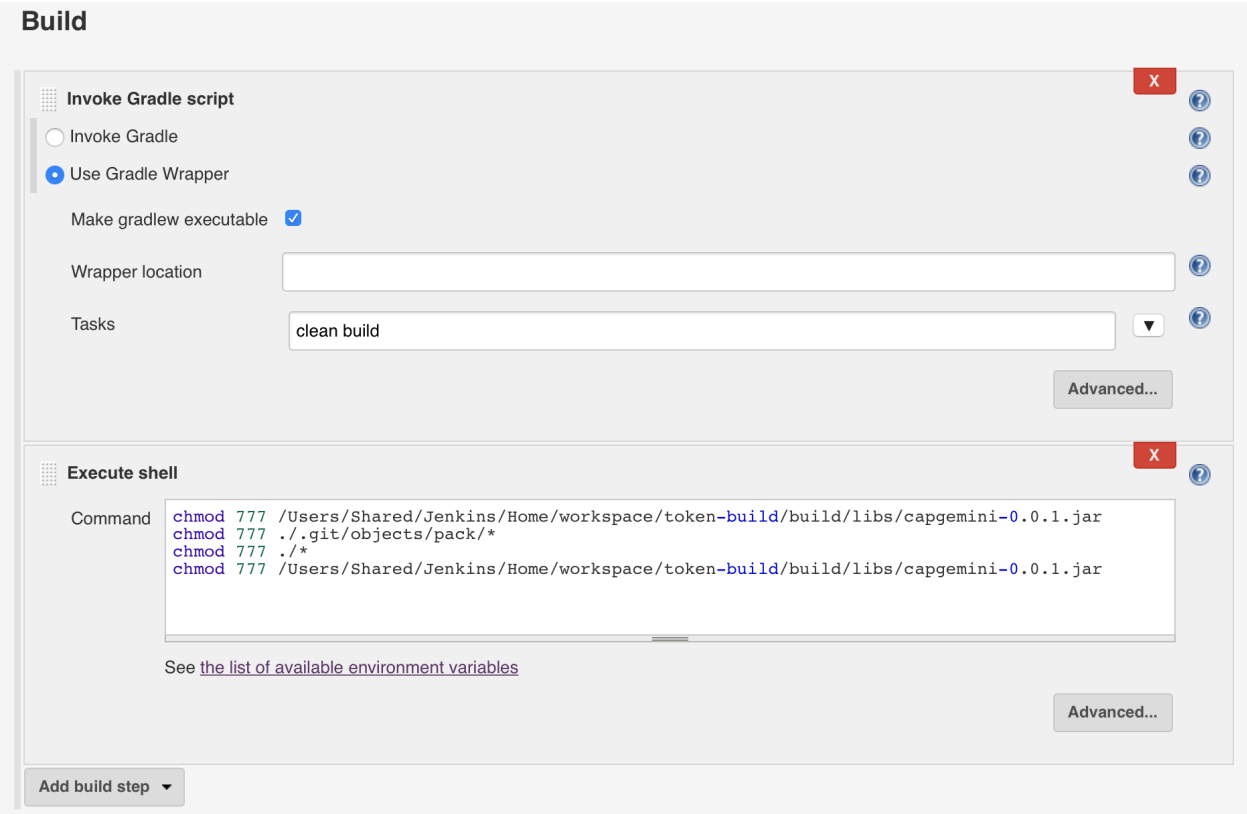


on the sours code management tab, please fill out



here you do not need add user and password to the github, because for the clone free project password is not required.

next section is build



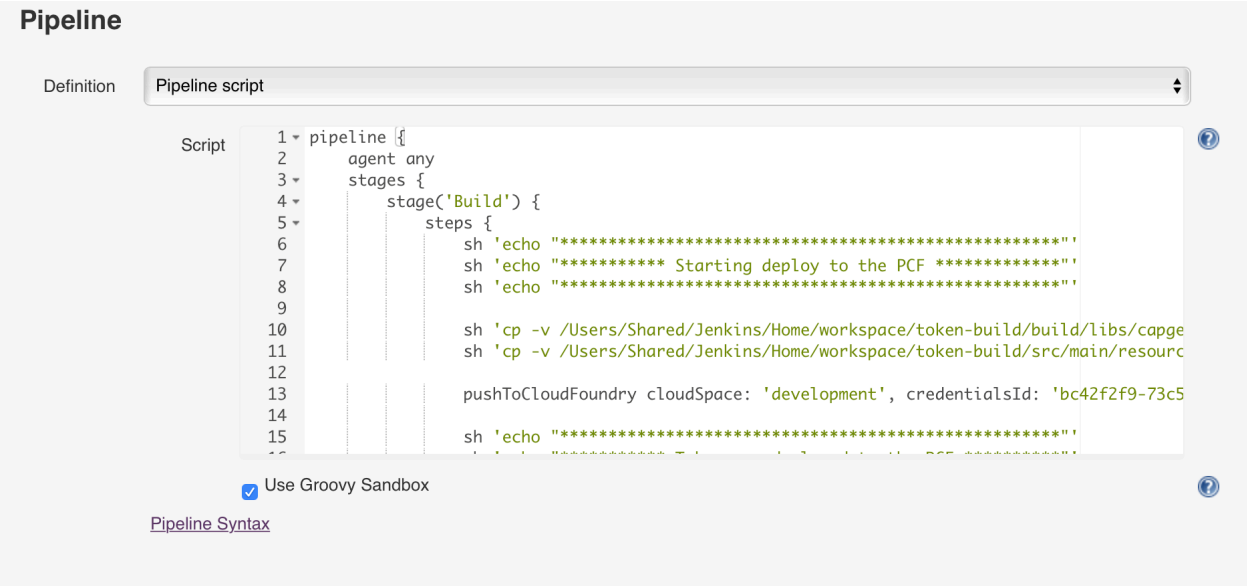
here we have to chose: gradle, please be accurate and make all like on the picture

then, we are changing privileges to the project files, in this case we added shell command

Result of this step: two files (jar and manifest) copied to one folder, and privileges 777 set up to those files.

```
sergeys-mbp:libs Sergey$ ls -la
total 33320
drwxr-xr-x  3 jenkins  jenkins        96 Jun 11 13:39 .
drwxrwxrwx  9 jenkins  jenkins       288 Jun 11 13:39 ..
-rwxrwxrwx  1 jenkins  jenkins  17058738 Jun 11 13:39 capgemini-0.0.1.jar
sergeys-mbp:libs Sergey$
```

create deploy to the PCF task:
goto: jenkins - new item - pipeline
next our task is deployment to the PCF with Jenkins



here we have to add this script, below i will explain what we are doing.

Whole script:

```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                sh 'echo
"************************************************"'
                sh 'echo "********** Starting deploy to the PCF
************"'
                sh 'echo
"************************************************"'

                sh 'cp -v /Users/Shared/Jenkins/Home/workspace/
token-build/build/libs/capgemini-0.0.1.jar /Users/Shared/
Jenkins/Home/workspace/deploy/capgemini-0.0.1.jar'
                sh 'cp -v /Users/Shared/Jenkins/Home/workspace/
token-build/src/main/resources/manifest.yml /Users/Shared/
Jenkins/Home/workspace/deploy/manifest.yml'
```

```
                pushToCloudFoundry cloudSpace: 'development',
credentialsId: 'bc42f2f9-73c5-4c23-9e70-30e0461970d1',
organization: 'orgCap', selfSigned: 'true', target: 'https://
api.run.pivotal.io'

                sh 'echo
"**************************************************"'
                sh 'echo "********** Token was deployed to the
PCF **********"'
                sh 'echo
"**************************************************"'
            }
        }
    }
}
```

lest go thru:

this is only log, because we have to know what is happing:

```
sh 'echo "**************************************************"'
                sh 'echo "********** Starting deploy to the PCF
************"'
                sh 'echo
“**************************************************"'
```

next:

```
sh 'cp -v /Users/Shared/Jenkins/Home/workspace/token-build/
build/libs/capgemini-0.0.1.jar /Users/Shared/Jenkins/Home/
workspace/deploy/capgemini-0.0.1.jar'
                sh 'cp -v /Users/Shared/Jenkins/Home/workspace/
token-build/src/main/resources/manifest.yml /Users/Shared/
Jenkins/Home/workspace/deploy/manifest.yml'
```

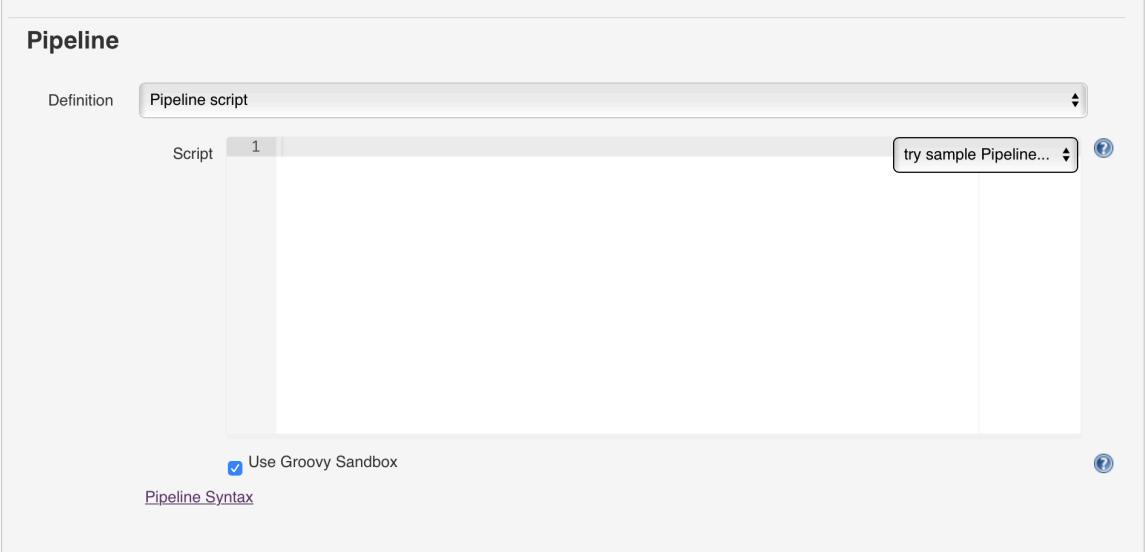
here we are copying to files to the one folder, because for the pushing jar to PCF we have to have jar and manifest together. (of course we can use whole PATH but it is extra headache).

```
pushToCloudFoundry cloudSpace: 'development', credentialsId:
'bc42f2f9-73c5-4c23-9e70-30e0461970d1', organization: 'orgCap',
selfSigned: 'true', target: 'https://api.run.pivotal.io'
```

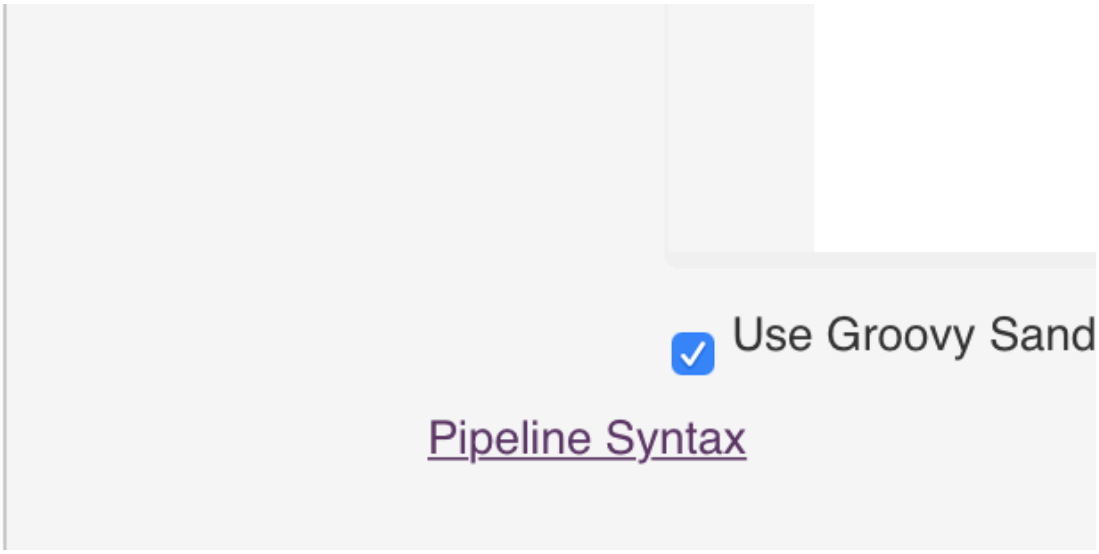
this is main part of our script:
here we are asking Jenkins to push jar.

for creating this script we are recommended to use helper:

create blank new item pipeline,

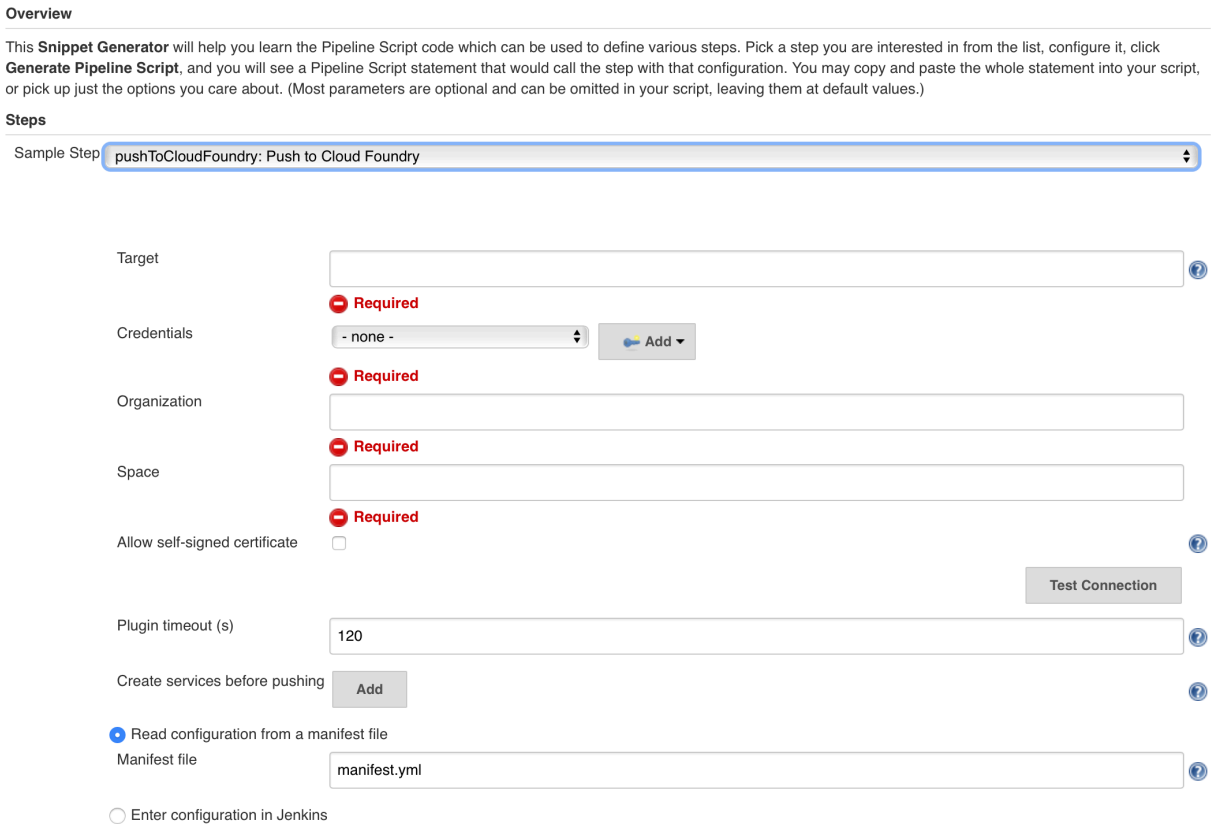


press Pipeline Syntax

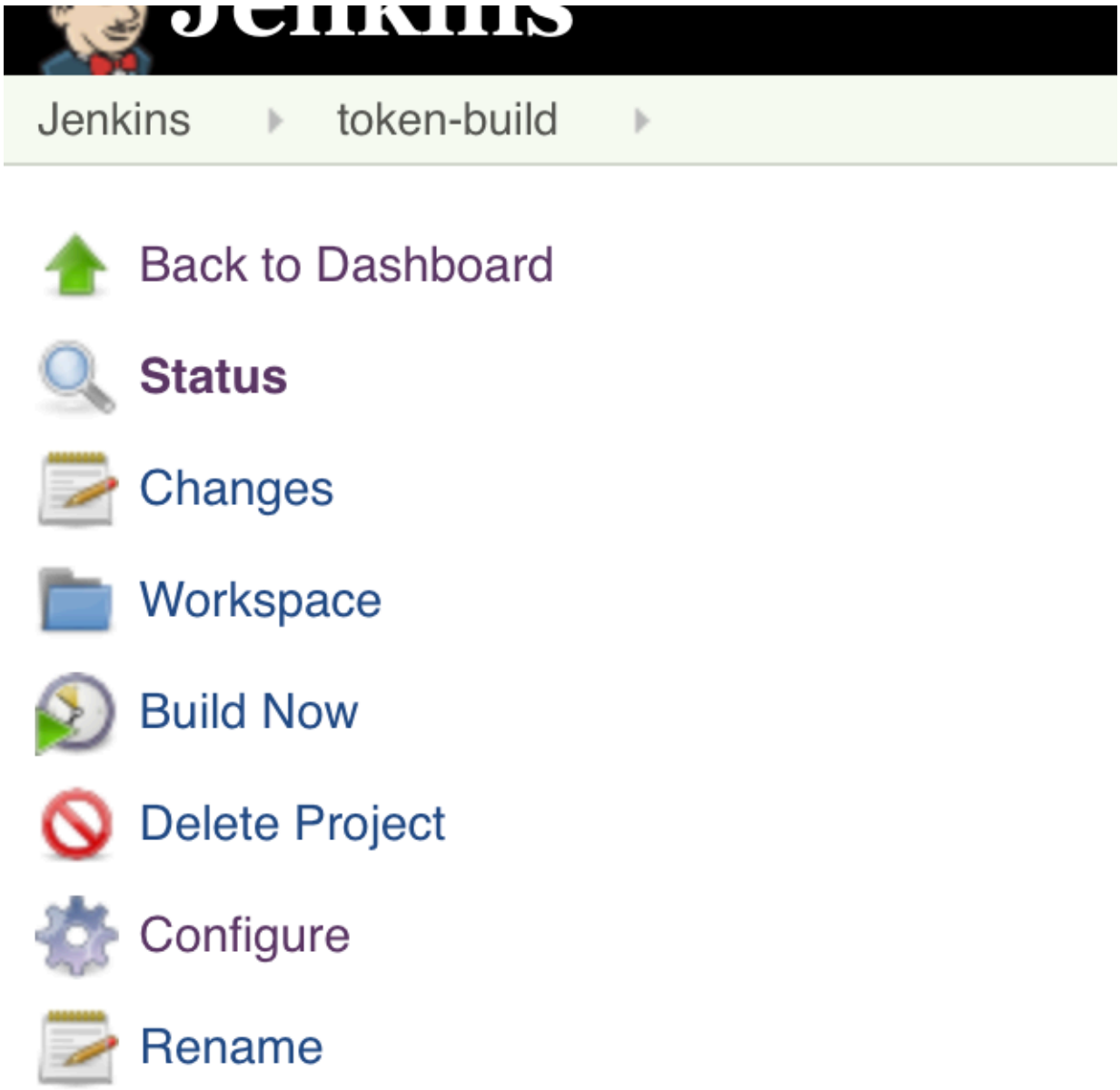


on the sample step we can chose:

fill in fields, and system will generate script for us.



then we need to ram our tasks



build run, and enjoy application on the cloud