

Tutorial

| | |
|--|----|
| User Story (main task): | 2 |
| Tasks: | 2 |
| Requirement: | 2 |
| Expected result: | 3 |
| Implementation: | 4 |
| Task 0: installation | 4 |
| Task 1: develop microservices | 6 |
| Task 2: Code have to be pushed to GitHub | 8 |
| Task 3: deploy onto PCF | 19 |
| Task 4: install Jenkins, make automatization deployment with Jenkins | 38 |

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>

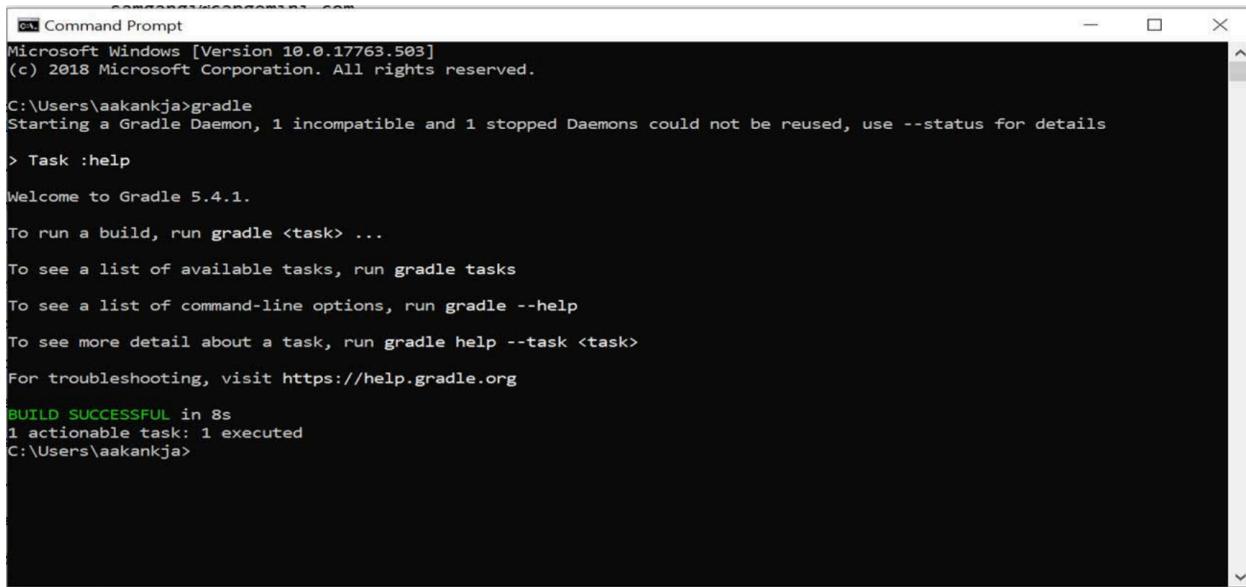
after installation you have to check on the console (terminal)
command: git --version

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/>



```
Command Prompt
Microsoft Windows [Version 10.0.17763.503]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\akankja>gradle
Starting a Gradle Daemon, 1 incompatible and 1 stopped Daemons could not be reused, use --status for details
> Task :help

Welcome to Gradle 5.4.1.

To run a build, run gradle <task> ...

To see a list of available tasks, run gradle tasks

To see a list of command-line options, run gradle --help

To see more detail about a task, run gradle help --task <task>

For troubleshooting, visit https://help.gradle.org

BUILD SUCCESSFUL in 8s
1 actionable task: 1 executed
C:\Users\akankja>
```

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.

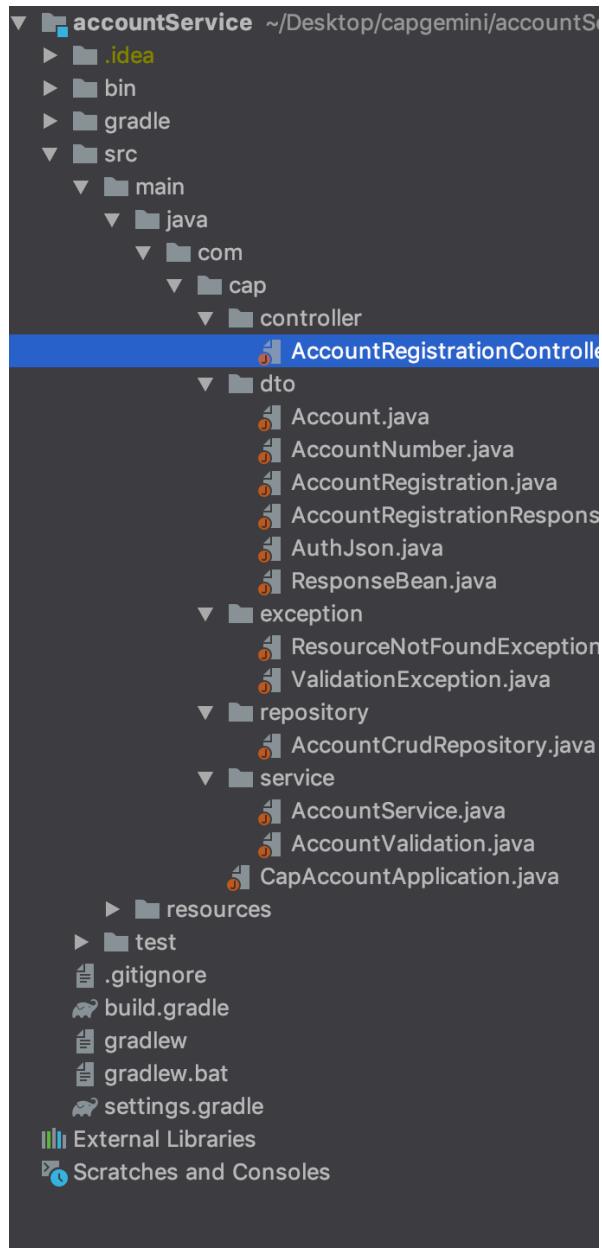
Microservice always have to return correct response, newer return exception.

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:



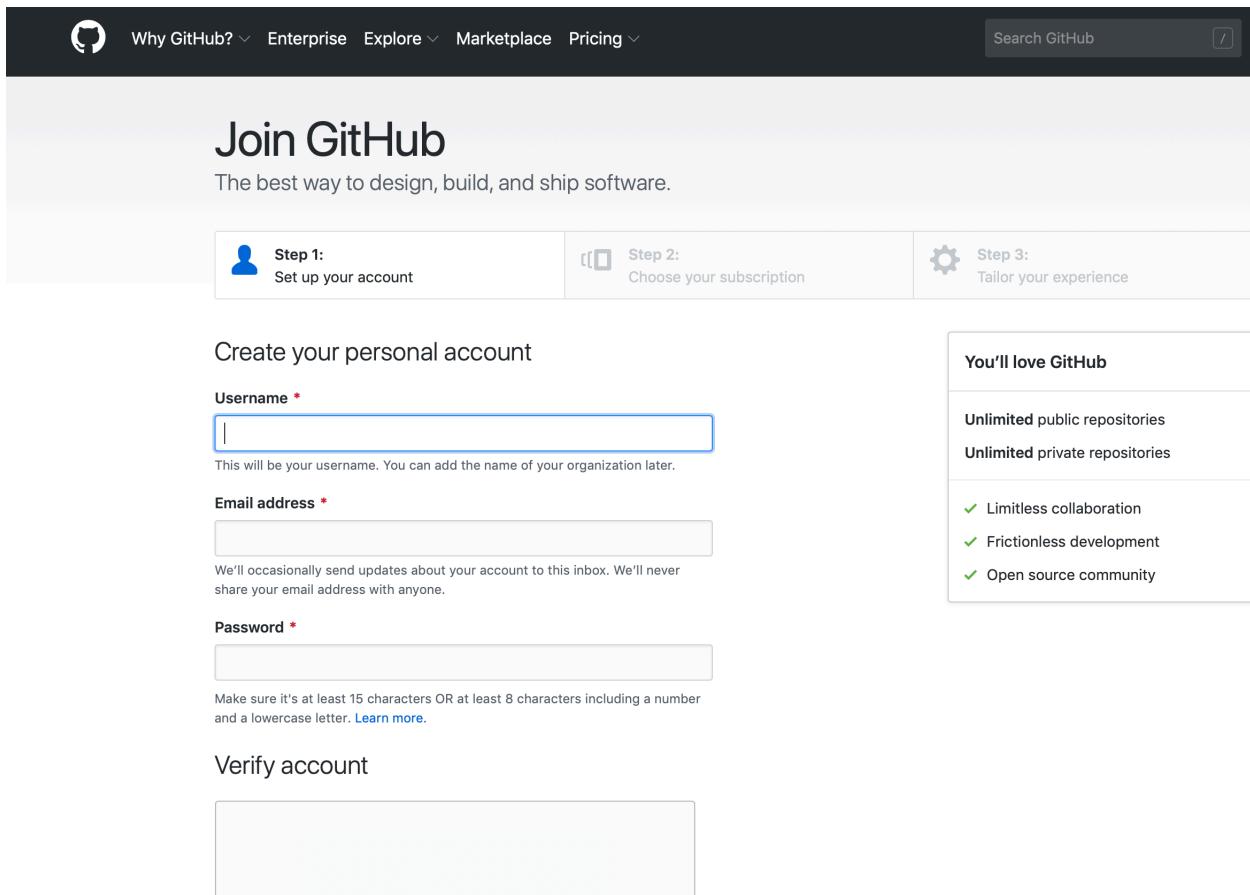
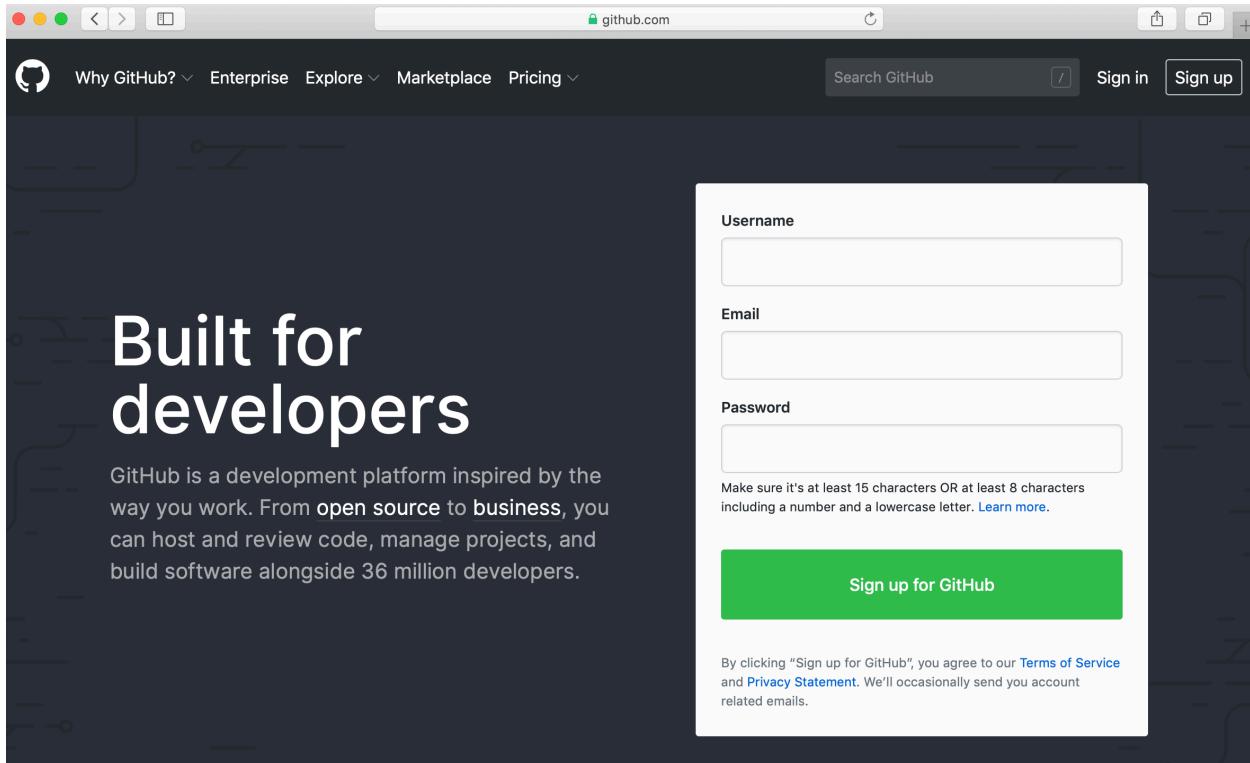
Task 2: Code have to be pushed to GitHub

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

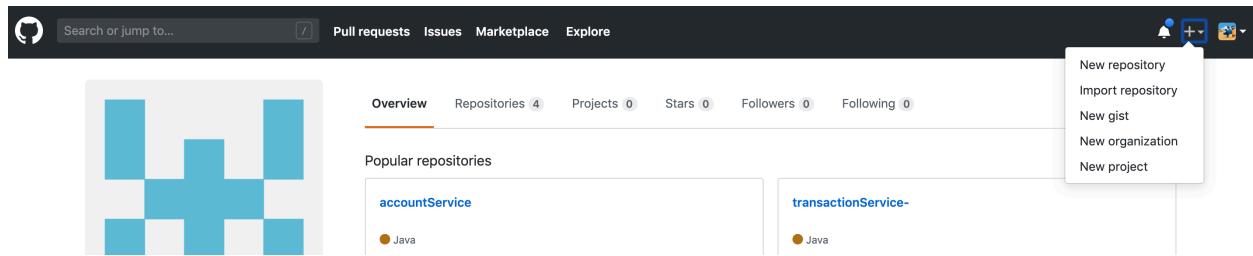
those steps will be described in depth

- 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 

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

The screenshot shows a GitHub repository page. At the top, the repository name is 'svolchenkov / testRepository'. To the right are buttons for 'Watch' (0), 'Star' (0), and 'Fork' (0). Below the header is a navigation bar with tabs: 'Code' (selected), 'Issues 0', 'Pull requests 0', 'Projects 0', 'Wiki', 'Security', 'Insights', and 'Settings'. A message 'No description, website, or topics provided.' is displayed, with an 'Edit' button to its right. Below this is a 'Manage topics' section. Key statistics are shown: '1 commit', '1 branch', '0 releases', and '1 contributor'. Below the stats are buttons for 'Branch: master ▾', 'New pull request', 'Create new file', 'Upload files', 'Find File', and a green 'Clone or download ▾' button. A list of files shows 'svolchenkov Initial commit' and 'README.md'. The 'README.md' file is expanded, showing the text 'testRepository'. There is a small edit icon at the end of the README content.

2.Clone repository

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

The screenshot shows a GitHub repository page for 'svolchenkov / testRepository'. At the top, there are buttons for Watch (0), Star (0), and Fork (0). Below that is a navigation bar with links for Code, Issues (0), Pull requests (0), Projects (0), Wiki, Security, Insights, and Settings. A message says 'No description, website, or topics provided.' with an 'Edit' button. Below the message are statistics: 1 commit, 1 branch, 0 releases, and 1 contributor. A dropdown menu for 'Branch: master' is open, showing 'New pull request'. On the right, there are buttons for Create new file, Upload files, Find File, and Clone or download (highlighted in green). A modal window titled 'Clone with HTTPS' shows the URL <https://github.com/svolchenkov/testRe> and options to Open in Desktop or Download ZIP.

step2: and copy link

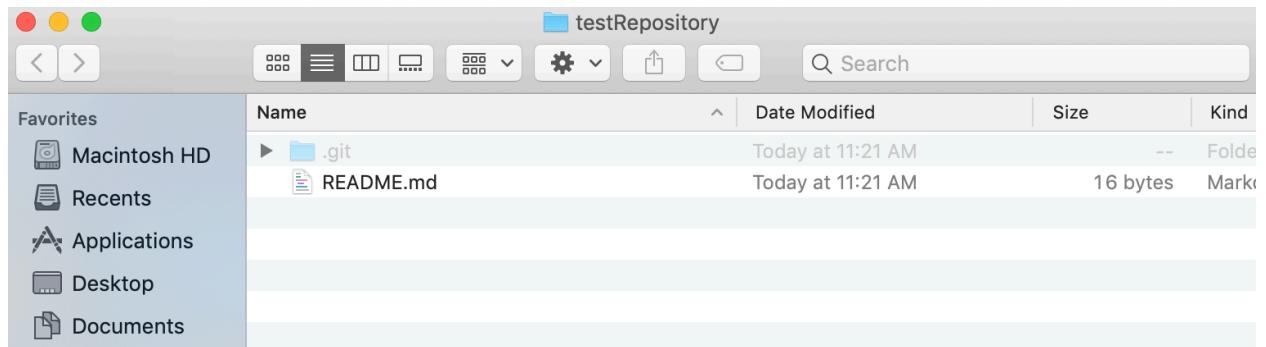
step3:

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.

The screenshot shows a GitHub repository page for 'svolchenkov / CapgeminiS'. The repository name is 'CapgeminiS / CapgeminiSergey /'. The branch is 'master'. The commit history shows a single commit from 'svolchenkov' with the message 'revert'. The commit was made on April 24 and is dated '2 months ago'. The commit details show the following files:

| File | Commit Type | Date |
|-----------------|-------------|--------------|
| .settings | first | 2 months ago |
| bin | revert | 2 months ago |
| gradle/wrapper | first | 2 months ago |
| src | revert | 2 months ago |
| .classpath | first | 2 months ago |
| .gitignore | first | 2 months ago |
| .project | first | 2 months ago |
| build.gradle | revert | 2 months ago |
| gradlew | first | 2 months ago |
| gradlew.bat | first | 2 months ago |
| settings.gradle | first | 2 months ago |

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: Create account on the PCF

goto: <https://pivotal.io/>

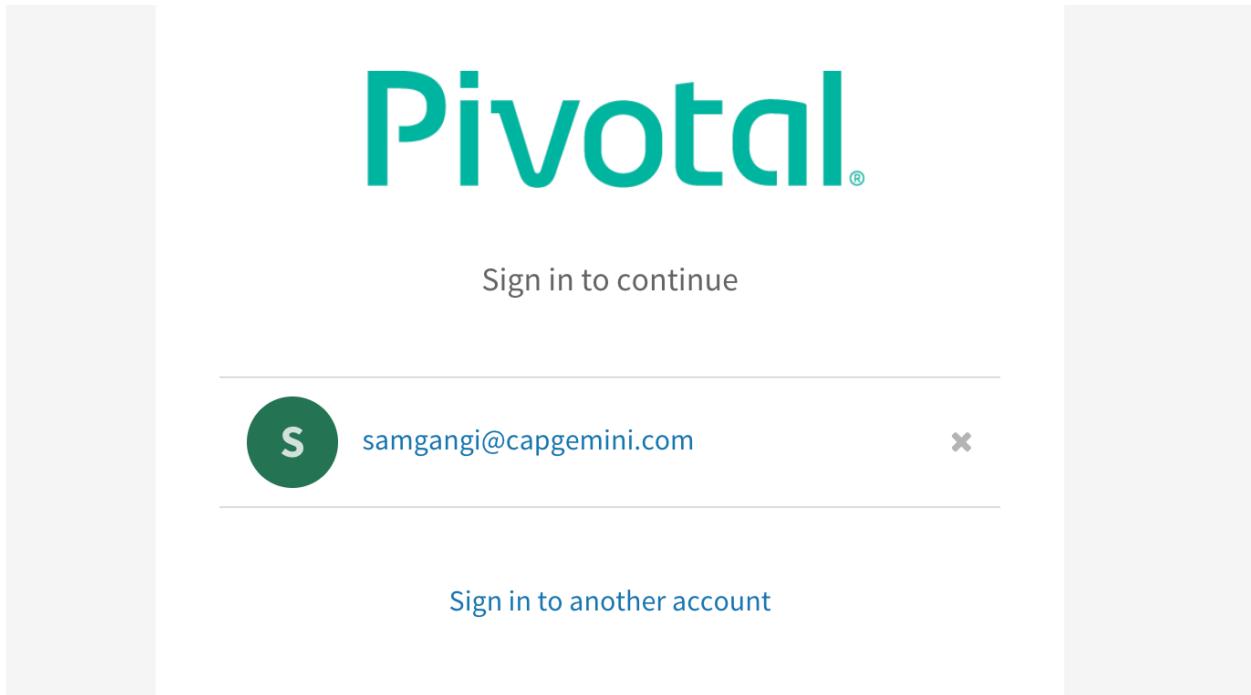
goto create account

fill out form

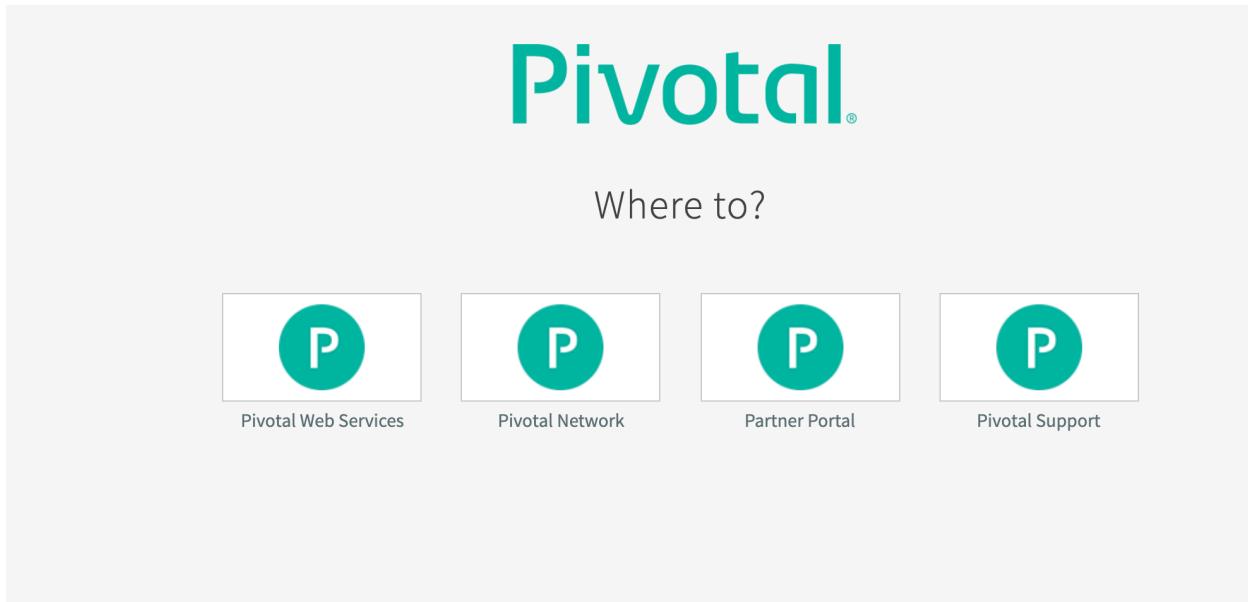
! Your account will be bind to your phone #

! 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

The screenshot shows the Pivotal Web Services dashboard. On the left is a dark sidebar with links: Home, Marketplace, Tools, Docs (with a link), and Support (with a link). The main area has a search bar at the top with placeholder text "Search apps, services, spaces, & orgs". Below the search bar is a "Recently Accessed Apps" section. A table lists two apps: "cap-accountmanagement" and "cappgemini-0.0.1". The table columns are Name, Org/Space, Status, Instances, and Last Push. "cap-accountmanagement" is listed under "orgCap / development" and is "Stopped" with 1 instance, last pushed 13 days ago. "cappgemini-0.0.1" is listed under "orgCap / development" and is "Running" with 1 instance, last pushed 12 days ago. At the bottom right of the main area is a blue button labeled "CREATE ORG".

| Name | Org/Space | Status | Instances | Last Push |
|-----------------------|----------------------|-----------|-----------|-------------|
| cap-accountmanagement | orgCap / development | ● Stopped | 1 | 13 days ago |
| cappgemini-0.0.1 | orgCap / development | ● Running | 1 | 12 days ago |

3.Create DB on the PCF

Here very interesting and difficult step, please be attention:
goto marketplace:

The screenshot shows the Pivotal Web Services Marketplace interface. At the top, there is a navigation bar with icons for back, forward, refresh, and home, followed by a URL field containing <https://console.run.pivotal.io/organizations/21616437-7297-48c>. Below the URL are several links: Apps, pay, all, OCPJP, AC/DC, Kindle Cloud Read..., and weather. On the left, a sidebar menu includes Home, Marketplace (which is currently selected), Tools, and Docs. The main content area has a search bar at the top with the placeholder "Search apps, services, spaces, & orgs". A blue info box displays the message: "Info: Your org, 'orgCap', is still in trial. To get access". Below the search bar, the word "Marketplace" is displayed, followed by a dropdown menu showing "orgCap". A call-to-action button says "Get started with our free marketplace services. Upgrade". There is also another search bar at the bottom with the placeholder "Search by name, description, or tags".

search mysql

Pivotal Web Services

Home Marketplace Tools Docs ↗ Support ↗ Blog ↗ Status ↗ GIVE FEEDBACK

Search apps, services, spaces, & orgs

Info: Your org, "orgCap", is still in trial. To get access, upgrade to a paid plan.

Marketplace orgCap

Get started with our free marketplace services. Upgrade

my

Services ^

 ClearDB MySQL Database
Highly available MySQL for your Apps.

chose that DB

The screenshot shows the Pivotal Web Services marketplace interface. On the left is a dark sidebar with navigation links: Home, Marketplace (which is selected), Tools, Docs, Support, Blog, Status, and a GIVE FEEDBACK button. The main content area has a search bar at the top. A blue banner at the top right says "Info: Your org, 'orgCap', is still in trial. To get access to 25GB of memory and paid service plans, [upgrade now](#)". Below this, the "ClearDB MySQL Database" service is listed under the "SERVICE" category. It is described as "Highly available MySQL for your Apps." To the right, there is "ABOUT THIS SERVICE" text: "ClearDB is a reliable, fault tolerant, geo-distributed database-as-a-service for your MySQL powered applications". Below the service listing, there are four other database options: "Spark DB" (free), "Boost DB" (\$10.00/MONTH), "Amp DB" (\$50.00/MONTH), and "Shock DB" (\$100.00/MONTH). On the far right, there is company information: "COMPANY" and "SuccessBr". At the bottom right of the main content area is a blue "SELECT THIS PLAN" button.

Pivotal
Web Services

Search apps, services, spaces, & orgs

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](#)

SERVICE

ClearDB MySQL Database

Highly available MySQL for your Apps.

Docs [Support](#)

Spark DB
free

Boost DB
\$10.00/MONTH

Amp DB
\$50.00/MONTH

Shock DB
\$100.00/MONTH

ABOUT THIS SERVICE

ClearDB is a reliable, fault tolerant, geo-distributed database-as-a-service for your MySQL powered applications

COMPANY

SuccessBr

Spark DB free

- Price: FREE!
- DB size: up to 5 MB
- Connections: 4
- I/O Performance: Low
- Daily Backups
- Perfect for proof-of-concept and initial development.

SELECT THIS PLAN

press select this plan (free)

The screenshot shows a service configuration interface for ClearDB MySQL Database. On the left, a sidebar menu includes Home, Marketplace (selected), Tools, Docs, Support, Blog, Status, and a GIVE FEEDBACK button. The main content area displays information about the ClearDB MySQL Database service, noting it's in trial mode and available for 25GB of memory. It features a 'Spark DB' plan, described as highly available MySQL for apps, with a free tier. The 'Instance Configuration' section allows setting the instance name to 'development' and binding it to the 'development' space. There's an optional 'Bind To App' field set to '[do not bind]'. An 'Advanced Configuration' section with a 'SHOW ADVANCED OPTIONS' link is also present. At the bottom are CANCEL and CREATE buttons.

Info: Your org, "orgCap", is still in trial. To get access to 25GB of memory and paid service plans, [upgrade now](#)

SERVICE
ClearDB MySQL Database
Highly available MySQL for your Apps.

ABOUT THIS SERVICE
ClearDB is a reliable, fault tolerant, geo-distributed database-as-a-service for your MySQL powered applications

COMPANY
SuccessBricks, Inc. DBA ClearDB

Docs [Support](#)

Spark DB
free

- Price: FREE!
- DB size: up to 5 MB
- Connections: 4
- I/O Performance: Low
- Daily Backups
- Perfect for proof-of-concept and initial development.

Instance Configuration

Instance Name

Add To Space
development

Bind To App (Optional)
[do not bind]

Advanced Configuration [SHOW ADVANCED OPTIONS](#)

[CANCEL](#) [CREATE](#)

fill in one field
press create

The screenshot shows the Pivotal Web Services dashboard. On the left, a sidebar lists navigation options: Home, Marketplace, Tools, Docs, Support, Blog, Status, and a feedback link. The main content area displays a service card for 'ClearDB MySQL Database'. The card includes a 'SERVICE' section with a database icon and the text 'Highly available MySQL for your Apps.', and an 'ABOUT THIS SERVICE' section describing ClearDB as a reliable, fault tolerant, geo-distributed database-as-a-service. A blue banner at the top states, 'Info: Your org, "orgCap", is still in trial. To get access to 25GB of memory and paid service plans, [upgrade now](#)'. Below the service card is a 'Spark DB' section with a 'free' badge and a list of features: Price: FREE!, DB size: up to 5 MB, Connections: 4, I/O Performance: Low, Daily Backups, and Perfect for proof-of-concept and initial development.

Instance Configuration

Instance Name: Capinstanse

Add To Space: development

Bind To App (Optional): [do not bind]

Advanced Configuration: [SHOW ADVANCED OPTIONS](#)

Buttons: CANCEL, CREATE

then you will create successful

The screenshot shows the ServiceNow interface. On the left is a dark sidebar with links: Home, Marketplace, Tools, Docs, Support, Blog, and Status. A 'GIVE FEEDBACK' button is at the bottom. The main area has a blue header bar with an info message: 'Info: Your org, "orgCap", is still in trial. To get access to 25GB of memory and paid service plans, [upgrade now](#)'. Below it is a green success message: 'Success: Service instance "CapInstanse" was successfully created'. The URL 'Home / orgCap / development' is shown. Under 'development', there's a summary: SPACE (1), RUNNING (3), STOPPED (0), CRASHED (0). Below this are tabs: Apps (4), Services (3) (which is selected), Route (1), Member (1), and Settings. A blue 'ADD A SERVICE' button is on the right. The 'Services' table lists one item:

| Service | Name | Bound Apps | Plan | Last Operation |
|------------------------|-------------|------------|-----------------|------------------|
| ClearDB MySQL Database | CapInstanse | 0 | free - Spark DB | create succeeded |

press link close the DB

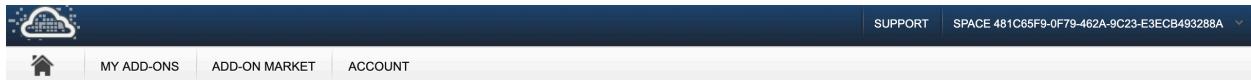
The screenshot shows the Heroku dashboard for the organization "orgCap". A success message at the top states: "Success: Service instance "CapInstanse" was successfully created". Below this, the "development" space summary shows 1 RUNNING service, 3 STOPPED services, and 0 CRASHED services. The "Services" tab is selected, displaying a table with one row:

| Service | Name | Bound Apps | Plan | Last Operation |
|------------------------|-------------|------------|-----------------|------------------|
| ClearDB MySQL Database | CapInstanse | 0 | free - Spark DB | create succeeded |

this window will be opened
goto manage in the right

The screenshot shows the Heroku mobile application interface. At the top, there are navigation links: "Docs" with a gear icon, "Support" with a gear icon, and "Manage" with a gear icon. Below these is a large button labeled "BIND APP".

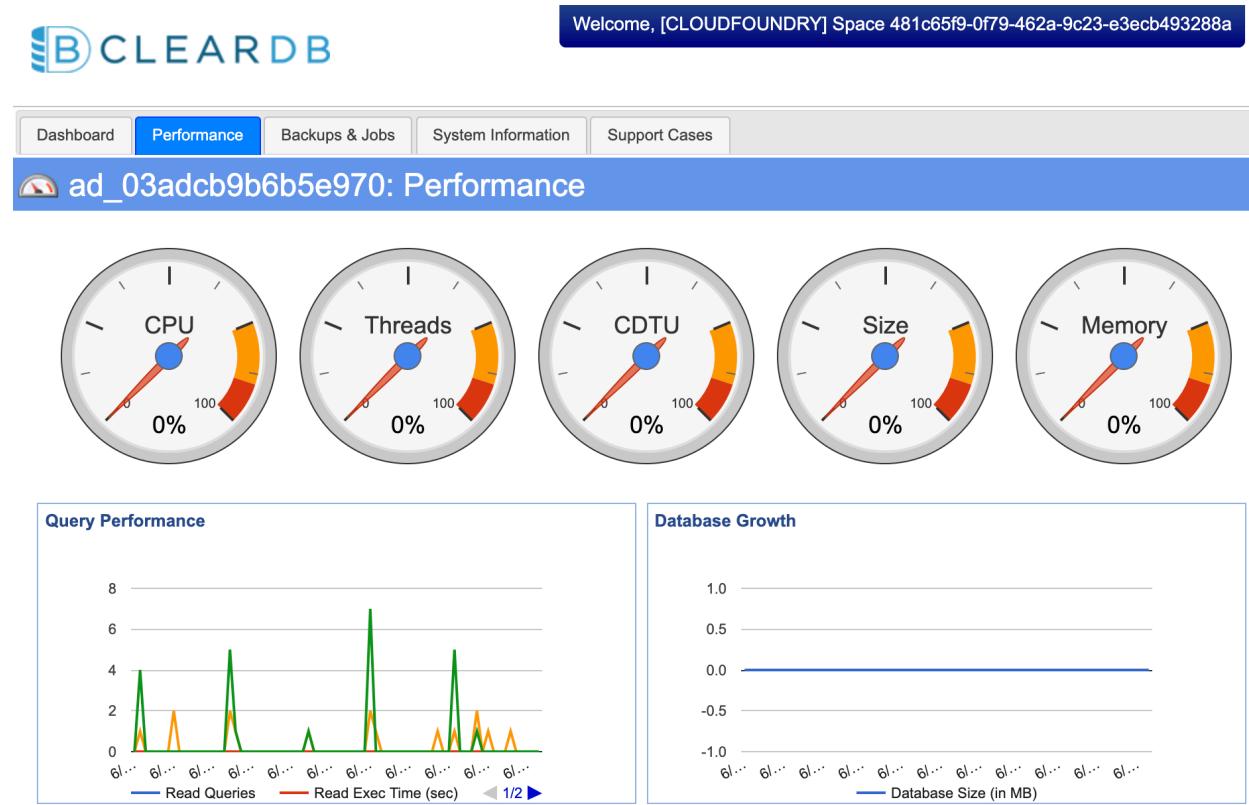
press allow



press link under the name

A screenshot of the ClearDB 'My Databases' page. At the top, there's a header with the ClearDB logo and a welcome message: 'Welcome, [CLOUDFOUNDRY] Space 481c65f9-0f79-462a-9c23-e3ecb493288a'. Below the header, there's a blue navigation bar with the text 'My Databases'. Underneath, there's a table with the following columns: Name, Cloud, Region, Status, Type, Last Updated, and Actions. One row is visible, showing a database named 'ad_03adcb9b6b5e970' with 'Other' as the Cloud provider, 'US-East' as the Region, 'Online' as the Status, 'spark' as the Type, and 'Fri May 31 2019 21:19:26 UTC' as the Last Updated date. The 'Actions' column shows 'N/A'.

you will see this one



goto System information tab

Welcome, [CLOUDFOUNDRY] Space 481c65f9-0f79-462a-9c23-e3ecb493288a

ad_03adcb9b6b5e970: System Information

Cluster Gateway Information

| Hostname | Max Connections | Max Queries/Hour |
|----------------------------------|-----------------|------------------|
| us-cdbr-iron-east-02.cleardb.net | 4 | 3600 |

Cluster Nodes

| Type | Provider | Region | Name | Status |
|-------|---------------------|---------|------------------------------------|--------|
| Cloud | Amazon Web Services | US-East | us-iron-auto-dca-02-a.cleardb.net | Online |
| Cloud | Amazon Web Services | US-East | us-iron-auto-dca-02-b.cleardb.net | Online |
| Cloud | Amazon Web Services | US-West | us-iron-auto-sfo-02-bh.cleardb.net | Online |

and, bingo, we see connection parameters

Cluster Gateway Information

Hostname

us-cdbr-iron-east-02.cleardb.net

Access Credentials

Username: b798ea040f416f

Password: 541c41db (Reset)

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 two 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-79i
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_64
----> Downloading Open Jdk JRE 1.8.0_212 from https://java-buildpack.cloudfoundry.org/openjdk/bionic/x86_64/open
  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-ri
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:$PWD/.java-buildpack/open_jdk_jre/bin/jvmkill-1.16.0_RELEASE=printHeapHistogram
               -Djava.ext.dirs=$PWD/.java-buildpack/container_security_provider:$PWD/.java-buildpack/open_jdk_jre,
               $JAVA_OPTS" && CALCULATED_MEMORY=$($PWD/.java-buildpack/open_jdk_jre/bin/java-buildpack-memory-calculator
               -stackThreads=250 -vmOptions="$JAVA_OPTS") && echo JVM Memory Configuration: $CALCULATED_MEMORY &&
               exec $PWD/.java-buildpack/open_jdk_jre/bin/java $JAVA_OPTS -cp $PWD/. org.springframework.boot.loader.Launcher
               #0   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:

The screenshot shows the Postman interface with a successful API call. The request URL is `https://capgemini-001-exhausted-gerenuk.cfapps.io/gettoken`. The Body tab contains a JSON payload:

```
1 + [  
2     "email": "email",  
3     "pwd": "pwd"  
4 ]
```

The response status is 200 OK, time is 80 ms, and size is 604 B. The response body is displayed in JSON format:

```
1 + {  
2     "token": "eyJhbGciOiJIUzI1NiJ9.eyJqdGkiOiJ0b2tlbiIsImlhCI6MTU2MDI3OTYxMywic3ViIjoiZW1haWwiLCJpc3MiOiJwd2QiLCJleHAiOjE1NjAyODE0MTN9  
.6w2asD6cMznLlvFp02SL8oaY7ujCsXsALMwcDVHaA",  
3     "status": "success",  
4     "tokenException": null  
5 }
```

and PCF

The screenshot shows the Pivotal Web Services dashboard. On the left is a sidebar with links: Home, Marketplace, Tools, Docs (with a dropdown), Support (with a dropdown), Blog (with a dropdown), Status (with a dropdown), and a GIVE FEEDBACK button. The main area has a search bar at the top with placeholder text "Search apps, services, spaces, & orgs". Below the search bar is a message: "Info: Your org, 'orgCap', is still in trial. To get access to 25GB of memory and paid service plans, [upgrade now](#)". The URL in the address bar is "Home / orgCap / development". Below this, there's a summary of the space: "development" with counts: RUNNING 1, STOPPED 3, CRASHED 0. A navigation bar below the summary includes "Apps (4)", "Services (3)", "Route (1)", "Member (1)", and "Settings". The "Apps" section is currently selected. It displays a table of four applications:

| Status | Name | Instances | Memory | Last Push | Route |
|---------|-----------------------|-----------|--------|---------------|---|
| Stopped | accountService | 1 | 768 MB | loading... | <i>no bound route</i> |
| Stopped | cap-accountmanagement | 1 | 1 GB | 13 days ago | <i>no bound route</i> |
| 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:

The screenshot shows the Jenkins dashboard. On the left, there's a sidebar with various links: New Item, People, Build History, Manage Jenkins, My Views, Lockable Resources, Credentials, and New View. Below this is a 'Build Queue' section which says 'No builds in the queue.' To the right is a main content area. At the top of this area is a navigation bar with 'All' and a '+' button. Below it is a table titled 'Name ↓' with four rows. The first row has a blue circle icon, a yellow sun icon, and the name 'deploy'. The second row has a grey circle icon, a yellow sun icon, and the name 't'. The third row has a grey circle icon, a yellow sun icon, and the name 'test'. The fourth row has a blue circle icon, a yellow sun icon, and the name 'token-build'. Below this table is another table with columns 'W' and 'Description', showing a yellow sun icon and the text 'Build stability: No recent builds'. At the bottom of the main content area, there's a section titled 'Build Executor Status' with two entries: '1 Idle' and '2 Idle'.

| S | W | Name ↓ |
|---|---|-------------|
| | | deploy |
| | | t |
| | | test |
| | | token-build |

Icon: [S](#) [M](#) [L](#)

| W | Description |
|---|-----------------------------------|
| | Build stability: No recent builds |

| Build Executor Status | |
|-----------------------|------|
| 1 | Idle |
| 2 | Idle |

create freestyle project with name: build

Jenkins > All >

Enter an item name

build

» Required field



Freestyle project

This is the central feature of Jenkins. Jenkins will build your code and something other than software build.



Pipeline

Orchestrates long-running activities that can span multiple nodes. Suitable for organizing complex activities that do not easily fit in free-style projects.



Multi-configuration project

Suitable for projects that need a large number of different configurations.

on the source code management tab, please fill out

Source Code Management

None
 Git

Repositories

| | | | |
|----------------|---|-----|---|
| Repository URL | <input type="text" value="https://github.com/capgeminiAccountProject/token.git"/> | X | ? |
| Credentials | - none - | Add | |
| Advanced... | | | |

| | | | |
|----------------|---|-----|---|
| Repository URL | <input type="text" value="https://github.com/capgeminiAccountProject/token.git"/> | X | ? |
| Credentials | - none - | Add | |
| Advanced... | | | |

Add Repository

Branches to build

| | | | |
|------------------------------------|---------------------------------------|---|---|
| Branch Specifier (blank for 'any') | <input type="text" value="*/master"/> | X | ? |
| Add Branch | | | |

Repository browser (Auto)

Additional Behaviours Add ▾

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

The screenshot shows the Jenkins build configuration interface. It consists of two main sections:

- Invoke Gradle script**:
 - Configuration:
 - Invoke Gradle (radio button)
 - Use Gradle Wrapper (radio button, selected)
 - Make gradlew executable (checkbox checked)
 - Wrapper location: (empty input field)
 - Tasks: clean build
 - Buttons: Advanced...
- Execute shell**:
 - Configuration:
 - Command:

```
chmod 777 /Users/Shared/Jenkins/Home/workspace/token-build/build/libs/capgemini-0.0.1.jar
chmod 777 ./git/objects/pack/*
chmod 777 ./*
chmod 777 /Users/Shared/Jenkins/Home/workspace/token-build/build/libs/capgemini-0.0.1.jar
```
 - Buttons: Advanced...

At the bottom left, there is a "Add build step" button.

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

```

Pipeline

Definition Pipeline script

Script

```

1 pipeline {
2     agent any
3     stages {
4         stage('Build') {
5             steps {
6                 sh 'echo *****'
7                 sh 'echo ***** Starting deploy to the PCF *****'
8                 sh 'echo *****'
9
10                sh 'cp -v /Users/Shared/Jenkins/Home/workspace/token-build/build/libs/capge
11                sh 'cp -v /Users/Shared/Jenkins/Home/workspace/token-build/src/main/resource
12
13                pushToCloudFoundry cloudSpace: 'development', credentialsId: 'bc42f2f9-73c5
14
15                sh 'echo *****'

```

Use Groovy Sandbox

[Pipeline Syntax](#)

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'
```

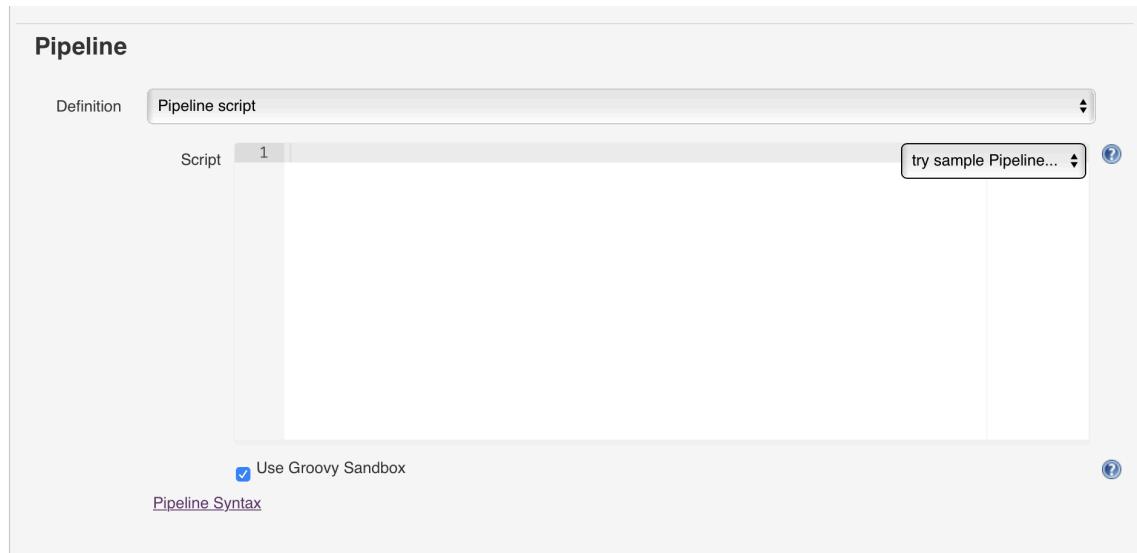
here we are copying to files to the one folder, because for the pushing jar to PCF we have to have jar and manifest together. (of course we can use whole PATH but it is extra headache).

```
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
```

Use Groovy Sand

[Pipeline Syntax](#)

on the sample step we can chose:

Overview

This **Snippet Generator** will help you learn the Pipeline Script code which can be used to define various steps. Pick a step you are interested in from the list, configure it, click **Generate Pipeline Script**, and you will see a Pipeline Script statement that would call the step with that configuration. You may copy and paste the whole statement into your script, or pick up just the options you care about. (Most parameters are optional and can be omitted in your script, leaving them at default values.)

Steps

Sample Step | archiveArtifacts: Archive the artifacts

Files to archive

Advanced...

Generate Pipeline Script

The screenshot shows a user interface for generating Pipeline Script snippets. At the top, there's a header with sections for 'Overview' and 'Steps'. Below the 'Steps' section, a 'Sample Step' is selected: 'archiveArtifacts: Archive the artifacts'. This step has a configuration field 'Files to archive' containing '[]'. There's also an 'Advanced...' button with a gear icon. A prominent blue button at the bottom left says 'Generate Pipeline Script'. To the right of this button is a large, empty text area with a scroll bar, intended for displaying the generated Pipeline Script code.

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

Overview

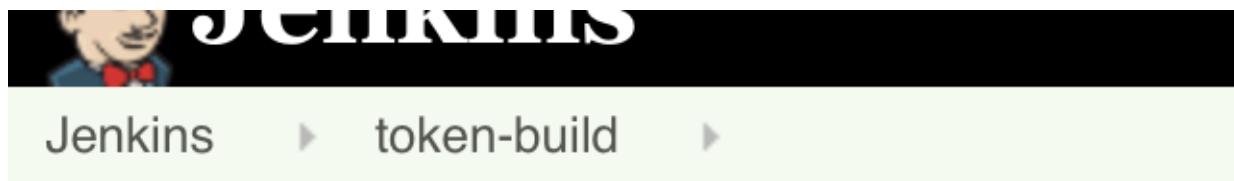
This **Snippet Generator** will help you learn the Pipeline Script code which can be used to define various steps. Pick a step you are interested in from the list, configure it, click **Generate Pipeline Script**, and you will see a Pipeline Script statement that would call the step with that configuration. You may copy and paste the whole statement into your script, or pick up just the options you care about. (Most parameters are optional and can be omitted in your script, leaving them at default values.)

Steps

Sample Step pushToCloudFoundry: Push to Cloud Foundry

| | | |
|--|---|--|
| Target | <input type="text"/> | (?) |
| Credentials | <input type="button" value="- none -"/> Add ▾ | (?) |
| Organization | <input type="text"/> | (?) |
| Space | <input type="text"/> | (?) |
| Allow self-signed certificate | <input type="checkbox"/> | <input type="button" value="Test Connection"/> |
| Plugin timeout (s) | <input type="text" value="120"/> | (?) |
| Create services before pushing | Add | (?) |
| <input checked="" type="radio"/> Read configuration from a manifest file | | |
| Manifest file | <input type="text" value="manifest.yml"/> | (?) |
| <input type="radio"/> Enter configuration in Jenkins | | |

then we need to ram our tasks



The image shows a screenshot of the Jenkins web interface. At the top, there is a dark header bar with the Jenkins logo on the left and the word "JENKINS" in white capital letters. Below the header, the URL "Jenkins" is followed by a right-pointing arrow, then "token-build", and another right-pointing arrow. The main content area has a light gray background. On the left side, there is a vertical list of options, each with an icon and text: "Back to Dashboard" (green arrow icon), "Status" (magnifying glass icon), "Changes" (document with pencil icon), "Workspace" (blue folder icon), "Build Now" (hourglass icon), "Delete Project" (red circle with slash icon), "Configure" (purple gear icon), and "Rename" (document with pencil icon). The "Status" option is currently selected, as indicated by its bolded text.

-  Back to Dashboard
-  **Status**
-  Changes
-  Workspace
-  Build Now
-  Delete Project
-  Configure
-  Rename

build run, and enjoy application on the cloud

The screenshot shows the Pivotal Web Services dashboard. On the left is a sidebar with links: Home, Marketplace, Tools, Docs (selected), Support, Blog, Status, and a GIVE FEEDBACK button. The main area has a search bar at the top. A blue banner at the top right says "Info: Your org, 'orgCap', is still in trial. To get access to 25GB of memory and paid service plans, [upgrade now](#)". Below the banner, the URL is "Home / orgCap / development". It shows a summary of the "development" space with counts for RUNNING (1), STOPPED (3), and CRASHED (0) instances. A table below lists four apps: accountService (Stopped), cap-accountmanagement (Stopped), capgemini-0.0.1 (Running), and capgemini-0.1.0 (Stopped). The capgemini-0.0.1 app has a link to its route: <https://capgemini-001-exhausted-gerenuk.cfapps.io>.

| Status | Name | Instances | Memory | Last Push | Route |
|-----------|-----------------------|-----------|--------|-------------|---|
| ● Stopped | accountService | 1 | 768 MB | loading... | <i>no bound route</i> |
| ● Stopped | cap-accountmanagement | 1 | 1 GB | 13 days ago | <i>no bound route</i> |
| ● Running | capgemini-0.0.1 | 1 | 768 MB | an hour ago | https://capgemini-001-exhausted-gerenuk.cfapps.io |
| ● Stopped | capgemini-0.1.0 | 1 | 768 MB | loading... | <i>no bound route</i> |