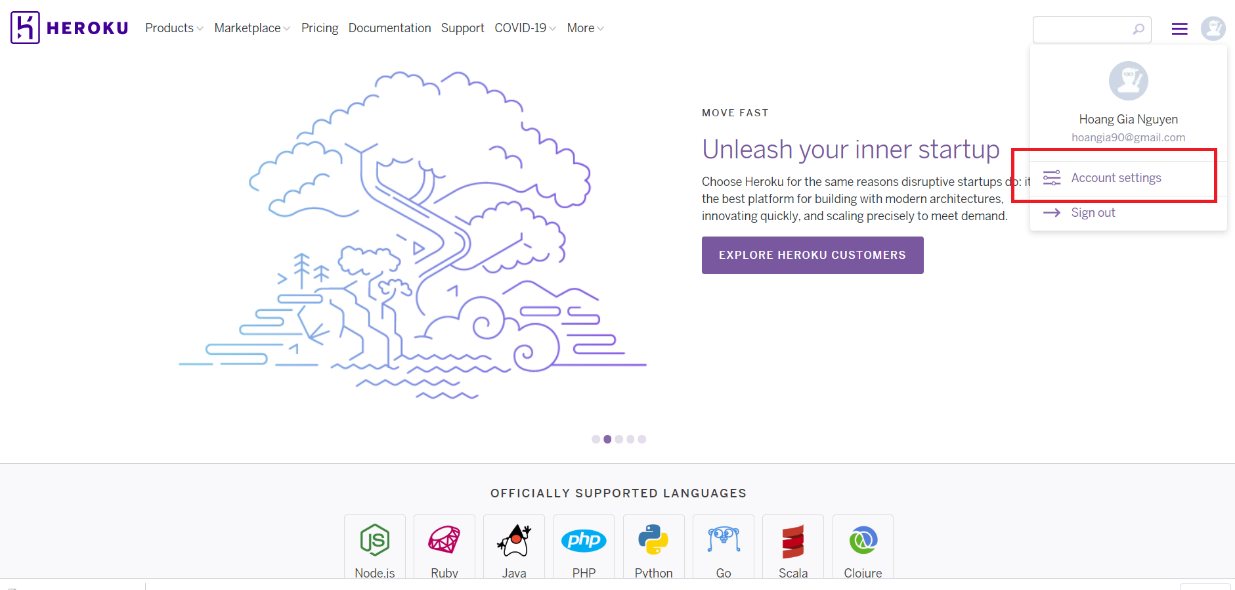
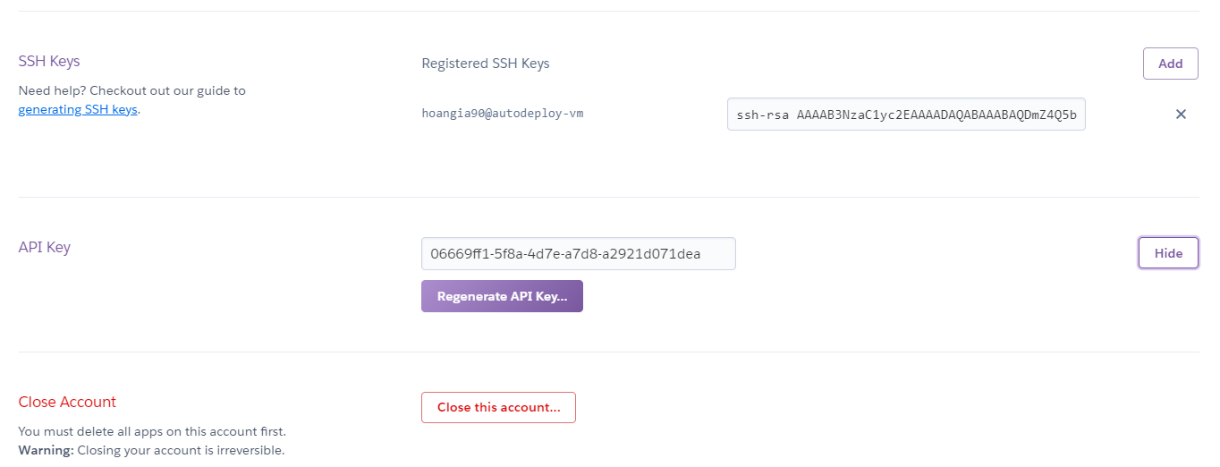
JavaBIP for Cloud Tutorial

JavaBIP for Cloud is an extended from the legacy JavaBIP running on local machine. JavaBIP for Cloud is used for cloud such as PaaS, IaaS. This Demo is a micro-service which is used for deploying automatically a web application/micro-service. The demo includes:

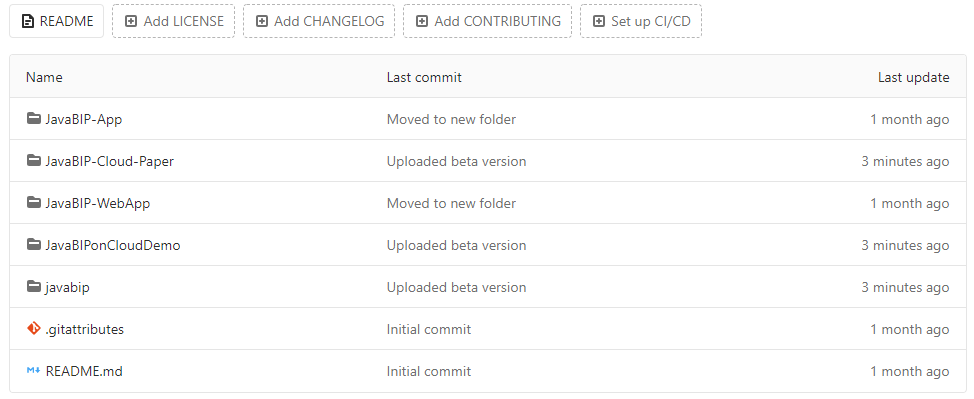
* 1. BIPDeployer micro-service:
     1. Run on IaaS (Google Compute) or local machine
     2. This micro-service in the demo not only calls Heroku APIs but also executes Heroku CLI. Therefore, it needs to run on a virtual machine (such as Google Compute) where BIPDeployer can access the terminal for executing Heroku CLI commands
  2. Compute micro-service: run on PaaS (Heroku) or local machine
     1. A simple web application/micro-service can be run on PaaS (like Heroku) or any PaaS and IaaS

1. Setup BIPDeployer
   1. Get Heroku authentication token/API key by creating an Heroku account with your e-mail on <https://www.heroku.com/home>, and then go to the Account Setting to get the key as in photo below:

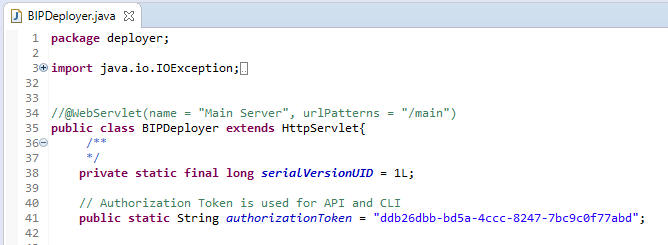




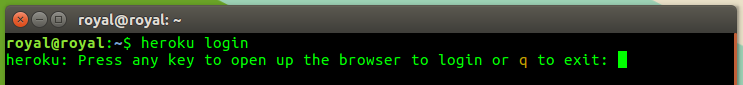
* 1. Clone the source code to your local machine with git at <https://gitlab.inria.fr/hoanggng/JavaBIP.git> or <https://github.com/hoangia90/JavaBIP.git>:



* + 1. JavaBIP-App: legacy JavaBIP with updated libraries running as Java application
    2. JavaBIP-WebApp: Extended version of JavaBIP running on Web (Java Servlet)
    3. JavaBIPonCloudDemo: A demo of JavaBIP-WebApp.
    4. JavaBIP-Cloud-Paper: A paper for JavaBIP on cloud
  1. Change Heroku authentication key with your key associated with your account in BIPDeployer.java

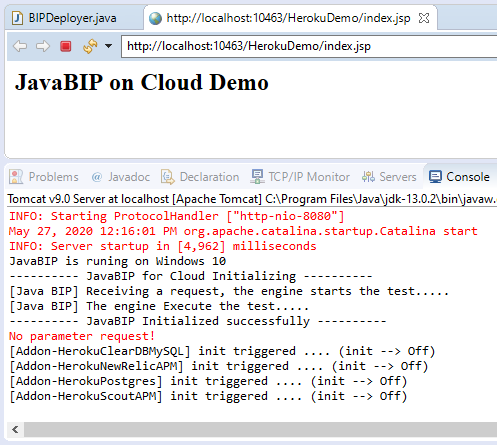


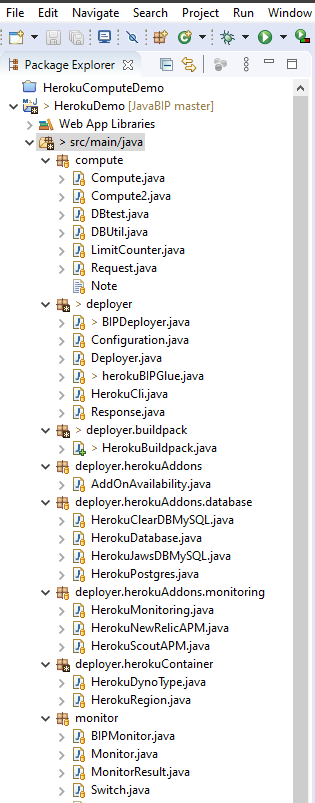
* 1. Install Heroku for your local machine at <https://devcenter.heroku.com/articles/heroku-cli#download-and-install> and sign in with your created account by command “Heroku login” at your local machine. Note that the authentication key must be generated with the account.



* 1. Set Heroku CLI directory and the file path of a web application/micro-service in war format that you want to deploy on Heroku



* 1. Open and run JavaBIPonCloudDemo by running BIPDeployer.java. To run it correctly, you need to setup first the Tomcat server (<http://tomcat.apache.org/>) and Java JDK. In this time of writing the demo, we are running it with Tomcat 9 and JDK 14 with using JRE 1.8 for JavaBIP libraries.
  2. Packages explication

From the top down to the bottom of the package explorer on the right side are:

* 1. Compute: a simple web application/micro-service generating a random integer number.
  2. Deployer: a JavaBIP based web application/micro-service deploying automatically a web application/micro-service on Heroku platform. It contains mainly deployer BIP model, BIP configuration. Sub-packages contain BIP models for

Container or Heroku Dyno (<https://devcenter.heroku.com/categories/dynos>)

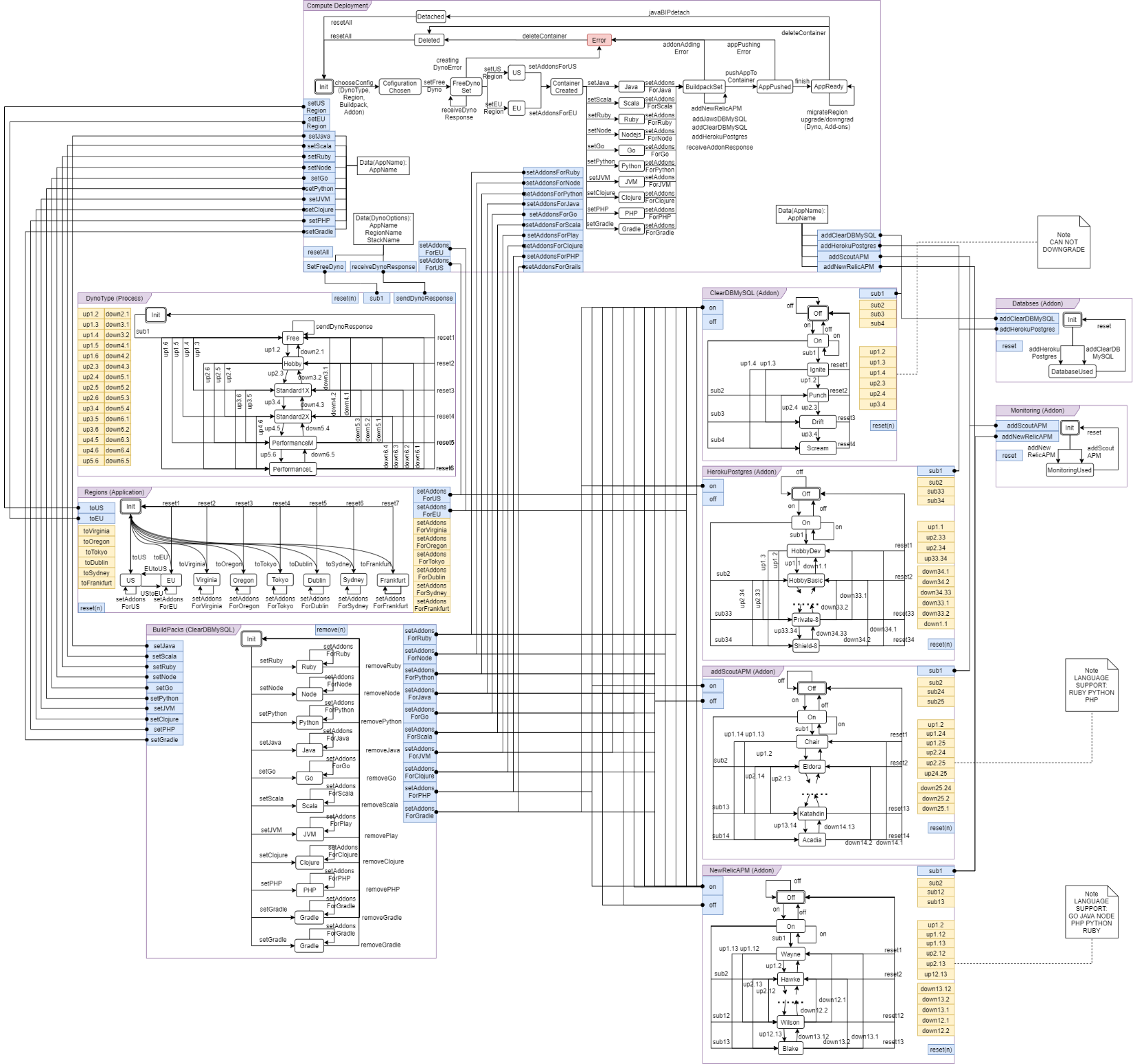
Buildpack or Heroku buildpack (<https://devcenter.heroku.com/articles/buildpacks>)

Add-ons or Heroku Add-ons (<https://elements.heroku.com/addons>)

* 1. Monitor: another JavaBIP service for monitoring the compute micro-service and it will be not involved in this demo

In this demo we use HerokuComputeDemo.war for deploying

Note: for understanding more about Heroku APIs and CLI commands, its references published at <https://devcenter.heroku.com/articles/heroku-cli> and <https://devcenter.heroku.com/articles/platform-api-reference>

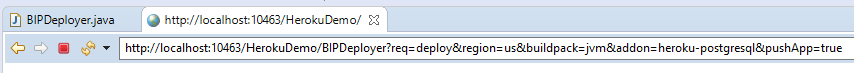


BIPDeployer model

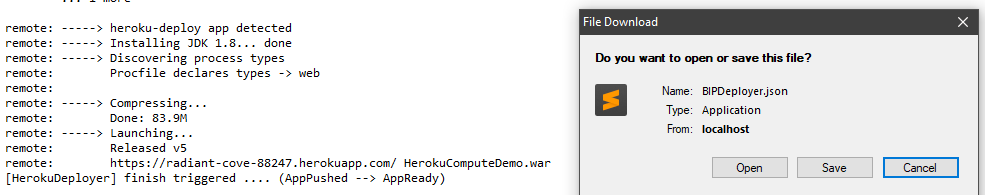
1. Run the demo

In order to use BIPDeployer, we use APIs which are using HTTP Get methods, in the demo we will show you how to deploy a Compute application onto Heroku free container (Dyno) located in Europe or US together with native Heroku Postgre database addon.

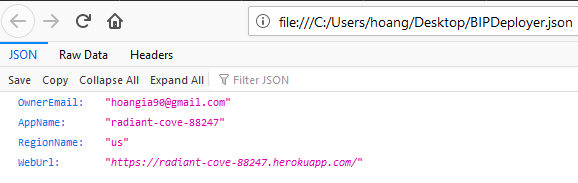
* 1. Enter the following url <http://localhost:10463/HerokuDemo/BIPDeployer?req=deploy&region=us&buildpack=jvm&addon=heroku-postgresql&pushApp=true>. With “req” is “deploy”, “region” is “us” or “eu”, “buildpack” is “jvm” (java virtual machine), “addon” is “Heroku-postgre” and “pushApp” is “true” or “false”. The last parameter aims to skip the deploying the app to deployed container which takes quite long time, we recommended set it “false” for checking if the container was created in your Heroku account.



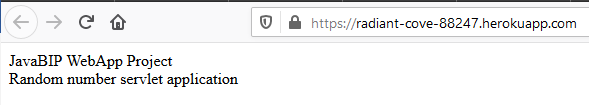
When it is deployed, you will return a json file



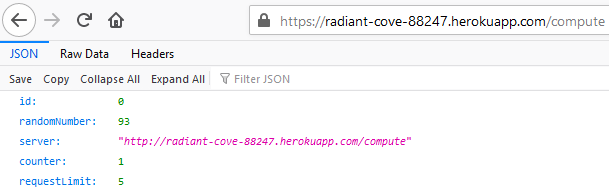
Open the json file you will see the information of container name or web application name, region and the application url



* 1. Open the application on deployed container
     1. Follow the url in the previous step to go to the application. If you could see the content below then the Compute application is successfully deployed on a Heroku container.

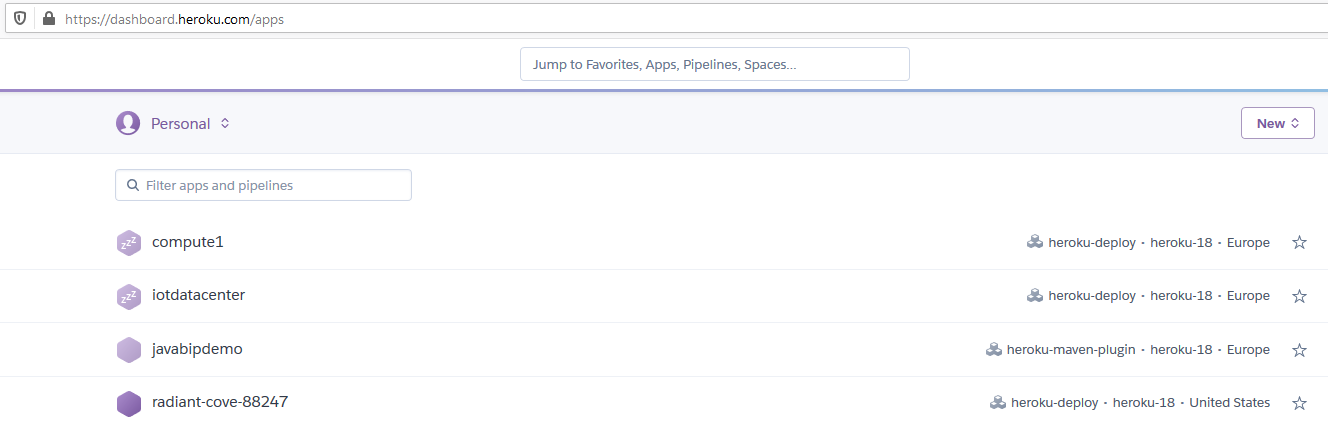


* + 1. Entering your servlet url to run the application, we name our servlet in Compute application is “compute”. After requesting, the server will show the content in JSON format as below

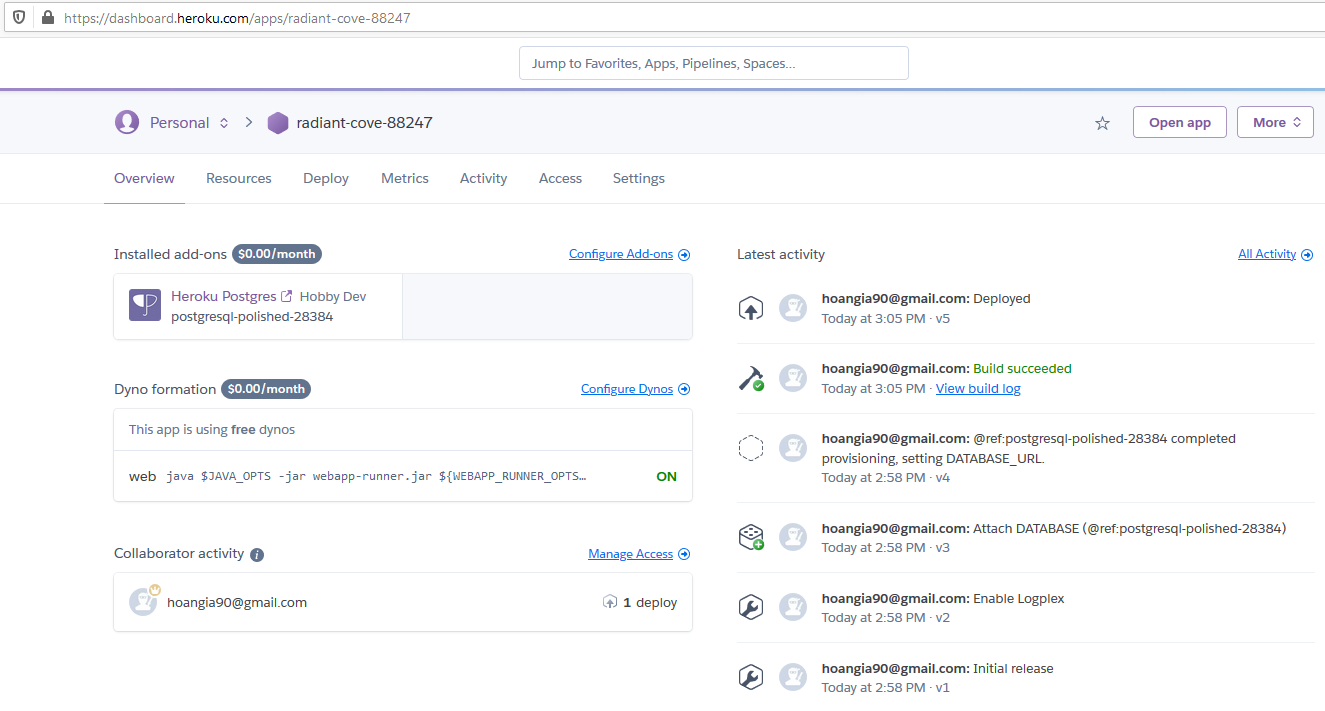


1. Check the configuration of the deployed container

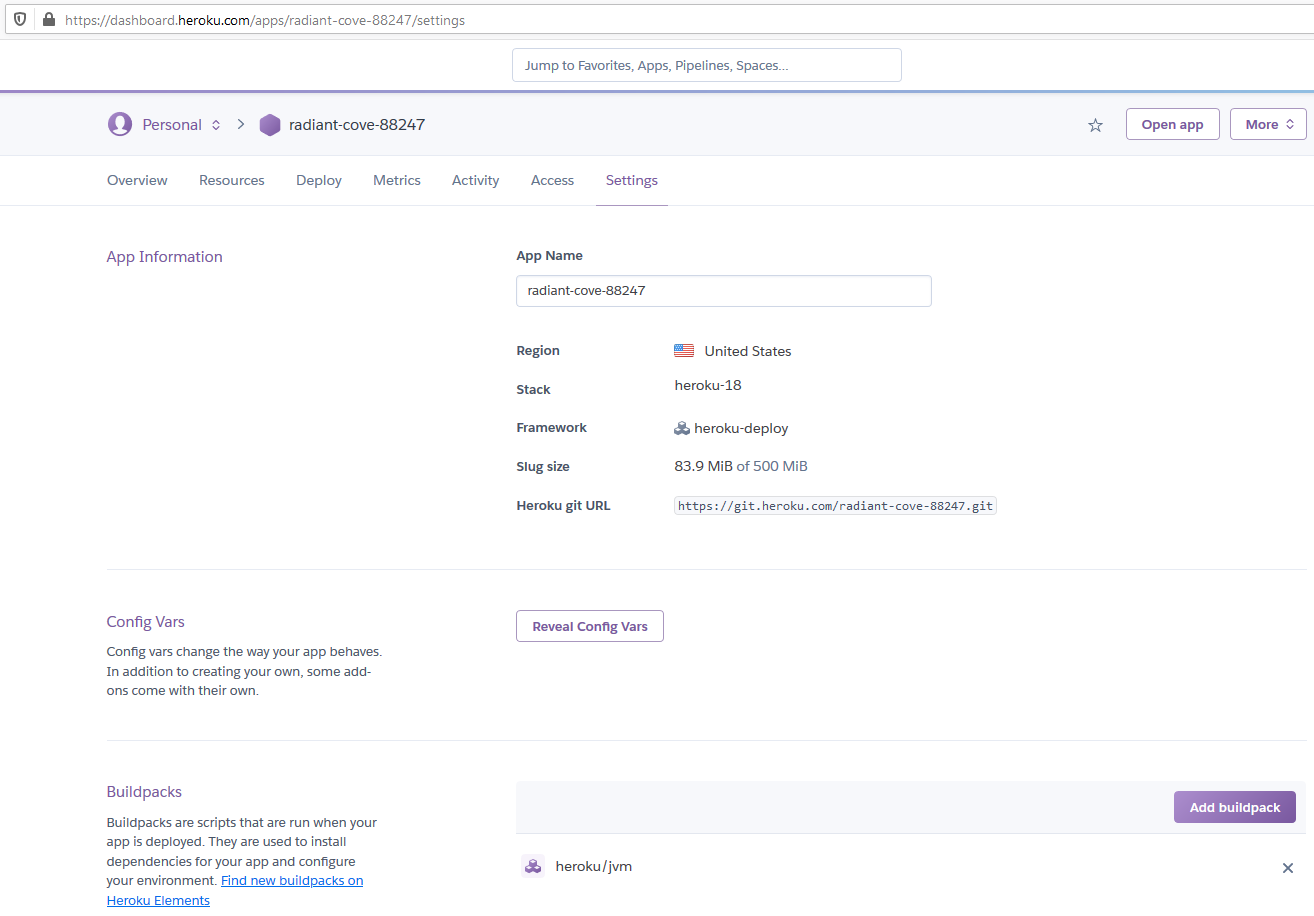
We can check the container configuration on (<https://dashboard.heroku.com/apps>). At this page, we can see the created container is located in US location



Dyno plan we created here is free with a free Postgre database addon plan



Our builpack is Heroku/jvm

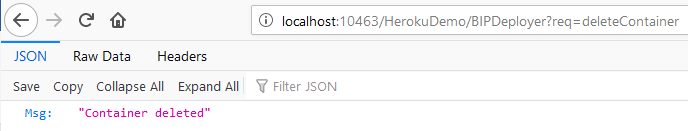


1. Control, monitor, detach application

When a application or micro-service is deployed, JavaBIP engine is still running in order to control and monitoring it as in the BIPDeployer model above. We currently implemented few functions for showing in the demo.

* 1. Delete the container: <http://localhost:10463/HerokuDemo/BIPDeployer?req=deleteContainer>
  2. Detach the JavaBIP engine, which detaches the JavaBIP from controlling the container and leaves the application: <http://localhost:10463/HerokuDemo/BIPDeployer?req=detach>

We will try with delete the container



1. APIs:
   1. Start JavaBIP engine: <http://localhost:port/HerokuDemo/BIPDeployer?req=start>
   2. Stop JavaBIP engine: <http://localhost:port/HerokuDemo/BIPDeployer?req=stop>
   3. Restart JavaBIP engine: <http://localhost:port/HerokuDemo/BIPDeployer?req=restart>
   4. Delete container: <http://localhost:port/HerokuDemo/BIPDeployer?req=deleteContainer>
   5. Detach container: <http://localhost:port/HerokuDemo/BIPDeployer?req=detach>
   6. Deploy a web application or micro-service <http://localhost:port/HerokuDemo/BIPDeployer?req=deploy&region=us(eu)&buildpack=jvm&addon=heroku-postgresql&pushApp=true(false)>
2. Showing Error

In case any error in deploying, the server will send back an error information in JSON