Basically I want to outline two things,

- What problems I'm facing
- What the solution is.

So I'm sure you guys know my concept.

- Build Sogaco from source
- Build our code from source
- CI to build/test on git push
- Deploy to some server.

The problem?

Building Sogaco requires 2 binaries to be installed.

- HG(mercurial) to clone the repository
- mvn(Maven to install Sogaco and generate the WAR files)

On a local machine thats fine, you can install them.
On a remote VM (Some linux VM) you can also install them.
On a web deployer like Heroku, you can't.

Why?

Because Heroku works off It's own theory of a VM.
Basically you:

- Create a build pack (Or in most cases use one).
- Tell Heroku to use your buildpack when Deploying
- And that's it, automatic deployment.

The problem is that buildpacks are designed to be as lightweight as humanly possible. Which means they have installed on the Buildpack VM's effectively nothing, and you have to add everything through bash scripts. This is not ideal, since you don't have sudo access to these shell's its extremely hard to add anything. It can be done but requires a lot of work. So I started looking at other possibilities. In particular AWS because it's free for 12 months and integrates with Travis well.

The problem with AWS:

Using AWS for this specific project is like using a 50. cal to kill a mouse. it's super overkill. AWS is great in the fact it's modular, you can deploy to different servers at the same time so you have backup servers, you keep the file system on a separate server using "buckets" to the actual runtime environment with multiple levels of backup and security. But, our application needs

none of this. It’s a very simple webapp, It doesn't need to scale much, we don't have many users and aren't going to have to deploy to a cluster to manage everything. The overhead of creating an EC2 server is immense, Well, i lied. To create the server is super simple, you click create and select AWs and all of a sudden you have a VM to ssh into with everything from python to ruby to yum installed with passwordless sudo. Although this involves skipping about 50 steps of which each one is more complex and requires more research than the last. At the end of the day you end up realising unless you are Someone like Twitter, you are going to use 0.01% of what's available on AWS in terms of features. But the main problem lies in deployment. You generate a few SSH keys and some encrypted passwords, Travis CI even encrypts them again so when the deploy information is in Git no one can actually decode it (That’s great and easy). But the you have to specify what group of servers you're deploying to, what Bucket you need, etc. etc. The list goes on. The deploy on heroku vs AWS is about 50 lines difference in a basic Travis config. Following KISS (Keep it Simple Stupid) AWS is far too complex for our needs and would make a maintenance nightmare, as well as the fact that no one other than me would ever be able to use it other than me would be able to use it without extensive research.

What about the others.

Mostly travis has no inbuilt support for these. i am looking at potentially using Deploybot (A sort of CI on steroids) as a replacement for Travis, deploy bot is free and has support for digital ocean droplets among others. I will investigate this further over the next day.

The Solutions:

1. Still use heroku but keep Sogaco build artifacts in the repository so Heroku can use them. (We will still build from source for development so we have access to the Sogaco source for reference and to test potential Sogaco changes)

2. Make the complex Heroku Buildpack

3. Create the complex AWS setup and use AWS codedeploy

4. Deploybot + Digital ocean.

5. Get Jens to publish artifacts on a private maven repository (I think this is unlikely, but dreams are free)

5 is the best option and makes the entire process much simpler, however it is the least likely to happen.

I’m going to temporarily deploy to heroku using option 1. It requires the least effort and is the easiest to maintain. It allows us to get builds deploying to live and everyone is happy.

In the long term I will look towards using something liem deploybot + Digital ocean. This makes it easier to swap versions of SoGaCo and makes it a uniform process across the entire pipeline.