**What is Sassy?**

It's SAS's sexy connection between any front-end and CAS/VIYA, so it's named Sassy. It's also just right to type in whenever you want it up and running, like:

**TBD** (cannot be replicated and setup easily by others until enterprise git and true VIYA connectivity is up.)

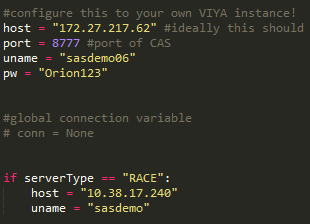
~~pip install sassy\_viya~~

Long story short, connect to your SAS Viya instance easily!

**Quickstart**

1. Copy and paste the whole Sassy package into the run path. As admin, cd to the 'python' folder (e.g. Sassy\python), and run 'pip install -r requirements.txt'.

2. If using an AuthInfo file change authinfo.txt to your own authentication details and to point to your VIYA server IP and port.

If not, change them in Sassy.py, change 'host','uname' and 'port' accordingly. Alternatively if you are using a RACE image, state serverType as "RACE" and change the host name to that accordingly. This is in case you need to change to a RACE image halfway (it happens)  


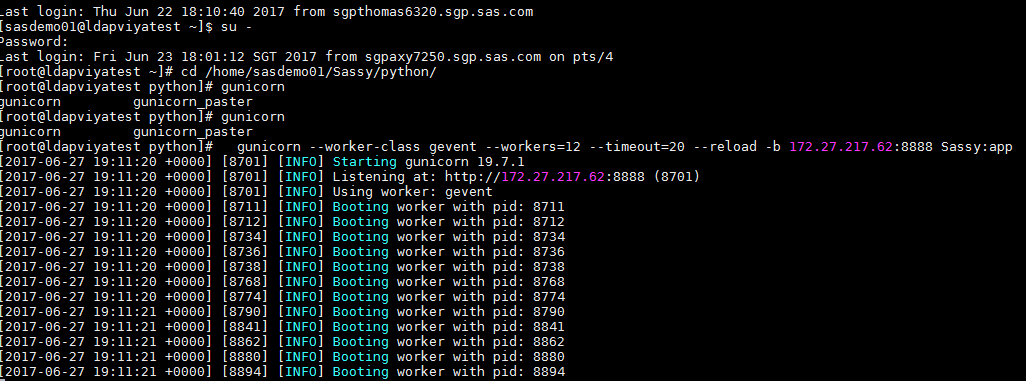
For ease of setup, it is good to host the API server on the same IP as your VIYA instance (Linux-based), as this makes the SWAT wrapper and configs behave well.

Then run the server:

gunicorn --workers=NUM\_CONCURRENT\_USERS --reload -b API\_SERVER\_IP\_HERE:8888 Sassy:app

e.g.

gunicorn --worker-class gevent --workers=12 --timeout=20 --reload -b 172.27.217.62:8888 Sassy:app



If needed, use --log-level:DEBUG to see the existing config and a stacktrace of errors.

3. Within same server and on another network, check initial connection:

PostMan:

API\_SERVER\_IP\_HERE:8888/testconn  
Note: Ensure you are sending form-data

OR

Linux/Mac:

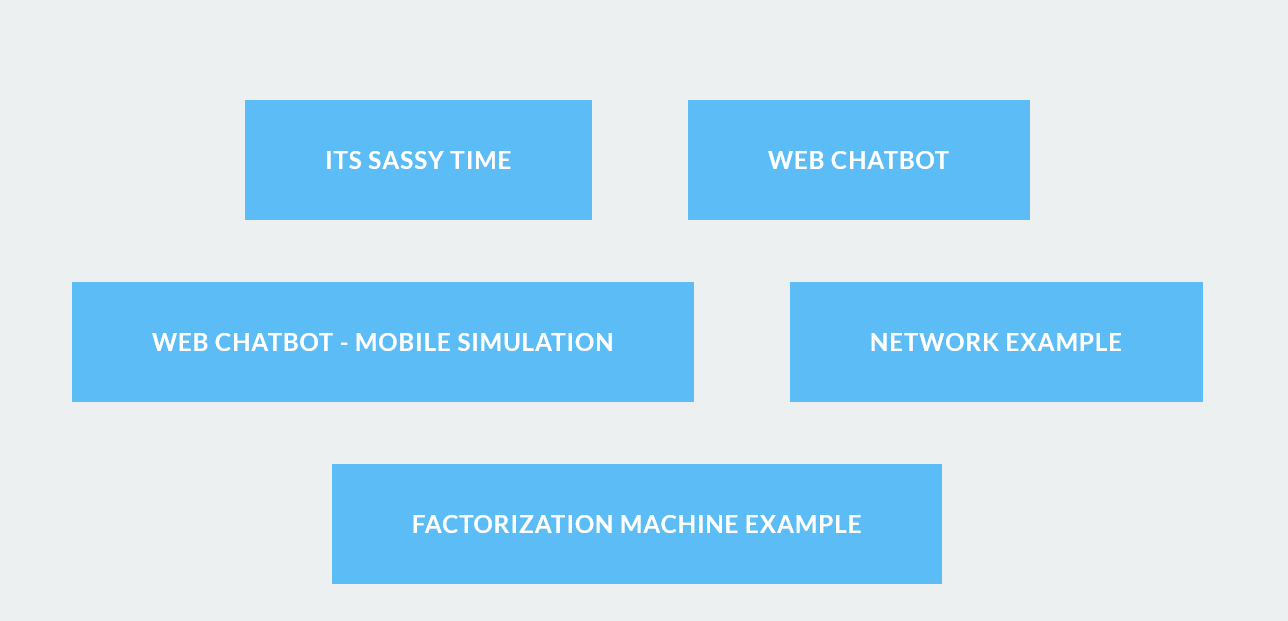
curl -d POST 172.27.217.62:8888/testconn

OR

Windows:

http --form POST 172.27.217.62:8888/testconn

(pip install -U httpie)

If these work, you're good to go! Go to a sample frontend folder (i.e. Chrome) and run node server.js. Go to localhost:8080 and select one of the demos. You should see this:  


**MUST READ:**  
You **need** to change the IPs of **each** demo page to where they are located!! To build out new buttons and pages, you can refer to \Sassy\sample frontends\Chrome\public\boilerplate.html for rudimentary pages. As of this time of writing, no templating engines were used, so the pages were manually copy-pasted with JSON calls. I know, I know…  
  
In other words, if you are using a RACE demo image, every renewal and subsequently IP address needs to be reflected on the demos which call where Sassy is hosted. And Sassy is usually hosted alongside CAS…  
   
Maintaining the frontend code will be much easier if the demo pages reference a pure public cloud URL. I call this ‘pure’ public as it can be universally accessed without limitation of network, just as a RESTful call to motion.ai. This means less changes with each new build of Sassy or the demo, and consistency throughout all versions of Sassy.  
  
However, if you do not have a pure public cloud URL or have limitations in hosting VIYA on public cloud, this cannot be done. Some demos would not work, especially those with frontend logic sitting on their own servers, such as motion.ai. An exception to this is Amazon Alexa, which uses the Lambda service. Lambda service has a special VPC connection back to a private cloud instance if you have one. Be warned though, this may prove costly over the long run.

**DEMO USER GUIDE:**

1. Ensure server is running. If Sassy and CAS is on a local box (aka AFT), you need to ensure VPN to SAS network (Cisco AnyConnect). Ensure environment and auth details are correct. If it is hosted on pure public cloud (i.e. link to CAS and API server can be called from any network), this is not required.

2. Go into specific demo url, login if necessary. If run from local, it will be localhost:8080. If on pure public cloud, go to the published link.

3. Proceed to demo by clicking on the respective buttons :)

LIVE DEPLOYMENT:

**TBD**  
*Deployment to Cloud Foundry:*

*Please refer to this link for implementation details. As of 05/31/2017, this is still a WIP!*

*github.com/ihuston/python-cf-examples*

(Note: The idea is to deploy easily to cloudfoundry via commandline. If you have used heroku before, the experience is very similar. Think Python buildpacks and integration with your git to push to master.)

*For inquiries, POST them (pun intended) to thomas.thio@hotmail.com*