Kevin Wang

Structure of SC_CPC_workshop

codes/data that can run on your laptop

+

GitHub

≠

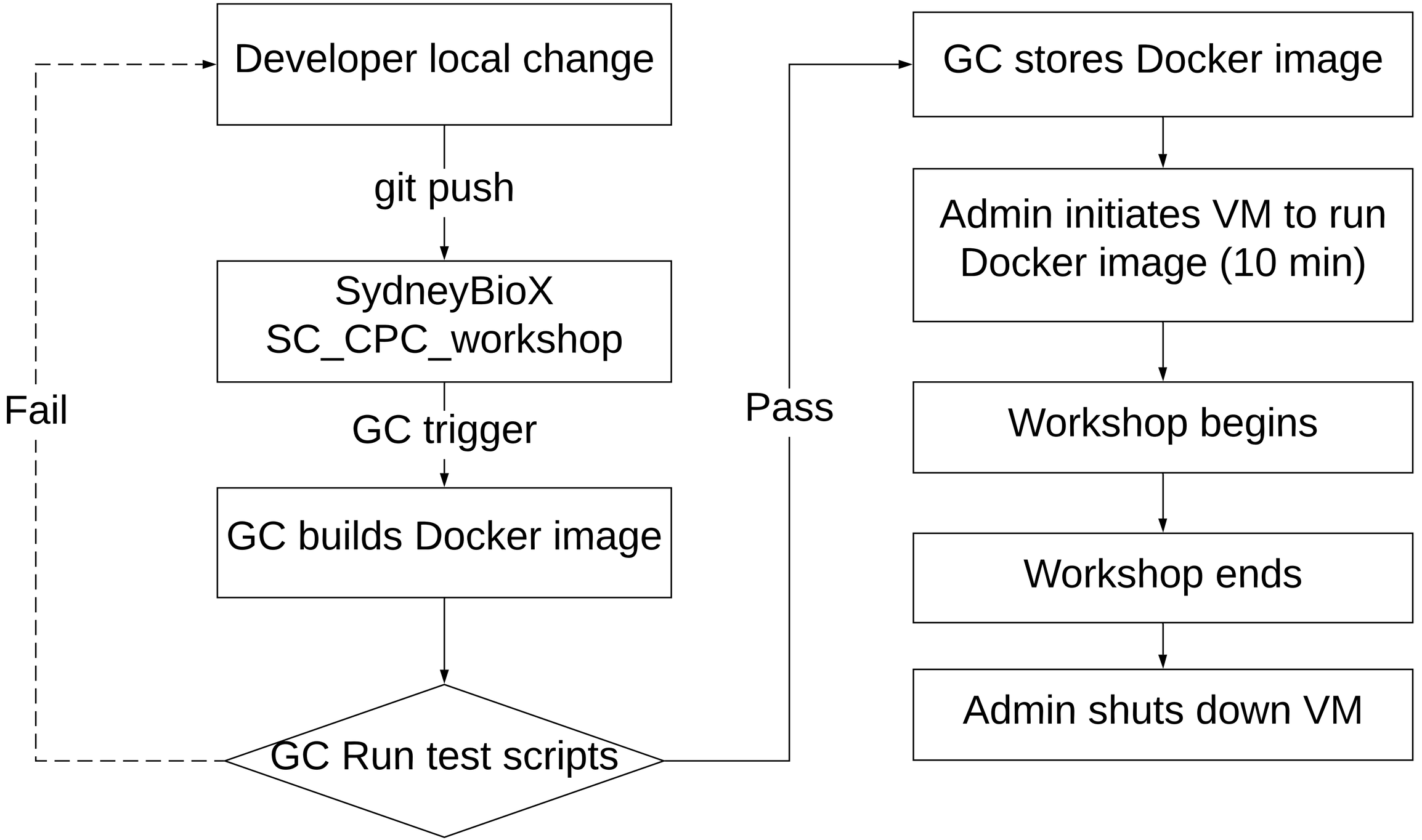
open source

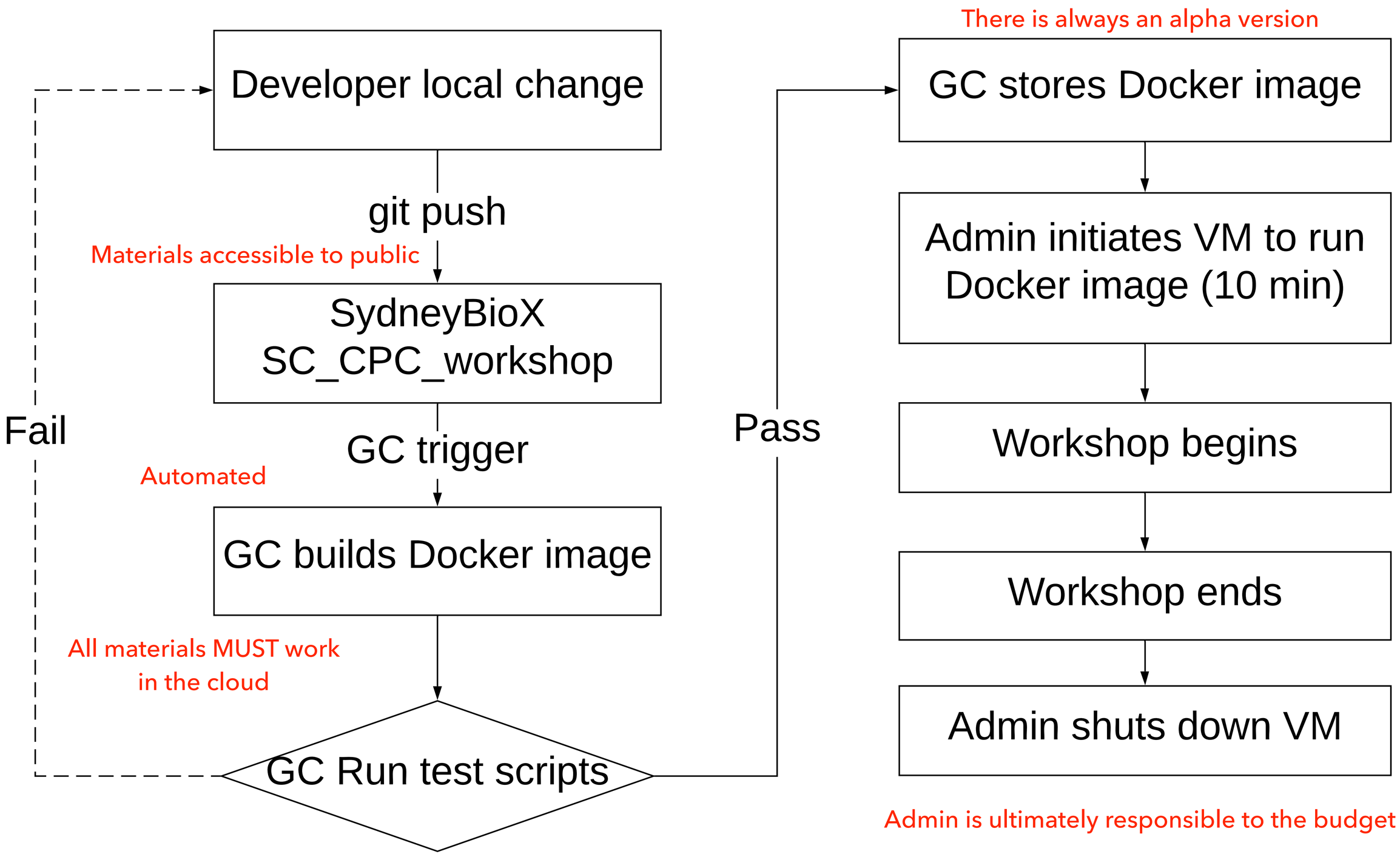
Why Google Cloud?

- June:
  - Developers write materials, compile, push to GitHub. We can only hope that the same code can be ran on another laptop
  - Installing packages took about 1 hour
- Inconsistent folder structures can break the code
- High computational requirements
- Our group got $2,000 from Google. Amazon gave me $30

Google Cloud is a collection of services

- Our GC account currently hosts
  - data.zip (Storage Bucket)
  - Cloud Build triggers
  - Docker images (Container Registry)
  - Virtual machines (VM instances)


Developer local change
git push
SydneyBioX
SC_CPC_workshop
GC trigger
GC builds Docker image
GC Run test scripts
Fail
Pass
GC stores Docker image
Admin initiates VM to run
Docker image (10 min)
Workshop begins
Workshop ends
Admin shuts down VM


There is always an alpha version
Developer local change
git push
Materials accessible to public
SydneyBioX
SC_CPC_workshop
GC trigger
Automated
GC builds Docker image
All materials MUST work
in the cloud
GC Run test scripts
Fail
Pass
GC stores Docker image
Admin initiates VM to run
Docker image (10 min)
Workshop begins
Workshop ends
Admin shuts down VM
Admin is ultimately responsible to the budget

Why this design?

- Clarity: Developers only need to push*, admin only need to deploy
- Portable: Admin can initiate the workshop from a mobile app
- Community: Multiple developers can commit to GitHub and trigger build
- Backup: Admin always have access to a working copy of Docker + website
- Accountable: Admin is solely responsible for deployment and budgeting
- Rigorous: Everything that is available in a Docker image must be functional
- Trackable: Every version is linked to a GitHub commit with timestamp

*Developers also need to build the website

Specific files and functionalities

https://github.com/SydneyBioX/SC_CPC_workshop

- All teaching materials are identical to https://github.com/SydneyBioX/SingleCellPlus, except:
  - `Dockerfile` and other set-up scripts
  - `deployment` folder with all the passwords
  - `.gitignore` (to prevent public access to sensitive information)

Building Docker image

- `Dockerfile` list out build commands:
  - docker pull from Bioconductor Docker image
  - Add and run build scripts into Docker (install.R and docker_setup.sh)
  - Add `docker_test.R`, `user_setup.R` and `omg.R`

`install.R`, `docker_setup.sh` and `docker_test.R`

- `install.R` controls how all R packages should be installed
- `docker_setup.sh` does two things:
  - git pull from SydneyBioX/SC_CPC_workshop
  - download data.zip from https://storage.googleapis.com/scp_data/data.zip
- `docker_test.R` renders the main RMarkdown files to check if they can be successfully compiled

`user_setup.R` and `omg.R`

- When the workshop begins, attendees should type `source("/home/user_setup.R")` into the R console.
- This copies all the files needed to run that workshop.
- Anything goes wrong during the workshop
  - Minor: ask attendees to fix it themselves
  - Major: type `source("/home/omg.R")` into the console (this is the last resort)

Initiating the workshop

- `deployment/password.Rmd` uses
  - number of users
  - number of virtual machines needed

to generate passwords and store those as `deployment/users.csv`. Print and distribute to attendees.

- `deployment/GCE.R` must be ran by a person authorised to `scpworkshop` GC account. Basic configurations (CPU, RAM or HD) are needed.