Leibniz-Institut für
Astrophysik Potsdam

Behind the scenes of a Daiquiri powered archive

Building a DevOp environment

A. Galkin, O. Michaellis / AG Tagung 2022, Bremen / 15. September 2022

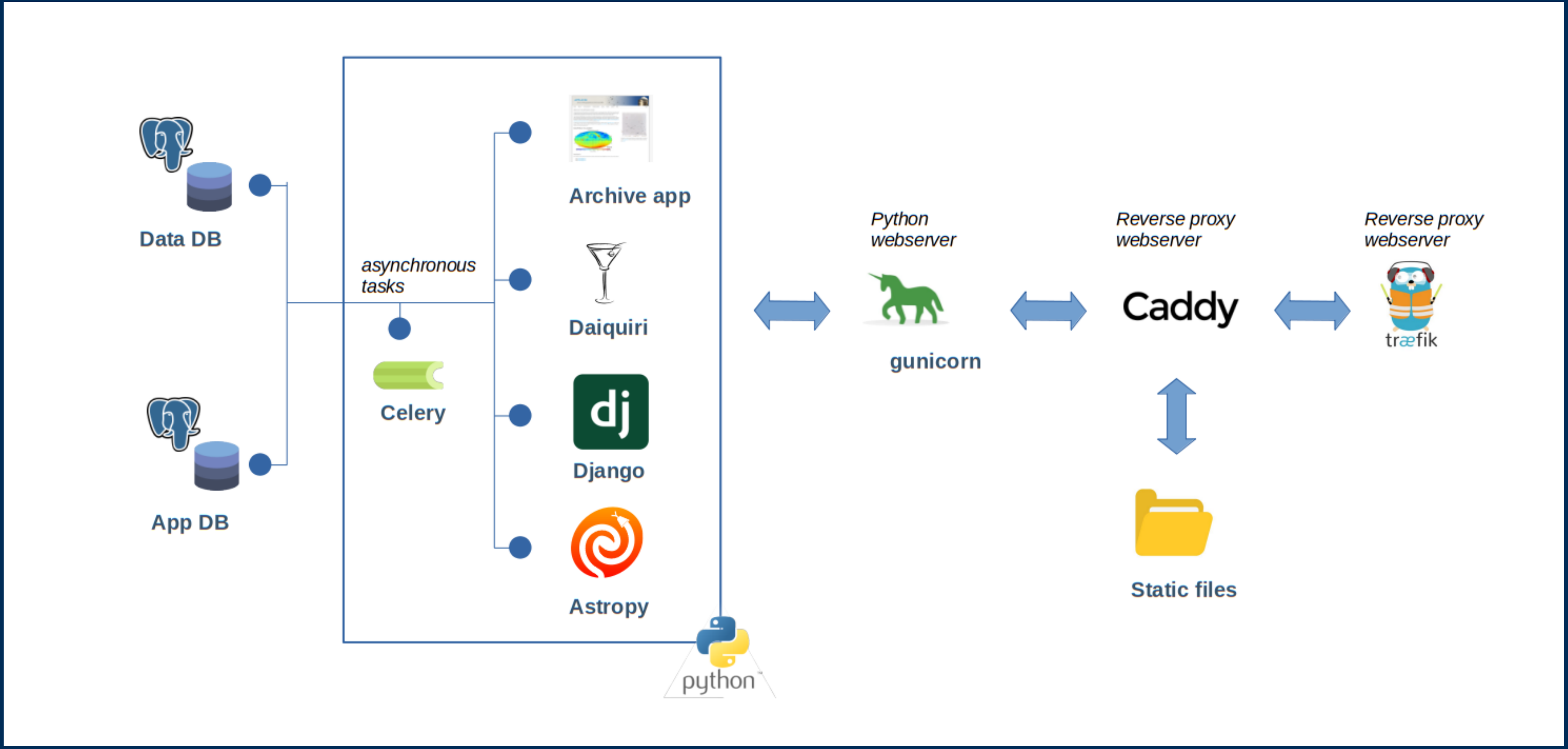
Archive components

Data is the primary focus of an archive. But to deliver it, we need

- Software
  - Web technologies
  - Data ingest and quality control scripts
  - Metadata scripts and tools
  - User management
  - Database technologies (SQL)
  - Data analysis / transformation packages
- Hardware
  - Storage raids
  - DB servers
  - Web servers

A couple of data curators also happen to be very useful. ;)

An archive instance



From development to operation

Development on local machine

→

Development on a dev server:

- test data
- storage

→

Test on the test instance:

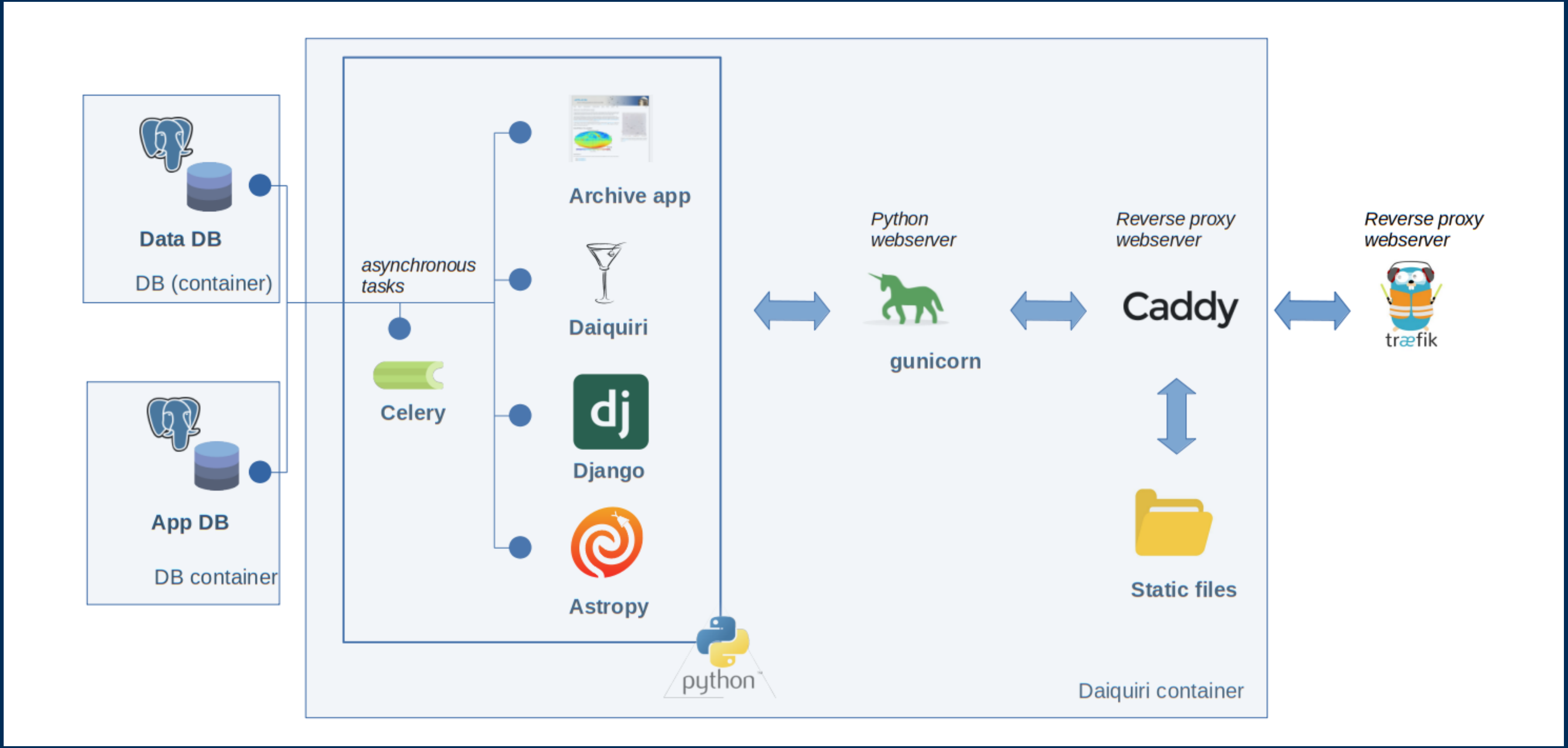
- identical to the prod instance plus the new feature
- Test data
- storage

→

Deployment to the productive instance

Ideally, a developer has the exact same environment as on the production instance.

An archive instance



Setup on the server

On a multi instance system – create user with the storage permissions, needs to have permissions to run docker.

Folder structure - repositories

- App
- Content
- Daiquiri core
- Dq-dev

Prerequisites:

- Docker version 20.10.17
- Docker compose V2
- Python 3.6.8

dq-dev technologies

- Bash scripts
- Python 3

The configuration is stored in the `conf.toml` and `secrets.toml`

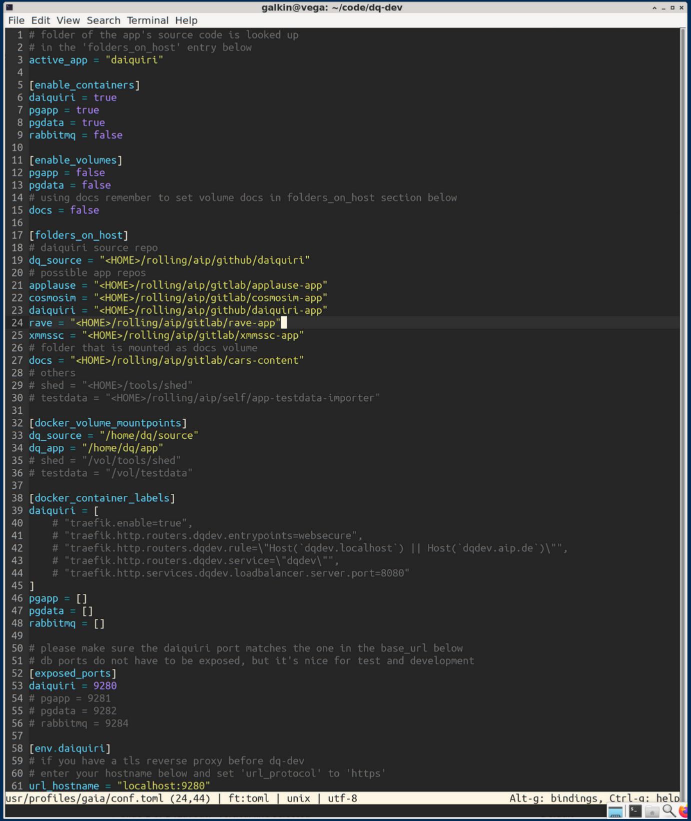
The highs…

- The setup is reproducible. So are the bugs.
- Development, test and deployment setups are the same environment.
- Less downtime
- No more „changing configs“ on the productive instances.
- As many as dev or test instances as needed.
- Additional features for maintanance can be built into the setup – data volumes, custom installs for the app, cron jobs, etc.

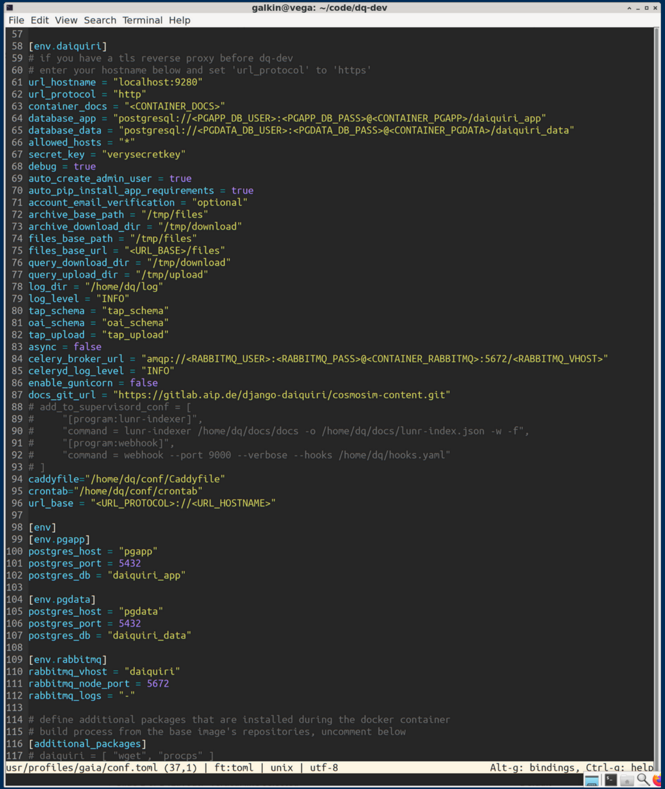
Docker is a very fast developing technology – continious updates are needed, more difficult to deploy on older OS. Same goes for web services.

With docker any updates or combinations of packages and features are possible and can be tested beforehand, as many dev or test instances as needed.

… and the lows ;)



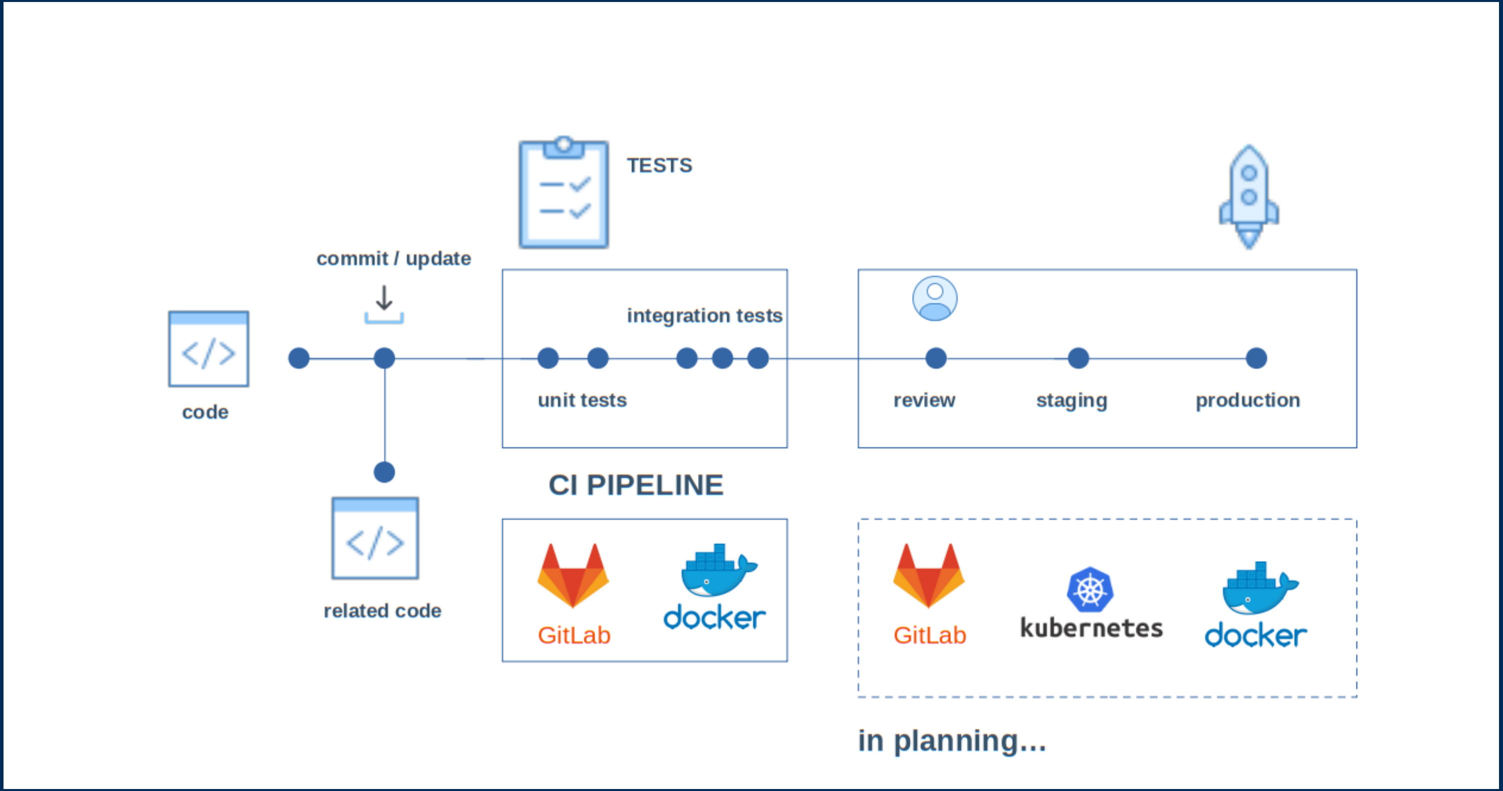
```
galkin@vega: ~/code/dq-dev
File  Edit  View  Search  Terminal  Help
# folder of the app's source code is looked up
# in the 'folders_on_host' entry below
active_app = "daiquiri"

[enable_containers]
daiquiri = true
pgapp = true
pgdata = true
rabbitmq = false

[enable_volumes]
pgapp = false
pgdata = false
# using docs remember to set volume docs in folders_on_host section below
docs = false

[folders_on_host]
# daiquiri source repo
dq_source = "<HOME>/rolling/aip/github/daiquiri"
# possible app repos
applause = "<HOME>/rolling/aip/gitlab/applause-app"
cosmosim = "<HOME>/rolling/aip/gitlab/cosmosim-app"
daiquiri = "<HOME>/rolling/aip/github/daiquiri-app"
rave = "<HOME>/rolling/aip/gitlab/rave-app"
xmmssc = "<HOME>/rolling/aip/gitlab/xmmssc-app"
# folder that is mounted as docs volume
docs = "<HOME>/rolling/aip/gitlab/cars-content"
# others
# shed = "<HOME>/tools/shed"
# testdata = "<HOME>/rolling/aip/self/app-testdata-importer"

[docker_volume_mountpoints]
dq_source = "/home/dq/source"
dq_app = "/home/dq/app"
# shed = "/vol/tools/shed"
# testdata = "/vol/testdata"

[docker_container_labels]
daiquiri = [
    # "traefik.enable=true",
    # "traefik.http.routers.dqdev.entrypoints=websecure",
    # "traefik.http.routers.dqdev.rule=\"Host(`dqdev.localhost`) || Host(`dqdev.aip.de`)\"",
    # "traefik.http.routers.dqdev.service=\"dqdev\"",
    # "traefik.http.services.dqdev.loadbalancer.server.port=8080"
]
pgapp = []
pgdata = []
rabbitmq = []

# please make sure the daiquiri port matches the one in the base_url below
# db ports do not have to be exposed, but it's nice for test and development
[exposed_ports]
daiquiri = 9280
# pgapp = 9281
# pgdata = 9282
# rabbitmq = 9284

[env.daiquiri]
# if you have a tls reverse proxy before dq-dev
# enter your hostname below and set 'url_protocol' to 'https'
url_hostname = "localhost:9280"
usr/profiles/gaia/conf.toml (24,44) | ft:toml | unix | utf-8          Alt-g: bindings, Ctrl-g: help
```



```
galkin@vega: ~/code/dq-dev
File  Edit  View  Search  Terminal  Help

[env.daiquiri]
# if you have a tls reverse proxy before dq-dev
# enter your hostname below and set 'url_protocol' to 'https'
url_hostname = "localhost:9280"
url_protocol = "http"
container_docs = "<CONTAINER_DOCS>"
database_app = "postgresql://<PGAPP_DB_USER>:<PGAPP_DB_PASS>@<CONTAINER_PGAPP>/daiquiri_app"
database_data = "postgresql://<PGDATA_DB_USER>:<PGDATA_DB_PASS>@<CONTAINER_PGDATA>/daiquiri_data"
allowed_hosts = "*"
secret_key = "verysecretkey"
debug = true
auto_create_admin_user = true
auto_pip_install_app_requirements = true
account_email_verification = "optional"
archive_base_path = "/tmp/files"
archive_download_dir = "/tmp/download"
files_base_path = "/tmp/files"
files_base_url = "<URL_BASE>/files"
query_download_dir = "/tmp/download"
query_upload_dir = "/tmp/upload"
log_dir = "/home/dq/log"
log_level = "INFO"
tap_schema = "tap_schema"
oai_schema = "oai_schema"
tap_upload = "tap_upload"
async = false
celery_broker_url = "amqp://<RABBITMQ_USER>:<RABBITMQ_PASS>@<CONTAINER_RABBITMQ>:5672/<RABBITMQ_VHOST>"
celeryd_log_level = "INFO"
enable_gunicorn = false
docs_git_url = "https://gitlab.aip.de/django-daiquiri/cosmosim-content.git"
# add_to_supervisord_conf = [
#     "[program:lunr-indexer]",
#     "command = lunr-indexer /home/dq/docs/docs -o /home/dq/docs/lunr-index.json -w -f",
#     "[program:webhook]",
#     "command = webhook --port 9000 --verbose --hooks /home/dq/hooks.yaml"
# ]
caddyfile="/home/dq/conf/Caddyfile"
crontab="/home/dq/conf/crontab"
url_base = "<URL_PROTOCOL>://<URL_HOSTNAME>"

[env]
[env.pgapp]
postgres_host = "pgapp"
postgres_port = 5432
postgres_db = "daiquiri_app"

[env.pgdata]
postgres_host = "pgdata"
postgres_port = 5432
postgres_db = "daiquiri_data"

[env.rabbitmq]
rabbitmq_vhost = "daiquiri"
rabbitmq_node_port = 5672
rabbitmq_logs = "-"

# define additional packages that are installed during the docker container
# build process from the base image's repositories, uncomment below
[additional_packages]
# daiquiri = [ "wget", "procps" ]
usr/profiles/gaia/conf.toml (37,1) | ft:toml | unix | utf-8          Alt-g: bindings, Ctrl-g: help
```

Contenairization does not mean simplicity. A lot of expertise is needed, but it is completely scripted / documented now.

An ideal DevOp world



Questions?

Anastasia Galkin

agalkin@aip.de

•

github.com/django-daiquiri