Git repos i've worked on
https://gitlab.docking.org/mar/cartblanche22
https://gitlab.docking.org/mar/AMIS
https://gitlab.docking.org/mtsukanov/Bioisostere
https://gitlab.docking.org/mar/tin-manager
https://gitlab.docking.org/mar/ZINC
https://gitlab.docking.org/mar/zinc21
https://github.com/docking-org/zinc22-3d
https://github.com/docking-org/zinc22-2d
https://gitlab.docking.org/mtsukanov/ABBB

USEFUL WIKI LINKS
https://wiki.docking.org/index.php?title=Cartblanche22_Build_Instructions
https://wiki.docking.org/index.php?title=Tin_Manager
https://wiki.docking.org/index.php?title=Init_partitioned_search.py_(aka_Search_zinc22.py)

ZINC22
ZINC22 Manual
Design Changes & Reasoning for zinc25
ben left plans for zinc25

Cartblanche

Cartblanche dev environment

NOTE: I recommend running the dev environment inside the cluster, otherwise you're going to have to manually tunnel every tin machine. To make live changes, set up VS Code SSH (https://code.visualstudio.com/docs/remote/ssh). There is a copy of the current version on epyc2 in /home/s_jcastanon/cartblanche22.

- To connect with vs code, forward the 22 port on the machine that you'd like to use using the following command
- ssh -CL localhost:2222:epyc:22 youruser@portal3.ucsf.bkslab.org

- Connect to vs code, navigate to your home directory in a terminal
- Clone repoGit clone https://gitlab.docking.org/chinzo/cartblanche22
    - Cd cartblanche22
- Install node 16.
    - Then, run 'npm i –force' inside root dir of the project
- Install conda for your user if not yet installed
    - https://docs.anaconda.com/miniconda/miniconda-install/ (look for linux installer)

- Create conda environment with python 3.7, install dependencies
    - conda create -c rdkit -n cartblanche22 rdkit python=3.7

- pip install -r backend/requirements.txt

A .env file is required inside the backend/ folder. Here is a sample file taken from my current dev environment

```
COMMON_DATABASE=postgresql://zincuser:@10.20.1.17:5534/zinc22_common
ZINC22_DATABASE=postgresql+psycopg2://test:@10.20.0.38:5432/zinc22
TIN_DATABASE=postgresql+psycopg2://tinuser:usertin@10.20.1.17:5437/tin
CELERY_BROKER_BACKEND="pyamqp://guest:guest@localhost/"
CELERY_CACHE_BACKEND="db+sqlite:///celery.sqlite"
CELERY_RESULT_BACKEND="redis://localhost:6379/1"
GOOGLE_CLIENT_ID=655763979132-hbv1uk9ipqk3ee60a1v3hmossud56s08.apps.googleusercon
tent.com
GOOGLE_API_KEY=AIzaSyBOgEaWPe4LlZqAzAqxe6MsBNwFl405tGA
SECRET_KEY=you-will-never-guess
SMALLWORLD_MAP_PATH=/nfs/db3/private_smallworld_5th_gen/maps/
SMALLWORLD_PUBLIC_MAP_PATH=/nfs/db3/public_smallworld_5th_gen/maps/
SMALLWORLD_CONFIG_PATH=/nfs/db3/private_smallworld_5th_gen/smallworld.cfg
SMALLWORLD_JAR_PATH=/nfs/exj/smallworld-java
SMALLWORLD_DATABASE=/nfs/db2/smallworld_anon_23Q3
SW_DB_VIRTUAL_PATH=/nfs/db5/swp_6th_gen/anon
SWDIR=/nfs/db5/swp_6th_gen_cb
MAIL_PASSWORD=biksudzabwuolarv
MAIL_USERNAME=cartblanche20
MAIL_SERVER=smtp.gmail.com
ZINC20_DATABASE=postgresql://zincread:readonly@mem2:5433/zinc21
MAIL_PASS=molecule20
```

To start the app, 3 different commands will need to be run in separate terminals::

1. In root folder, **npm start**
2. Inside backend folder, **celery -A cartblanche.run_celery.celery worker -l INFO -n worker3 --concurrency=5 (or more)**
3. Inside backend folder, **gunicorn -b :5000 -w 3 --log-level=DEBUG --access-logfile - --error-logfile - application --timeout 36000 --reload**

The app will refresh on any changes either to the front end or backend.

Front end (React) code can be found inside the src/ folder. Backend (flask) code is inside backend/cartblanche. All api endpoints can be found in these files

- backend/cartblanche/data/carts.py
- backend/cartblanche/data/search.py
- backend/cartblanche/data/tranches.py

These files will often call celery tasks located in

- backend/data/tasks/

Deployment Information

- **Production site runs on three machines (all on port 5068)**
  - **Epyc2**
  - **N-1-18**
  - **n-9-38**

- Useful Commands (Use these in the respective server to debug):
  - **$ docker restart molecule-shopping-cart** (restart the docker container)
  - **$ docker logs molecule-shopping-cart** (view app logs)
  - **$ docker exec -it molecule-shopping-cart /bin/bash** (open a shell inside of the docker container)

- Some troubleshooting tips:
  - Cartblanche relies on other outside services (smallworld, arthor, databases) so it's good to check the status of those before looking into the code.
  - For smallworld, check the status of api calls, or run a sample query.
  - For arthor, check the status of api calls, or run a sample query.
  - For search errors, the docker app logs are very clear as to where the data is being searched.
  - Cartblanche relies on other services for task management. Also check redis and redis-sentinel processes on each of the servers.
    - **$ sudo service redis(or redis-sentinel) status**
    - This should only be a problem if all three are down, but it is useful to check them if something weird is happening. Refer to list above on which servers to check

- Deployment Process
  - Deployment is as simple as pushing changes to the respective branch. A docker image will be created with the new version of the code, which will then replace the current version.
  - To redeploy manually, go to https://gitlab.docking.org/chinzo/molecule-shopping-cart/-/pipelines/new,
  - Then select 'master' and 'run pipeline'

ZINC 15

Setting up ZINC

1. Create conda environment
   ```
   conda create -c rdkit -n zinc15 rdkit python=2.7.18 -y
   ```

   Activate env

2. ```
   conda activate zinc15
   ```
3. ```
   pip install -r requirements.txt
   ```
4. ```
   pip install gevent==22.10.2
   ```
5. ```
   ./boot-dev.sh
   ```
6. Go to port 5000, if 5000 is taken you can change the port in boot-dev.sh

Zinc 21

Setting up ZINC

1. Create conda environment
   ```
   conda create -c rdkit -n zinc21 rdkit=2019.09.3.0 python=3.6.13 -y
   ```

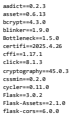
   Activate env

2. ```
   conda activate zinc21
   ```
3. ```
   pip install -r requirements.txt
   ```
4. ```
   ./boot-dev.sh
   ```
5. Go to port 5000, if 5000 is taken you can change the port in boot-dev.sh

Tin Manager

Running Tin Manager

- Tin manager runs in two separate screens, one for the front end and one for the backend.
- Install the requirements.txt using pip for the backend.
  - For me the existing req txt didn't work this is what I used - Kyle


   ```

```
Flask-SQLAlchemy==3.0.2
fonttools==4.38.0
globre==0.1.5
greenlet==2.0.1
importlib-metadata==6.0.0
itsdangerous==2.1.2
Jinja2==3.1.2
jsmin==3.0.1
kiwisolver==1.4.4
lessc==0.1.3
lesscpy==0.15.1
MarkupSafe==2.1.1
matplotlib==3.5.3
numexpr==2.10.2
numpy==2.0.2
packaging==25.0
pandas==1.3.5
paramiko==3.5.1
Pillow==9.3.0
ply==3.11
psycopg2==2.9.5
pycparser==2.22
PyNaCl==1.5.0
pyparsing==3.0.9
python-dateutil==2.9.0.post0
python-dotenv==0.21.0
pytz==2025.2
six==1.17.0
SQLAlchemy==1.4.46
typing_extensions==4.4.0
urllib3==1.26.14
webassets==2.0
Werkzeug==3.0.1
WTForms==3.0.1
zipp==3.11.0
```

- To run the app, run two separate commands in two separate screens

  In one screen in the tin-manager-frontend folder:
  - $ `npm install`
  - $ `npm start`

  Then in another:
  - If you don't have an env:
    - $ `conda create -n tin_manager python=3.9`
  - $ `conda activate tin_manager`
  - $ `flask run`
- This app still needs to be dockerized for use with gitlab deployment

AMIS

https://amis.docking.org/

- tiling scripts in git repo slice.py
- run the same way as cartblanche with separate gunicorn/react server
- Images stored in
  - /nfs/mammoth/idg-images
  - /nfs/exj/idg-images