Design Review III: Fossilized Controller

Fossilized Containers: Emily Ramirez Serrano, Jeremy Klein, Jadon Fowler, and Mumbi Mbuthia

Background

- Paleoclimatology is the study of past climates
- Climate Reconstructions show models of climate over time
- Combating climate change
  - Better understand the past for the future

Nicholas McKay
Associate Professor
Paleoclimate Dynamics Laboratory

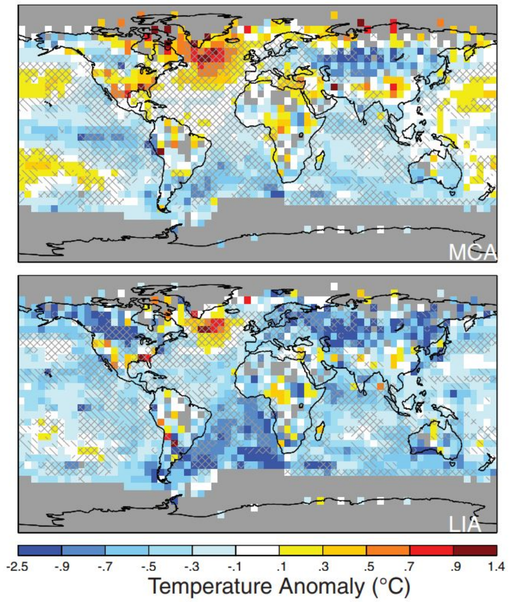


Figure: Paleoclimate Reconstruction

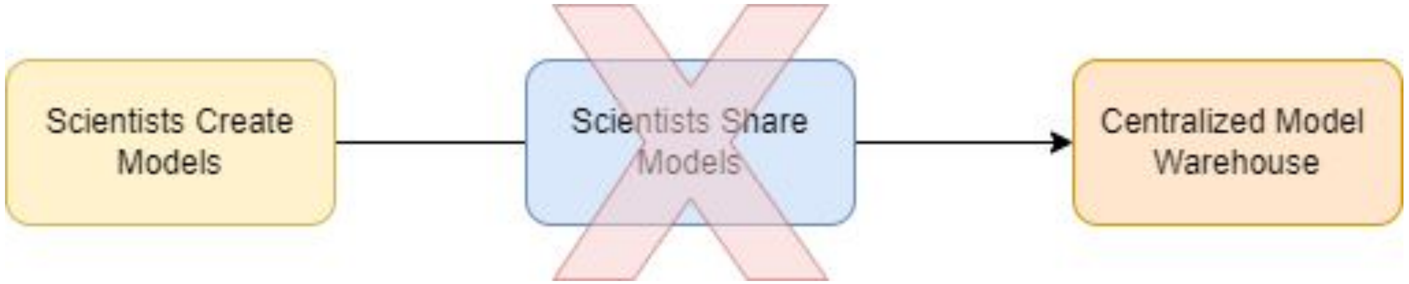
Problem Statement

No centralized storehouse for climate reconstruction models

- No standardization for submissions

No easy way to share climate reconstruction models

- Containerization
    - Scientists do not know how to containerize reconstruction models



Workflow Diagram

Solution Overview

PReSto

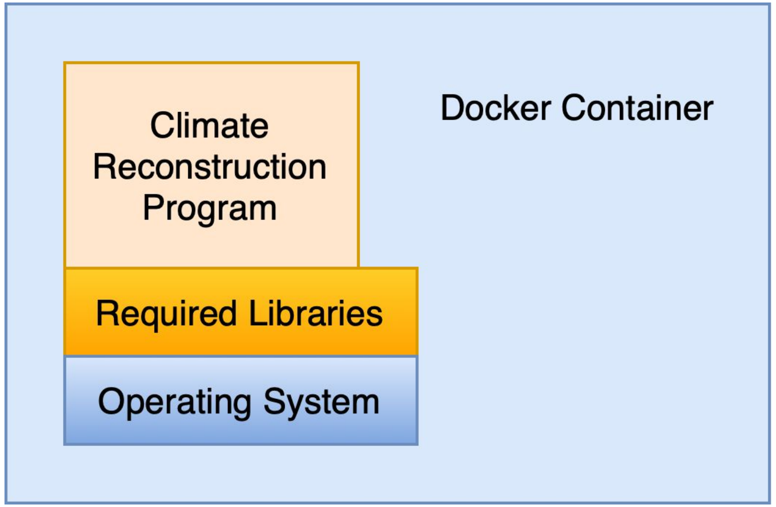
- Paleoclimate Reconstruction Storehouse

Docker Container

- Removes assumptions of installed libraries and operating system

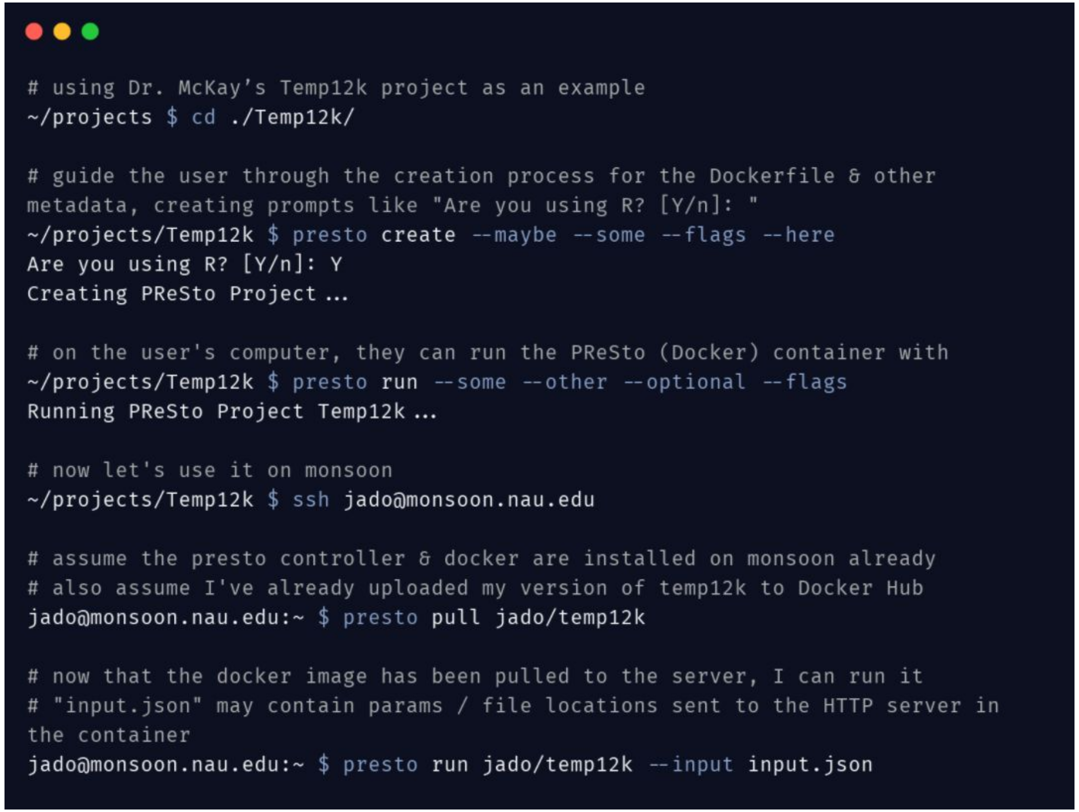
Fossilized Controller

- User Interface for building containers



Requirements/ Specs Review

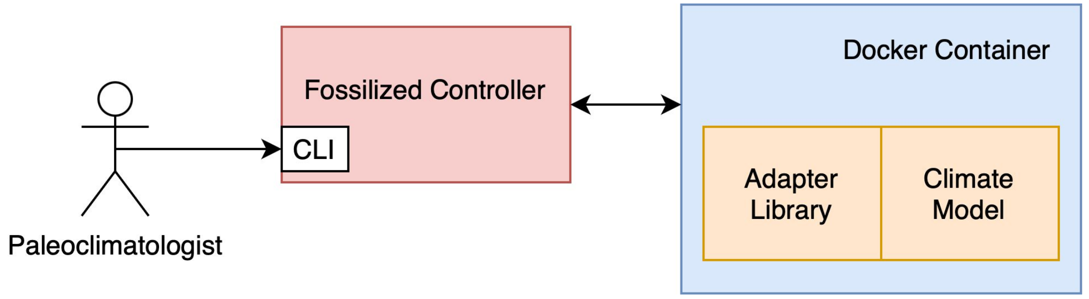
- *Simple* to containerize and run climate models
- Language Agnostic
- Quick integration for projects written in Python and R



```
# using Dr. McKay's Temp12k project as an example
~/projects $ cd ./Temp12k/

# guide the user through the creation process for the Dockerfile & other
metadata, creating prompts like "Are you using R? [Y/n]: "
~/projects/Temp12k $ presto create --maybe --some --flags --here
Are you using R? [Y/n]: Y
Creating PReSto Project ...

# on the user's computer, they can run the PReSto (Docker) container with
~/projects/Temp12k $ presto run --some --other --optional --flags
Running PReSto Project Temp12k ...

# now let's use it on monsoon
~/projects/Temp12k $ ssh jado@monsoon.nau.edu

# assume the presto controller & docker are installed on monsoon already
# also assume I've already uploaded my version of temp12k to Docker Hub
jado@monsoon.nau.edu:~ $ presto pull jado/temp12k

# now that the docker image has been pulled to the server, I can run it
# "input.json" may contain params / file locations sent to the HTTP server in
the container
jado@monsoon.nau.edu:~ $ presto run jado/temp12k --input input.json
```

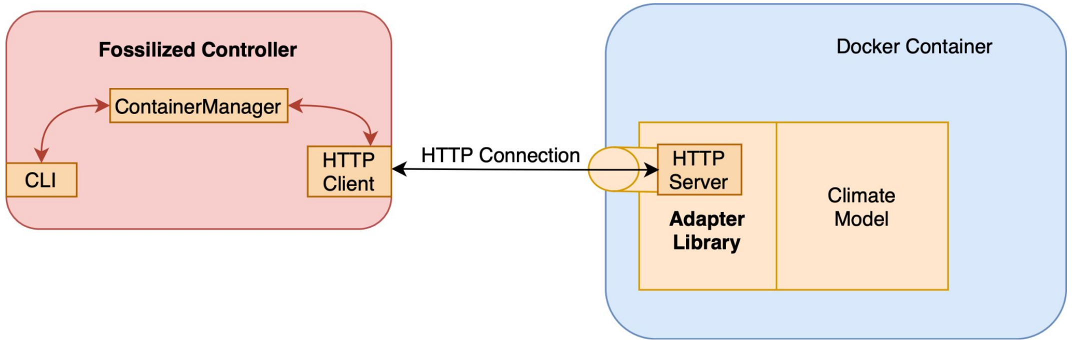
Architecture Overview

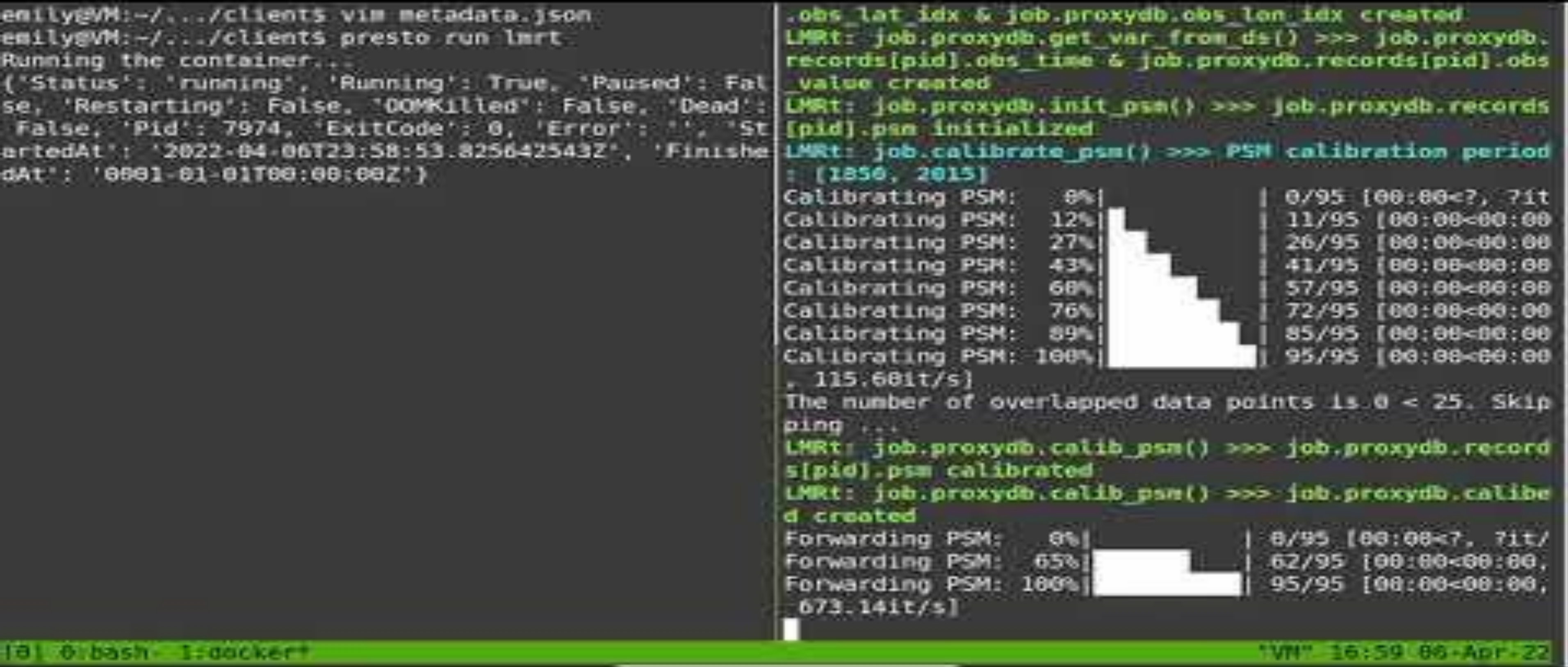
- Fossilized Controller contains Command Line Interface
- Controller communicates with Docker Containers
- Adapter Library for Scientists to use



Implementation Review

- Fossilized Controller is written in Python, using *Click* and *Docker SDK*
- Controller & containers connect over HTTP
- Adapter Libraries are written in Python and R





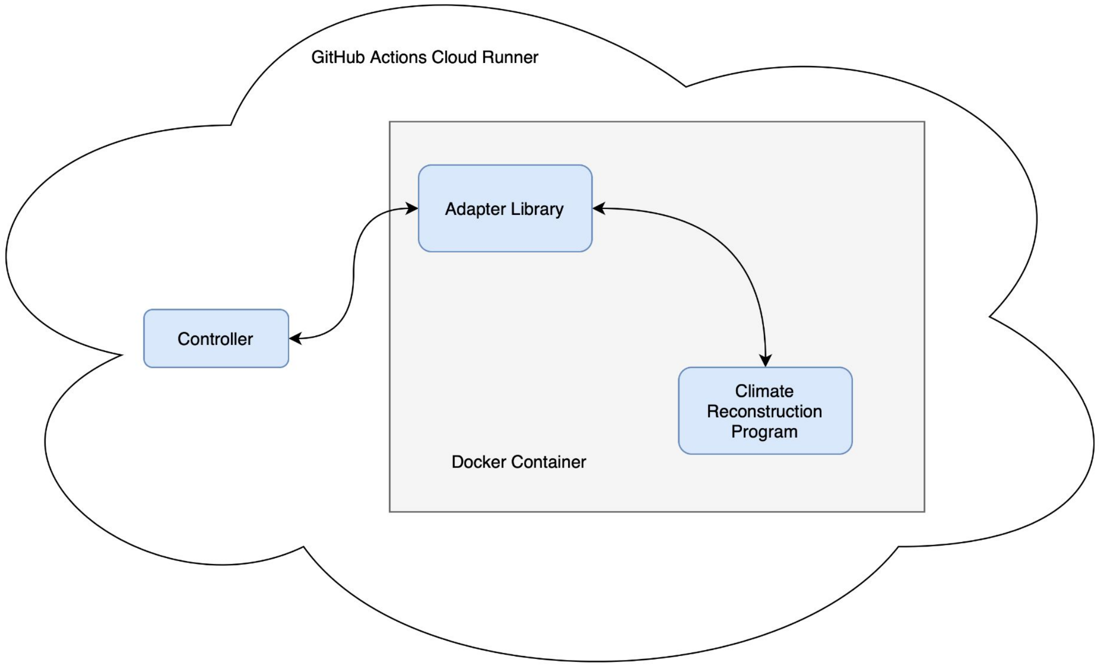
```
emily@VM:~/.../clients$ vim metadata.json
emily@VM:~/.../clients$ presto run lmrt
Running the container...
{'Status': 'running', 'Running': True, 'Paused': Fal
se, 'Restarting': False, 'OOMKilled': False, 'Dead':
 False, 'Pid': 7974, 'ExitCode': 0, 'Error': '', 'St
artedAt': '2022-04-06T23:58:53.825642543Z', 'Finishe
dAt': '0001-01-01T00:00:00Z'}
```

```
.obs_lat_idx & job.proxydb.obs_lon_idx created
LMRt: job.proxydb.get_var_from_ds() >>> job.proxydb.
records[pid].obs_time & job.proxydb.records[pid].obs
_value created
LMRt: job.proxydb.init_psm() >>> job.proxydb.records
[pid].psm initialized
LMRt: job.calibrate_psm() >>> PSM calibration period
: [1850, 2015]
Calibrating PSM:   0%|          | 0/95 [00:00<?, ?it
Calibrating PSM:  12%|█         | 11/95 [00:00<00:00
Calibrating PSM:  27%|██▋       | 26/95 [00:00<00:00
Calibrating PSM:  43%|████▎     | 41/95 [00:00<00:00
Calibrating PSM:  60%|██████    | 57/95 [00:00<00:00
Calibrating PSM:  76%|███████▌  | 72/95 [00:00<00:00
Calibrating PSM:  89%|████████▉ | 85/95 [00:00<00:00
Calibrating PSM: 100%|██████████| 95/95 [00:00<00:00
, 115.60it/s]
The number of overlapped data points is 0 < 25. Skip
ping ...
LMRt: job.proxydb.calib_psm() >>> job.proxydb.record
s[pid].psm calibrated
LMRt: job.proxydb.calib_psm() >>> job.proxydb.calibe
d created
Forwarding PSM:   0%|          | 0/95 [00:00<?, ?it/
Forwarding PSM:  65%|██████▌   | 62/95 [00:00<00:00,
Forwarding PSM: 100%|██████████| 95/95 [00:00<00:00,
 673.14it/s]
```

```
[0] 0:bash- 1:docker*                                    "VM" 16:59 06-Apr-22
```

Testing Plan

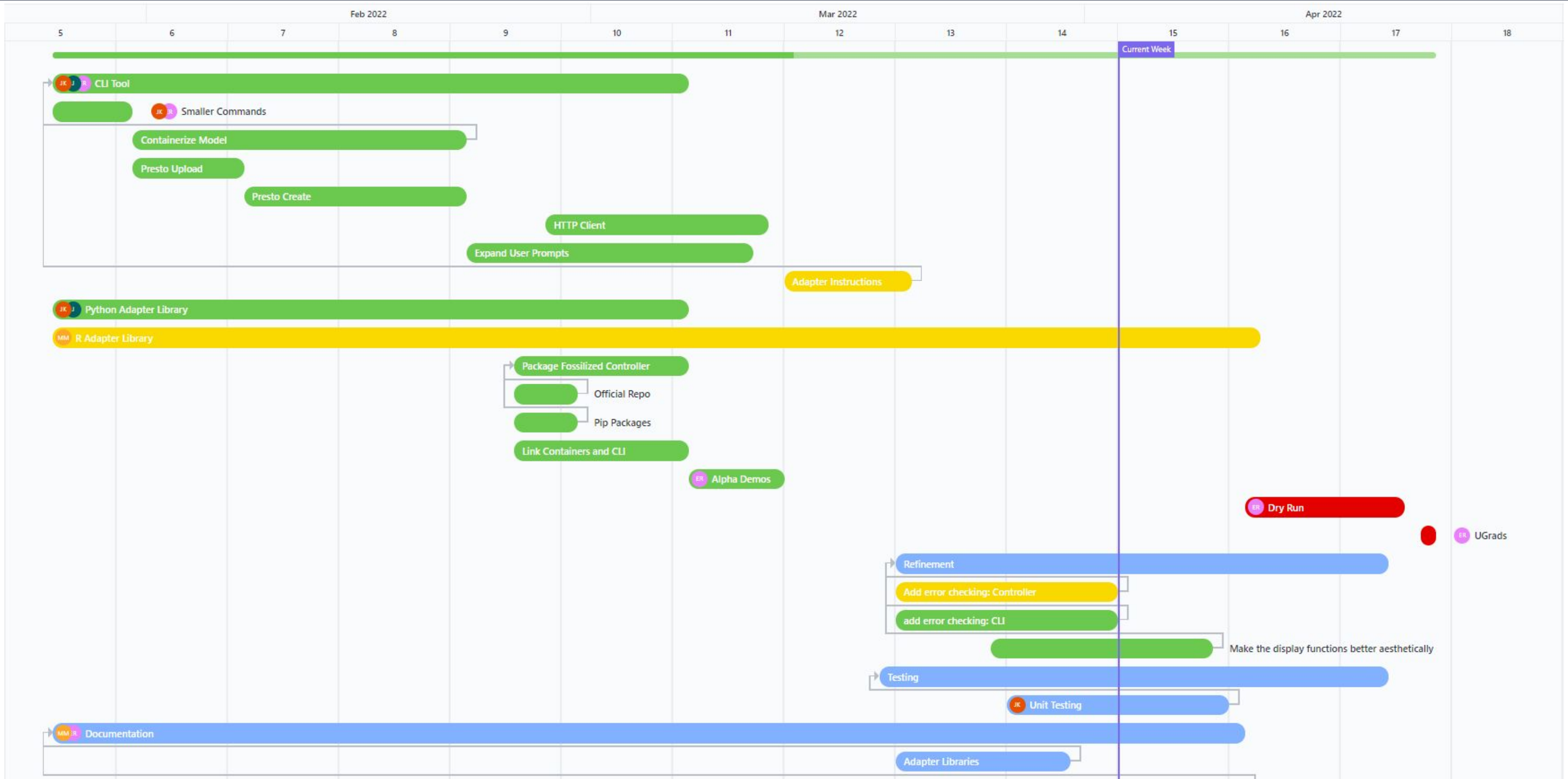
- **Unit Tests** for Controller and Adapter Libraries
- **Integration Tests** run full climate models
- **Usability Tests** of prototyped Command Line Interface



Challenges and Resolutions

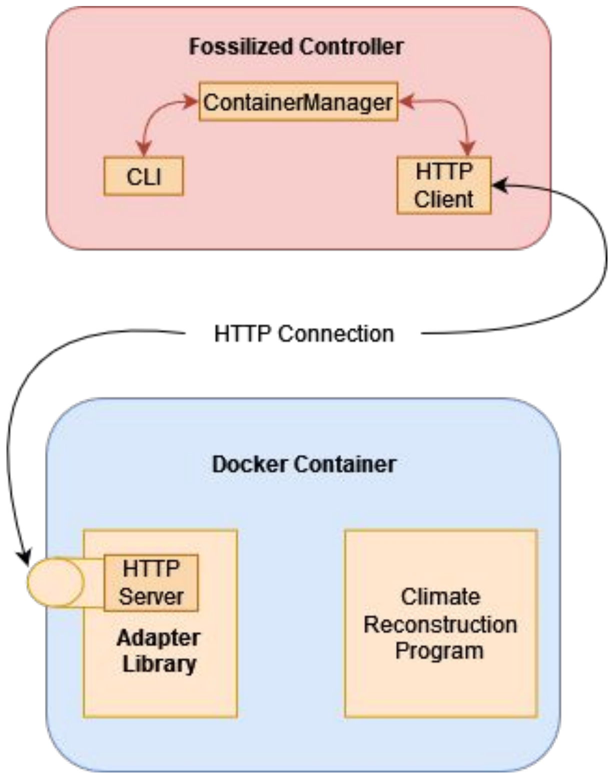
Risk	Difficulty	Solution
Live Logs in the CLI From A Docker Container	**High**	**For Building An Image, Return Old Logs; For Running A Container, Access Separate Terminal**
R Adapter Implementation	**Moderate**	**Research and Preparation**
Amount of Public, Testable Reconstructions	**Low**	**Thorough Usability Testing**

Schedule


Feb 2022
Mar 2022
Apr 2022
5
6
7
8
9
10
11
12
13
14
15
16
17
18
Current Week
CLI Tool
Smaller Commands
Containerize Model
Presto Upload
Presto Create
HTTP Client
Expand User Prompts
Adapter Instructions
Python Adapter Library
R Adapter Library
Package Fossilized Controller
Official Repo
Pip Packages
Link Containers and CLI
Alpha Demos
Dry Run
UGrads
Refinement
Add error checking: Controller
add error checking: CLI
Make the display functions better aesthetically
Testing
Unit Testing
Documentation
Adapter Libraries

Conclusion

- Paleoclimatology and model sharing
- Problem and Solution Overview
- Where We Are Now
- Challenges and Schedule
- Next:
  - Integrating our modules to work as one product
  - Refine and test modules
  - Update documentation



Thank You