Containerized Web Processing Services - Why you want that (and why not)

Nils Brinckmann[1], Matthias Rüster[1], Massimiliano Pittore[1,2], Benjamin Proß[3]

1: GFZ German Research Centre for Geosciences, Potsdam, Germany
2: Eurac Research, Bolzano, Italy
3: 52°North, Münster, Germany

Geospatial Sensing Virtual 2020, 1 September

Outline

What are containerized web processing services?

We need to talk about:

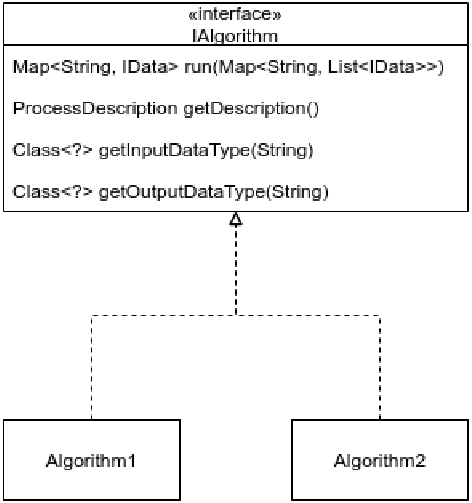
- Web Processing Services
- Container
- Putting it together

All in the context of the RIESGOS project.

Web Processing Services

- Interface for arbitrary remote processing computations
- OGC Standard
- Currently version 2.0
- Operations: GetCapabilities, DescribeProcess, Execute, GetStatus, GetResult
- In the future: OGC API Processes

In the Java World


«interface»
IAlgorithm
Map<String, IData> run(Map<String, List<IData>>)
ProcessDescription getDescription()
Class<?> getInputDataType(String)
Class<?> getOutputDataType(String)
Algorithm1
Algorithm2

Container

- Lightweight virtual machines
- Most prominent: Docker
- Others as well (Singularity, LXD, FreeBSD Jails)

Common use cases for docker

- DB in one container, web server in another
- One container per microservice

Rather uncommon:

- Run pdflatex for beamer template in a container due to missing dependencies and broken package manager on host system :-)

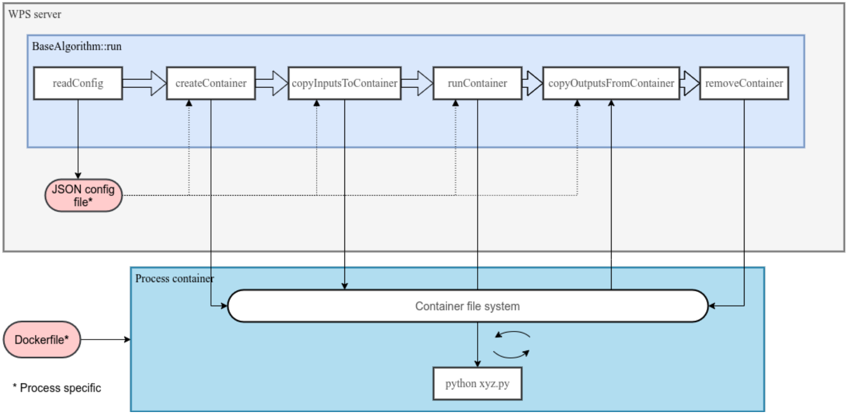
What have we done differently?

- Server in one image
- Each service ifself in its own image
- Similar to one image per post api route
- One generic base process initialized by configurations

Why did we do this? (Aka Benefits)

- To learn how docker works
- Several **real** reasons

Architecture



Docker images

- Existing images for all possible kinds of processes
- Service integrity (docker image IDs)
- Dependency management (Dockerfiles)

Docker file system

- Layered
- Copy on write
- Temporary file system per container
- docker rm $container $\rightarrow$ garbage collection

Encapsulation

- One generic base process maintained by us
- Scientists wrote code for command line applications
- Support on writing dockerfile & configurations by us

$\rightarrow$ Code can run as WPS

- Any programming language can be supported

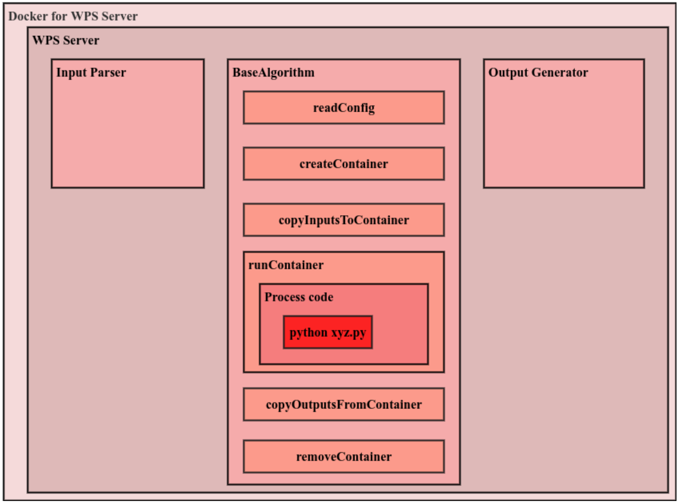
Problems

Docker isn't the silver bullet.
We have several drawbacks that must be mentioned.

Complexity & Lasagna Code

- One generic base process for all kind of computations is hard
- Erros are hard to find (is it in the server? The base process? The scientists code? The IO between them? ...)

Complexity & Lasagna Code



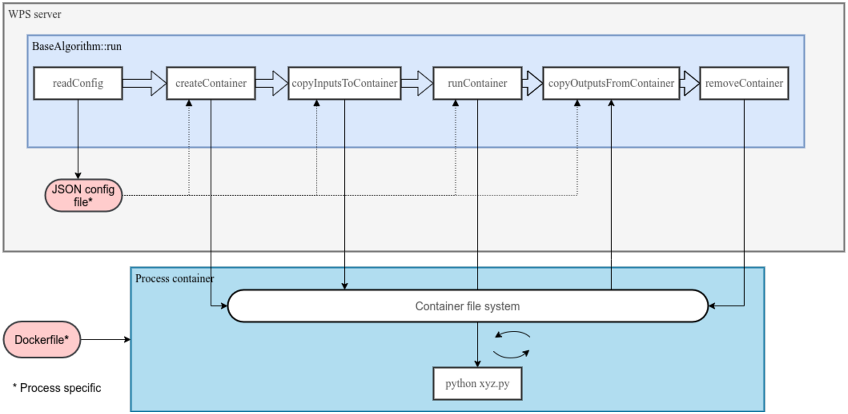
Debugging

Brian W. Kernighan:
Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it.

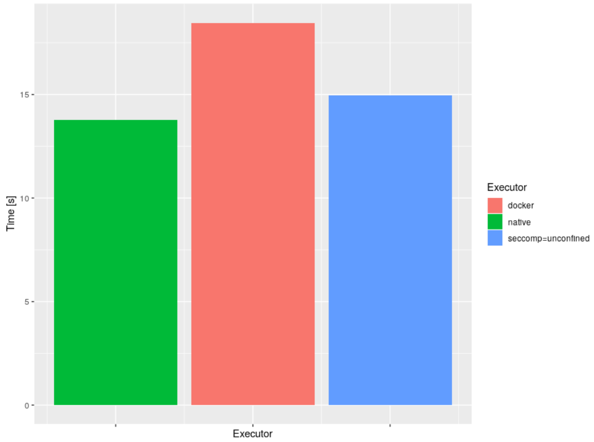
Performance

- More IO to get the data from the base process into the container
- Docker overhead itself

Architecture



Performance - How bad is it?



- Docker adds an overhead of ≈ 30 - 50 %
- With docker security option *seccomp=unconfined* only ≈ 8 %

What to do?

For complexity & too many layers:

- Server & base process layers are always used
  $\rightarrow$ identifiying errors faster
  $\rightarrow$ improving stability
- Test driven development on scientific code
- Integration tests
- Monitoring & logging

What to do?

For performance:

- Approach with volumes (52°North implementation)
- Plans to tests with Singularity or native runners
- Check task granularity & parallel processing

Future developments

- Migrating to the javaPS
- Support for OGC API Processes
- Test Singularity & native runners
- Improve composibility (improved configuration files & event driven architecture)

Thank you for your attention

Are there any question?