Hydro

SHOP: Microsoft interface

Nils Flaten Ræder

Hell, 2019-03-13

What we want (SHOP to do)

- We want …
  - SHOP to run in a way that will facilitate increased automation of the production planning (and re-planning) processes

- Project to test SHOP in the cloud

- We need …
  - Fresh results available at all times
  - Automatic detection and fixing of (potential) errors in inputs and results, and metering of result quality
  - To run multiple scenarios for each model, both for inputs, and permutations of running units
  - To run in an environment where the number of parallel optimizations doesn’t effect calculation time
  - Easy access to results and process status
  - Manually trigger and adjusting model runs
  - A robust test environment, and continuous deployments

Source

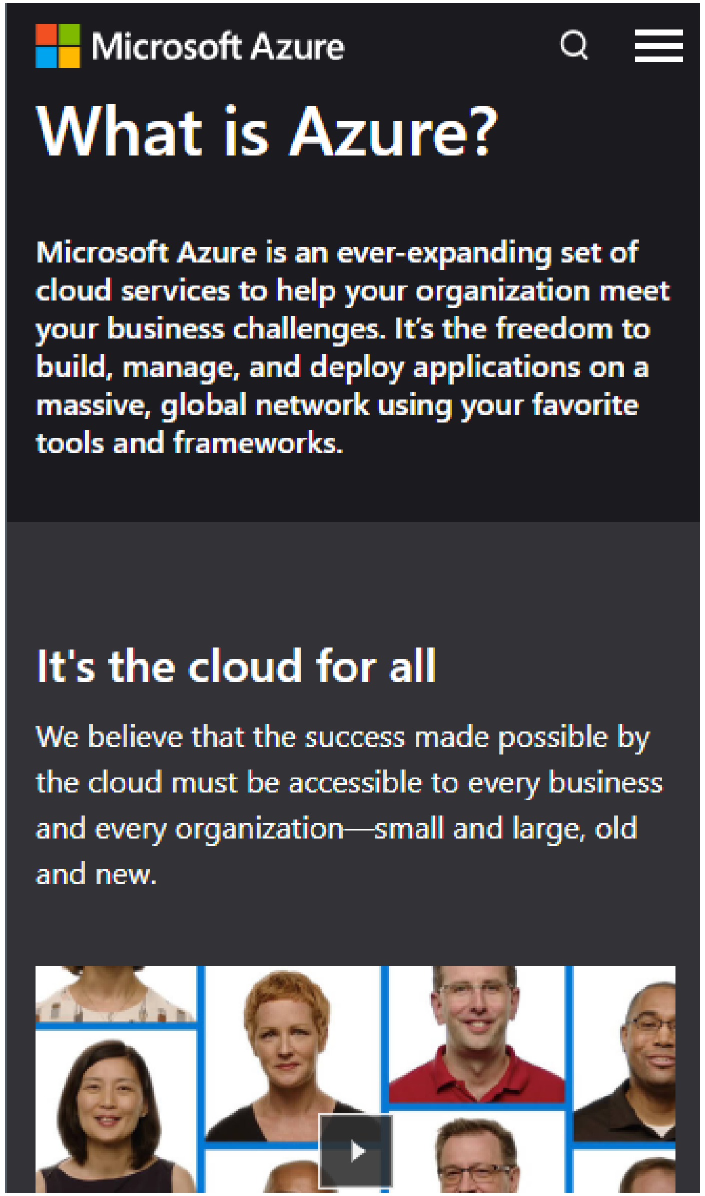
Running SHOP in Microsoft Azure

- Finished a pilot project to test feasibility of running SHOP in the cloud Autumn 2018

- Foreseen benefits of the cloud were mostly confirmed
  - On-demand computing power and storage
  - Parallelization without impacting calculation time
  - Potentially easy testing and deployment
  - No/limited on-premise infrastructure, and easy monitoring of infrastructure costs

- Started by running PyShop on a virtual server in Azure
  → (extremely) easy, but no benefits

- Need to use cloud specific services



What is Azure?

Microsoft Azure is an ever-expanding set of cloud services to help your organization meet your business challenges. It's the freedom to build, manage, and deploy applications on a massive, global network using your favorite tools and frameworks.

It's the cloud for all

We believe that the success made possible by the cloud must be accessible to every business and every organization—small and large, old and new.

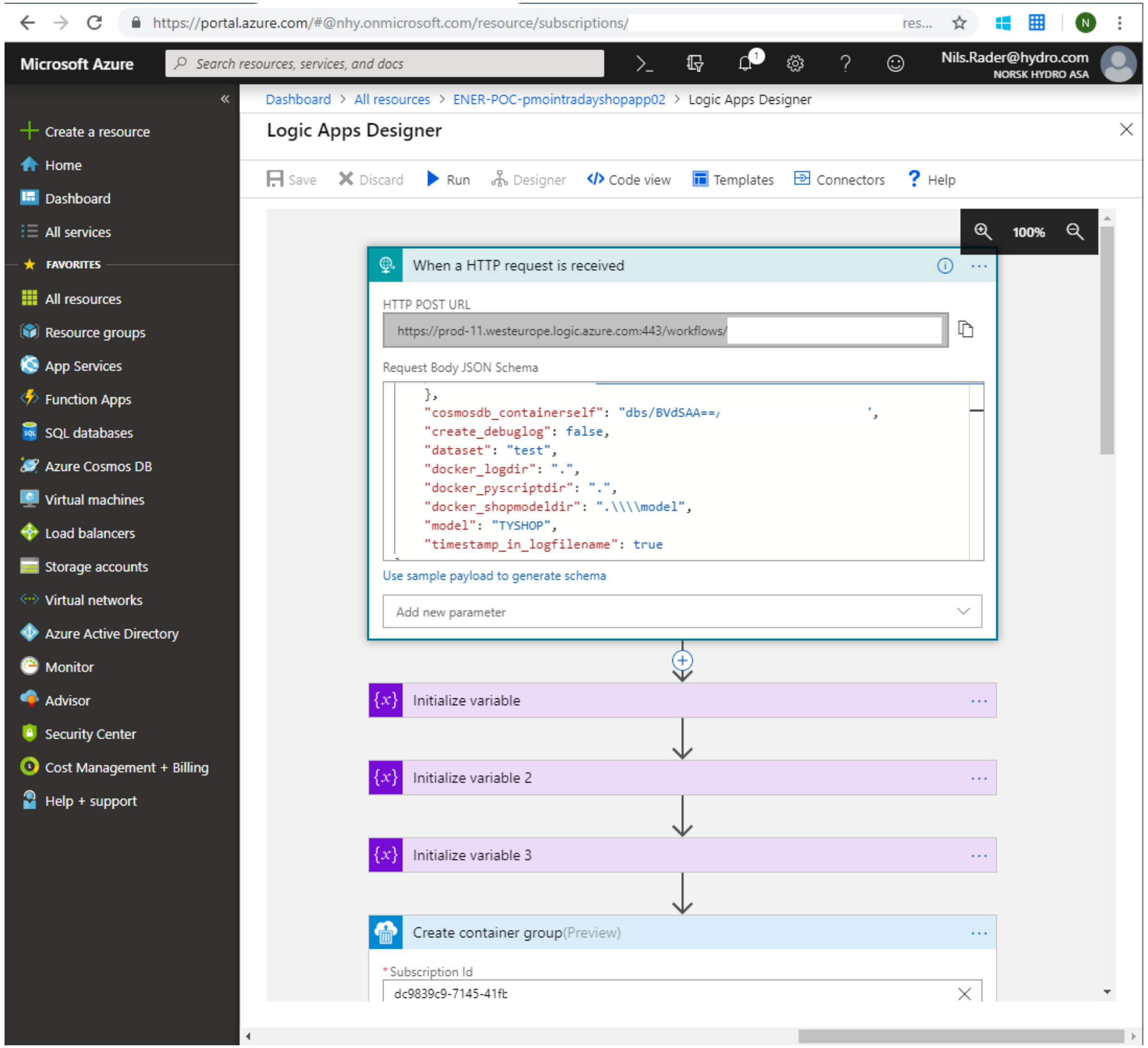
Source: https://azure.microsoft.com/en-us/overview/what-is-azure/

Source

Running SHOP in Microsoft Azure

- Cloud specific services include *serverless* and *stateless* functions, databases and container services

- Cloud services lend themselves to easy partitioning of functionality
  - Function apps for starting/stopping/moving etc.
  - Repository for code
  - Serverless storage
  - Webapps for user interface
  - Container instances for calculations



When a HTTP request is received

HTTP POST URL: https://prod-11.westeurope.logic.azure.com:443/workflows/

Request Body JSON Schema

```
    },
    "cosmosdb_containerself": "dbs/BVdSAA==/                    ',
    "create_debuglog": false,
    "dataset": "test",
    "docker_logdir": ".",
    "docker_pyscriptdir": ".",
    "docker_shopmodeldir": ".\\\\model",
    "model": "TYSHOP",
    "timestamp_in_logfilename": true
```

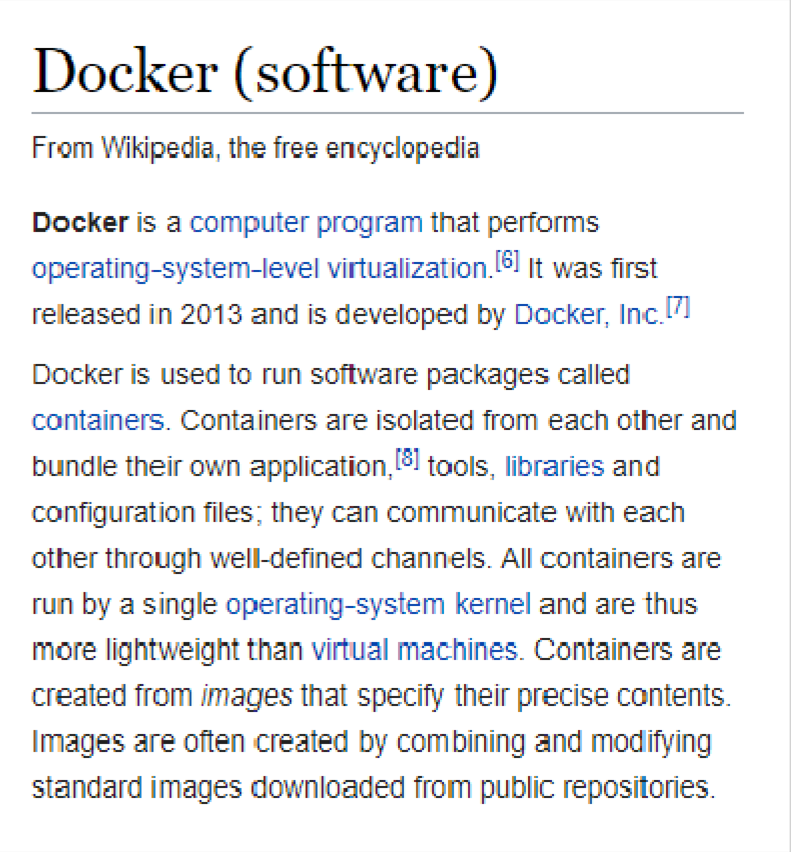
*Subscription Id: dc9839c9-7145-41fc

Source

Running SHOP in Microsoft Azure

- Containers are one of two key components
  - Mini virtual servers created from pre-defined images
  - Each started for a single SHOP optimization, then killed
  - Parallelization is just a matter of starting multiple container instances

- Took some time to make it work
  - We were initially unaware of SHOP and PyShop dependencies not available in the standard Windows images
  - Installing dependencies massively inflated the size of the docker images
  - Start-up time of minutes for a calculation time of 30 seconds
  - Good help from Sintef in reducing size of images, new dependencies mostly solve the size problem
  - A Linux version of SHOP would further speed up the system



Docker (software)

From Wikipedia, the free encyclopedia

Docker is a computer program that performs operating-system-level virtualization.[6] It was first released in 2013 and is developed by Docker, Inc.[7]

Docker is used to run software packages called containers. Containers are isolated from each other and bundle their own application,[8] tools, libraries and configuration files; they can communicate with each other through well-defined channels. All containers are run by a single operating-system kernel and are thus more lightweight than virtual machines. Containers are created from *images* that specify their precise contents. Images are often created by combining and modifying standard images downloaded from public repositories.

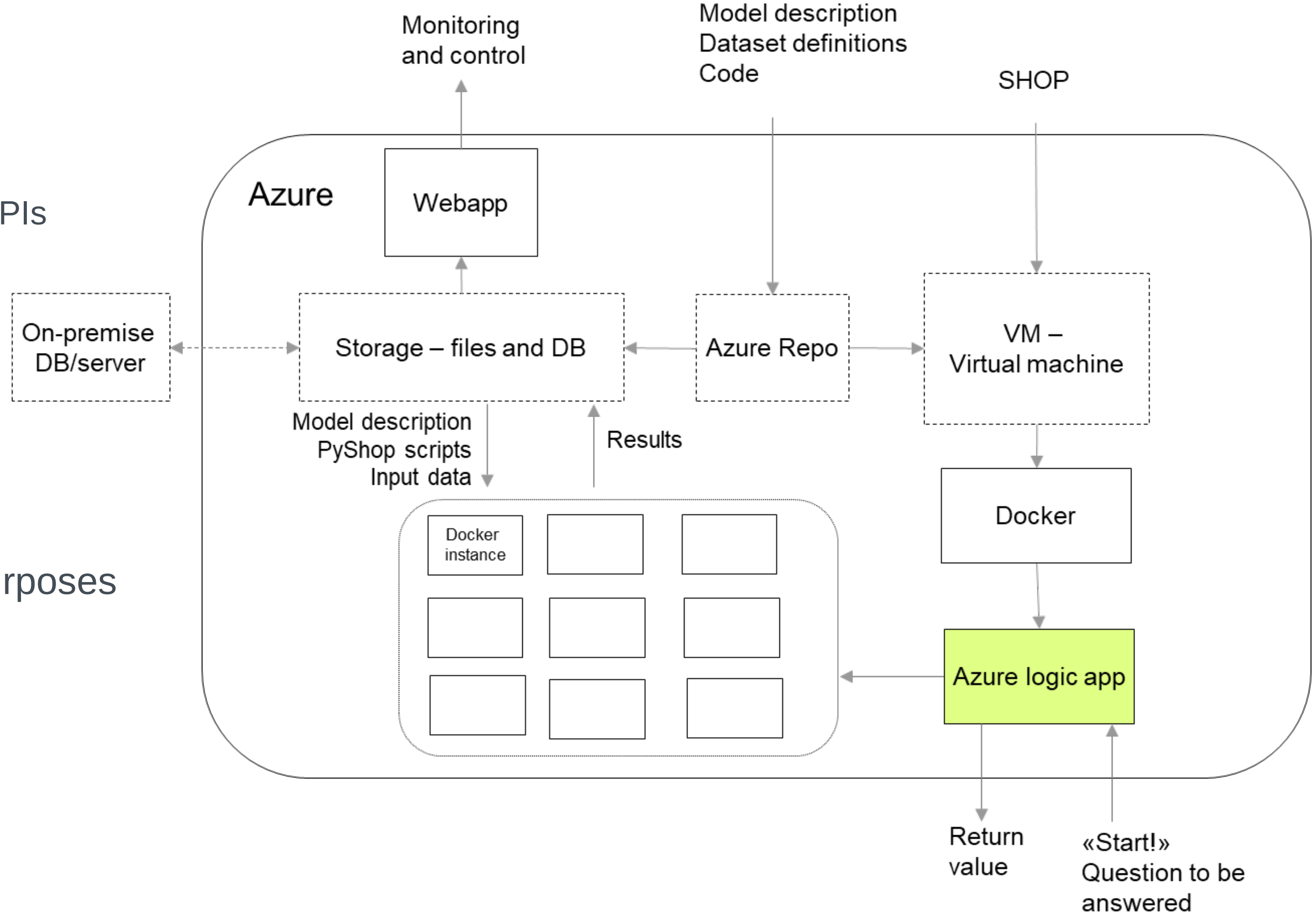
Source

Running SHOP in Microsoft Azure

- Services loosely connected by APIs
  - Also accessible from on-premise PCs using Azure APIs
  - API payloads as JSON

- Storage of all data from each optimization
  - All results, all logs, all inputs, full model description
  - Possible to recreate any results if errors detected

- Possible to duplicate entire set-up for testing purposes

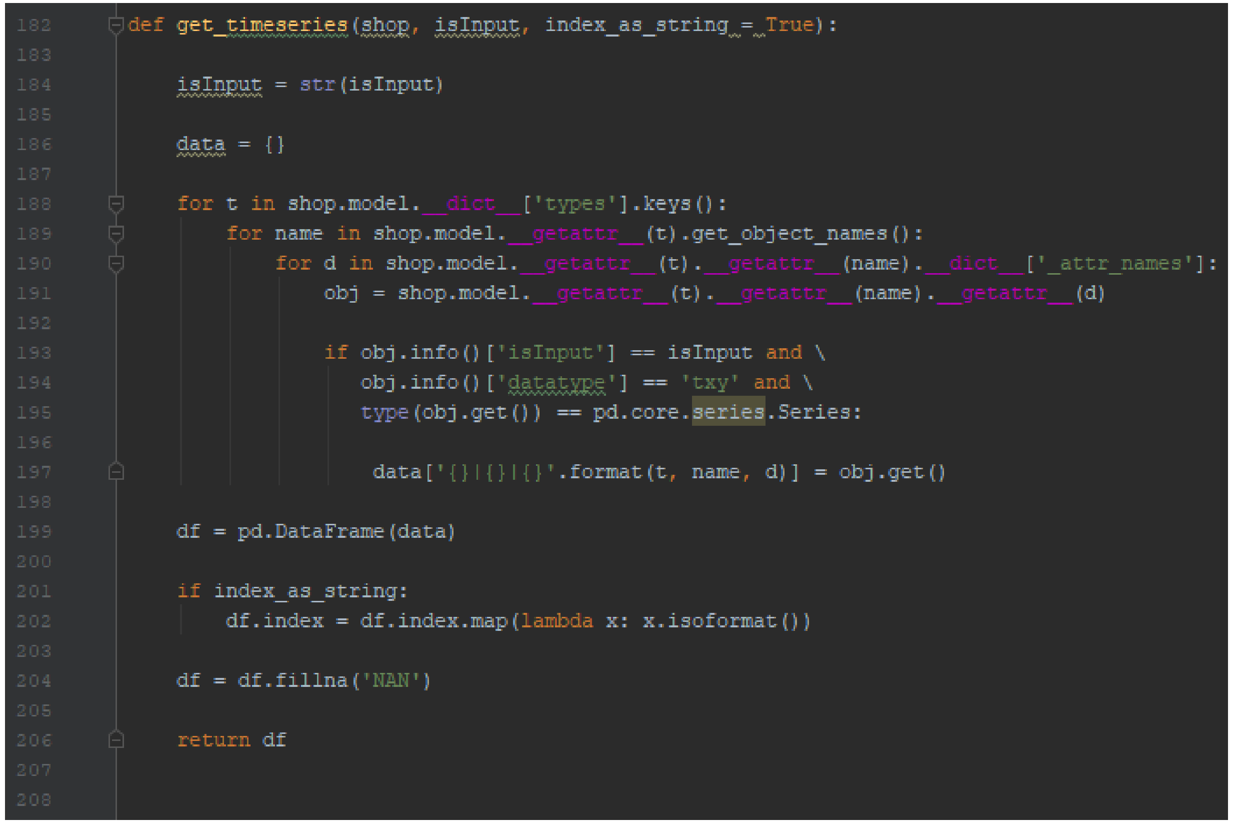


Source

PyShop and challenges

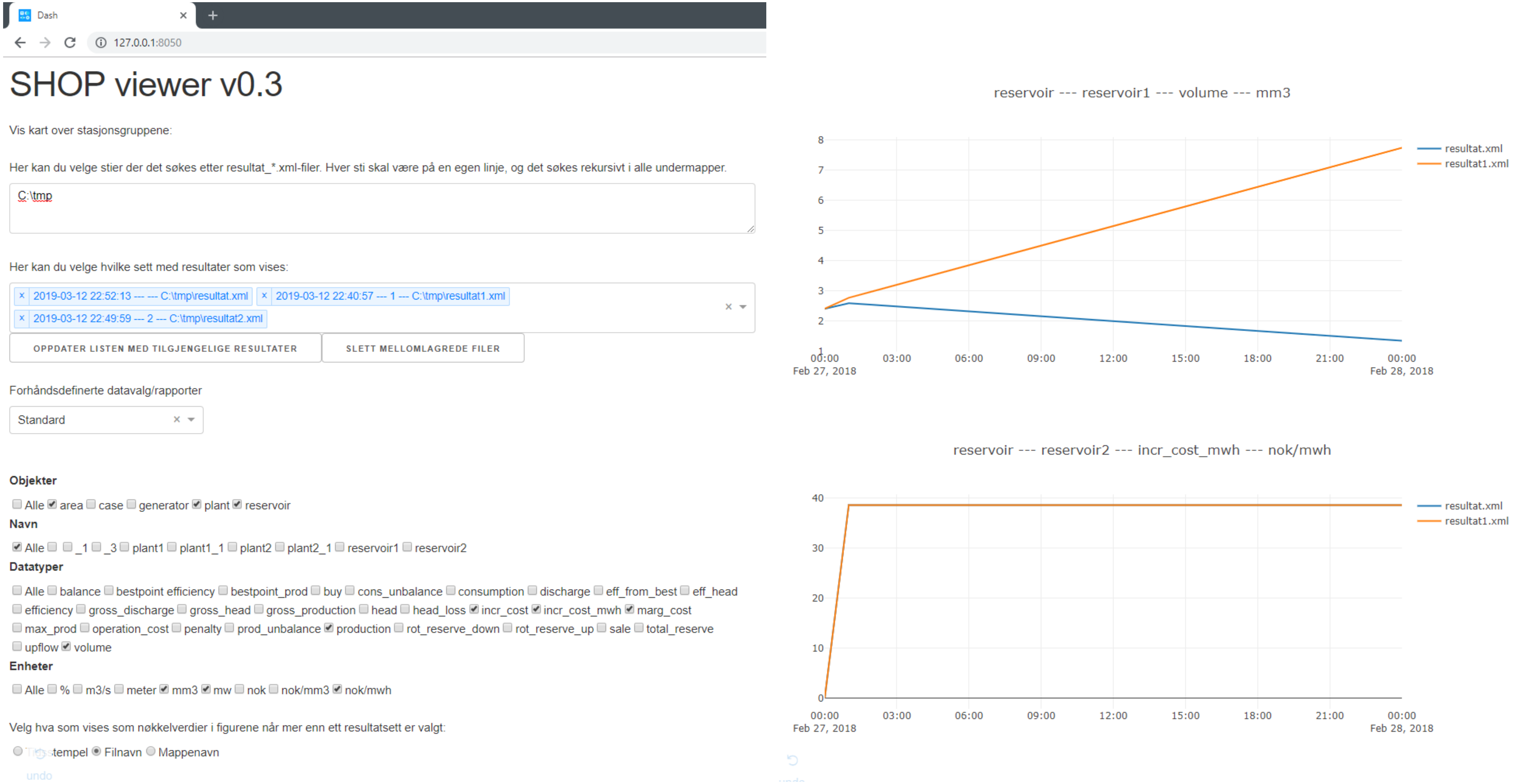
- PyShop was the second key component
  - Not practical to run SHOP in docker without PyShop
  - PyShop makes it easy to instantly verify the integrity of the results, change the inputs, rerun the models…
  - We used cmd and ascii files, with additional set-up in PyShop

- Main challenges with running SHOP in the cloud was integration with existing legacy/on-premise systems



```python
def get_timeseries(shop, isInput, index_as_string = True):

    isInput = str(isInput)

    data = {}

    for t in shop.model.__dict__['types'].keys():
        for name in shop.model.__getattr__(t).get_object_names():
            for d in shop.model.__getattr__(t).__getattr__(name).__dict__['_attr_names']:
                obj = shop.model.__getattr__(t).__getattr__(name).__getattr__(d)

                if obj.info()['isInput'] == isInput and \
                   obj.info()['datatype'] == 'txy' and \
                   type(obj.get()) == pd.core.series.Series:

                    data['{}|{}|{}'.format(t, name, d)] = obj.get()

    df = pd.DataFrame(data)

    if index_as_string:
        df.index = df.index.map(lambda x: x.isoformat())

    df = df.fillna('NAN')

    return df
```

Source



SHOP viewer v0.3

Vis kart over stasjonsgruppene:

Her kan du velge stier der det søkes etter resultat_*.xml-filer. Hver sti skal være på en egen linje, og det søkes rekursivt i alle undermapper.

C:\tmp

Her kan du velge hvilke sett med resultater som vises:

× 2019-03-12 22:52:13 --- --- C:\tmp\resultat.xml × 2019-03-12 22:40:57 --- 1 --- C:\tmp\resultat1.xml
× 2019-03-12 22:49:59 --- 2 --- C:\tmp\resultat2.xml

OPPDATER LISTEN MED TILGJENGELIGE RESULTATER SLETT MELLOMLAGREDE FILER

Forhåndsdefinerte datavalg/rapporter

Standard

Objekter

☐ Alle ☑ area ☐ case ☐ generator ☑ plant ☑ reservoir

Navn

☑ Alle ☐ ☐ _1 ☐ _3 ☐ plant1 ☐ plant1_1 ☐ plant2 ☐ plant2_1 ☐ reservoir1 ☐ reservoir2

Datatyper

☐ Alle ☐ balance ☐ bestpoint efficiency ☐ bestpoint_prod ☐ buy ☐ cons_unbalance ☐ consumption ☐ discharge ☐ eff_from_best ☐ eff_head
☐ efficiency ☐ gross_discharge ☐ gross_head ☐ gross_production ☐ head ☐ head_loss ☑ incr_cost ☑ incr_cost_mwh ☑ marg_cost
☐ max_prod ☐ operation_cost ☐ penalty ☐ prod_unbalance ☑ production ☐ rot_reserve_down ☐ rot_reserve_up ☐ sale ☐ total_reserve
☐ upflow ☑ volume

Enheter

☐ Alle ☐ % ☐ m3/s ☐ meter ☑ mm3 ☑ mw ☐ nok ☐ nok/mm3 ☑ nok/mwh

Velg hva som vises som nøkkelverdier i figurene når mer enn ett resultatsett er valgt:

○ Tidsstempel ● Filnavn ○ Mappenavn

Hydro

We are aluminium