



Automated deployment for better return on investment

Long Vu, DevOps at Ouranos

vu.long@ouranos.ca

2021-July-06

Canadian Research Software Conference



canarie





PAVICS (Platform for the Analysis and Visualization of Climate Science)

- <https://pavics.ouranos.ca>
- Climate Science as a Service
- Project started in 2016
- Ouranos, CRIM, UofT and PCIC
- Funding: Canarie, ECCCC, CEDA, DACCS



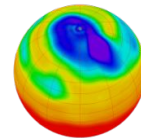
UNIVERSITY OF
TORONTO



canarie



DACCS
Data Analytics for Canadian
Climate Services



Centre for Environmental
Data Analysis


SCIENCE AND TECHNOLOGY FACILITIES COUNCIL
NATURAL ENVIRONMENT RESEARCH COUNCIL



Environment and
Climate Change Canada

Environnement et
Changement climatique Canada

PAVICS homepage




PAVICS

Power Analytics and Visualization for Climate Science

Login / Register

PAVICS is a virtual laboratory facilitating the analysis of climate data. It provides access to several data collections ranging from observations, climate projections and reanalyses. It also provides a Python programming environment to analyze this data without the need to download it. This working environment is constantly updated with the most efficient libraries for climate data analysis, in addition to ensuring quality control on the provided data and associated metadata.

- Data and files in netCDF format
- Metadata following CF-conventions
- Parallel computing environment (xarray + dask)



ANALYSE CLIMATE DATA

PAVICS provides methods for finding and analysing climate data, offering tools that span the range of common data analysis steps

1. Access Data
2. Subsetting
3. Climate indices
4. Ensemble statistics
5. Visualization

This content was generated from a Jupyter notebook and analyses cannot be run directly on this page. [Open this notebook in the PAVICS JupyterHub](#)

Climate data analysis

1. Finding and accessing datasets

PAVICS climate datasets are hosted on a THREDDS data server at <https://pavics.ouranos.ca/thredds>. Although THREDDS provides a user-interface for browsing datasets, it is often more practical to navigate the catalogue programmatically. The tutorial introduces the `siphon` library to browse the THREDDS catalog, and `xarray` to open a streaming connection to the remote data.

More specifically, the tutorial demonstrates how to access an ensemble of climate simulations, namely Ouranos' standard ensemble of bias-adjusted climate scenarios version 1.0 (cb-oura-1.0), using python commands. The ensemble contains 22 bias-adjusted CMIP5 simulations in netCDF format, each with three variables (tasmin, tasmax, pr) and dimensions of longitude, latitude and time (1064, 700, 55175). The server can provide multiple [data access](#) and [metadata services](#) but in the following steps the tutorial will focus on the `OPeNDAP` service, where instead of downloading huge volumes of data locally, only the relevant portion is accessed using a standard called Data Access Protocol (DAP).

To conserve any modifications to tutorial notebooks in the PAVICS JupyterLab they need to be copied into your writable-workspace directory.

```
In [1]: from siphon.catalog import TDSCatalog
url = "https://pavics.ouranos.ca/twiter/ows/proxy/thredds/catalog/datasets/simulations/bias_adjust

# Create Catalog
cat = TDSCatalog(url)

# List of datasets
print(f"Number of datasets: {len(cat.datasets)}")

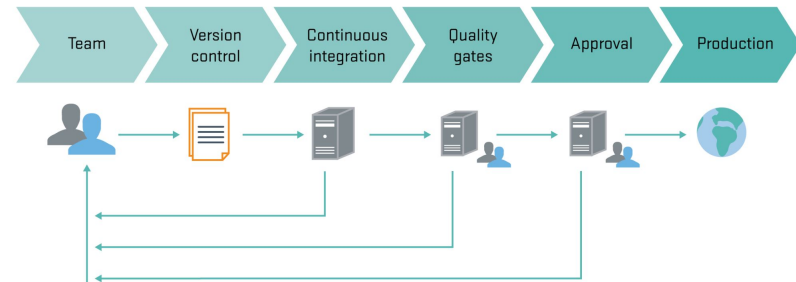
# Access mechanisms - here we are interested in OPeNDAP, a data streaming protocol
cids = cat.datasets[0]
print(f"Access URLs: {tuple(cids.access_urls.keys())}")

Number of datasets: 22
```

Contact: pavics@ouranos.ca Twitter: @PAVICSOuranos Legal: PAVICS Conditions

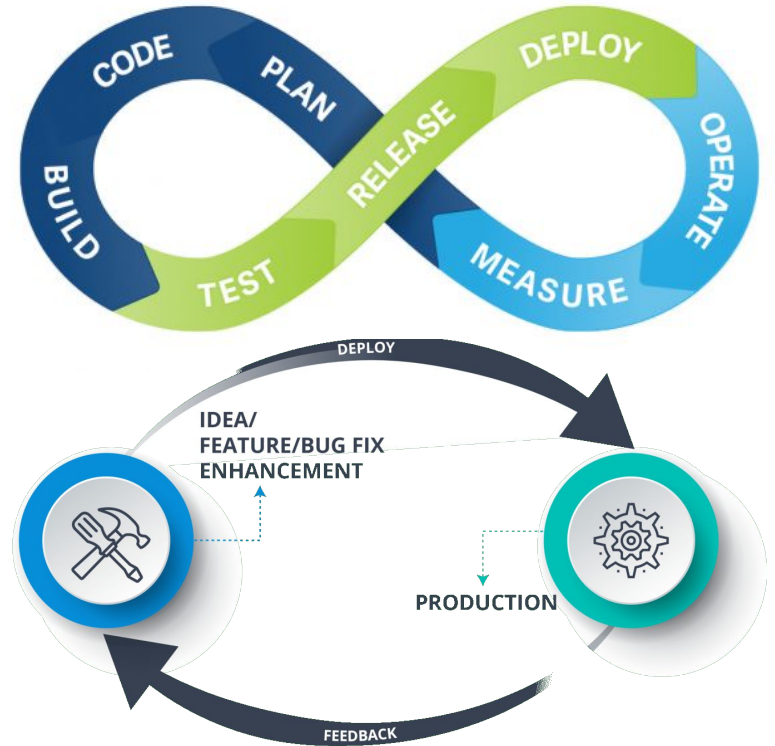
Cashing in with automated deployment

- Order online, getting ROI only when order arrives
- Car dealer, getting ROI only when car is sold
- Time spent fixing bugs, implementing feature, getting ROI only when end user actually gets it (deployed, in production). A package or build not deployed is like a car unsold, still in the dealer parking lot, taking up space.



Automated deployment advantages

- Can deploy much more frequently
 - PAVICS try to deploy nightly
- Smaller changes go into production = less surprise = less stress
 - Each PR is almost deployed by itself, unless 2 PR merged in same day
- If something goes wrong, root cause is much easier to find/revert



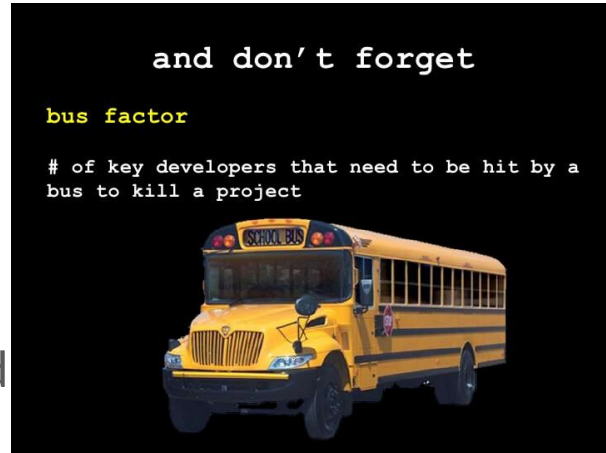
Automated deployment advantages for Devs

- Since a PR merge means a deployment, Devs basically have deployment privilege without needing access to production server
- Autonomy: Devs have full control of what and when to deploy, without needing to wait on someone



Automated deployment advantages for Org

- Resilience: increase the bus factor: more than one person can perform the deployment
- Traceability: automated deployment have logs, instead of being done by someone, from his personal machine without any trace



TRACEABILITY

Automated deployment advantages for DevOps

- Less interruption
- Can focus on upgrading other components, delegating upgrade of in-house components to Devs
- Avoid being the bottleneck slowing others down
- Have freedom to take off anytime



Automated deployment requirements

- Rigorous code review
- Comprehensive integration test suite
 - PAVICS tests real user end to end workflow using notebooks
- Which means reproducible deployment for testing
 - PAVICS has Vagrant support for spawning a complete VM locally
 - PAVICS has a staging server that checks for deployment every 5 minutes so we can catch error almost instantly after a PR merge

```
//Load values from database store with channel select
NotificationClient NotificationClient = null; // = NotificationClient.G
IF (NotificationClient == null)
{
    NotificationClient = new bl.desktop.NotificationClient() { Deny = fi
}
//NotificationClient.Insert();
}
else
{
    NotificationClient.LastRequest = DateTime.Now;
    NotificationClient.RequestCount = NotificationClient.RequestCount +
}
//NotificationClient.Update();
}
IF (NotificationClient.Deny == false)
{
    NotificationClient.Request NotificationClient = new bl.desktop.Notification
    NotificationClient.RequestCount = NotificationClient.RequestCount + 1;
    NotificationRequest request = new NotificationRequest { UserHostAddress =
    NotificationRequest.Timestamp = DateTime.Now;
}
//Redirect message
for (int i = 0; i <= NotificationClient.RequestCount; i++)
```



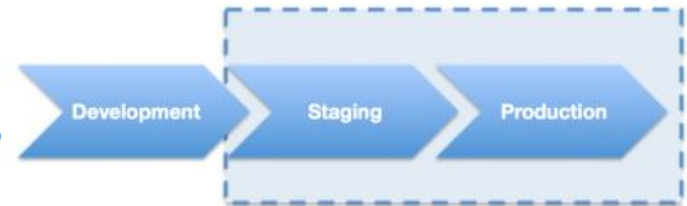
Jenkins



docker



VAGRANT



How we did it



Scheduler

Trigger

Autodeploy from git repos

Detect new
commits

Stop
Server

Pull

Deploy new
code

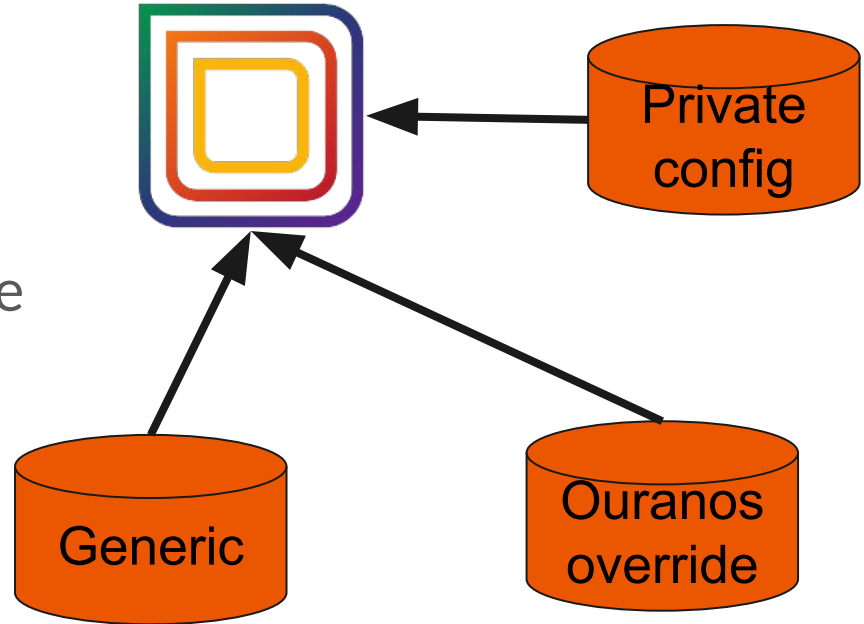
For all repos
If at least one
repo has new
commit on same
branch, continue

Stop before pull to
use current code
matching current
old state

For all repos

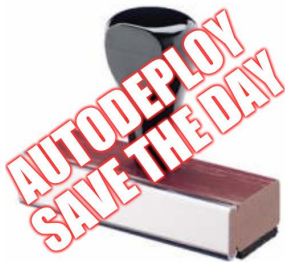
PAVICS platform deployment

- Platform is made of 3 git repos
 - Pluggable
 - Infrastructure-as-code

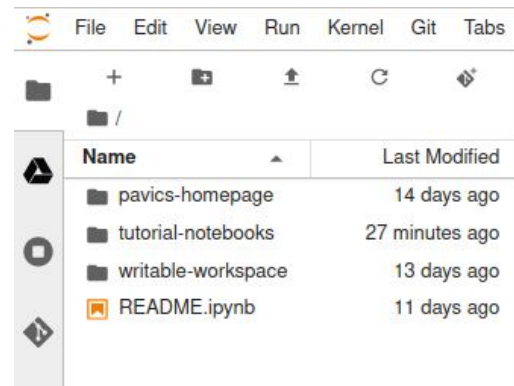
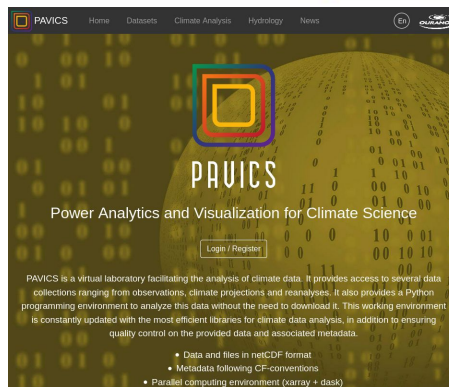


PAVICS other deploy jobs

- NcML and testdata on Thredds
- The homepage
- Tutorial notebooks in JupyterLab
- GEPS forecast (ECCC)



NcML



Questions?



- Contact: vu.long@ouranos.ca
- PAVICS: <https://pavics.ouranos.ca>
- Resources:
 - Deployment: <https://github.com/bird-house/birdhouse-deploy> + <https://github.com/bird-house/birdhouse-deploy-ouranos>
 - Test: <https://github.com/Ouranosinc/jenkins-config> + <https://github.com/Ouranosinc/PAVICS-e2e-workflow-tests>

Reference

- Deployment if new changes on server:
 - <https://github.com/bird-house/birdhouse-deploy/blob/1.13.9/birdhouse/deployment/triggerdeploy.sh> and deployment/deploy.sh
- Triggered nightly by cronjob:
 - <https://github.com/bird-house/birdhouse-deploy/blob/1.13.9/birdhouse/components/scheduler/config.yml.template> and scheduler/docker-compose-extra.yml
- All config centralized in
 - <https://github.com/bird-house/birdhouse-deploy/blob/1.13.9/birdhouse/env.local.example>

Reference

- Documentation:
 - <https://github.com/bird-house/birdhouse-deploy/tree/1.13.9/birdhouse/components#scheduler>
- Vagrantfile and config:
 - <https://github.com/bird-house/birdhouse-deploy/blob/1.13.9/Vagrantfile> and `vagrant_variables.yml.example`
- Battle tested, in production more than a year

Reference



- Generic deployment any files from any git repos
 - <https://github.com/bird-house/birdhouse-deploy/blob/1.13.9/birdhouse/deployment/deploy-data> and <https://github.com/bird-house/birdhouse-deploy/blob/1.13.9/birdhouse/deployment/deploy-data.config.example.yml>
- Testdata deployment
 - <https://github.com/bird-house/birdhouse-deploy/blob/1.13.9/birdhouse/env.local.example#L176-L198>

Reference



- Homepage and tutorial notebook deployment
 - https://github.com/bird-house/birdhouse-deploy-ouranos/blob/5dd7608e0639feec4b70cdee6325fad137bc5993/scheduler-jobs/deploy_pavics_landing_notebooks.yml
- Trigger other scripts (GEPS forecast)
 - https://github.com/bird-house/birdhouse-deploy-ouranos/blob/5dd7608e0639feec4b70cdee6325fad137bc5993/scheduler-jobs/retrieve_geps_forecasts.yml