



# **PopperCI: Continuous Validation of Scientific Experiments**

**Ivo Jimenez, Sina Hamedian (UC Santa Cruz)**

**Carlos Maltzahn (UC Santa Cruz)**

**Andrea Arpaci-Dusseau, Remzi Arpaci-Dusseau (UW  
Madison)**

**Kathryn Mohror (LLNL)**

**Jay Lofstead (SNL)**

**Rob Ricci (Utah)**

# Problem of Reproducibility in Computation and Data Exploration

The UI is capable of graphing the ratio  $\frac{getFeedback(FIT)}{getCost(FIT)}$  (Figure 2) in parallel with the analysis of the query streams (A high ratio indicates that WFIT generates good recommendations.) It also makes available the recommendations that are generated at each step, as well as the internal bookkeeping that the algorithm maintains. We will show some of this information as part of this scenario.

**Scenario #2.** We delve a little bit more into the details of our tool by allowing the candidate-index set to be automatically maintained but again keeping the feedback feature "off". At this point, the candidate-index set can dynamically grow/shrink and be repartitioned over time based on the calculations of index interactions associated with each statement. This brings the tool into a completely online mode where it can operate autonomously without any user intervention.

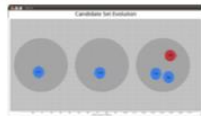


Figure 3: Evolution of the candidate set with respect to partitioning (by calculating index interactions at each step). Each set corresponds to phases 1, 2 and 3 respectively.

We will see again how the algorithm generates a configuration at each step, however, in this scenario the partitioning of the candidate set will evolve for each of the three phases of the workload (Figure 3). We will show that this feature actually improves the quality of the recommendations.

**Scenario #3.** We complete the picture and show the effect that feedback has on the performance of WFIT by demonstrating one of the key contributions of our work: a principled feedback mechanism that is tightly integrated with the logic of the on-line algorithm (WFA').

By inspecting the recommended set of indexes at any point in time, the DBA can decide whether to up- or down-vote any candidate index according to her criteria for not vote at all). For the small test workload, it is easy to come up with reasonable "good" and "bad" votes that the audience can interactively send as feedback to the recommendation engine. We will execute three instances of WFIT concurrently with distinct feedback (good, bad, and no-feedback) and show the difference in performance for each (Figure 4).

The audience will see how, in the case of "good" feedback, the performance of WFIT increases in relation to the performance of the "no-feedback" instance (using the performance of OPT as baseline). In contrast, with "bad" feedback, the performance of WFIT will decrease; however, and more importantly, we will witness how WFIT is able to recover from poor feedback. This recovery mechanism is another important feature of the WFIT algorithm.

**Scenario #4.** The last scenario executes the *Refler* workload suite of the *Online Index Selection Benchmark* [10] on *Kaizen*. This is a complex workload consisting of approximately 1600 statements (queries and updates) that refer-

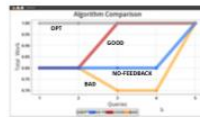


Figure 4: Multiple instances of WFIT running in parallel. The vote for the "good" and "bad" instances is done at step 1, causing the divergence in their behavior with respect to the "no-feedback" instance.

ence several datasets (TPC-C, TPC-DS, TPC-E, TPC-H and NREF).

We will show two WFIT variants: one with a stable and fixed candidate set partitioning; another whose candidate set is allowed to be automatically maintained. Similarly to scenario #1, we will graph the OPT vs. WFIT ratio in real-time as the workload is processed (Figure 5).

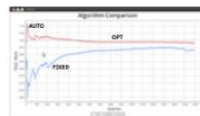


Figure 5: Two instances of WFIT running the Online Index Selection Benchmark. One with a fixed and stable candidate set (FIXED); another one with an automatically maintained candidate set (AUTO).

## 5. REFERENCES

- [1] S. Aravamudan, L. Collier, A. Marathe, V. Narasayana, and M. Svemala. Database tuning advisor for microsoft SQL server 2005. In *SIGMOD conference*, pages 930–932, 2005.
- [2] S. Agrawal, E. Cho, and V. Narasayana. Automatic physical design tuning: workload as a sequence. In *SIGMOD Conference*, pages 483–494, 2006.
- [3] A. Bonella and R. El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, 1998.
- [4] N. Bruno and S. Chaudhuri. An online approach to physical design tuning. In *ICDE*, pages 826–835, 2007.
- [5] N. Bruno and S. Chaudhuri. Constrained physical design tuning. *Proceedings of the VLDB Endowment*, 1:4–15, 2008.
- [6] N. Bruno and S. Chaudhuri. Interactive physical design tuning. In *ICDE*, pages 1161–1164, 2010.
- [7] D. Zilio et al. DB2 Design Advisor: Integrated Automatic Physical Database Design. In *VLDB*, pages 1997–2004, 2004.
- [8] B. Dagville, D. Das, K. Das, K. Yagoubi, M. Zaki, and M. Zoukri. Automatic SQL Tuning in Oracle 10g. In *VLDB*, pages 1189–1199, 2004.
- [9] K. Schmeiter, S. Alshabool, T. Milo, and N. Polyzotis. On-Line index selection for sliding workloads. In *ICDE*, pages 459–468, 2007.
- [10] K. Schmeiter and N. Polyzotis. A Benchmark for Online Index Selection. In *ICDE*, pages 1781–1788, 2009.
- [11] K. Schmeiter and N. Polyzotis. Semi-automatic index tuning: Keeping DBAs in the loop. *FVLDB*, 5(5):478–489, 2012.
- [12] K. Schmeiter, N. Polyzotis, and L. Gettoer. Index interactions in physical design tuning: modeling, analysis, and applications. *FVLDB*, 2(1):1234–1245, 2009.

- What compiler was used?
- Which compilation flags?
- How was subsystem X configured?
- How does the workload look like?
- What if I use input dataset Y?
- And if I run on platform Z?
- ...

# Lab Notebook

torpor-popper (branch: master)

View Branch Create Branch Search

Subject	Author	Date
<a href="#">master</a> <a href="#">origin/HEAD</a> <a href="#">origin/master</a> Add...	Ivo Jimenez	2016-10-05...
Adds torpor as a submodule	Ivo Jimenez	2016-08-25...
Results of running base-vs-tar...	Ivo Jimenez	2016-08-25...
Makes use of 'baseliner' role fo...	Ivo Jimenez	2016-08-25...
Removes unused line	Ivo Jimenez	2016-08-21...
Will add later	Ivo Jimenez	2016-08-21...
Re-adds baseliner ansible role	Ivo Jimenez	2016-08-21...
renames ansible folder to exper...	Ivo Jimenez	2016-08-21...
benchmarks now have their ow...	Ivo Jimenez	2016-08-21...
Adds latest version of baseliner...	Ivo Jimenez	2016-06-12...
Adds experiment on all stress-...	Ivo Jimenez	2016-06-12...
Adds same-platform variability...	Ivo Jimenez	2016-06-10...
WIP on using baseliner role	Ivo Jimenez	2016-06-10...
Checks first if docker is already...	Ivo Jimenez	2016-05-16...
Tuning cpu-quota of 5 machines	Ivo Jimenez	2016-05-16...
Fixes creation of parameters an...	Ivo Jimenez	2016-05-16...
Adds logic to install statically li...	Ivo Jimenez	2016-05-16...

SHA: af13570c1d700f85d2a9de348beb37215eb978c4

Author: Ivo Jimenez <[ivo.jimenez@gmail.com](mailto:ivo.jimenez@gmail.com)>

Date: Thu Aug 25 2016 01:29:21 GMT-0700 (PDT)

Subject: **Results of running base-vs-targets for stressing on 4 machines**

Parent: [7a13185872bc73719048a1bd8a76f68a06798e13](#)

Results of running base-vs-targets for stressing on 4 machines

Base machine is 'issdm-12' and targets were tuned with the 'crafty' and 'c-ray' benchmarks.

The main difference between these results and the ones that appear on our VarSys '16 paper is that we are reflecting the speedup function for  $x=1$ , for both (1) tuning targets and (2) displaying results. This allows us to show better the reduction in variability without having to deal with different scales (slowdowns that lie between the [0-1] range are instead reflected and treated as speedups).

In short, we now unambiguously observe reduced variability when targets are limited. A couple of outliers, in particular stress-ng's memfd stressor, when limited, is very slow on target machines.

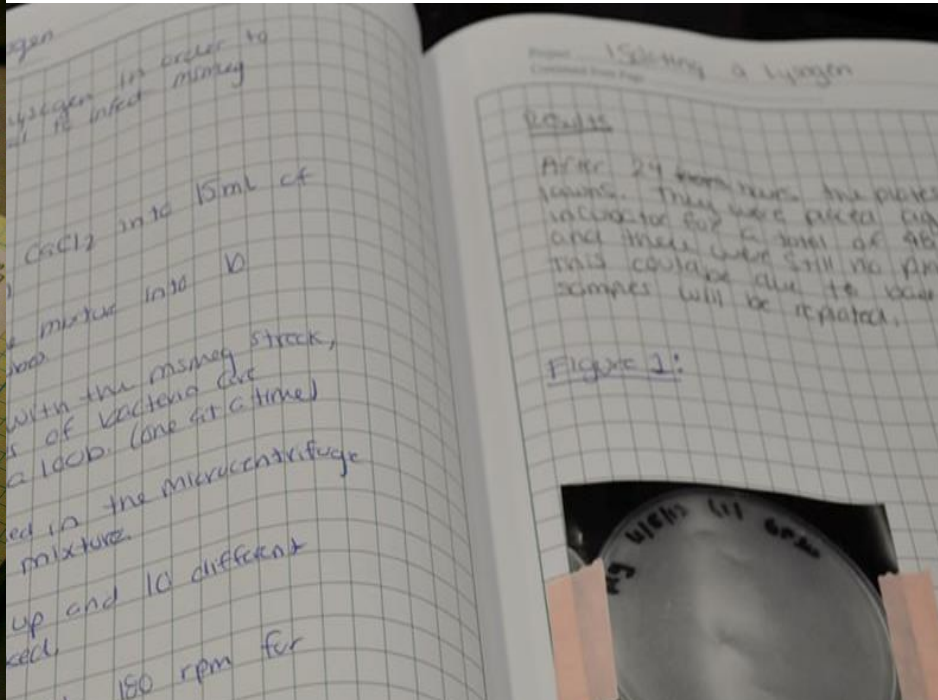
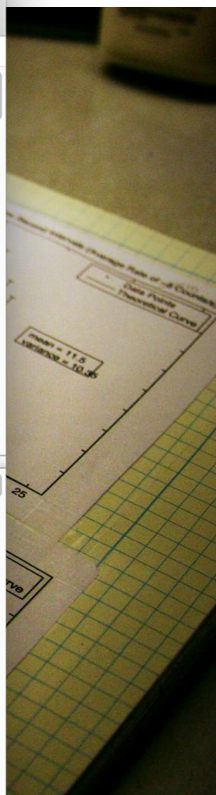
created [experiments/base-vs-limited-targets/all\\_results.csv](#)

created [experiments/base-vs-limited-targets/all\\_results.json](#)

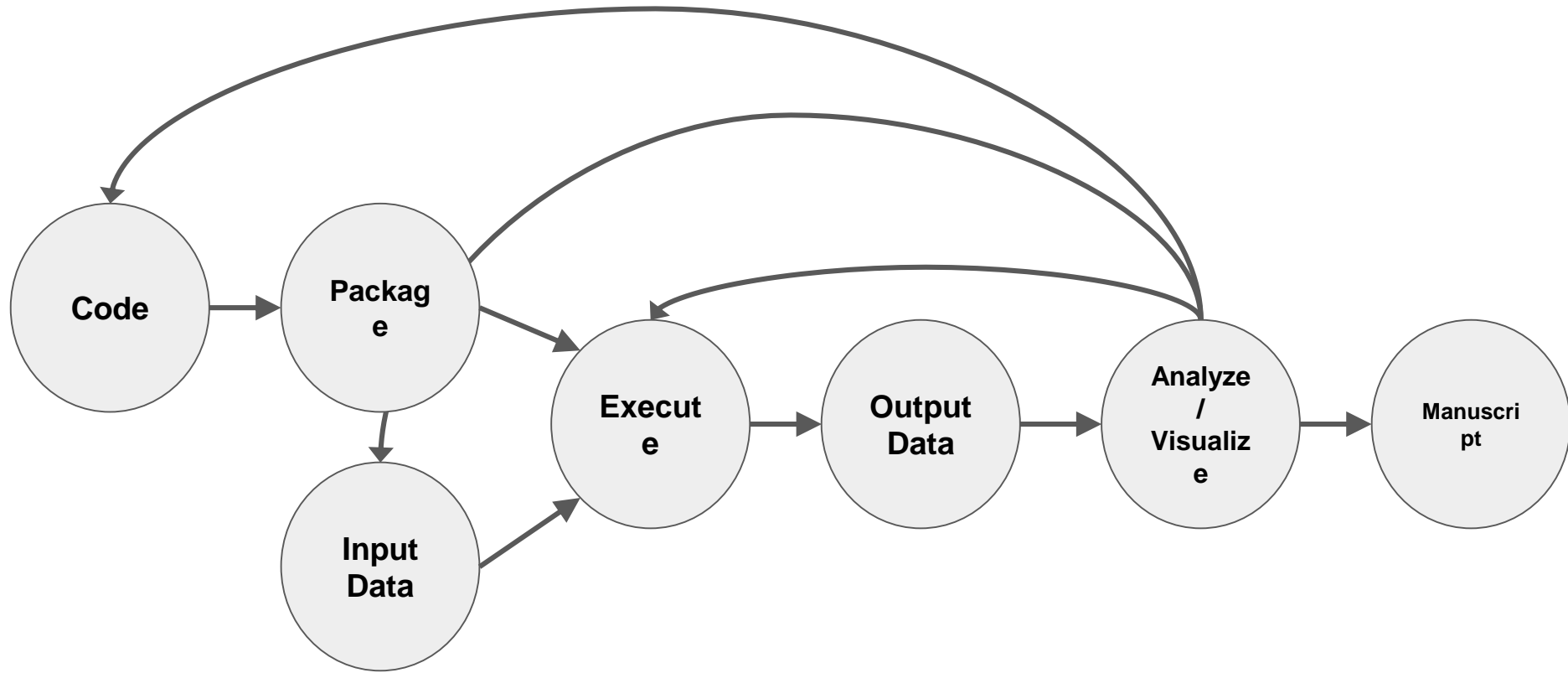
created [experiments/base-vs-limited-targets/ansible.log](#)

created [experiments/base-vs-limited-targets/facts/192.168.140.81.json](#)

129 commits loaded



# Common Experimentation Workflow

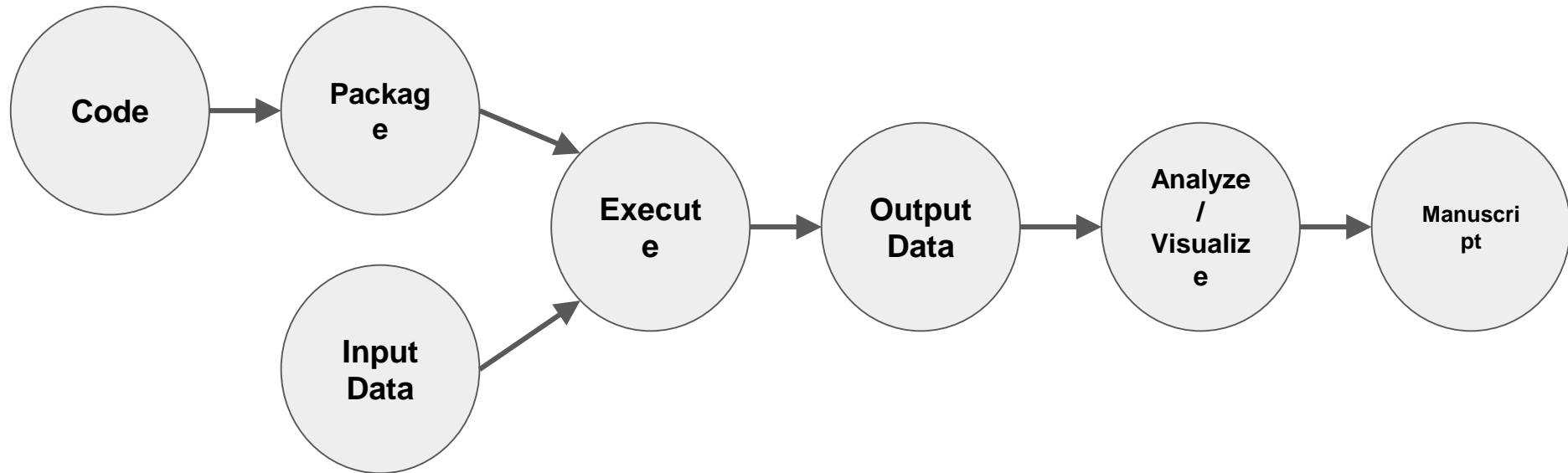


# Analogies with DevOps Practice

Scientific exploration	Software project
Experiment code	Source code
Input data	Test examples
Analysis / visualization	Test analysis
Validation	CI / Regression testing
Manuscript / note book	Documentation / reports

**Key Idea behind The Popper Protocol:** manage a scientific exploration like software projects

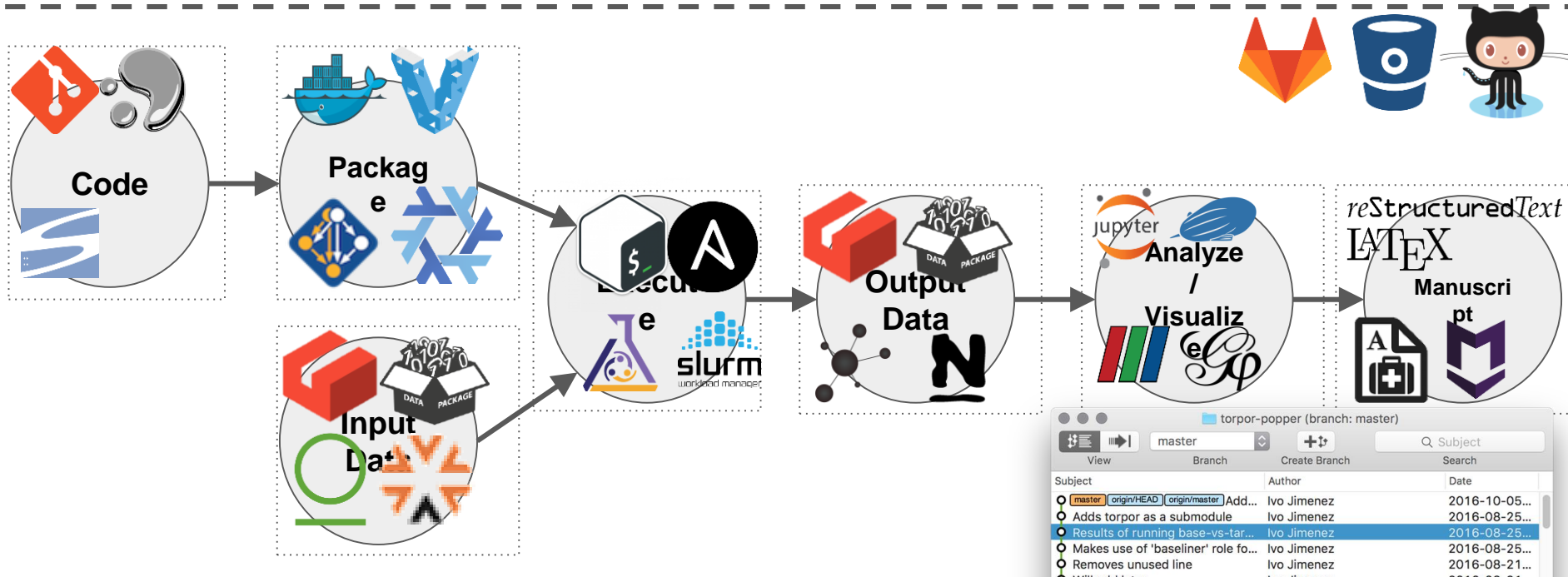
# Common Experimentation Workflow





# Popper

falsifiable.us



torpor-popper (branch: master)

View: master | Create Branch | Search: Subject

Subject	Author	Date
master   origin/HEAD   origin/master   Add...	Ivo Jimenez	2016-10-05...
Adds torpor as a submodule	Ivo Jimenez	2016-08-25...
Results of running base-vs-tar...	Ivo Jimenez	2016-08-25...
Makes use of 'baseliner' role fo...	Ivo Jimenez	2016-08-25...
Removes unused line	Ivo Jimenez	2016-08-21...
Will add later	Ivo Jimenez	2016-08-21...
Re-adds baseliner ansible role	Ivo Jimenez	2016-08-21...
renames ansible folder to exper...	Ivo Jimenez	2016-08-21...
benchmarks now have their ow...	Ivo Jimenez	2016-08-21...
Adds latest version of baseliner...	Ivo Jimenez	2016-06-12...
Adds experiment on all stress-...	Ivo Jimenez	2016-06-12...
Adds same-platform variability...	Ivo Jimenez	2016-06-10...
WIP on using baseliner role	Ivo Jimenez	2016-06-10...
Checks first if docker is already...	Ivo Jimenez	2016-05-16...
Tunning cpu-quota of 5 machines	Ivo Jimenez	2016-05-16...
Fixes creation of parameters an...	Ivo Jimenez	2016-05-16...
Adds logic to install statically li...	Ivo Jimenez	2016-05-16...

Gist it

# DevOps in Practice

## Typical



```
ivo@mbp:~  
$ hostname  
mbp  
ivo@mbp:~  
$ date  
Tue Jan 31 09:31:14 PST 2017  
ivo@mbp:~  
$ cat src/popper/popper/README.md  
# Popper-CLI  
  
A CLI tool to help bootstrap projects that follow the  
[Popper](https://github.com/systemslab/popper) convention.  
  
## Install  
  
Download from  
[popper/releases](https://github.com/systemslab/popper/releases). Note  
that we have only tested on OSX and Linux (Windows coming soon). Once  
downloaded, uncompress and place the binary in a folder that is  
included in your `PATH` (e.g. `/usr/bin`).  
  
## Usage  
  
To get an overview and list of commands check out the command line  
help:  
  
```bash  
popper help  
```\br/>  
ivo@mbp:~  
$ find $HOME/tmp -path "**login*"  
ivo@mbp:~  
$ find $HOME/tmp -path "**git-meme*"  
ivo@mbp:~  
$ find $HOME/tmp -path "**git-meme.jpg*"  
ivo@mbp:~  
$ find $HOME/tmp -path "**meme.jpg*"   
  
ivo@mbp:~  
$ find $HOME/tmp/ -path "**meme.jpg*"   
  
/Users/ivo/tmp/git-meme.jpg  
ivo@mbp:~  
$ clear  
ivo@mbp:~  
1: [tmux] 2: bash- 3: bash
```

## DevOps



myscript.sh

```
$ bash myscript.sh
```





1. Pick one or more DevOps tools.
  - At each stage of experimentation workflow.
2. Put all associated scripts in version control.
  - Make experiment self-contained.
  - For external dependencies (code and data), reference specific versions.
3. Document changes as experiment evolves.
  - In the form of commits.

[1]: Jimenez et al. *Standing on the Shoulders of Giants by Managing Scientific Experiments Like Software*, ;login: Winter 2016, Vol. 41, No. 4.

[2]: Jimenez et al. *The Popper Convention: Making Reproducible Systems Evaluation Practical*, REPPAR 2017.

# Popper-compliant Experiments

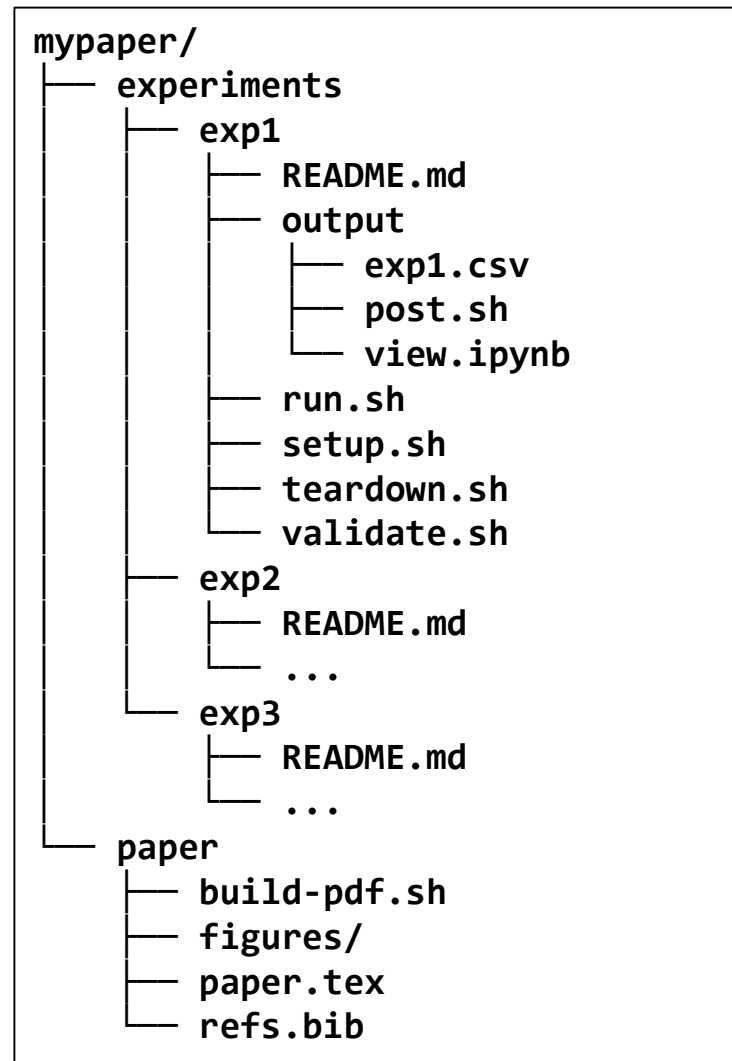
- An experiment is *Popper-compliant* if all of the following is available (self-contained):
  - Experiment code.
  - Data dependencies.
  - Parameterization.
  - Results.
  - Validation.

```
$ cd mypaper-repo
$ popper init
-- Initialized Popper repo mypaper-repo

$ popper experiment list
-- available templates -----
ceph-rados      proteustm      mpi-comm      adam
cloverleaf     gassyfs       zlog          bww
spark-stand    torpor        malacology    genevo
hadoop-yarn    kubsched      alg-encycl    macrob

$ popper add gassyfs
-- Added gassyfs experiment to mypaper-repo
```

# Popper-compliant Experiments



# PopperCl

# PopperCI: Continuous Validation of Scientific Experiments

- Project structure follows a convention.
  - One experiment per subfolder
  - Optional paper folder
- Bash-oriented interface to execution:
  - **setup.sh**: hardware allocation/configuration, software deployment.
  - **run.sh**: run experiment, obtain results.
  - **teardown.sh**: cleanup, release resources.
- Goal: automate end-to-end execution

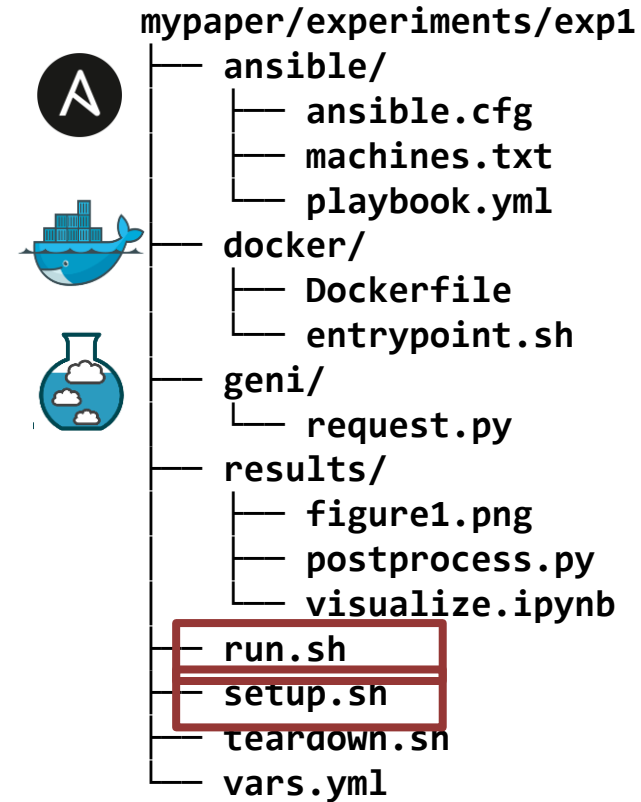
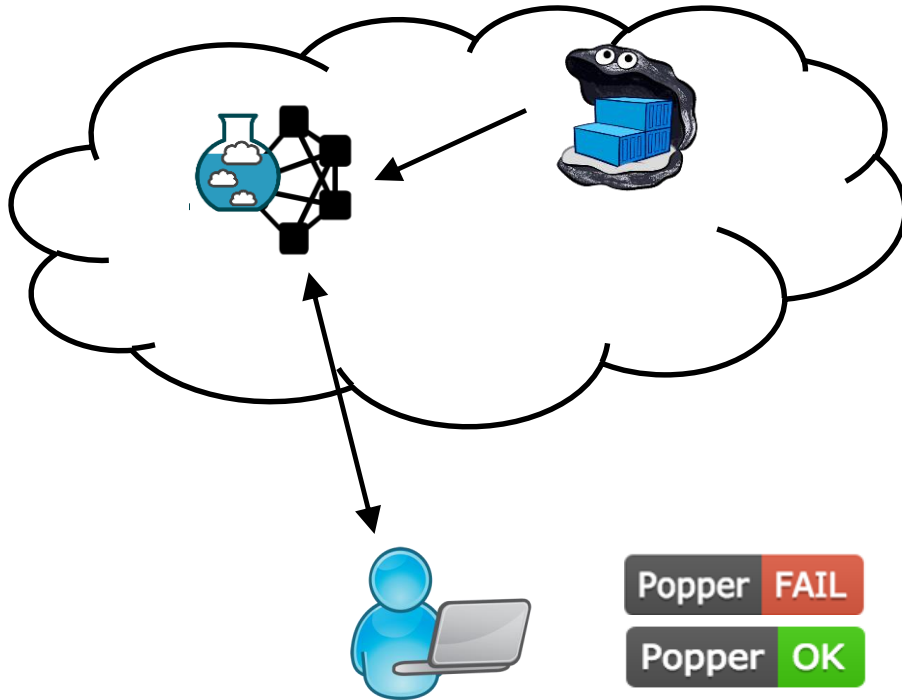
```
$ popper experiment init myexp
-- Initialized exp1 experiment.
```

```
$ ls -l experiments/myexp/
total 20K
-rw-r----- 1 ivo ivo   8 Apr 29 23:58 README.md
-rwxr-x--- 1 ivo ivo 210 Apr 29 23:58 run.sh
-rwxr-x--- 1 ivo ivo 206 Apr 29 23:58 setup.sh
-rwxr-x--- 1 ivo ivo  61 Apr 29 23:58 teardown.sh
```

```
#!/bin/bash
#!/bin/bash
#
# trigger execution of experiment
docker run google/kubectl run ...
...

```

# Automating Execution Stages





```
$ popper check exp1
```

```
Popper check started
```

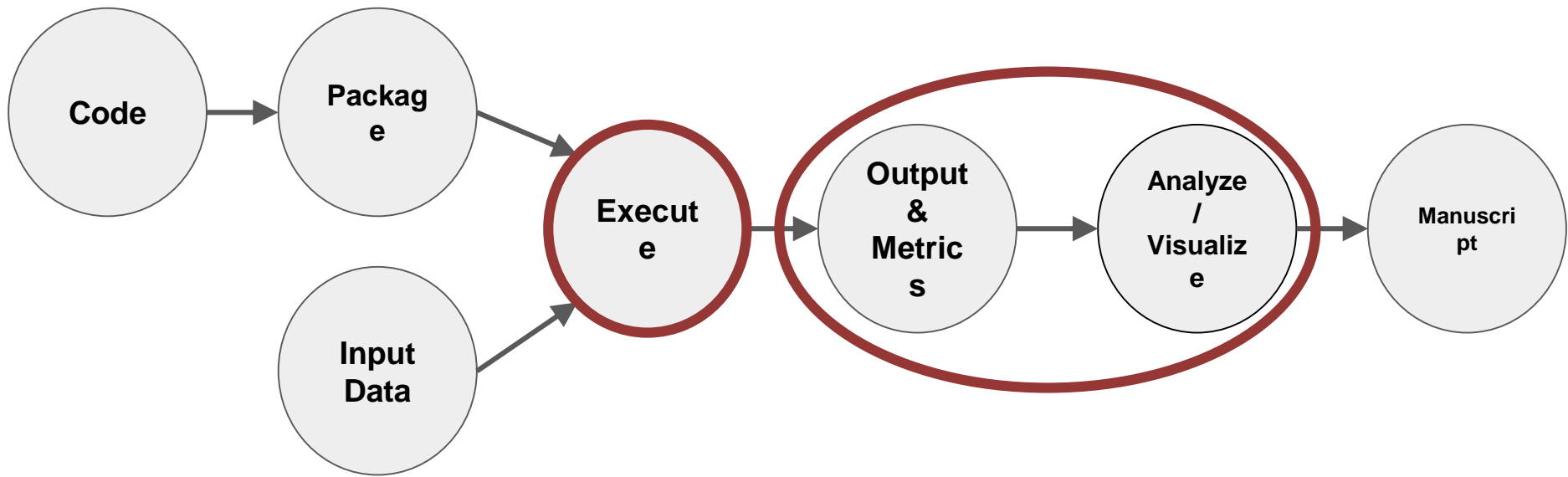
```
Stage: setup.sh .....
```

```
Stage: run.sh .....
```

```
Stage: teardown.sh ..
```

```
Popper check finished
```

```
Status: SUCCESS
```



# Codified Validations



`num_nodes, throughput, raw_bw, net_saturated`

```
Src,Eqid,Version,Datetime,Lat,Lon,Magnitude,Depth,NST,Region
ci,14692356,1,"Tuesday, May 4, 2010 03:21:38 UTC",32.6443,-1
ci,14692348,1,"Tuesday, May 4, 2010 03:19:38 UTC",32.1998,-1
ci,14692332,1,"Tuesday, May 4, 2010 03:16:56 UTC",32.6756,-1
ci,14692324,1,"Tuesday, May 4, 2010 03:08:47 UTC",32.6763,-1
ci,14692316,1,"Tuesday, May 4, 2010 03:08:08 UTC",32.6778,-1
ci,14692308,1,"Tuesday, May 4, 2010 03:06:20 UTC",32.7071,-1
ci,14692300,1,"Tuesday, May 4, 2010 03:01:52 UTC",32.1948,-1
ak,10047267,1,"Tuesday, May 4, 2010 03:01:04 UTC",61.2695,-1
ci,14692284,1,"Tuesday, May 4, 2010 02:58:51 UTC",32.7016,-1
ci,14692276,1,"Tuesday, May 4, 2010 02:57:46 UTC",32.6998,-1
ak,10047263,1,"Tuesday, May 4, 2010 02:56:28 UTC",63.5779,-1
ak,10047261,1,"Tuesday, May 4, 2010 02:52:00 UTC",60.4986,-1
ci,14692268,1,"Tuesday, May 4, 2010 02:48:40 UTC",32.6813,-1
ci,14692260,1,"Tuesday, May 4, 2010 02:35:27 UTC",32.2006,-1
nc,71392116,0,"Tuesday, May 4, 2010 02:15:24 UTC",38.8415,-1
ci,14692244,1,"Tuesday, May 4, 2010 02:05:07 UTC",33.5248,-1
ci,14692228,1,"Tuesday, May 4, 2010 01:57:08 UTC",32.6823,-1
ci,14692220,1,"Tuesday, May 4, 2010 01:53:28 UTC",32.6881,-1
ci,14692212,1,"Tuesday, May 4, 2010 01:48:53 UTC",32.6398,-1
ci,14692188,1,"Tuesday, May 4, 2010 01:26:58 UTC",32.5003,-1
ci,14692180,1,"Tuesday, May 4, 2010 01:19:44 UTC",32.6836,-1
ci,14692172,1,"Tuesday, May 4, 2010 01:12:01 UTC",32.5321,-1
ci,14692164,1,"Tuesday, May 4, 2010 01:08:24 UTC",32.6833,-1
```

- Log file
- CSV
- DB Table
- TSDB
- ...

**expect**

`linear(num_nodes, throughput)`

**when**

`not net_saturated`

**expect**

`throughput >= (raw_bw * 0.9)`

[1]: Jimenez et al. *Tackling the reproducibility problem in storage systems research with declarative experiment specifications*, PDSW '15.

[2]: Jimenez et al. *Aver: Providing Declarative Experiment Specifications Facilitates the Evaluation of Computer Systems Research*, TinyTOCS, Vol. 3,.

# One More Experiment Stage

## 1. Setup

- Resource allocation, software deployment.

## 2. Execution

- Run experiment, obtain results.

## 3. **Validation**

- **Verify claims by checking validation statements against result datasets.**

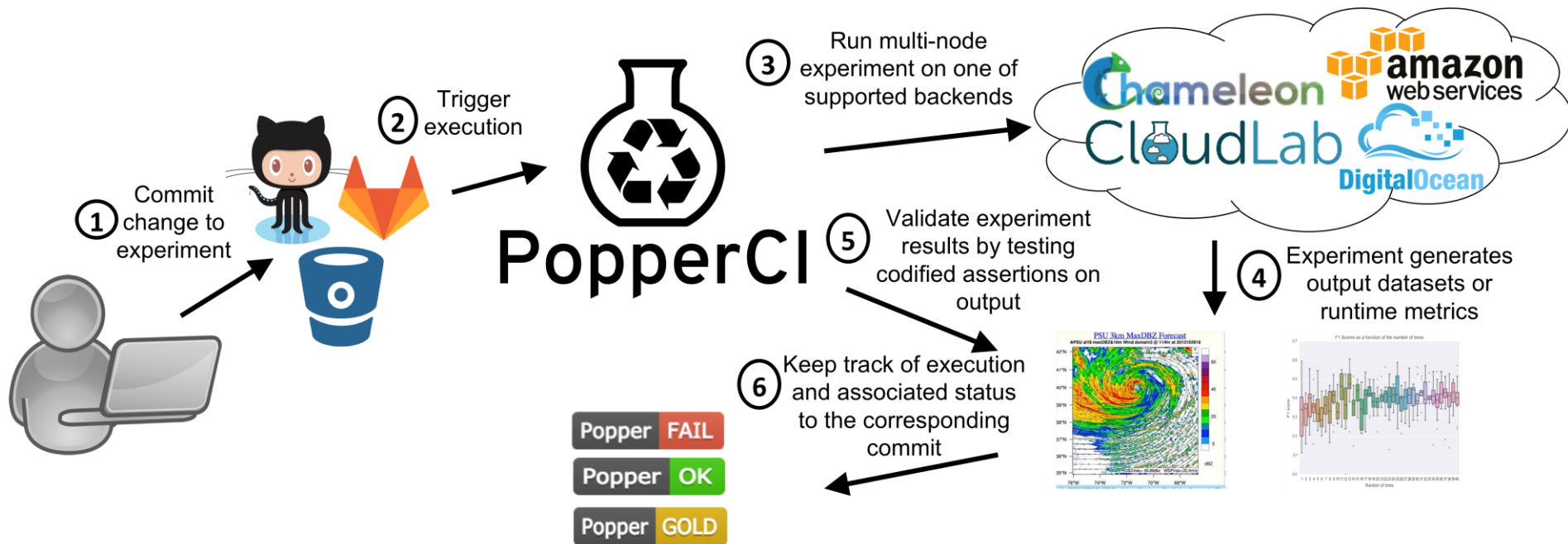
## 4. Teardown

- Cleanup, release resources.

# One More Type of Status

- **FAIL**
  - Any failure along execution pipeline.
  - Ignore teardown errors.
- **SUCCESS**
  - Experiment runs OK end-to-end.
- **GOLD**
  - **Experiment runs OK *and* all validations pass.**






# PopperCI Web Service



README.md

build **passing** coverage **72%** downloads **12** stable **No Release** unstable **dev-master** license **GNU**

# ACM's Three Rs of Reproducibility<sup>[1]</sup>

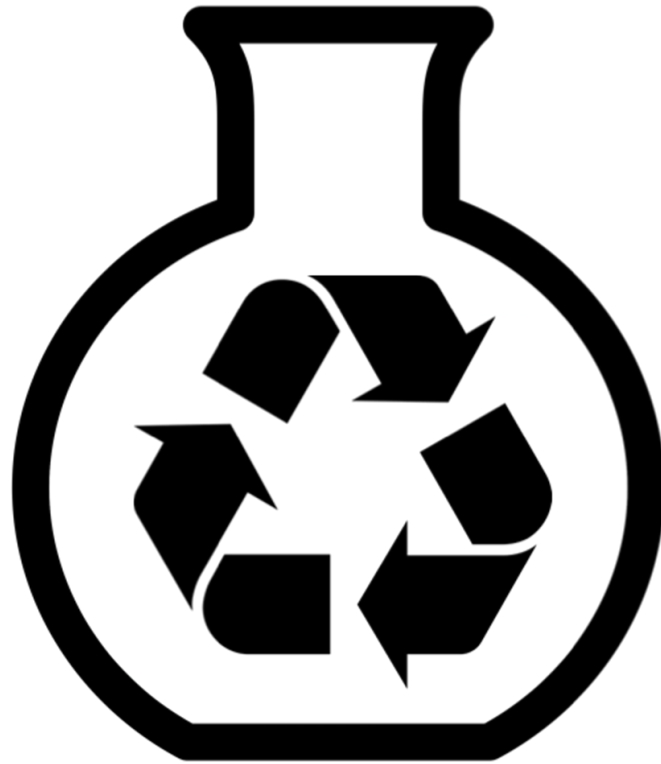
Result Status	Re-executed By	Artifacts	ACM Badge	PopperCI Badge
Repeatability	Author(s)	Original		
Replicability	Nonauthor(s)	Original		
Reproducibility	Anyone	Re-implemented		

[1]: <https://www.acm.org/publications/policies/artifact-review-badging>

# Conclusion

- Popper Experimentation Protocol
  - Three high-level steps for generating experiments that are easy to re-execute.
- PopperCI
  - Convention for structuring Popper repositories.
- Popper CLI **check** command
  - CLI tool to test (locally) for PopperCI-compliance.
- PopperCI Web Service
  - Track experiment status; share/re-run experiments.
- Repeatability/Replicability Badges
  - “Compatible” with ACM’s policy on reproducibility.





**Popper**  
falsifiable.us