



www.chameleoncloud.org

REPRODUCIBILITY AS SIDE-EFFECT

Kate Keahey

Mathematics and CS Division, Argonne National Laboratory

Computation Institute, University of Chicago

keahey@anl.gov

October 25, 2018

GEFI 2018

SEPTEMBER 17, 2019

I



CHAMELEON IN A NUTSHELL

- ▶ **Deeply reconfigurable:** “As close as possible to having it in your lab”
 - ▶ Deep reconfigurability (bare metal) and isolation
 - ▶ Power on/off, reboot from custom kernel, serial console access, etc.
 - ▶ But also – modest KVM cloud for ease of use
- ▶ **Combining large-scale and diversity:** “Big Data, Big Compute research”
 - ▶ **Large-scale:** ~660 nodes (~15,000 cores), 5 PB of storage distributed over 2 sites connected with 100G network...
 - ▶ ...and **diverse:** ARMs, Atoms, FPGAs, GPUs, Corsica switches, etc.
 - ▶ **Coming soon:** more storage, more accelerators
- ▶ Blueprint for a **sustainable** production testbed: “cost-effective to deploy, operate, and enhance”
 - ▶ Powered by OpenStack with bare metal reconfiguration (Ironic)
 - ▶ Chameleon team contribution recognized as official OpenStack component
- ▶ **Open, collaborative, production** testbed for **Computer Science Research**
 - ▶ Started in 10/2014, testbed available since 07/2015, renewed in 10/2017
 - ▶ Currently 2,700+ users, 400+ projects, 100+ institutions

REPRODUCIBILITY DILEMMA

Should I invest in making my experiments repeatable?



Should I invest in more new research instead?

- ▶ Reproducibility as side-effect: lowering the cost of repeatable research
 - ▶ Example: Linux “history” command
 - ▶ From a meandering scientific process to a recipe
- ▶ Documenting the process: interactive papers

REPEATABILITY MECHANISMS IN CHAMELEON

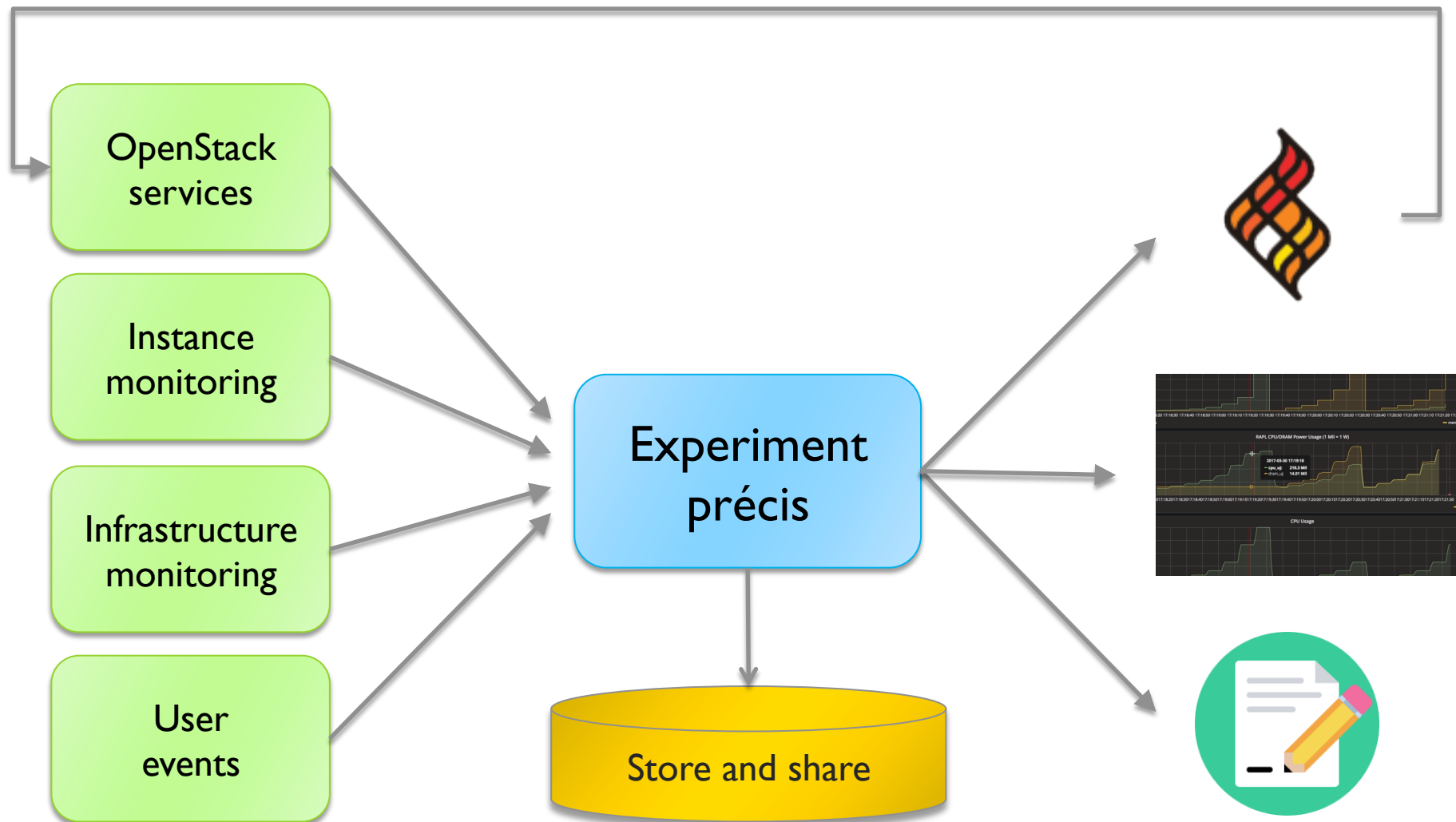
- ▶ Testbed versioning (collaboration with Grid'5000)
 - ▶ Based on representations and tools developed by G5K
 - ▶ >50 versions since public availability – and counting
 - ▶ Still working on: better firmware version management
- ▶ Appliance management
 - ▶ Configuration, versioning, publication
 - ▶ Appliance meta-data via the appliance catalog
 - ▶ Orchestration via OpenStack Heat
- ▶ Monitoring and logging
- ▶ **However... the user still has to keep track of this information**

KEEPING TRACK OF EXPERIMENTS

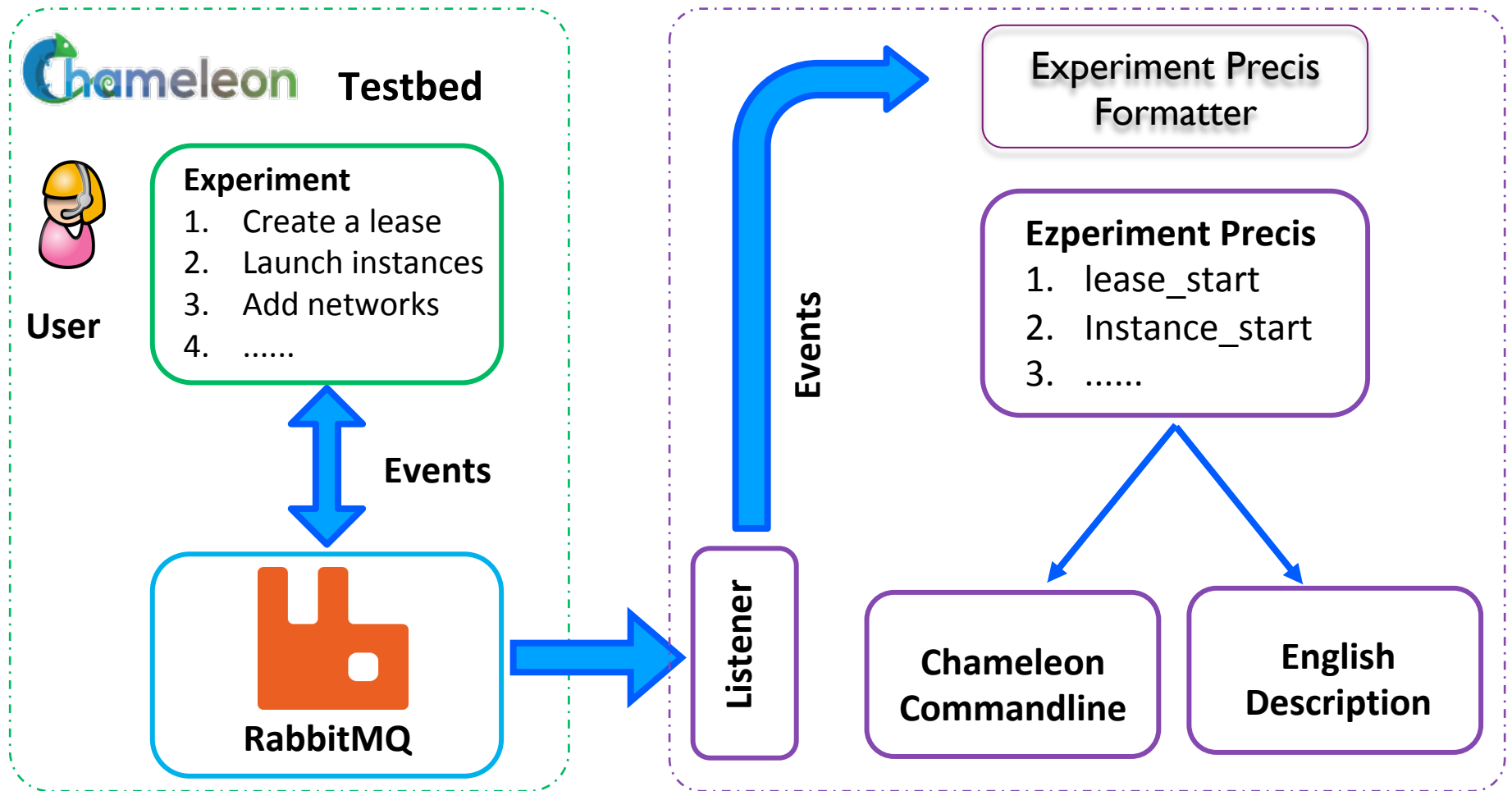
- ▶ Everything in a testbed is a recorded event
 - ▶ The resources you used
 - ▶ The appliance/image you deployed
 - ▶ The monitoring information your experiment generated
 - ▶ Plus any information you choose to share with us: e.g., “start power_exp_23” and “stop power_exp_23”
-

- ▶ **Experiment précis:** information about your experiment made available in a “consumable” form

REPEATABILITY: EXPERIMENT PRÉCIS

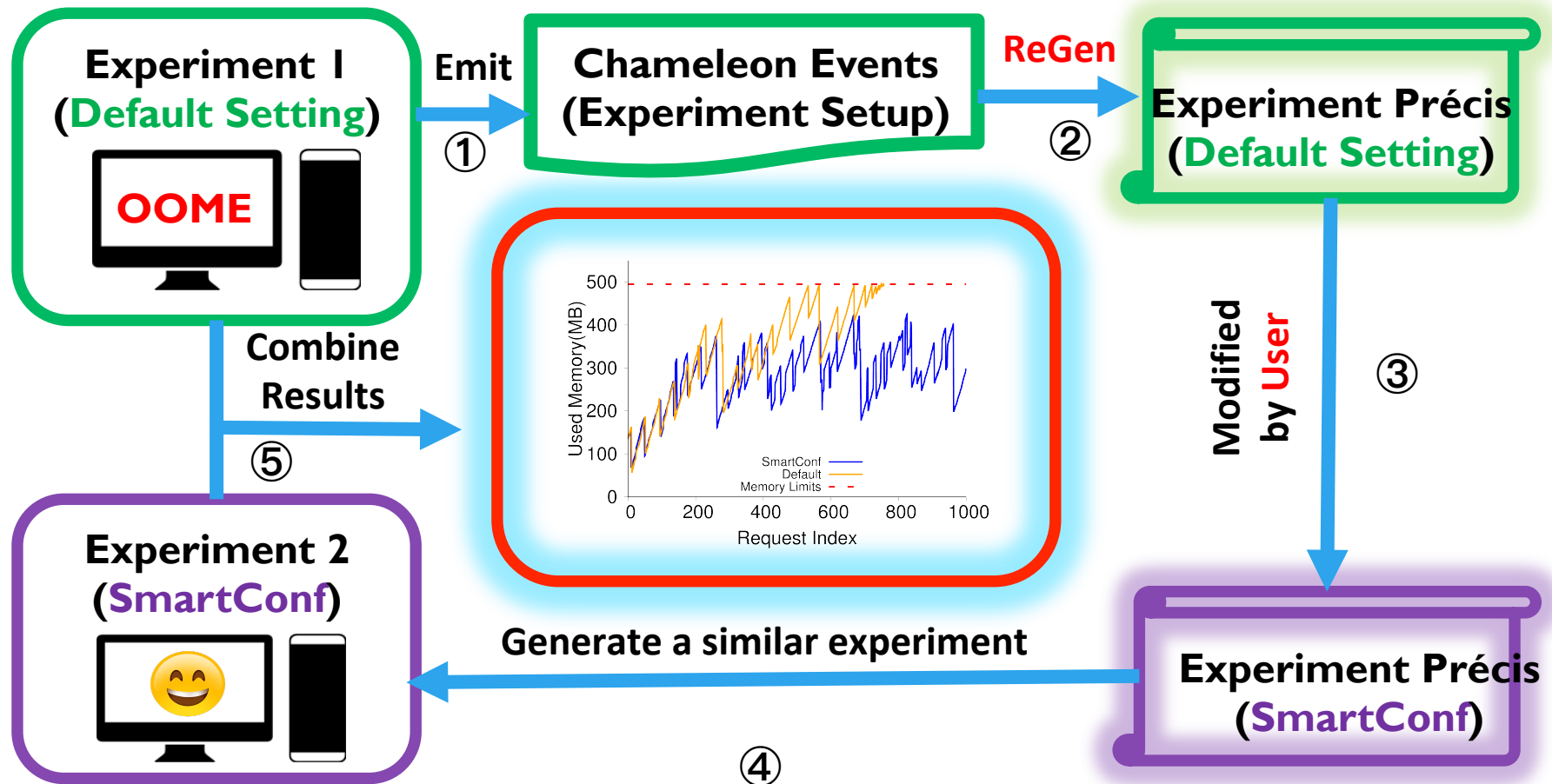


EXPERIMENT PRÉCIS IMPLEMENTATION



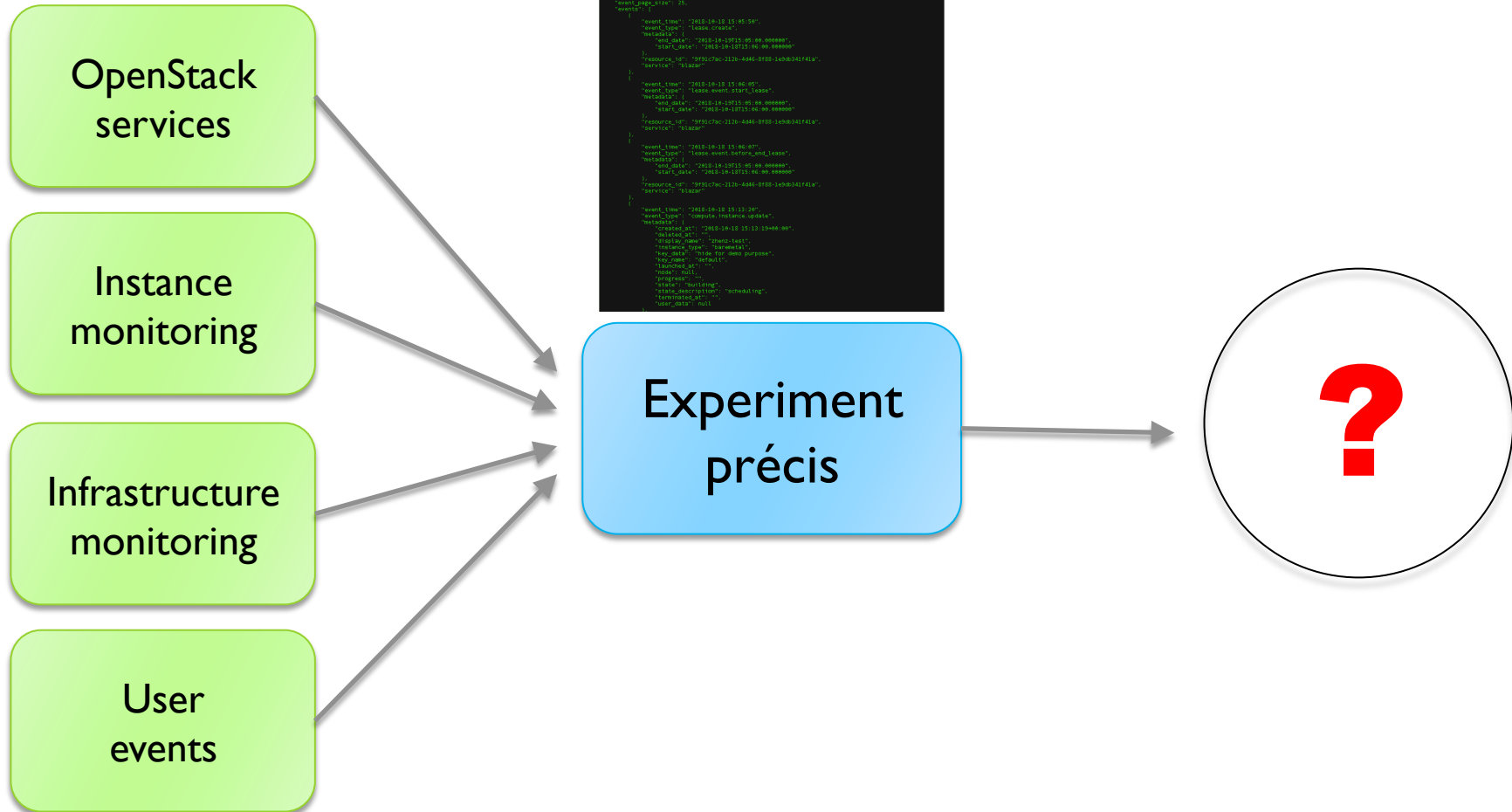
SCI8 poster: "Reproducibility as Side-Effect"

EXPERIMENT PRÉCIS: A CASE STUDY



Based on Wang et al., Understanding and Auto-Adjusting Performance-Sensitive Configurations. ASPLOS, 2018

REPEATABILITY: EXPERIMENT PRÉCIS



WHAT DOES IT MEAN TO DOCUMENT A PROCESS?

- ▶ Requirements
 - ▶ Human readable/modifiable format
 - ▶ Integrates well with ALL aspects of experiment management
 - ▶ Bit by bit replay – allows for bit by bit modification (and introspection) as well – element of interactivity
 - ▶ Support story telling: allows you to explain your experiment design and methodology choices
 - ▶ Has a direct relationship to the actual paper that gets written
 - ▶ Can be version controlled and easily shared
 - ▶ Sustainable, a popular open source choice
- ▶ Implementation options
 - ▶ Orchestrators: Heat, the dashboard, and Flame
 - ▶ Notebooks: Jupyter, Nextjournal

JUPYTER ON CHAMELEON



```
File Edit View Run Kernel Tabs Settings Help
Der... ip yr > | ... | jovyana@2: X | noteb... | FiveLineDr... X
Python 3
7.0 (3): In [3]: import paramiko

def exec_cmd(ssh, cmd):
    stdout, stderr = ssh.exec_command(cmd)
    return stdout.read().decode('utf-8')

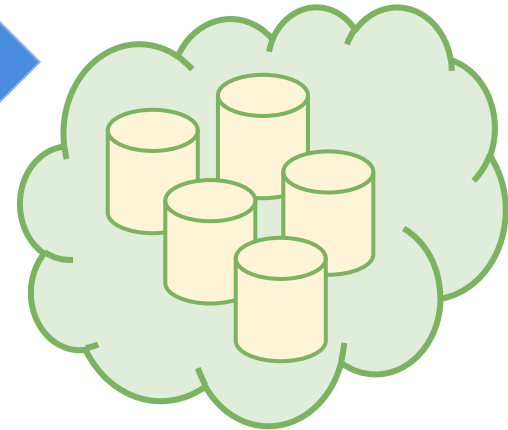
# Allocating floating IP... still to do ;)
# '129.114.108.67'
floating_ip = '129.114.108.28'

if floating_ip is None:
    raise Exception('Remember to set the floating IP okay')

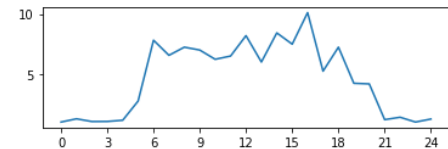
ssh = paramiko.SSHClient()
ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
try:
    print('Connecting!')
    ssh.connect(floating_ip, username='cc')
except paramiko.AuthenticationException:
    print('[-] Authentication Exception! ...')
except paramiko.SSHException:
    print('[-] SSH Exception! ...')

print('Connected! Want some proof?')
print(exec_cmd(ssh, 'ls -al'))

Connecting
Connected! Want some proof?
```

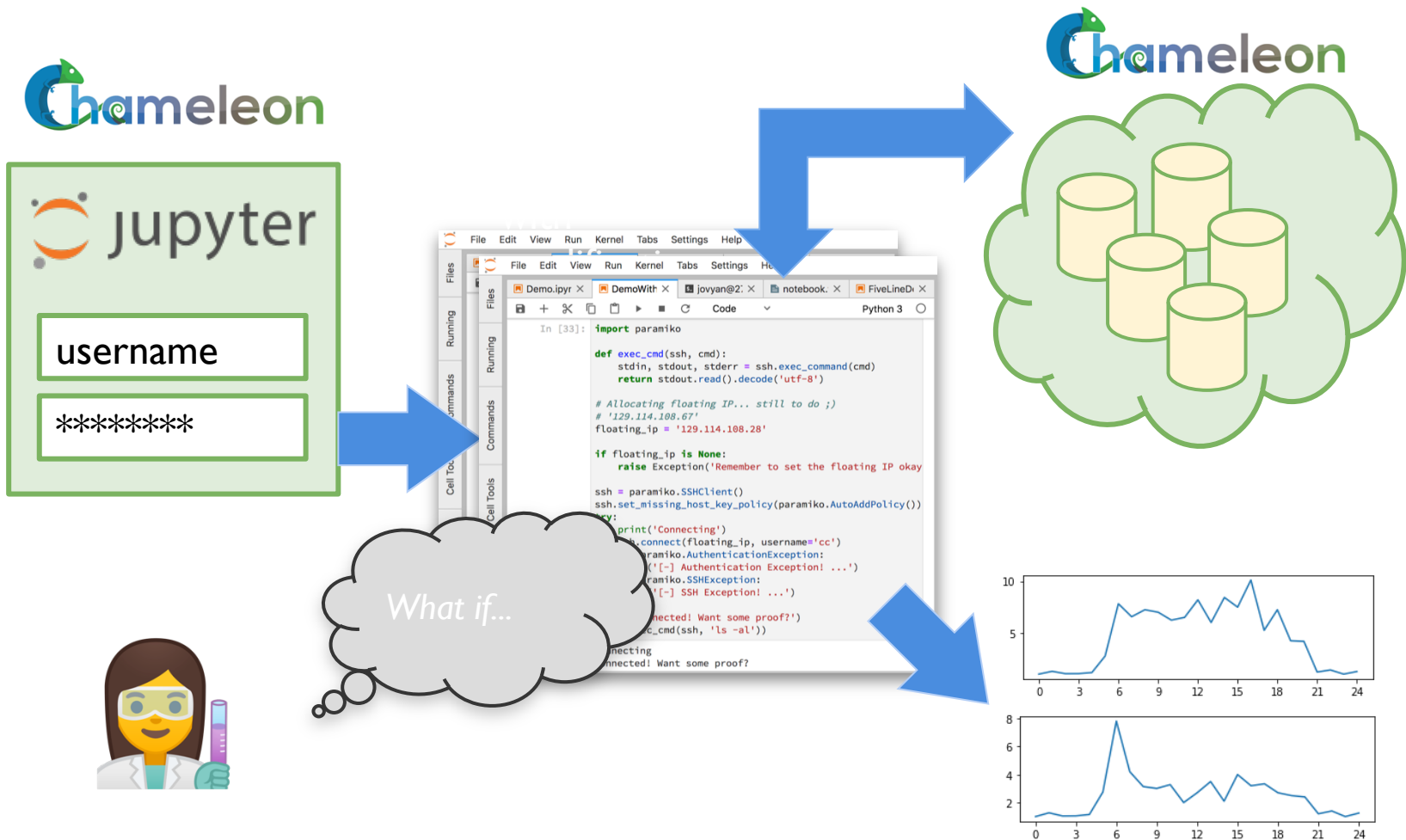


L^AT_EX



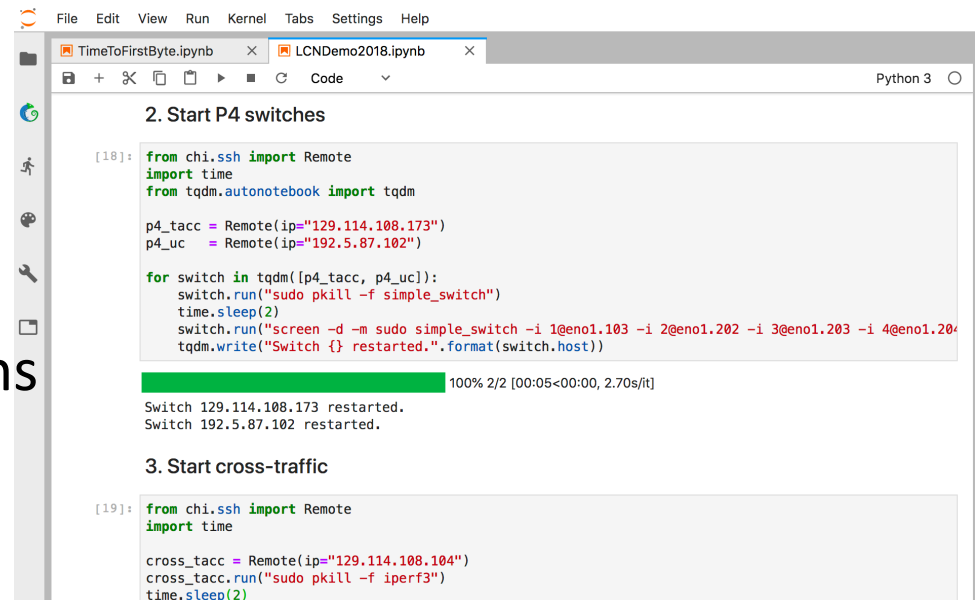
publishing

JUPYTER ON CHAMELEON



CHAMELEON JUPYTER INTEGRATION

- ▶ Storytelling with Jupyter
 - ▶ text, process, results
- ▶ Jupyter as an interface to Chameleon
 - ▶ All the main testbed functions
 - ▶ Jupyter.chameleoncloud.org
 - ▶ “Hello World” template
- ▶ Create and modify your experiment bit by bit
- ▶ Screencast of a complex experiment
 - ▶ <https://vimeo.com/297210055>



The screenshot shows a Jupyter Notebook window with two tabs: 'TimeToFirstByte.ipynb' and 'LCNDemo2018.ipynb'. The active tab is 'LCNDemo2018.ipynb', which contains two code cells. The first cell, labeled '[18]:', is titled '2. Start P4 switches' and contains Python code that uses the 'Remote' class to connect to two hosts (129.114.108.173 and 192.5.87.102) and restarts switches on both. A progress bar shows 100% completion. The second cell, labeled '[19]:', is titled '3. Start cross-traffic' and contains code to connect to a third host (129.114.108.104) and start a cross-traffic test using 'iperf3'.

```
[18]: from chi.ssh import Remote
import time
from tqdm.autonotebook import tqdm

p4_tacc = Remote(ip="129.114.108.173")
p4_uc = Remote(ip="192.5.87.102")

for switch in tqdm([p4_tacc, p4_uc]):
    switch.run("sudo pkill -f simple_switch")
    time.sleep(2)
    switch.run("screen -d -m sudo simple_switch -i 1@eno1.103 -i 2@eno1.202 -i 3@eno1.203 -i 4@eno1.204")
    tqdm.write("Switch {} restarted.".format(switch.host))

Switch 129.114.108.173 restarted.
Switch 192.5.87.102 restarted.

3. Start cross-traffic

[19]: from chi.ssh import Remote
import time

cross_tacc = Remote(ip="129.114.108.104")
cross_tacc.run("sudo pkill -f iperf3")
time.sleep(2)
```

PARTING THOUGHTS

- ▶ Testbeds are not just a place to do experimentation – they provide a common denominator for sharing of experimental work and artifacts
 - ▶ Documenting the process is one side of the coin
 - ▶ Maintaining conditions in which it can be carried out repeatedly is another
- ▶ We can do better than erase the reproducibility dilemma: if it makes my work easier to document the process – I will
- ▶ We would love your feedback!
 - ▶ Go to jupyter.chameleoncloud.org



www.chameleoncloud.org

Help us all dream big:

www.chameleoncloud.org

keahey@anl.gov

SEPTEMBER 17, 2019 15

