Experimentation problematics (on Grid'5000 but not only)

Joseph Emeras

sc-camp'11 – July 12 2011

Outline

Outline

Experimenting

How to ensure our experiments are valuable ?

Experimentation problematics

Experimentation dedicated research platforms

- Complex systems (even advanced users can be fooled)
- Heterogeneity
- Many users
  - regular experiment disturbance
  - system overload
  - users' silliness
- Hardware failures, bugs, configuration mistakes, platform maintenance, ...

How to improve experiments reliability

Where it begins

- You (read and respect the charter)
- Your code
- Your experiment plan
- The experiment environment

Outline

Context and Motivations

Experimentation dedicated research platforms

- Deploy and execute some code on a set of nodes at a time
- Some case – deploy user software environment
  Users :
  - + can control their experiment environment
  - − have to manage it
- Flexibility / Constraints for the user

Experimental process

Experiment

- Operation w/ a Specific Process (under control)
- Conditions (are they really controlled ?)
  - the experiment instance parameters (input data, configuration settings)
  - its execution environment : software and hardware (and its state)
- Results comparison and analysis (experiments conditions may differ)

Experimental process

Experiment

- Operation w/ a Specific Process (under control)
- Conditions (are they really controlled ?)
  - the experiment instance parameters (input data, configuration settings)
  - its execution environment : software and hardware (and its state)
- Results comparison and analysis (experiments conditions may differ)

Experiment results comparison

Experimental process

Experiment

- Operation w/ a Specific Process (under control)
- Conditions (are they really controlled ?)
  - the experiment instance parameters (input data, configuration settings)
  - its execution environment : software and hardware (and its state)
- Results comparison and analysis (experiments conditions may differ)

Experiment results comparison

- Reproducibility of the results (mandatory for comparison, matter of trust)

Experimental process

Experiment

- Operation w/ a Specific Process (under control)
- Conditions (are they really controlled ?)
  - the experiment instance parameters (input data, configuration settings)
  - its execution environment : software and hardware (and its state)
- Results comparison and analysis (experiments conditions may differ)

Experiment results comparison

- Reproducibility of the results (mandatory for comparison, matter of trust)
- Reproducibility of the experiment (experiment plan)

Experimental process

Experiment

- Operation w/ a Specific Process (under control)
- Conditions (are they really controlled ?)
  - the experiment instance parameters (input data, configuration settings)
  - its execution environment : software and hardware (and its state)
- Results comparison and analysis (experiments conditions may differ)

Experiment results comparison

- Reproducibility of the results (mandatory for comparison, matter of trust)
- Reproducibility of the experiment (experiment plan)
- Recreate the experiment observation conditions

Experimentation dedicated research platforms

Controlling the software environment

- Software environment built for the experiment : well adapted.
- Fine tuning of what impact the experiment run : deep analysis.

Experimentation dedicated research platforms

Controlling the software environment

- Software environment built for the experiment : well adapted.
- Fine tuning of what impact the experiment run : deep analysis.

Experiment results reproducibility, needs :

- Redo the experiment
- Reproduce the experiment conditions
- Be able to get the same environment $\rightarrow$ Software
  - history : keep trace of the build
  - replay : keep the environment or the ability to recreate it

Problem Details and Background

Experiment software environment creation

- generally not a “one shot” process
- from the basis to the final version : setup and debug stages

Problem Details and Background

Experiment software environment creation

- generally not a "one shot" process
- from the basis to the final version : setup and debug stages

Reusing the user environment – annoyances

- user environment "aging" $\rightarrow$ may become obsolete
- deployed on a totally different node configuration for comparison

Problem Details and Background

Experiment software environment creation

- generally not a "one shot" process
- from the basis to the final version : setup and debug stages

Reusing the user environment – annoyances

- user environment "aging" $\rightarrow$ may become obsolete
- deployed on a totally different node configuration for comparison
- need the ability to recreate it **in the same way**

Problem Details and Background

Experiment software environment creation

- generally not a "one shot" process
- from the basis to the final version : setup and debug stages

Reusing the user environment – annoyances

- user environment "aging" $\rightarrow$ may become obsolete
- deployed on a totally different node configuration for comparison
- need the ability to recreate it **in the same way**
- need to control what will change between 2 reconstructions

Problem Details and Background

Experiment software environment creation

- generally not a "one shot" process
- from the basis to the final version : setup and debug stages

Reusing the user environment – annoyances

- user environment "aging" $\rightarrow$ may become obsolete
- deployed on a totally different node configuration for comparison
- need the ability to recreate it **in the same way**
- need to control what will change between 2 reconstructions

Reconstructability of the experiment software environment

- **Keep track of what have been done during creation**
  - History
  - Reconstruction
- A tool is necessary !

Kameleon : a tool to generate software appliances

A tool that generates software appliances from an environment description :

- Recipe (high level, semantic, YAML)
- Steps (low level, technical, Shell commands + Power-K commands)

Kameleon combines this to :

- Model the software environment : recreate it **in the same way**
- Everything described : keep log of what has been done

Goals :

- Simple and easy to handle
- Reuse previous work

How does it work ?

- Step : one technical action (software installation, configuration . . .)
- Recipe : combination of steps that lead to environment construction
- Environment generation in an isolated environment (chroot method)
- **Execution contexts** (Power-K commands)
  action done :
  - machine running Kameleon
  - isolated environment

Recipes and Steps : Example

Recipe sample: Creation of a KVM Debian image

```
# ### debian.yaml Kameleon recipe sample ###
global:
  distrib: debian
  # Debian specific
  debian_version_name: squeeze
  distrib_repository: http://ftp.fr.debian.org/debian/
  #
  # Architecture
  arch: amd64
  kernel_arch: "amd64"
  #
 steps:
  - bootstrap
  - system_config
  - root_passwd
  - mount_proc
  - kernel_install
  - strip
  - umount_proc
  - build_appliance:
    - clean_udev
    - create_raw_image
    - create_nbd_device
    - mkfs
    - mount_image
    - copy_system_tree
    - install_grub
    - umount_image
    - save_as_raw
  - clean
```

Recipes and Steps : Example

Recipe sample: Creation of a KVM Debian image

```
# ### debian.yaml Kameleon recipe sample ###
global:
 distrib: debian
 # Debian specific
 debian_version_name: squeeze
 distrib_repository: http://ftp.fr.debian.org/debian/
 #
 # Architecture
 arch: amd64
 kernel_arch: "amd64"
 #
steps:
 - bootstrap
 - system_config
 - root_passwd
 - mount_proc
 - kernel_install
 - strip
 - umount_proc
 - build_appliance:
   - clean_udev
   - create_raw_image
   - create_nbd_device
   - mkfs
   - mount_image
   - copy_system_tree
   - install_grub
   - umount_image
   - save_as_raw
 - clean
```

Debian Kernel installation step

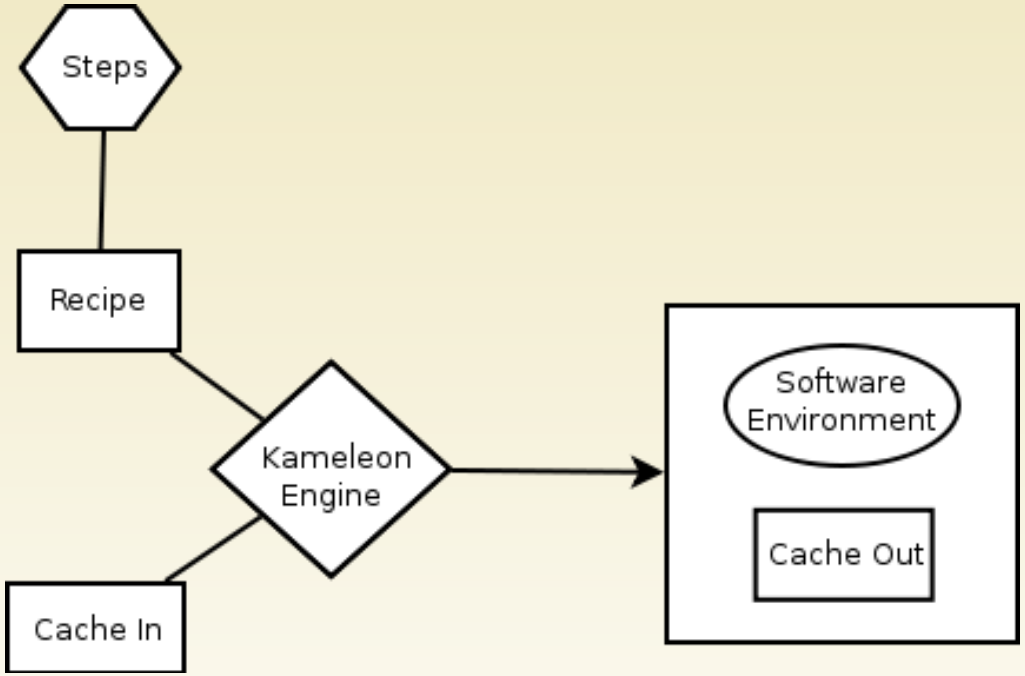
```
kernel_install:
- kernel-img_conf:
  - write_file:
    - /etc/kernel-img.conf
    - |
      do_symlinks = yes
      relative_links = yes
      do_bootloader = yes
      do_bootfloppy = no
      do_initrd = yes
      link_in_boot = no
- kernel_install:
  - exec_chroot: bash -c "DEBIAN_FRONTEND
    =noninteractive apt-get -y --force-yes
    install linux-image-$$kernel_arch"
```

Debian basis installation step

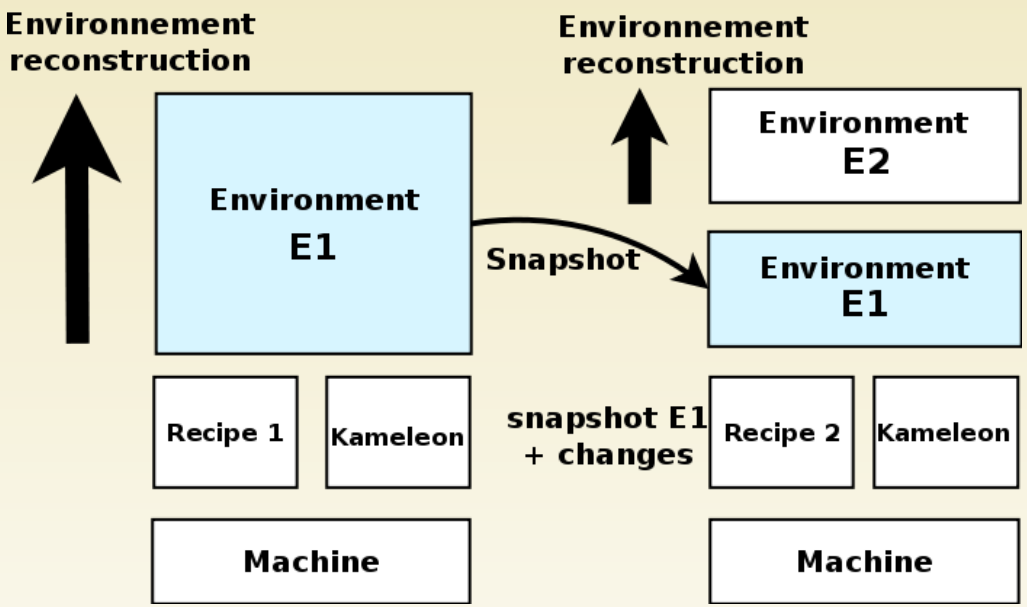
```
bootstrap:
 - debootstrap:
   - exec_appliance:  debootstrap --arch=
     $$arch $$debian_version_name
     $$chroot/ $$distrib_repository
```

Kameleon – in the core

Kameleon process



Snapshot Feature

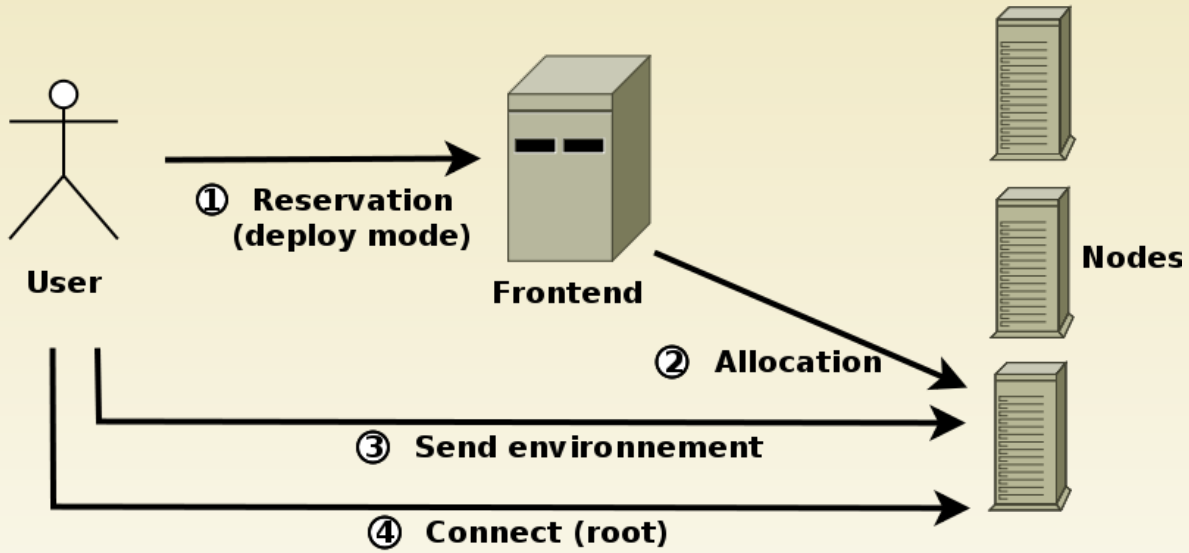


Kameleon features

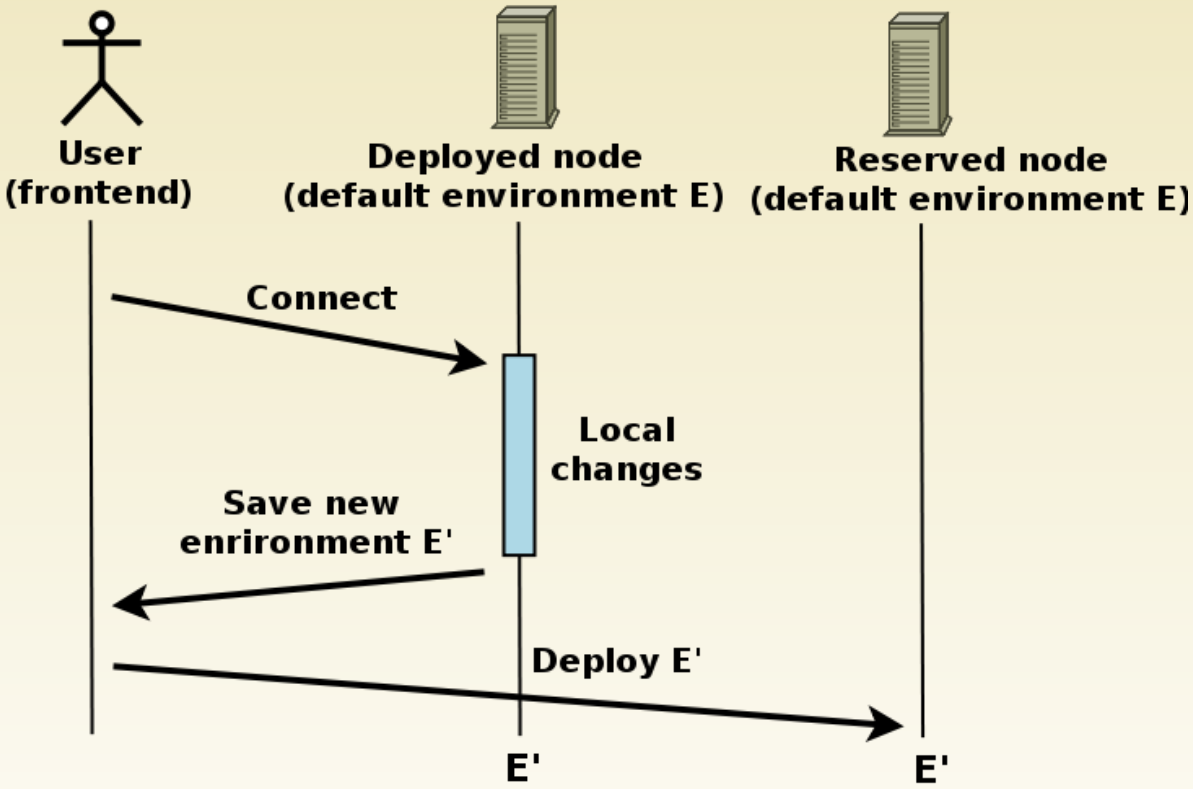
- File operations : append, create (Power-K commands)
- User control (shell)
- Steps dependencies
- Provided ingredients : recipes and steps for Debian, G5K, KVM, Xen ...

Use Cases : Grid'5000

Environment deployment on Grid'5000 (G5K)

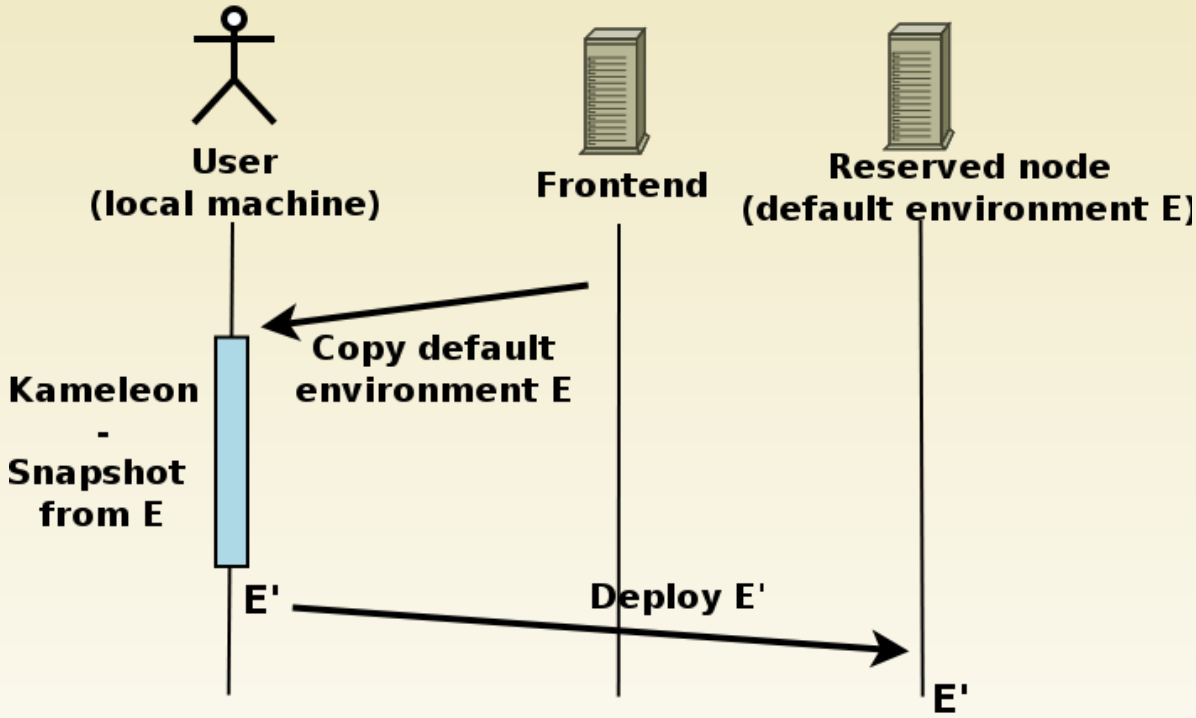


Regular environment customization on G5K

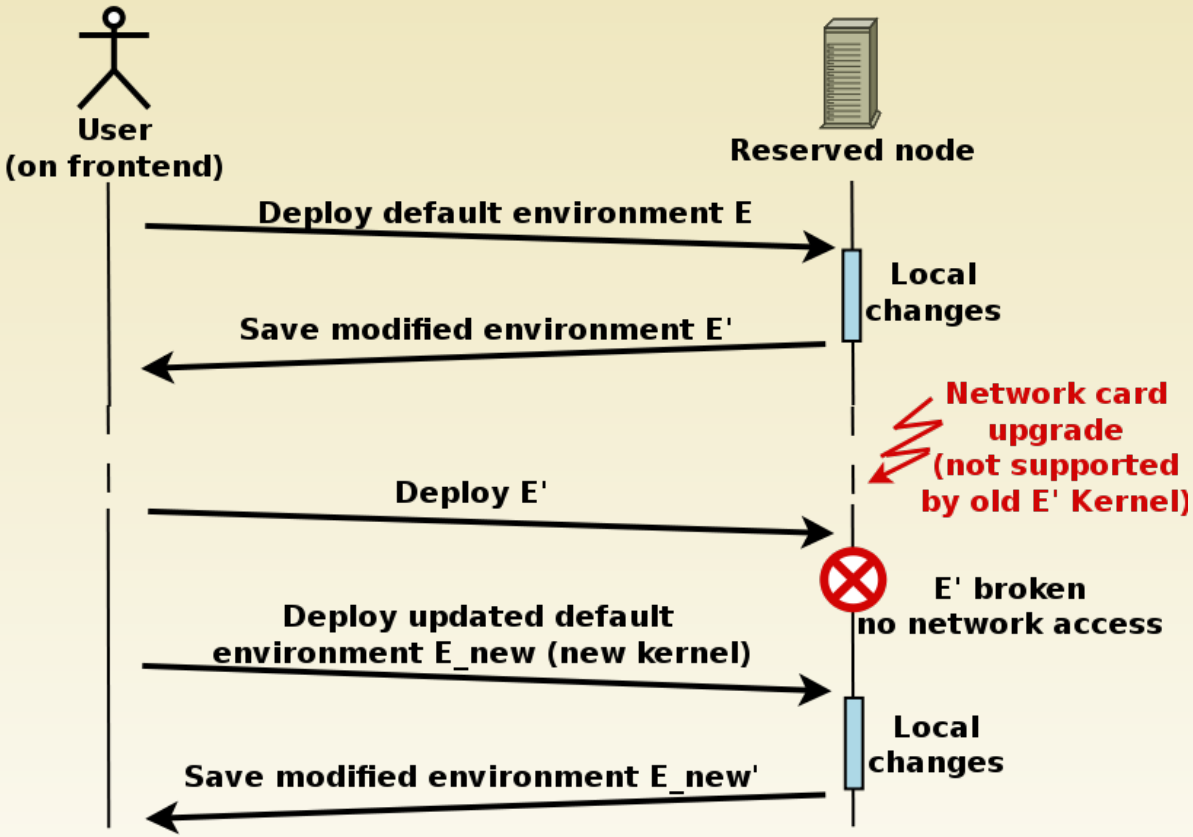


Use Cases : Grid'5000

From a reference image + Kameleon : customize and deploy

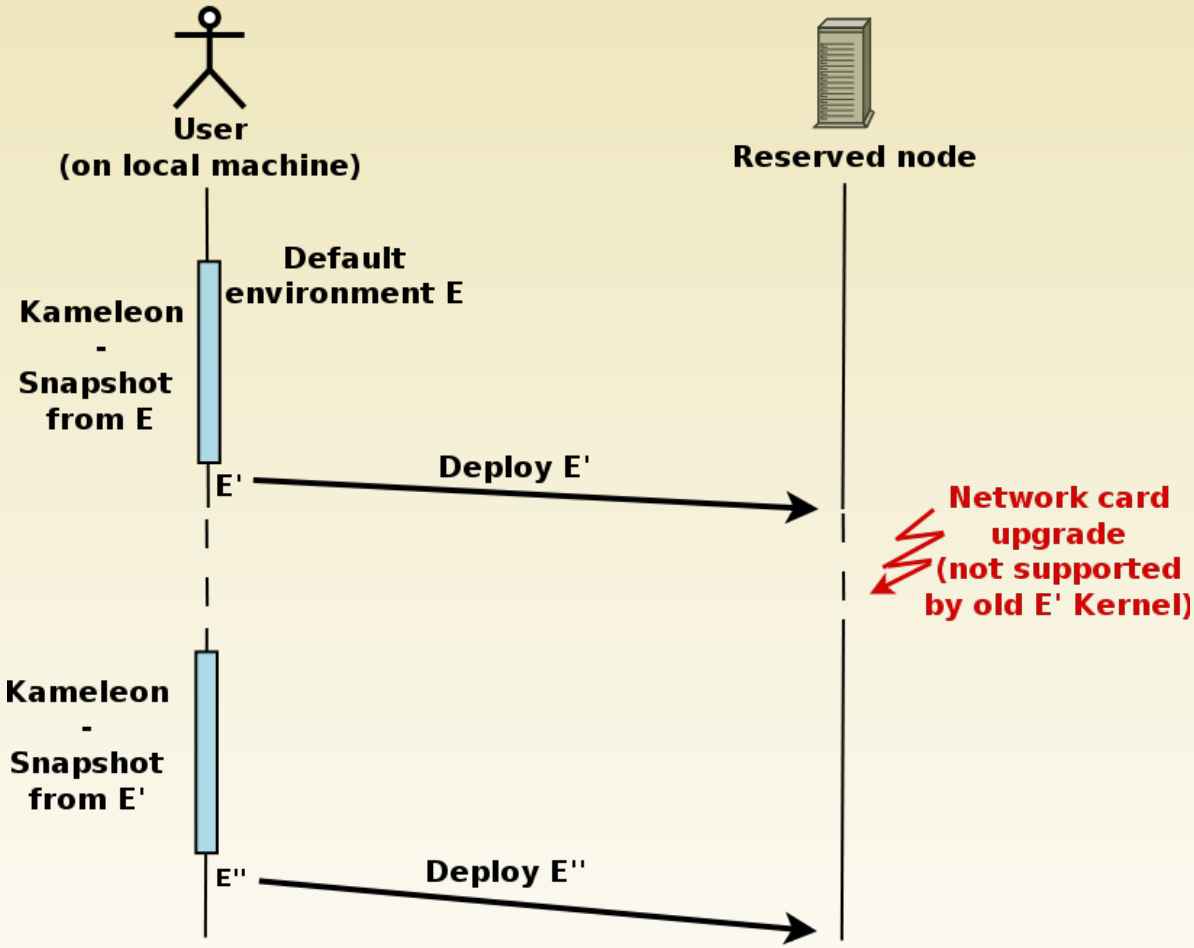


User environment aging problem – without Kameleon



E' ≠ E_new'
Software versions differ

User environment aging problem – with Kameleon



Conclusion and Perspectives

- Kameleon : one solution for reconstructability
  - as complete as possible
  - lightweight and easy to handle
  - many steps provided, **share** !
  - git available : scm.gforge.inria.fr/gitroot/kameleon
- Works in progress :
  - File System steps
  - Batch Scheduler steps
  - Fedora, SL5 steps

Outline

Experimenting on G5K : Tricks

- Platform heterogeneity (even inside a cluster)
- `https://www.grid5000.fr/mediawiki/index.php/Grid5000:Gotchas`
- Default environments – tendency to merge
- Users behaviours
- Code behaviours (network, NFS)
- Different softwares installed on frontends
- Ethernet network don't scale (better use IPoIB or IPoMyrinet)

Experimenting on G5K : Treats

- **Katapult** Wrapper around kadeploy. Handles retries when failures + execution of script after deployment
- **terminator** (or alternatives) – Several X terminals in one window
- **screen** Run experiment unattended, take control back when needed
- **xargs** Simple way to run commands in parallel (-P)
  cat nodeslist | xargs -P10 -I HOST -n1 ssh HOST hostname
- **Taktuk** Efficiently run commands on a large number of nodes
- `https://www.grid5000.fr/mediawiki/images/G5k_cheat_sheet.pdf`
- `https://www.grid5000.fr/mediawiki/index.php/User:Lnussbaum`

Experimenting on G5K : Good practices

Scripting complex experiments

- Very difficult process
- But :
  - Increases reproducibility of experiments
  - Better control of the experiment
- No unique Good Way
  We are not there yet, unfortunately
- Some advices to keep in mind :
  - Think your scripts for easy debugging
  - Split your experiments into steps
  - Start with independent and idempotent scripts
  - Use stable building blocks (standard tools)
  - Keep as much data as possible (verbosity, logs)

Experimenting on G5K : Good practices

- **Never** compile on the frontend !!!
- Read the charter
- Use vlans to isolate your network
  - global, local, isolated
  - no network disturbance
  - manage bandwith between nodes
  - `https://www.grid5000.fr/mediawiki/index.php/KaVLAN`
  - `https://www.grid5000.fr/mediawiki/index.php/Network_isolation_on_Grid'5000`
- Use all the preceding advices :-)

Thank you for your attention

That's all folks