PyConPl '13
Szczyrk, 17-20 X

The Anaconda Fedora and RHEL installer

Martin Kolman
Red Hat
http://www.modrana.org/pyconpl2013
martin.kolman@gmail.com
@M4rtinK

What is Anaconda ?

- *a large non-venomous snake found in tropical South America*
- *a movie*
- *a small town in Montana, USA*
- **the default installer for Fedora, Red Hat Enterprise Linux (RHEL) distributions and their derivatives**

Anacondas job

- it installs a Linux distribution to a computer
  - might or might not have an OS already installed
- has to detect hardware and configure it as needed
- prepares storage
- installs and configures software
- reboots the computer

Computer

- might be a X86 PC
- a server (or a few thousand of them)
- some ARM board (Raspberry Pi, Beaglebone, ...)
- a PowerPC computer
- a IBM S390 *mainframe*

Anaconda needs to be able to run on all of those.

Storage

- there are lots of storage devices and filesystems
- basic HDDs with normal filesystems, basic RAID
- advanced (enterprise) storage devices:
  - iSCSI & multipath devices
  - fibrechannel
  - enterprise grade RAID setups

Anaconda needs to be able to detect, partition, format and mount them.

Installing and configuring

- a community developed Linux distribution such as Fedora is a constantly changing landscape
  - *hostname* might just drop an option once in a while
  - *GTK* might just starts calling ***exit*** if X is not yet running instead of raising an exception
  - */tmp* gets moved on a ramdisk

Anaconda needs to keep up with all these changes to install the system correctly.

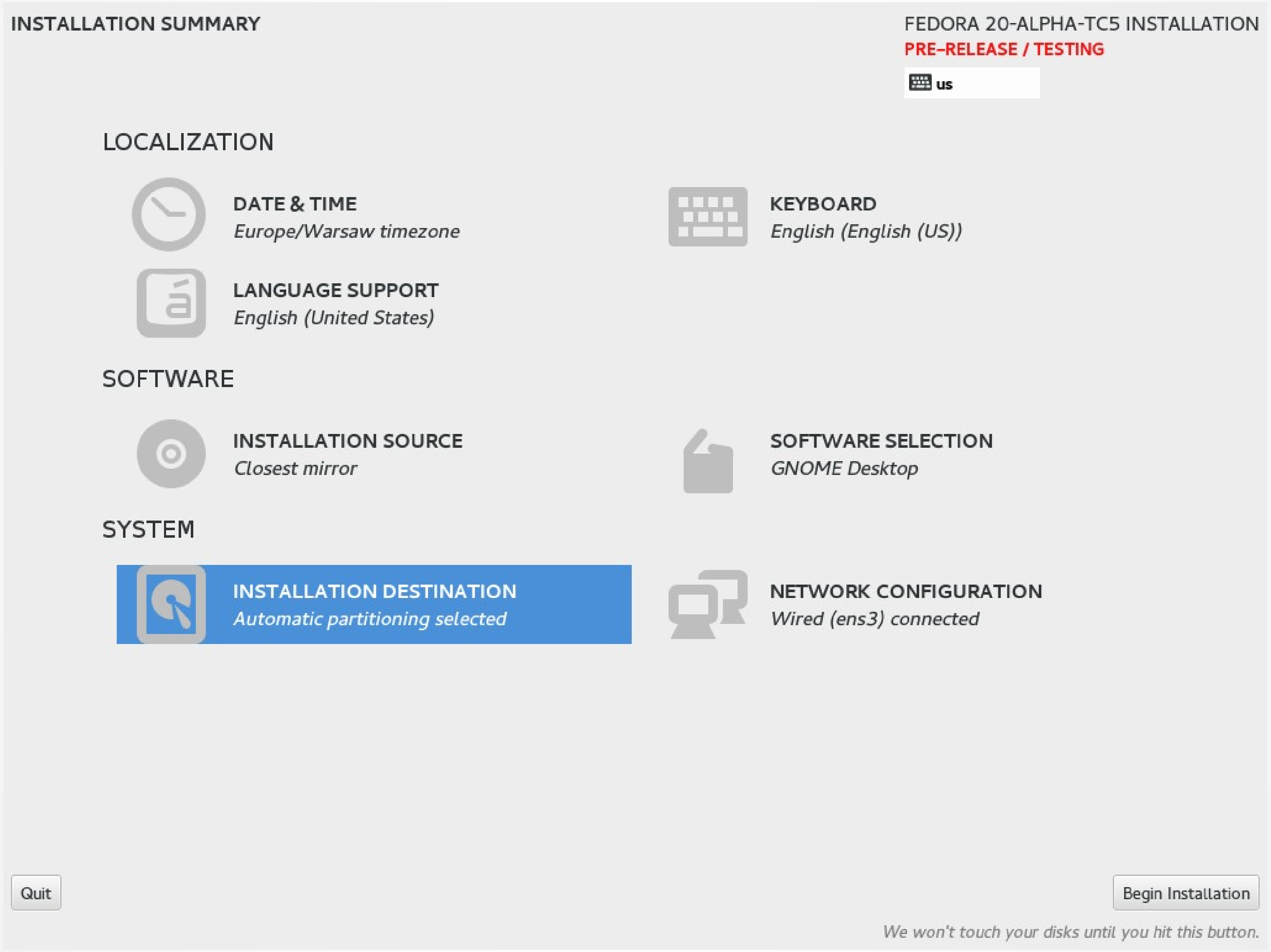
Two and half installations

Anaconda has 2.5 ways of installing the OS

1. interactive installation
   - configured through the UI by the user
2. automated install
   - running without user interaction
   - configured by a *kickstart* recipe

2.5. hybrid installation
   - some values are set from *kickstart*, the rest from UI

GUI


INSTALLATION SUMMARY

FEDORA 20-ALPHA-TC5 INSTALLATION
PRE-RELEASE / TESTING
us

LOCALIZATION

DATE & TIME
Europe/Warsaw timezone

KEYBOARD
English (English (US))

LANGUAGE SUPPORT
English (United States)

SOFTWARE

INSTALLATION SOURCE
Closest mirror

SOFTWARE SELECTION
GNOME Desktop

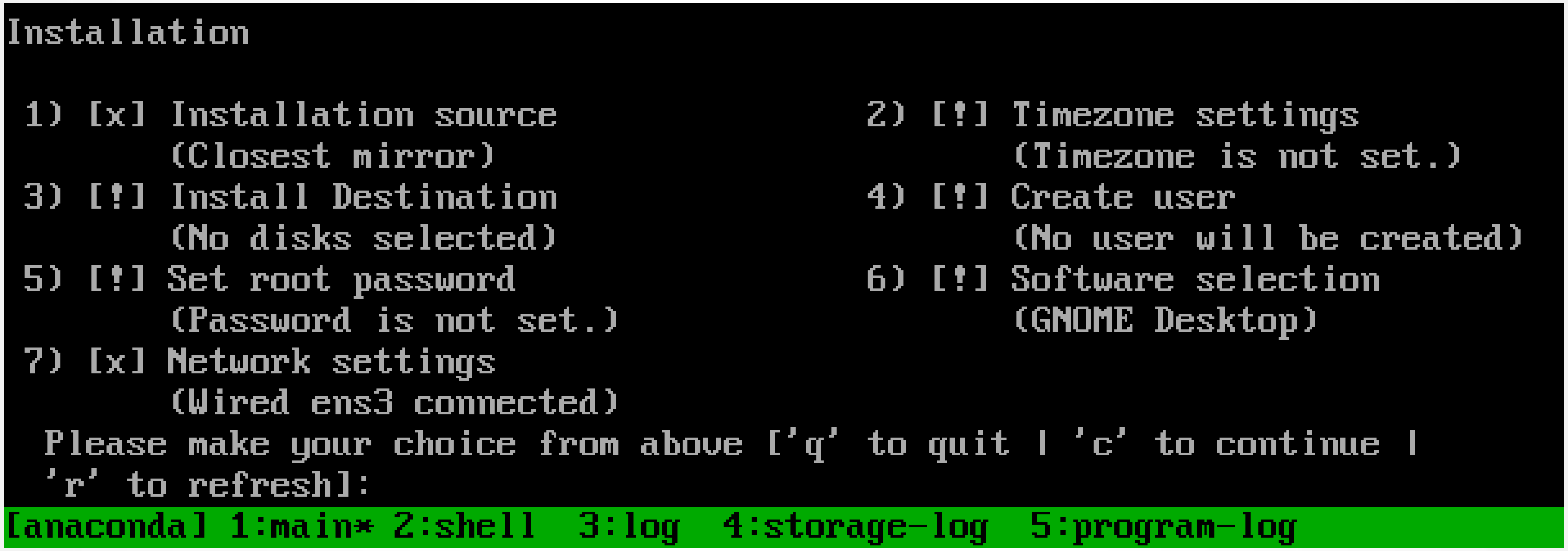
SYSTEM

INSTALLATION DESTINATION
Automatic partitioning selected

NETWORK CONFIGURATION
Wired (ens3) connected

We won't touch your disks until you hit this button.

TUI



```
Installation

 1) [x] Installation source                     2) [!] Timezone settings
        (Closest mirror)                               (Timezone is not set.)
 3) [!] Install Destination                     4) [!] Create user
        (No disks selected)                            (No user will be created)
 5) [!] Set root password                       6) [!] Software selection
        (Password is not set.)                         (GNOME Desktop)
 7) [x] Network settings
        (Wired ens3 connected)
  Please make your choice from above ['q' to quit | 'c' to continue |
  'r' to refresh]:
[anaconda] 1:main* 2:shell  3:log  4:storage-log  5:program-log
```

- write only
  - would even work on a teletype (*AKA the S390 terminal*)

History

- first commit in VCS dates back to 1999

```
commit 785d44bf73ccc1d57d771895cc94112a34857809
Author: Matt Wilson <msw@redhat.com>
Date:   Sat Apr 24 03:57:59 1999 +0000

    the very start of a gui frontend for anaconda
```

- that's just for the GUI support, the installer itself is probably even older
- continuously developed and used ever since

```
-# This toplevel file is a little messy at the moment...
+# This toplevel file is a little messy at the moment... (2001-06-22)
+# ...still messy (2013-07-12)
```

Some stats

- currently 34k combined lines of code
  - down from the record high of 88k
- 36 contributors with at least 40 commits
- 7 developers with >1000 commits
- there are ~26000 commits in the Git repository taking up about **85 MB**
  - CVS was used at the beginning and later imported to Git

Architecture

- Anaconda is written in Python 2.7
- is using GTK 3 through the GObject introspection interface
- the GUI layout is specified using XML files generated by the Glade Interface Designer
- custom GTK widgets (such as the timezone map) are written in C and used through GI
- the TUI is using a custom write-only toolkit

Fun with threads

- Anaconda is a multithreaded application
  - it needs to do a lot of stuff at once and still react to user input
- so it needs to be threadsafe
- turns out, most libraries it is using are not ! :)
  - **GTK is not threadsafe !**
  - most DBUS bindings are not threadsafe !
  - even the YUM package manager is not threadsafe !

Untangling the mess

- only one specific thread needs to use the resources at the same time
  - hello GTK !
  - solution: message queues feeding the main thread
  - can be turned into a decorator, that forwards the function to be called by the main thread and then optionally returns the result back

```
@gtk_action_nowait
def _restart_spinner(self):
    spinner = self.builder.get_object("progressSpinner")
    spinner.show()
    spinner.start()
```

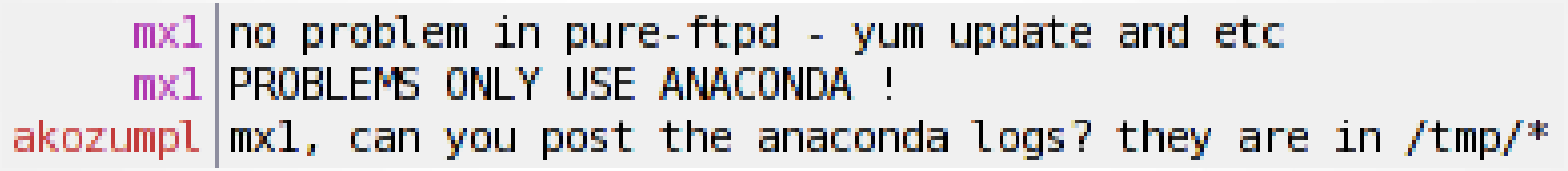
ThreadManager

- all Threads are started and tracked by a thread management module
- logs thread lifecycle
- provides easy checking if threads are running and waiting for threads to finish
- also handles exceptions getting caught by a random thread in Python

Debugging

- it is quite important that Anaconda works
- if an application breaks, the user is not able to use the application
- if Anaconda breaks, the users is not able to use their computer/server farm/mainframe

Logs

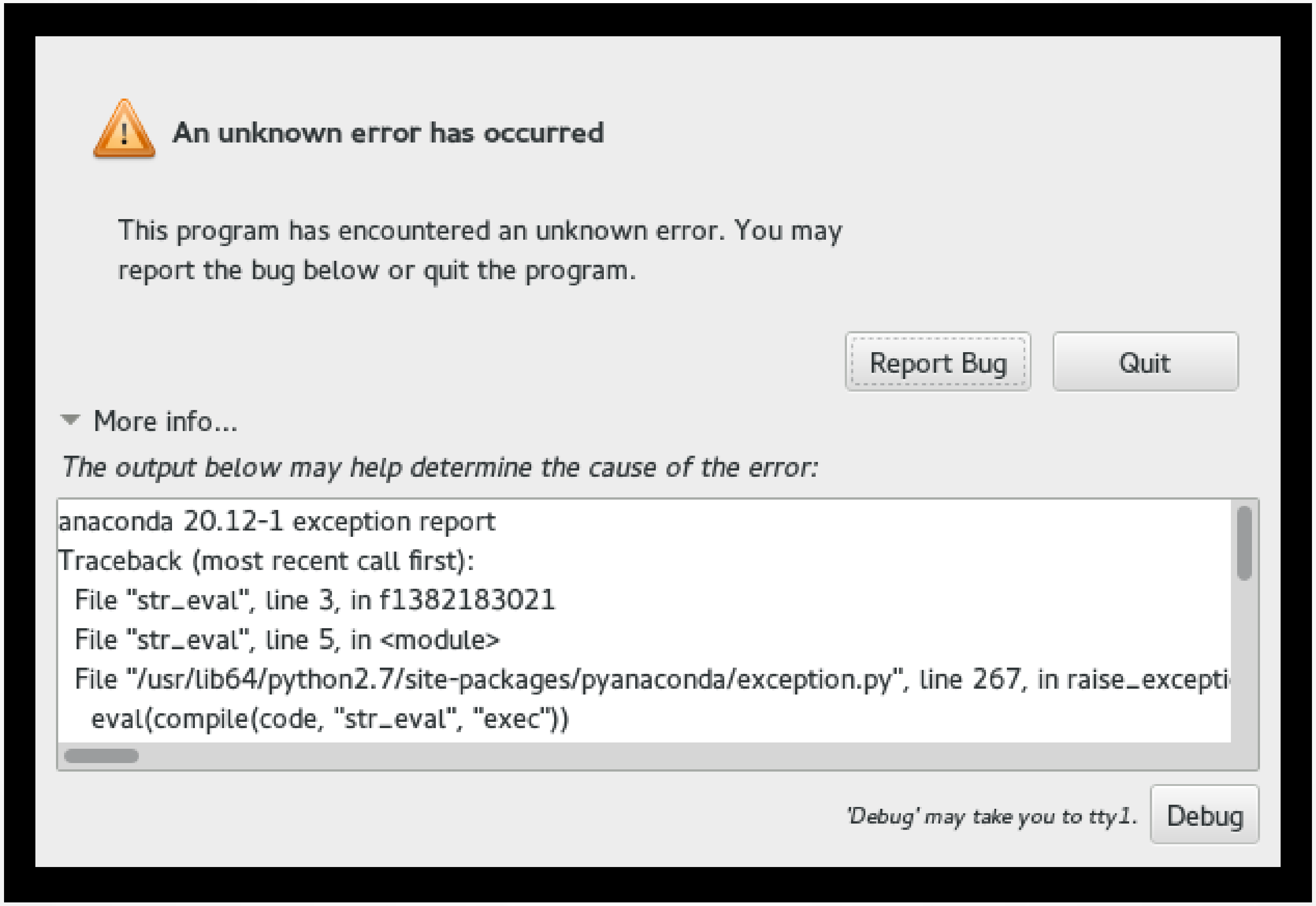


```
     mx1 | no problem in pure-ftpd - yum update and etc
     mx1 | PROBLEMS ONLY USE ANACONDA !
akozumpl | mx1, can you post the anaconda logs? they are in /tmp/*
```

- Anaconda logs quite a lot of stuff to /tmp/*
  - if installation succeeds, logs are copied to the installed system
- during the whole installation, there is a bash shell running on tty2, that can be used to check the log and generally probe the system

Interactive debugging

- Anaconda also has interactive debugging



An unknown error has occurred

This program has encountered an unknown error. You may report the bug below or quit the program.

The output below may help determine the cause of the error:

```
anaconda 20.12-1 exception report
Traceback (most recent call first):
  File "str_eval", line 3, in f1382183021
  File "str_eval", line 5, in <module>
  File "/usr/lib64/python2.7/site-packages/pyanaconda/exception.py", line 267, in raise_excepti
    eval(compile(code, "str_eval", "exec"))
```

'Debug' may take you to tty1.

Interactive debugging

- shows the traceback, stackframe and object dump in a scrollable window
- can submit debug data directly to Red Hat bugzilla or *scp* it to a server of your liking
- clicking the *debug* button switches you to tty1 and starts pdb

Development

- Anaconda is an open source project licensed under the GPLv2
- patches are submitted to the *anaconda-patches* mailing list
- they need to be *ACKed* by one of the core developers
- once *ACKed*, they are pushed to the Anaconda git repository and included in the next build

Development

- every distro release gets a Git branch, eventually
- fixes are backported back to supported branches
- RHEL5 (released 2007) and RHEL6 (released 2010) are still getting fixes
- RHEL7 branch is being stabilized

The children of Anaconda

All through the long history of the Anaconda project, one pattern always manifested itself.

New module would be added, grow with improvements and new features and then separate to live as independent libraries.

Not longer a part of Anaconda itself, they are still used as external dependencies, providing cleaner interface and allowing an independent release schedule and usage by other projects.

Pykickstart

- handles kickstart parsing and validation
- individual kickstart commands are defined as Python classes named after the Fedora release they were introduced
  - changed commands inherit the latest command class and change it's behavior
  - example:

    class `FC20_Firewall(FC13_Firewall)`
- only parsing, no command execution
- can also generate kickstart files from a list of properly configured command object instances
  - this is used by Anaconda to record all changes (even manual ones) done during installations into a comprehensive kickstart file, which is saved to the installed system
- has a comprehensive testsuite

blivet

“A blivet, also known as a poiuyt, devil's fork or widget, is an undecipherable figure, an optical illusion and an impossible object.”

- the aptly named blivet is a python storage library born from Anaconda storage handling code rewrite
- can handle a wide range of filesystems and storage devices
  - EXT2-4, XFS, BTRFS, HFS, NFS, NTFS, TMPFS, RAID, LVM, ...
- and do many operations on them
  - detection, creation, formatting, resizing, wiping, relabeling, fstab parsing and generation, ...

blivet

- is used as the Anaconda storage backend
- also used by the OpenLMI system management project for handling storage
- really really likes **kwargs
- has a comprehensive testsuite !
  - that needs needs to run as root
  - and does various partitioning operations on the current system :)

    (well, there is not really any other way of testing that all the operations blivet does are performed correctly without doing them on a real system and checking the result)

python-meh

- an exception handling module for Python
- once an exception happens, python-meh recursively dumps the complete object tree to a text file
- it also includes the traceback and variables from the current frame

pyparted

- provides python bindings the the *parted* partitioning utility

langtable

- while not direct Anaconda offspring, the langtable project was triggered due to need for advanced l10n handling in Anaconda
- maintains weighted tables of l10n data mappings
  - territory code<->languages, country<->keyboard layout<->language, translated language names, ...
- the table is a gziped XML parsed by *expat*
- there is a proposal to rewrite it to *Vala* and provide Python bindings through GObject introspection

Future plans

- using Python 3
  - Tentatively planed for Fedora 22
  - Depends on Python support in Python libs used
- using Wayland for the Anaconda minidistro
- improve documentation
- tests, test and more tests

Thanks!

- Questions ?