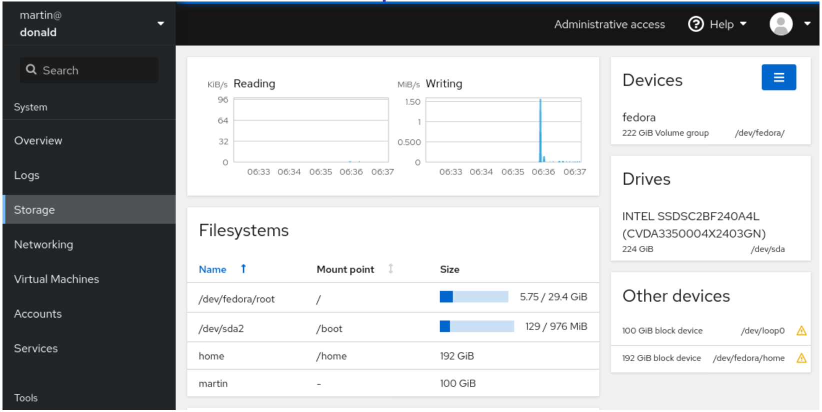
Cockpit Infrastructure

Martin Pitt, Sanne Raymaekers

Cockpit Team



Devices

fedora
222 GiB Volume group /dev/fedora/

Drives

INTEL SSDSC2BF240A4L (CVDA3350004X2403GN)
224 GiB /dev/sda

Filesystems

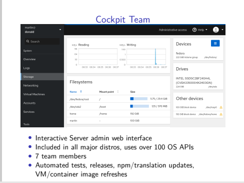
Name	Mount point	Size
/dev/fedora/root	/	5.75 / 29.4 GiB
/dev/sda2	/boot	129 / 976 MiB
home	/home	192 GiB
martin	-	100 GiB

Other devices

100 GiB block device /dev/loop0 ⚠

192 GiB block device /dev/fedora/home ⚠

- Interactive Server admin web interface
- Included in all major distros, uses over 100 OS APIs
- 7 team members
- Automated tests, releases, npm/translation updates, VM/container image refreshes



Cockpit Team

- Interactive Server admin web interface
- Included in all major distros, uses over 100 OS APIs
- 7 team members
- Automated tests, releases, npm/translation updates, VM/container image refreshes

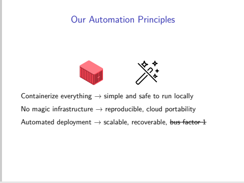
- Conceptually: Linux session running in a web browser; moral server equivalent of what GNOME is on a desktop
- talks to $>$ 100 system APIs, times $>$ 10 supported OSes $\rightarrow$ moving target, combinatorial explosion, things break all the time
- small team, heavily dependent on infrastructure
- automated testing, releasing, code hygiene, updating VM and container images

Our Automation Principles

Containerize everything → simple and safe to run locally

No magic infrastructure → reproducible, cloud portability

Automated deployment → scalable, recoverable, ~~bus factor 1~~



Our Automation Principles

Containerize everything → simple and safe to run locally
No magic infrastructure → reproducible, cloud portability
Automated deployment → scalable, recoverable, ~~bus factor 1~~

- Humans first: Make it simple and enjoyable to locally hack on tests, automation, CI
- happens that these very qualities make it easy to deploy to infra
- Containers are easy to reproduce, easy to run locally and on different cloud platforms/CI providers
- infra uses the exact same containers and commands as humans, just more powerful
- Deployed using publicly available ansible scripts

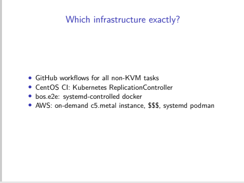
Which infrastructure exactly?

- GitHub workflows for all non-KVM tasks
- CentOS CI: Kubernetes ReplicationController
- bos.e2e: systemd-controlled docker
- AWS: on-demand c5.metal instance, $$$, systemd podman

2021-01-13

Cockpit Infrastructure

└─Which infrastructure exactly?


Which infrastructure exactly?

- GitHub workflows for all non-KVM tasks
- CentOS CI: Kubernetes ReplicationController
- bos.e2e: systemd-controlled docker
- AWS: on-demand c5.metal instance, $$$, systemd podman

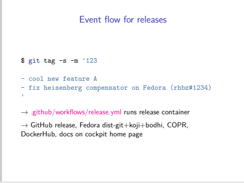
- GitHub's infra is unlimited, free, zero admin cost
- . . . . use for releases, container refreshes, and more
- tests need /dev/kvm access and internal network access depending on the tested OS
- CentOS CI ocp: powerful, free, many nodes; no internal tests; RCs
- e2e: 10 real-iron powerful machines; internal tests; systemd autorestart controlled docker instances
- AWS: on-demand test fallback in case e2e goes down, but pricey ($100/day)
- . . . permanent image server backup and log store; cheap virtualized instances

Event flow for releases

```
$ git tag -s -m '123

- cool new feature A
- fix heisenberg compensator on Fedora (rhbz#1234)
'
```

→ .github/workflows/release.yml runs release container

→ GitHub release, Fedora dist-git+koji+bodhi, COPR, DockerHub, docs on cockpit home page



Event flow for releases

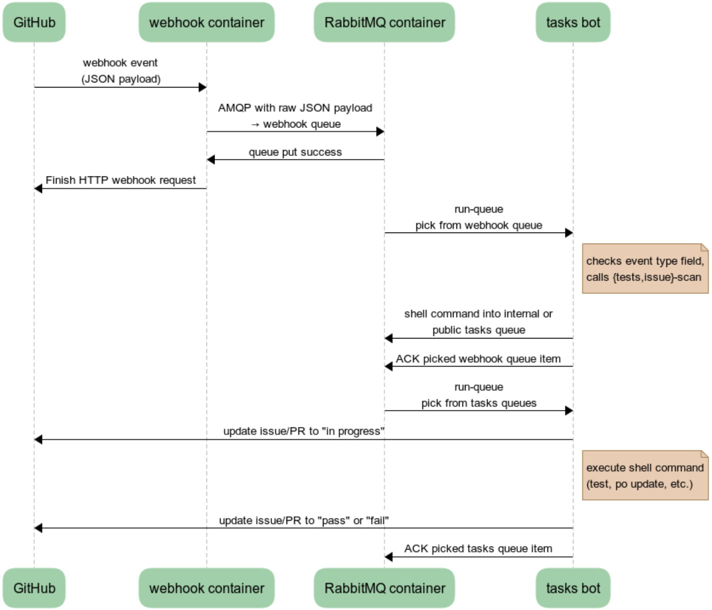
```
$ git tag -s -m '123

- cool new feature A
- fix heisenberg compensator on Fedora (rhbz#1234)
'
```

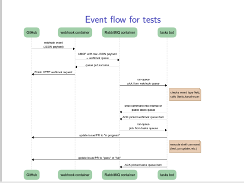
→ .github/workflows/release.yml runs release container

→ GitHub release, Fedora dist-git+koji+bodhi, COPR, DockerHub, docs on cockpit home page

- Explain a bit *what* we run on the infra; first example is releases
- minimized human work: summarize news, push signed tag, everything else is fully automated
- pushing tag triggers release workflow
- runs release container; looks at “cockpituous” script of the particular project, which controls what/where exactly to release

Event flow for tests


GitHub
webhook container
RabbitMQ container
tasks bot
webhook event (JSON payload)
AMQP with raw JSON payload → webhook queue
queue put success
Finish HTTP webhook request
run-queue pick from webhook queue
checks event type field, calls {tests,issue}-scan
shell command into internal or public tasks queue
ACK picked webhook queue item
run-queue pick from tasks queues
update issue/PR to "in progress"
execute shell command (test, po update, etc.)
update issue/PR to "pass" or "fail"
ACK picked tasks queue item
GitHub
webhook container
RabbitMQ container
tasks bot


Event flow for tests
GitHub
webhook container
RabbitMQ container
tasks bot
webhook event (JSON payload)
AMQP with raw JSON payload → webhook queue
queue put success
Finish HTTP webhook request
run-queue pick from webhook queue
checks event type field, calls (tests,issue)-scan
shell command into internal or public tasks queue
ACK picked webhook queue item
run-queue pick from tasks queues
update issue/PR to "in progress"
execute shell command (test, po update, etc.)
update issue/PR to "pass" or "fail"
ACK picked tasks queue item
GitHub
webhook container
RabbitMQ container
tasks bot

- tests infra is more complicated, no GH workflows yet
- starting point: GitHub event: something happens, like open PR; calls URL in your infra with JSON payload
- ephemeral → translate to distributed, transactional work queue: AMQP; very simple to use, robust, small; eats Jenkins for breakfast
- webhook pod is simple Python script + off-the-shelf rabbit container
- just single instance on CentOS CI; auto-recovers through PR/issue scanning (github is single source of truth)
- thus we can deal with few hours downtime, but not with days
- dozens of worker bots on various clouds connect to AMQP and run tasks

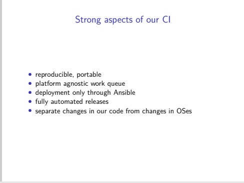
Strong aspects of our CI

- reproducible, portable
- platform agnostic work queue
- deployment only through Ansible
- fully automated releases
- separate changes in our code from changes in OSes

2021-01-13

Cockpit Infrastructure

└─Strong aspects of our CI


Strong aspects of our CI

- reproducible, portable
- platform agnostic work queue
- deployment only through Ansible
- fully automated releases
- separate changes in our code from changes in OSes

- make use of hybrid cloud; harness lots of powerful resources whereever we can get them
- for example, we can run our tests on Travis unmodified; just run it in our tasks container
- robust and simple work queue
- push-button releases
- Intro mentioned combinatorial explosion of OSes times APIs; we are everybody's OS regression test
- became good at isolating our changes from ever-changing/regressing OSes around us
- semi-auto-refreshed OS images, tests run completely offline
- find and investigate OS regressions at VM image update time, not in project PRs
- fully automatic tracking of OS regressions

Weak aspects/challenges of our CI

- arcane test logging and artifacts
- precarious e2e machines
- no monitoring/alerts
- hard to find public infra with /dev/kvm

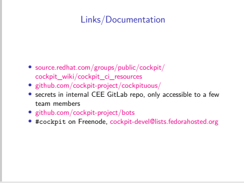
Weak aspects/challenges of our CI

- arcane test logging and artifacts
- precarious e2e machines
- no monitoring/alerts
- hard to find public infra with /dev/kvm

- our test logging/artifact infra is very arcane, too much custom logic; needs SSH, SPOF; want to move to standard infra (http post, s3, loki, etc.)
- e2e machines are ever more difficult to keep running; old RHEL 7 and no automation around Satellite; need well-maintained internal infra
- use host journal and k8s container logs for investigating failures; no automated monitoring (except for email on bot crash), notification, or prevention
- has not been a big enough pain point: no precious state in our CI, can re-scan github
- hard to find public infra with /dev/kvm: Travis for a while, but they stopped having free plans

Links/Documentation

- source.redhat.com/groups/public/cockpit/cockpit_wiki/cockpit_ci_resources
- github.com/cockpit-project/cockpituous/
- secrets in internal CEE GitLab repo, only accessible to a few team members
- github.com/cockpit-project/bots
- `#cockpit` on Freenode, cockpit-devel@lists.fedorahosted.org



Links/Documentation

- source.redhat.com/groups/public/cockpit/cockpit_wiki/cockpit_ci_resources
- github.com/cockpit-project/cockpituous/
- secrets in internal CEE GitLab repo, only accessible to a few team members
- github.com/cockpit-project/bots
- `#cockpit` on Freenode, cockpit-devel@lists.fedorahosted.org

- finally, where can you look at our stuff and steal or contribute
- top-level document on the source, describes available internal and external infra, lots of pointers
- public cockpituous repo has all our infra containers and Ansible scripts
- secrets like Fedora password, GitHub or COPR token are in a very restricted internal CEE GitLab repo
- bots is the code that runs inside containers; grab AMQP work queue item, invoke test, update translations, build VM image
- you can always say hi on IRC or ask on our mailing list