CI Testing with Beaker

Steven Pritchard — Sicura

How Beaker Works

Mental model

- Pick hosts (OS images + roles)
- Provision via a provider (Vagrant/Docker/cloud/…)
- Install an agent
- Apply code
- Test host status
- Tear down infrastructure

Why it's useful

- Tests real systems (not mocks)
- Catches packaging + service behavior
- Exercises modules the way users run them
- Same tests can run locally and in CI

Setting Up Beaker

- Add Beaker + a provider gem to your test bundle
- Define hosts + roles
- Run via `bundle exec rake beaker`

Gemfile

```
1 group :system_tests do
2   gem 'voxpupuli-acceptance', require: false
3 end
```

Rakefile

```
1 begin
2   require 'voxpupuli/acceptance/rake'
3 rescue LoadError
4   # only available if gem group system_tests is installed
5 end
```

spec/spec_helper_acceptance.rb

```
1 require 'voxpupuli/acceptance/spec_helper_acceptance'
2
3 configure_beaker(modules: :fixtures)
```

Writing Acceptance Tests

- RSpec + beaker-rspec gives you a DSL to run commands on hosts
- Test behavior: packages, services, files, idempotency, reboots, etc.
- Prefer small, focused tests that fail loudly

spec/acceptance/init_spec.rb

```
1  require 'spec_helper_acceptance'
2
3  describe 'my_class' do
4    it_behaves_like 'an idempotent resource' do
5      let(:manifest) { 'include myclass' }
6    end
7
8    it 'starts the service' do
9      on(host, 'systemctl -q is-active mysvc', acceptable_exit_codes: [0])
10   end
11 end
```

Beaker Providers (Hypervisors)

- beaker-aws
- beaker-docker
- beaker-gke
- beaker-google
- beaker-hcloud

- beaker-lima
- beaker-openstack
- beaker-vagrant
- beaker-vcloud
- beaker-vmware

beaker-vagrant

- Common choice for local testing
- Vagrant supports multiple providers (e.g., VirtualBox, libvirt)
- With vagrant-libvirt, you can run a small Beaker matrix in GitHub Actions

https://github.com/voxpupuli/gha-puppet/blob/bf8643f1399e85385d8ca66a0323afc5e5c94255/.github/workflows/beaker.yml#L181C1-L187C55

```
      - name: Setup libvirt for Vagrant
        if: ${{ inputs.beaker_hypervisor == 'vagrant_libvirt' && inputs.install_vagrant_dependencies
== true }}
        run: |
          sudo add-apt-repository ppa:evgeni/vagrant
          sudo apt-get update
          sudo apt-get install -y --no-install-recommends vagrant vagrant-libvirt
libvirt-daemon-system libvirt-daemon qemu-system-x86 qemu-utils dnsmasq
          sudo chmod 666 /var/run/libvirt/libvirt-sock
```

Use this when…

- You want local dev parity
- You need real VMs (kernel/systemd)

Watch out for…

- Runner resources (RAM/CPU)
- Provider setup time in CI

beaker-docker

- Runs “VM-like” containers as test nodes (sshd + provisioning)
- Works locally and in GitHub Actions; typically lower RAM than VMs
- Limitations: Linux-only; userland tests only (no kernel-level expectations)
- Not as instant as you’d expect: containers are treated like lightweight VMs

https://github.com/voxpupuli/gha-puppet/blob/bf8643f1399e85385d8ca66a0323afc5e5c94255/.github/workflows/beaker.yml#L176

```
      - name: Setup podman
        if: ${{ inputs.beaker_hypervisor == 'container_podman' }}
        run: |
          systemctl start --user podman.socket
          echo "DOCKER_HOST=unix:///run/user/$(id -u)/podman/podman.sock" >> "$GITHUB_ENV"
```

Use this when…

- You want lots of Linux nodes cheaply
- You’re testing userland behavior

Avoid this when…

- You need kernel/systemd/reboots
- You test firewall/sysctl modules

beaker-google

- What we use at Sicura
- Run acceptance tests on real cloud VMs
- Good fit for larger matrices and “realism” (kernel/systemd/reboots)

.gitlab-ci.yml

```
---
beaker:
  parallel:
    matrix:
      - BEAKER_PUPPET_COLLECTION: ['openvox8']
        NODESET: ['rhel8', 'rhel9']
  before_script:
    - mkdir -m 700 -p ~/.ssh
    - ssh-keygen -t ed25519 -f ~/.ssh/google_compute_engine < /dev/null
    - |
      cat >> Gemfile.local <<EOF
        gem "beaker-google", '~> 1.2', require: false
      EOF
    - bundle install
  script:
    - bundle exec rake spec_prep
    - bundle exec rake beaker:$NODESET
```

beaker-google

- What we use at Sicura
- Run acceptance tests on real cloud VMs
- Good fit for larger matrices and “realism” (kernel/systemd/reboots)

spec/acceptance/nodesets/rhel8.yml

```
---
HOSTS:
  rhel-8-x64:
    platform: el-8-x86_64
    roles:
      - agent
    hypervisor: <%= ENV.fetch('BEAKER_HYPERVISOR', 'vagrant') %>
    box: generic/rhel8
    family: rhel-cloud/rhel-8
    gce_machine_type: <%= ENV.fetch('BEAKER_MACHINE_TYPE', 'n2-standard-4') %>
CONFIG:
  type: aio
  puppet_collection: <%= ENV.fetch('BEAKER_PUPPET_COLLECTION', 'openvox8') %>
```

beaker-google

- What we use at Sicura
- Run acceptance tests on real cloud VMs
- Good fit for larger matrices and “realism” (kernel/systemd/reboots)

config.toml

```
[[runners]]
   environment = [
       "BEAKER_HYPERVISOR=google",
       "BEAKER_gce_project=my-beaker-project",
       "BEAKER_gce_zone=us-central1-a",
       "BEAKER_gce_network=default",
   ]
```

beaker-google

- What we use at Sicura
- Run acceptance tests on real cloud VMs
- Good fit for larger matrices and “realism” (kernel/systemd/reboots)
- **Caveat: metered resources — cleanup matters**

Use this when…

- You need “real VMs” in CI
- You want broad OS coverage

Watch out for…

- Cleanup failures ⇒ cost
- Quotas + credentials

New: beaker-kubevirt

- KubeVirt runs virtual machines on a Kubernetes cluster
- Scales from local KinD to on-prem clusters to cloud providers
- Workflow is similar to cloud providers (define VM spec, run tests, cleanup)
- Same cleanup caveat: orphaned VMs consume resources

Use this when…

- You already have Kubernetes
- You want VM realism + k8s scheduling

Watch out for…

- KubeVirt prerequisites
- Cleanup/quotas

How to Choose (Rules of Thumb)

Vagrant (VMs)

- Best local parity
- Kernel/systemd/reboot tests
- Heavier on CI resources

Docker (containers)

- Fast-ish + lightweight
- Many Linux nodes in CI
- Not for kernel-level tests

Google (cloud VMs)

- Broad OS coverage
- Scales well for matrices
- Cleanup matters (cost)

KubeVirt (k8s VMs)

- VM realism on Kubernetes
- Works from laptop → cloud
- Cleanup/quotas matter

Wrap-up

- Beaker gives you reproducible acceptance tests across real nodes
- Providers let you trade off speed, fidelity, and cost
- In CI: keep tests short, fail early, and automate cleanup
- If you remember one thing: pick the simplest provider that proves the behavior you care about

Questions?