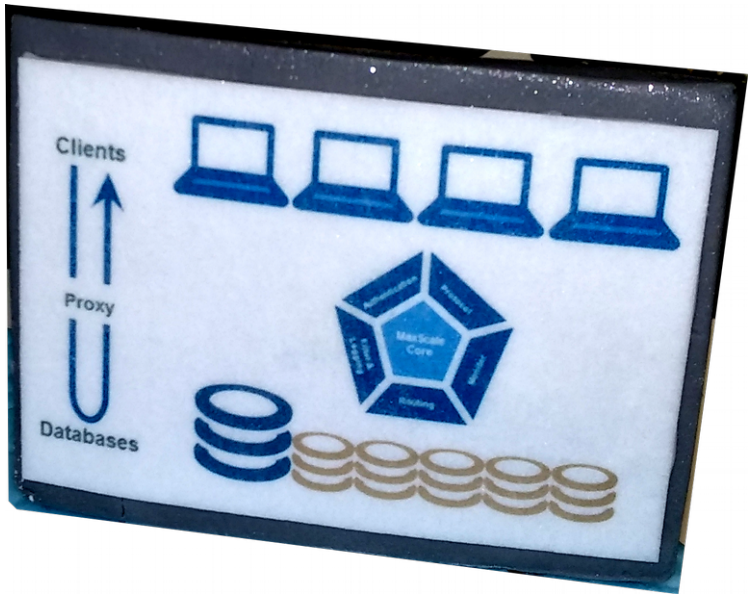
Build and test infrastructure as code

Open Source and Linux Lab

OSLL

The source of the problem - DB proxy server testing

- Product to build and test: Maxscale – database proxy
- Integration tests, performance tests, “scientific experiments”
- We need backend (N Virtual Machines), clients, network
  - lots of backend configurations
  - several virtual machines providers



MariaDB

VM management: e.g. Vagrant

- libvirt, AWS, Docker, LXC, DigitalOcean, OpenStack, …
- write Vagrantfile and 'vagrant up' …
- but
  - it can fail: restart logic is needed
  - Vagrantfile is not the simplest thing to write, “everything is in long long file”
- Provisioning:
  - lots of tools, let's just take simplest – Puppet
  - many version of MariaDB/MySQL – need a tool to configure paths to binary repos

MariaDB

Example of Vagrantfile

Amazon cloud config

One node descriptions (we need at least 9)

Amazon instance description

Specific User name (see ami-1c2e8b6b description...)

Instructions for Chef

Ruby loop

Magic ID

Hard coded server name

```
### Import AWS Provider access config ###
require 'yaml'
aws_config = YAML.load_file('/home/vagrant/mdbci/aws-config.yml')['aws']
## of import AWS Provider access config

### Vagrant configuration block  ###
#####################################
Vagrant.configure(2) do |config|

config.omnibus.chef_version = '12.9.38'

  ###          AWS Provider config block              ###
  ##################################################################
  config.vm.box = "dummy"

  config.vm.provider :aws do |aws, override|
    aws.keypair_name = "max-tst-02.mariadb.com_daily_test-382-2.1_1505122513"
    aws.region = aws_config["region"]
    aws.security_groups = aws_config["security_groups"]
    aws.user_data = aws_config["user_data"]
    override.ssh.private_key_path = "/home/vagrant/vms/daily_test-382-2.1/maxscale.pem"
    override.nfs.functional = false
    aws.aws_profile = "mdbci"
  end ## of AWS Provider config block

  ### Begin definition for machine: node_000
config.vm.define :node_000 do |node_000|

        node_000.vm.provider :aws do |aws,override|
                aws.ami = "ami-1c2e8b6b"
                aws.tags = { "hostname" => "max-tst-02.mariadb.com", "username" => "vagrant", "full_config_path" => "/home/vagra
                aws.instance_type = "t1.micro"
                override.ssh.username = "ec2-user"
        end
        node_000.vm.synced_folder "~/build-scripts/test-setup-scripts/cnf", "/home/vagrant/cnf_templates", type: "rsync"

        node_000.vm.provision 'shell', inline: 'curl -L https://omnitruck.chef.io/install.sh | sudo bash -s -- -v 12.9.38'        ### -- Chef
        node_000.vm.provision "chef_solo" do |chef|
                chef.cookbooks_path = "/home/vagrant/mdbci/recipes/cookbooks/"
                chef.roles_path = "."
                chef.add_role "node_000"
                chef.synced_folder_type = "rsync"
        end #<-- end of chef binding

end #  <-- End AWS definition for machine: node_000
```

MariaDB

Vagrant up …

- dependency on VM provider: it can easily **FAIL**
- 12 Linux distributions, 4 major MariaDB version, 3 major MySQL version, a lot of Maxscale plugins – **any combination should be testable** (see example of Vagrantfile)

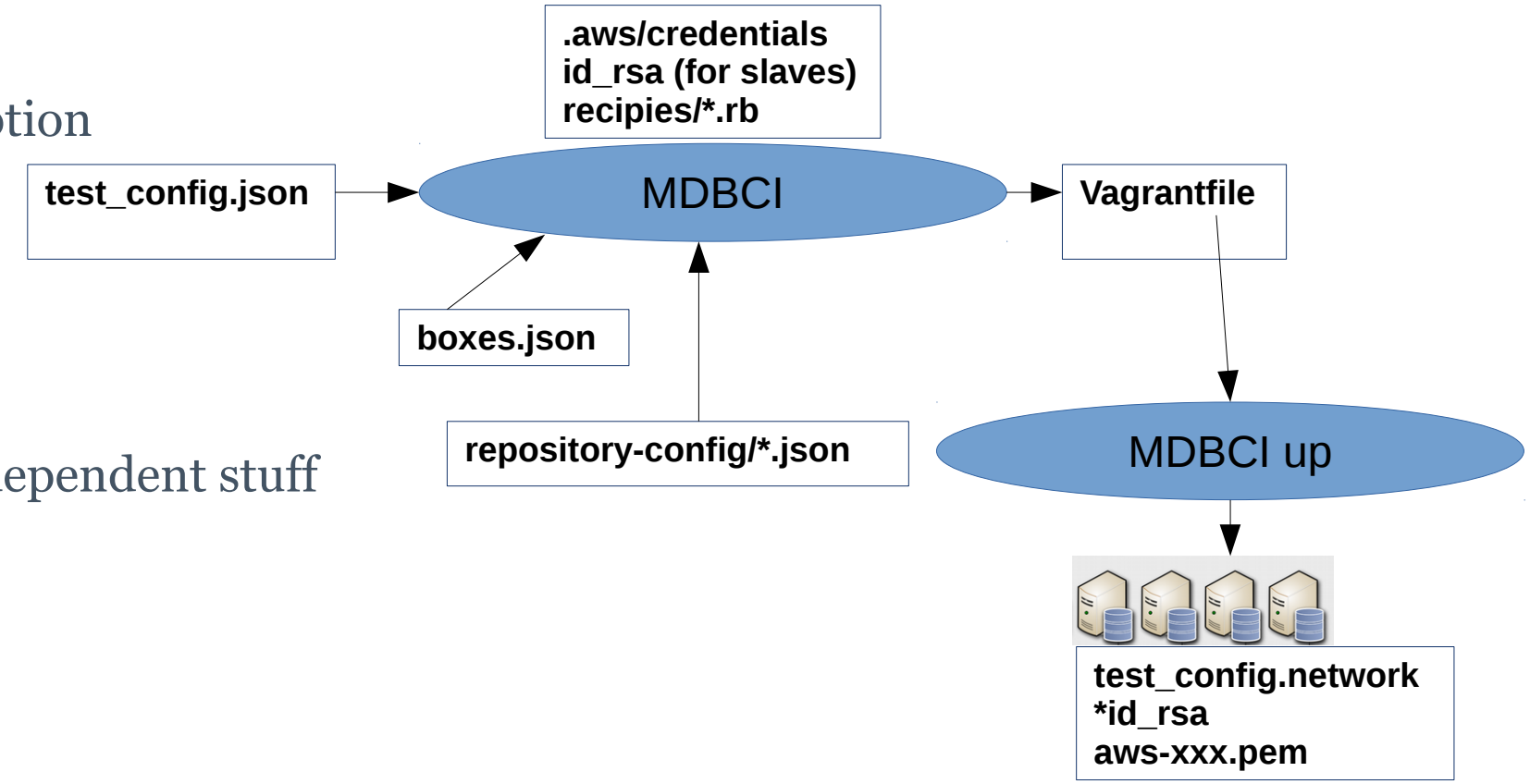
Example of random error:

```
INFO:  ==> galera_003: Starting domain.
ERROR:  There was an error talking to
Libvirt. The error message is shown
ERROR:  below:
ERROR:
ERROR:  Call to virDomainCreateWithFlags
failed: internal error: process exited while
connecting to monitor: warning: host
doesn't support requested feature:
CPUID.01H:EDX.ds [bit 21]
ERROR:  warning: host doesn't support
requested feature: CPUID.01H:EDX.acpi [bit
22]
ERROR:  warning: host doesn't support
requested feature: CPUID.01H:EDX.ht [bit
28]
...
```

MariaDB

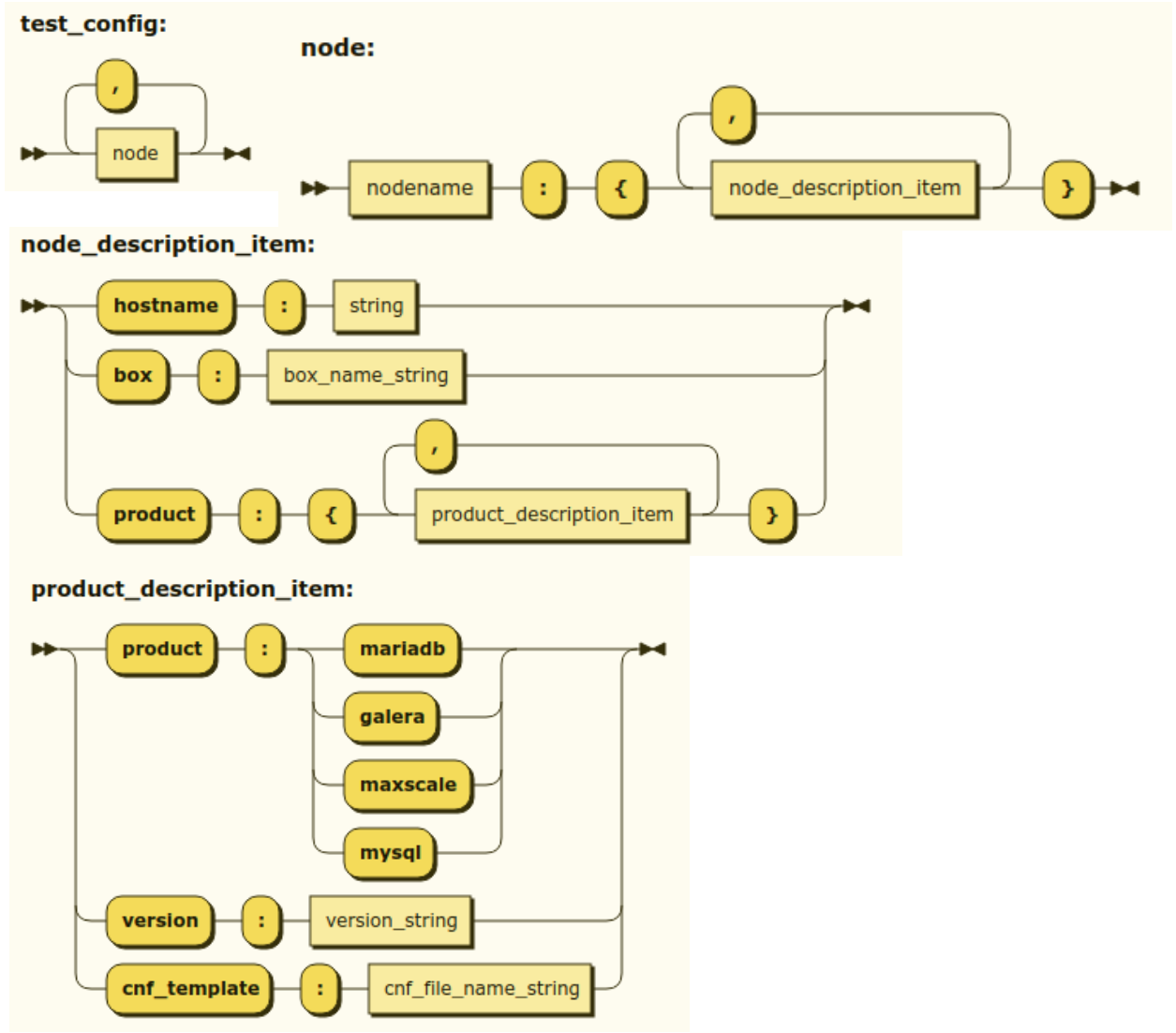
Our suggestion

- separate
  - high-level test setup description
  - VM box description
  - Repositories description
  - Chef recipes
  - Credentials
- hide
  - low-level provider-dependent stuff



MariaDB

Virtual Machiens description



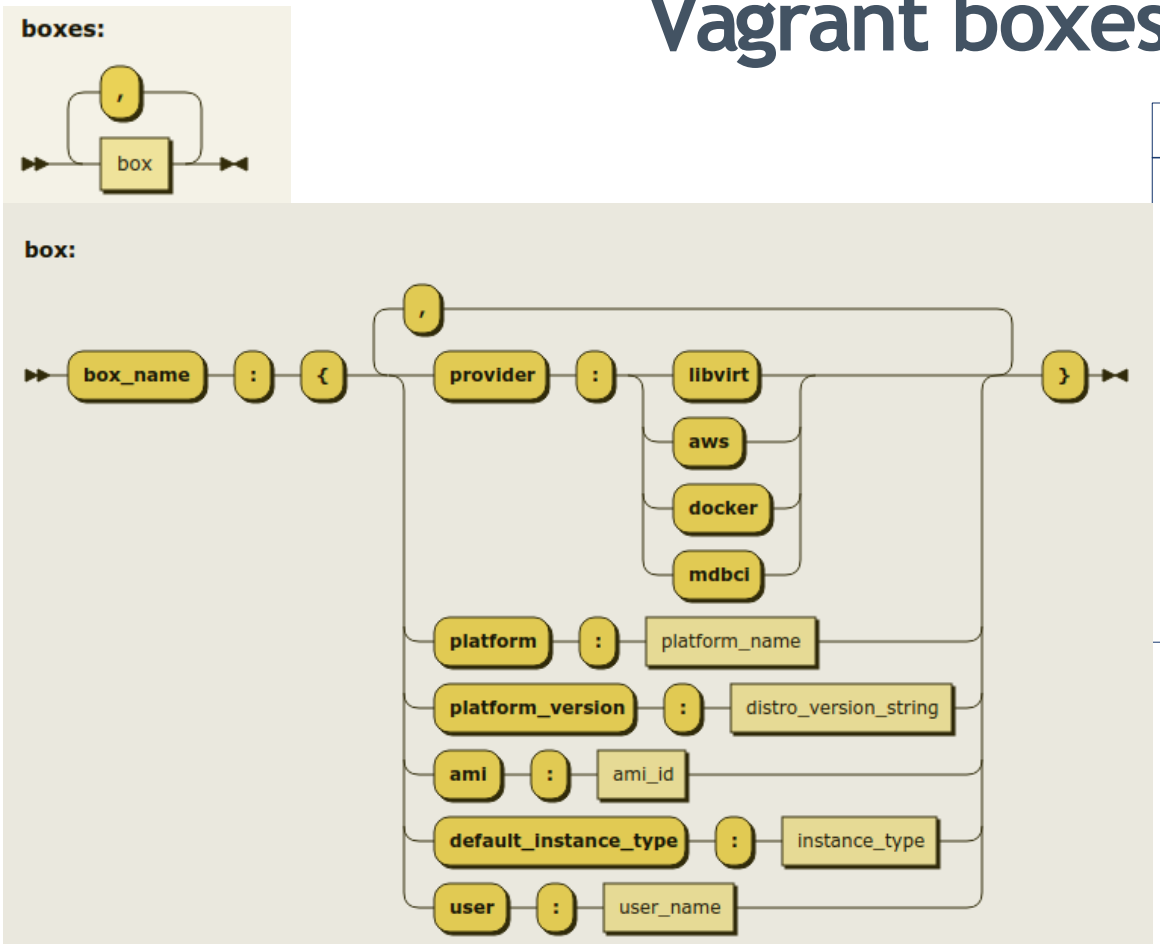
simple_test_config.json

```
"node_000" :
 {
      "hostname" : "node0",
      "box" : "centos_7_libvirt",
      "product" : {
            "name": "mariadb",
            "version": "10.0",
            "cnf_template" : "server1.cnf",
      }
 },
"node_001" :
 {
      "hostname" : "node1",
      "box" : "ubuntu_xenial_libvirt",
      "product" : {
            "name": "mariadb",
            "version": "10.2",
            "cnf_template" : "server2.cnf",
      }
 },
"maxscale" :
 {
      "hostname" : "maxscale",
      "box" : "ubuntu_xenial_libvirt",
      "product" : {
            "name": "maxscale",
            "version": "2.1.7"
      }
 }
```

Infrastructure as Code!

MariaDB

Vagrant boxes description



boxes_libvirt.json

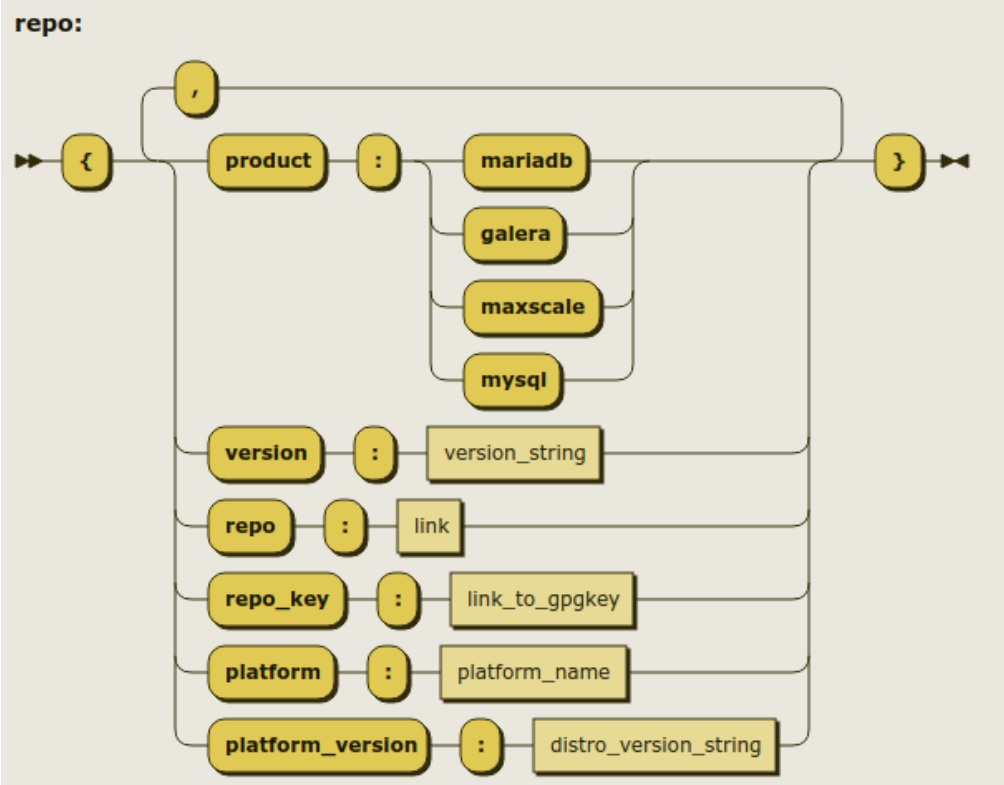
```
 "ubuntu_xenial_libvirt": {
    "provider": "libvirt",
    "box": "tnmt/xenial-alpha1",
    "platform": "ubuntu",
    "platform_version": "xenial"
  },
  "centos_7_libvirt": {
    "provider": "libvirt",
    "box": "centos/7",
    "platform": "centos",
    "platform_version": "7"
  },
....
```

boxes_aws.json

```
  "centos_7_aws_large" : {
    "provider": "aws",
    "ami": "ami-1c2e8b6b",
    "user": "ec2-user",
    "default_instance_type": "m1.large",
    "platform": "centos",
    "platform_version": "7"
  },
  "rhel_7_aws" : {
    "provider": "aws",
    "ami": "ami-25158352",
    "user": "ec2-user",
    "default_instance_type": "m3.medium",
    "platform": "rhel",
    "platform_version": "7"
  },
....
```

MariaDB

Repository description



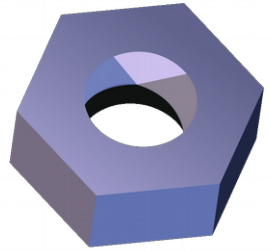
repository-config/maxscale-2.1.7-centos_7.json

```
{
  "product":          "maxscale",
  "version":          "default",
  "repo":             "https://downloads.mariadb.com/MaxScale//2.1.6/centos/7/$basearch",
  "repo_key":         "https://downloads.mariadb.com/MaxScale/MariaDB-MaxScale-GPG-KEY",
  "platform":         "centos",
  "platform_version":  "7"
}
```

MariaDB

Upper layer: BuildBot, Jenkins

- Buildbot
  - has been in use from very beginning (since middle of 90th)
  - do we want different tools for every our product?
  - there are many useful scripts for BuildBot
- Jenkins
  - popular tool
  - easy to start
  - let's try for Maxscale



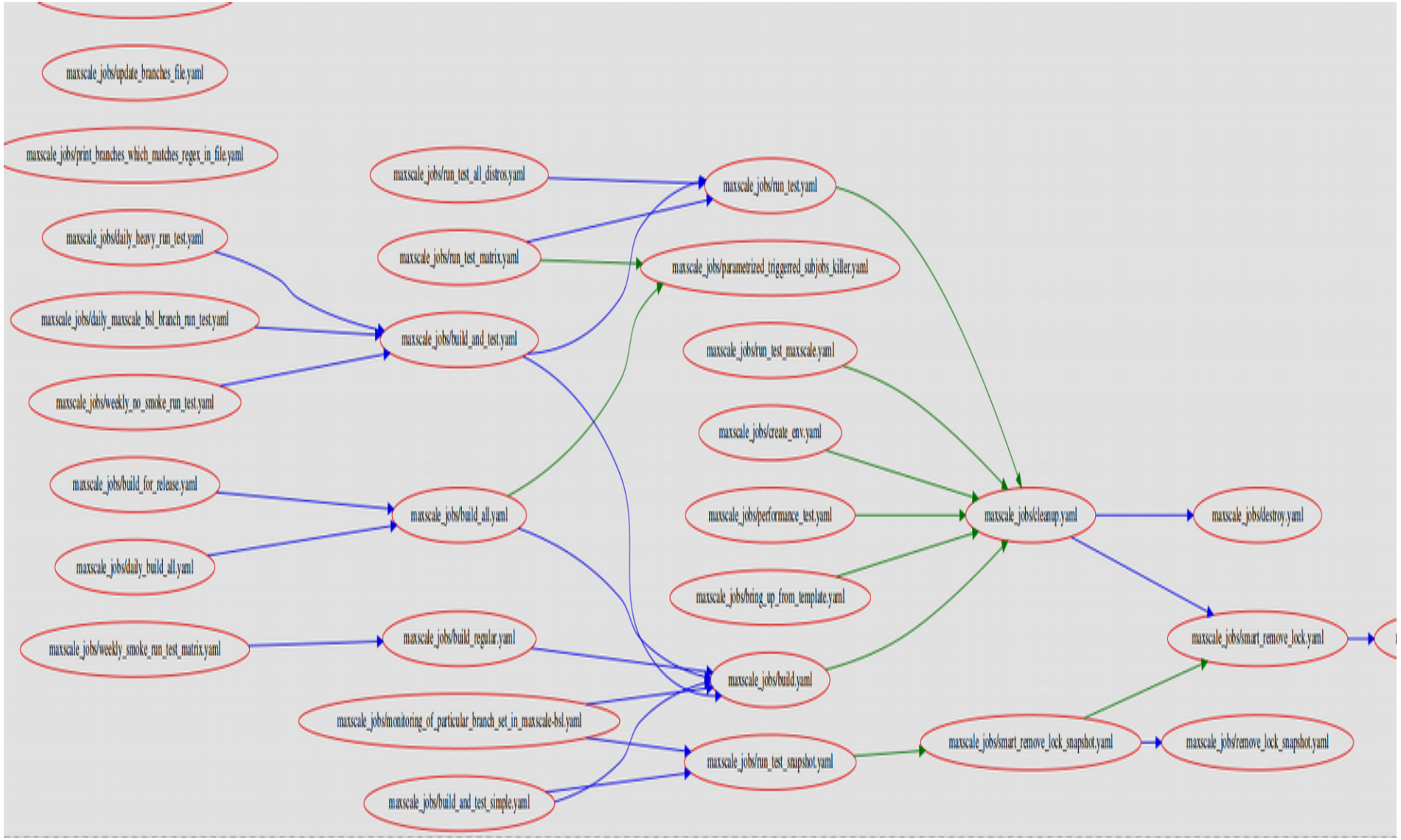
MariaDB

Jenkins: Job Builder

- A tool from OpenStack developers https://docs.openstack.org/infra/jenkins-job-builder/
- YAML
- Pipelines did not exist when we started
- our jobs: https://github.com/mariadb-corporation/maxscale-jenkins-jobs

MariaDB

Fail?



MariaDB

```
- job:
    name: run_test
    description: 'This job perform integration testing of maxscale'
    parameters:
        - !include: './maxscale_jobs/include/boxes_all_incl.yaml'
        - !include: './maxscale_jobs/include/products_incl.yaml'
        - !include: './maxscale_jobs/include/versions_incl.yaml'
        ....
        - !include: './maxscale_jobs/include/test_branch.yaml'
        - !include: './maxscale_jobs/include/slave.yaml'
    scm:
        - git:
            # TODO parametrize this url
            url: https://github.com/mariadb-corporation/maxscale-system-test.git
            branches:
                - $test_branch
    builders:
      - !include: './maxscale_jobs/include/build_parser/create_env_vars.yaml'
      - !include: './maxscale_jobs/include/build_parser/inject_initial_env.yaml'
      - !include: './maxscale_jobs/include/build_parser/run_test_and_collect.yaml'
      - !include: './maxscale_jobs/include/build_parser/parse_build_log.yaml'
      - !include: './maxscale_jobs/include/build_parser/inject_build_results.yaml'
      - !include: './maxscale_jobs/include/build_parser/create_env_coredumps.yaml'
      - !include: './maxscale_jobs/include/build_parser/inject_coredumps_var.yaml'
      - !include: './maxscale_jobs/include/build_parser/write_build_results.yaml'
    publishers:
      - !include: './maxscale_jobs/include/build_parser_mail_subject_with_name.yaml'
      - !include: './maxscale_jobs/include/call_cleanup.yaml'
    wrappers:
      - !include: './maxscale_jobs/include/workspace-cleanup.yaml'
      - !include: './maxscale_jobs/include/timeout.yaml'
    concurrent: true
```

boxes_all_incl.yaml

```
choice:
  name: box
  choices:
      !include: './maxscale_jobs/include/boxes_all.yaml'
  description: 'Virtual machine OS (name of Vagrant box)'
```

boxes_all_incl.yaml

```
- centos_7_libvirt
- centos_6_libvirt
- centos_5_libvirt
- ubuntu_wily_libvirt
- ubuntu_wily_aws
…
- suse_13_libvirt
```

boxes_all_incl.yaml

```
shell:
  '$HOME/build-
scripts/test/run_test.sh | tee
$WORKSPACE/build_log_$BUI
LD_ID; echo ${PIPESTATUS[0]}
> result_$BUILD_ID'
```

boxes_all_incl.yaml

```
email-ext:
  recipients: !include-raw: "./maxscale_jobs/include/mail_recipients.yaml"
  reply-to: $DEFAULT_REPLYTO
  content-type: default
  subject: $DEFAULT_SUBJECT ($name)
  body: !include-raw: "./maxscale_jobs/include/build_parser_email_body"
  attach-build-log: false
  ...
```

MariaDB

Results

- easy to create any number of VMs (described as easy JSON)
- MDBCI does all dirty job
- Jenkins is good for scheduling and logs storing
- **we got lost in Jenkins Jobs** (JJB YAMLs are not code, they are mess)
- **setup works!**
- results visualization is bad (Buildbot?)
- Pipelines or JobDSL – worth to try (if not BuildBot)

Build and test infrastructure as code

timofey.turenko@mariadb.com

https://mariadb.com/products/mariadb-maxscale

in cooperation with

Open Innovations Association FRUCT
http://fruct.org/

and

Open Source and Linux Lab
OSLL

https://osllblog.wordpress.com/