

11<sup>th</sup> Central and Eastern European Software  
Engineering Conference in Russia - CEE-SECR  
2015  
October 22 - 24, Moscow



Automatic tool for multi-configuration  
environment creation for database server  
and database proxy application testing



**Timofey Turenko**  
MariaDB Corporation AB

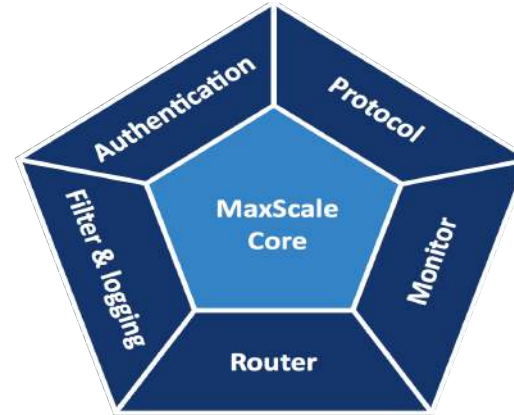
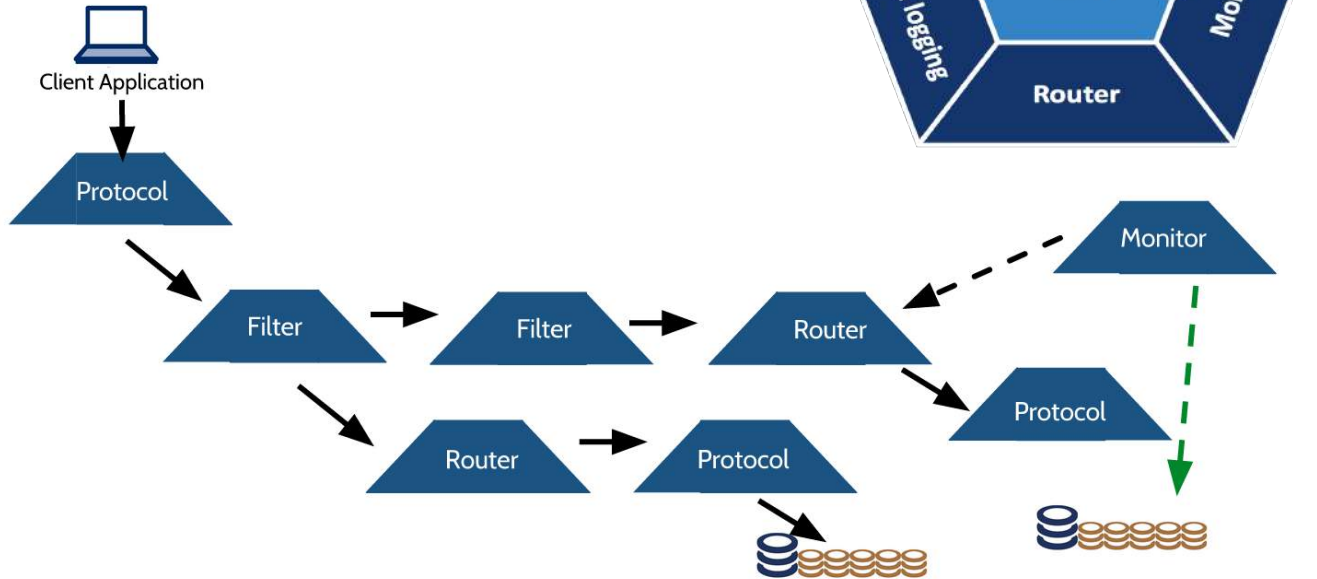
**Kirill Krinkin**

St-Petersburg Electrotechnical University



**MariaDB**

# MariaDB & Maxscale



# Test environment problem



- Server and proxy application testing:
  - backend is needed
  - lots of combination of machine configurations and version of backend servers

- Maxscale example:

- 12 major linux distributions



- MariaDB 5.5, 10.0, 10.1, 10.2



- MySQL 5.1, 5.5, 5.6, 5.7



- different topologies

- Maxscale itself from CI



**2015**  
**CEE-SEC(R)**

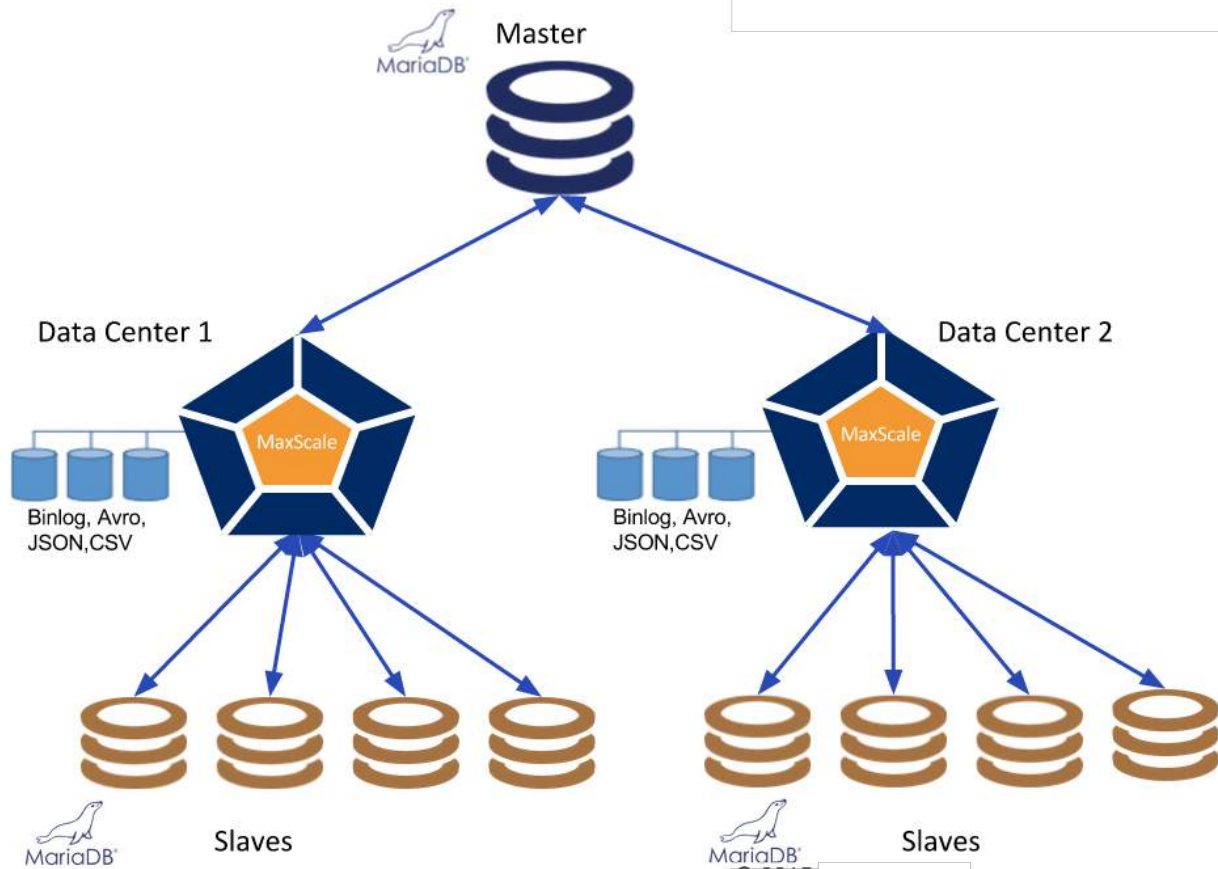
Software Engineering  
Conference in Russia

# Requirements



- arbitrary number of backend servers;
- any version of server software can be installed on every backend server;
- different virtual machine providers have to be supported (Virtual Box, Qemu, different cloud providers);
- deployment of test environment have to be automated;
- several test round can be executed in parallel;
- test environment description have to be clear and human readable.

# Configuration example



# VM managers, SW manager



- Vagrant, Terraform, Ansible
- Chef, Puppet

works? try to  
write/read it:

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

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

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

```
config.vm.provider :aws do |aws, override|  
  aws.access_key_id = aws_config["access_key_id"]  
  aws.secret_access_key = aws_config["secret_access_key"]  
  aws.keypair_name = aws_config["keypair_name"]  
  aws.region = aws_config["region"]  
  aws.security_groups = aws_config["security_groups"]  
  aws.user_data = aws_config["user_data"]  
  override.ssh.private_key_path = aws_config["pemfile"]  
  override.nfs.functional = false  
end ## of AWS Provider config block
```

```
config.vm.synced_folder "/home/vagrant/build-scripts/test-setup-...
```

```
config.vm.define :node0 do |node0|  
  node0.vm.provider :aws do |aws, override|  
    aws.ami = "ami-1c2e8b6b"  
    aws.instance_type = "t1.micro"  
    override.ssh.username = "ec2-user"  
  end  
  ##--- Chef binding ---  
  node0.vm.provision "chef_solo" do |chef|  
    chef.cookbooks_path = "../recipes/cookbooks/"  
    chef.roles_path = ""  
    chef.add_role "node0"  
    chef.synced_folder_type = "rsync"  
  end #<-- end of chef binding
```

```
end # -> End definition for machine: node0
```

```
config.vm.synced_folder "/home/vagrant/build-scripts/test-setup-...
```

```
# -> Begin definition for machine:  
config.vm.define :node1 do |node1|  
  node1.vm.provider :aws do |aws, override|  
    aws.ami = "ami-1c2e8b6b"  
    aws.instance_type = "t1"  
    override.ssh.username  
  end  
  ##--- Chef binding ---
```



2015

CEE-SEC(R)

Software Engineering  
Conference in Russia

# Make it readable



```
"node0" :
{
  "hostname" : "node0",
  "box" : "centos7",
  "product" : {
    "name": "mariadb",
    "version": "10.0",
    "cnf_template" : "server1.cnf",
    "cnf_template_path": "/home/vagrant/cnf"
  }
},
"node1" :
{
  "hostname" : "node1",
  "box" : "rhel7",
  "product" : {
    "name": "mysql",
    "version": "5.6",
    "cnf_template" : "server2.cnf",
    "cnf_template_path": "/home/vagrant/cnf"
  }
}
```

```
"node2" :
{
  "hostname" : "node2",
  "box" : "centos7",
  "product" : {
    "name": "galera",
    "version": "5.5",
    "cnf_template" : "server-galera3.cnf",
    "cnf_template_path": "/home/vagrant/cnf"
  }
}
```

# boxes.json

```
{
  "ubuntu_precise": {
    "provider": "virtualbox",
    "box": "chef/ubuntu-12.04",
    "platform": "ubuntu",
    "platform_version": "precise"
  },
  "centos7" : {
    "provider": "aws",
    "ami": "ami-1c2e8b6b",
    "user": "ec2-user",
    "default_instance_type": "t1.micro",
    "platform": "centos",
    "platform_version": "7"
  }
}
```





# Repository description



```
{ "product":      "mariadb",
  "version":      "10.0",
  "repo":         "http://yum.mariadb.org/10.0/centos5-amd64",
  "repo_key":     "https://yum.mariadb.org/RPM-GPG-KEY-MariaDB",
  "platform":     "centos",
  "platform_version": 5}

{ "product":      "mariadb",
  "version":      "10.0.17",
  "repo":         "http://yum.mariadb.org/10.0.17/centos5-amd64",
  "repo_key":     "https://yum.mariadb.org/RPM-GPG-KEY-MariaDB",
  "platform":     "centos",
  "platform_version": 5}

{ "product":      "mysql",
  "version":      "5.6",
  "repo":         "http://repo.mysql.com/yum/mysql-5.6-community/sles/12/x86_64",
  "repo_key":     "http://repo.mysql.com/RPM-GPG-KEY-mysql",
  "platform":     "sles",
  "platform_version": 12}
```

We have a generator for  
MariaDB, MariaDB enterprise, MariaDB Galera, Maxscale, MySQL



# Workflow

- write template
- './mdbc generate <name>'
- './mdbc up <name>'
- Get all needed info: './mdbc show ...'
- Access machines: './mdbc ssh' or directly via 'ssh'
- './mdbc destory <name>'



# Does it work?



- 9 and 25 machines configurations
- 40 test runs every day
- MariaDB, Maxscale, MariaDB enterprise, MariaDB-Galera
- VirtualBox, Amazon EC2, QEMU (plans: Docker, DigitalOcean)
- Developers like it: “use ./mdbci ssh and see all log, states, core dumps, use my env for debugging”

# Source

- <https://github.com/OSLL/mdbci>
- <https://github.com:mariadb-corporation/mdbci-repository-config>



## Maxscale

<https://github.com/mariadb-corporation/MaxScale>  
<https://mariadb.com/products/mariadb-maxscale>



MariaDB

