Archiving network element configuration with **Oxidized**

Trex Workshop 2014

Background

- Presenter
  - Anton Aksola (aakso@Twitter,IRCNet,Github)
  - Nebula Oy (AS29422 & AS13276)
  - Hosting and connectivity services
  - Contributing developer
- Founding developer
  - Saku Ytti (ytti@IRCNet,Github)
  - TDC Oy (AS3292)

What is Oxidized?

- Multi-vendor network element configuration backup and archiving software
- Fetches configuration, processes and saves it to a revision control system
- Attempt to remake Rancid and use more modern technologies and software
  - GIT vs CVS/SVN
  - Ruby vs Perl/TCL/Expect

Concurrent configuration polling

- Key features

- Currently oxidized is a single daemon process
- Uses thread per configuration poll worker
- Automatically adjusts the amount of threads
  - User can set maximum limit though
  - Example: 10000 nodes, 5 seconds on average per node, target time 1 hour -> 13 threads
- Only configuration poll and preprocessing are concurrent
  - Configuration is stored sequentially

Modular design

- Key features

- source
  - Loads the list of nodes and applies optional parameters
  - Supported: flat files, SQLite
- input
  - Interacts with network elements. Emulates CLI user if needed.
  - Supported: Telnet, SSH
- output
  - Stores configuration.
  - Supported: flat files, GIT
- model
  - Describes what to get from the node and optionally preprocesses the output
  - Support for devices from vendors such as: Cisco, Juniper, Arista, HP, Alcatel..
  - New models implemented quite often

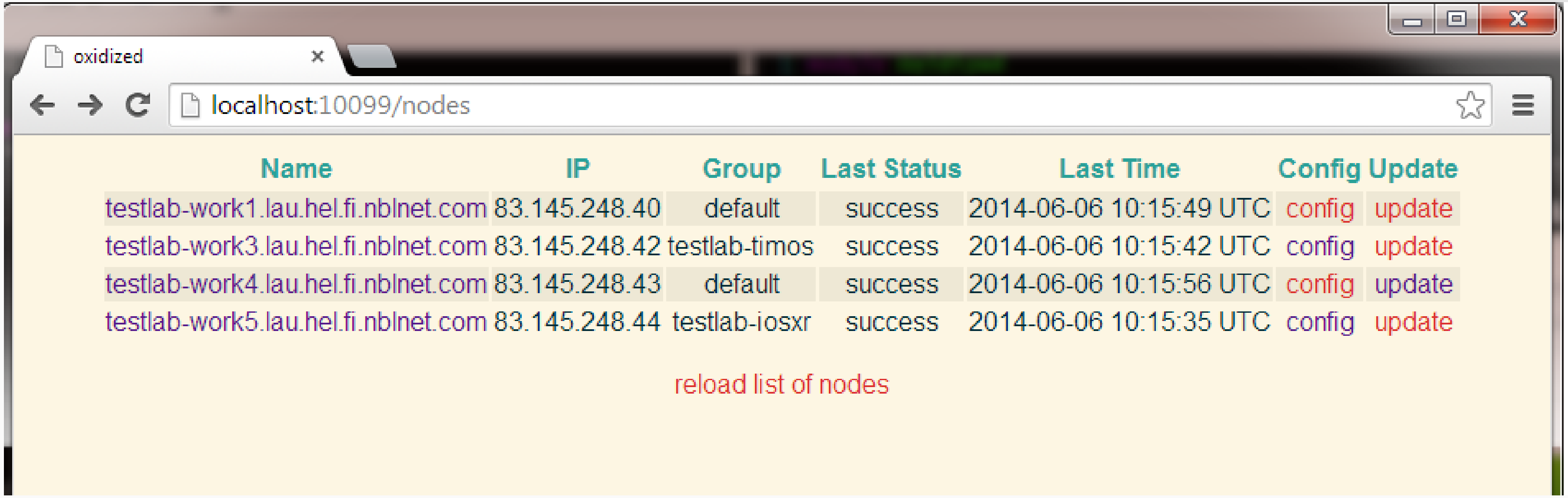
RESTful API

- Key features

- Optional feature. Separate gem ’oxidized-web’
- Simple web server, no authentication or anything fancy
- Provides a way to interact with the Oxidized daemon remotely
- Can be used to integrate Oxidized into other systems. For example:
    - Network monitoring (alerts)
    - Provisioning systems
- JSON input/output, HTML GUI for human interaction

RESTful API

- Node list



oxidized

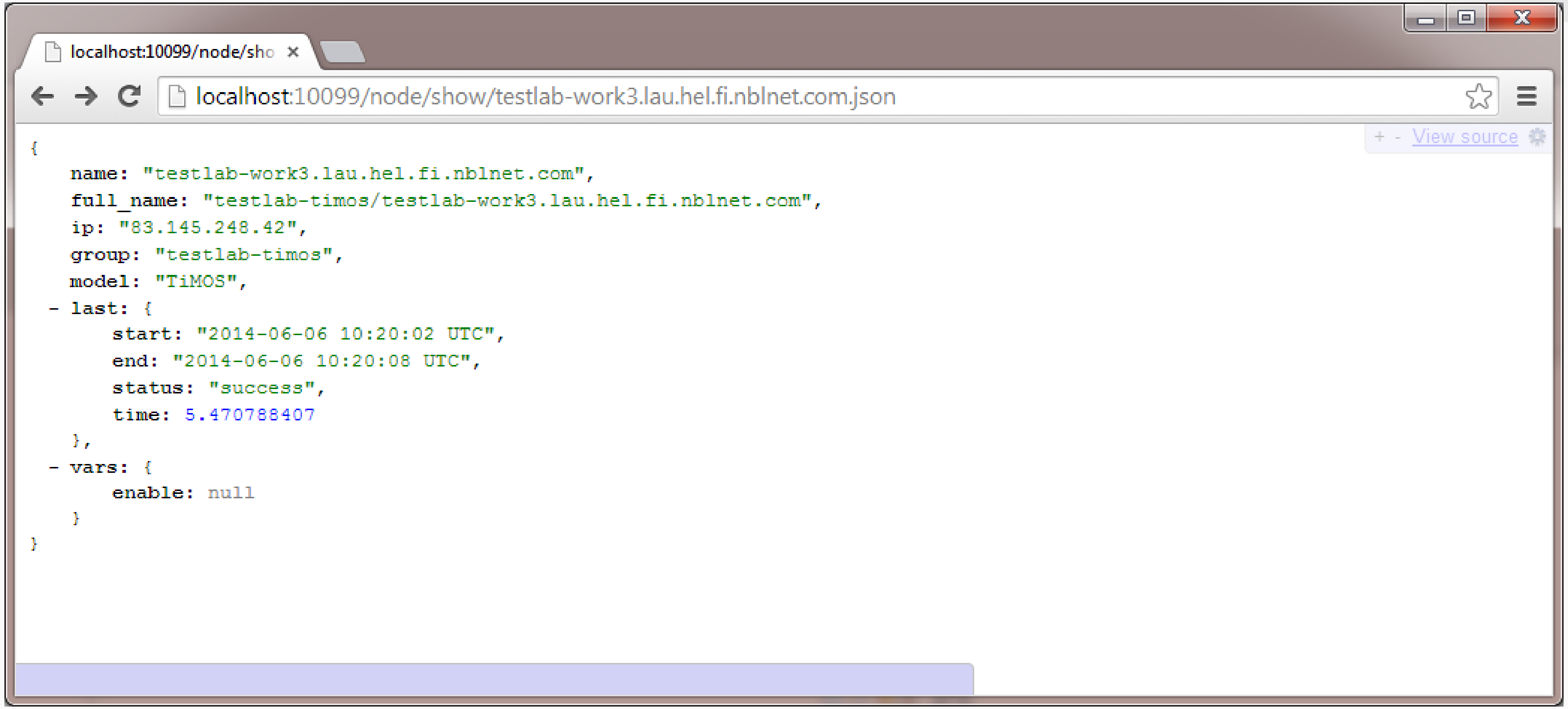
localhost:10099/nodes

Name	IP	Group	Last Status	Last Time	Config	Update
testlab-work1.lau.hel.fi.nblnet.com	83.145.248.40	default	success	2014-06-06 10:15:49 UTC	config	update
testlab-work3.lau.hel.fi.nblnet.com	83.145.248.42	testlab-timos	success	2014-06-06 10:15:42 UTC	config	update
testlab-work4.lau.hel.fi.nblnet.com	83.145.248.43	default	success	2014-06-06 10:15:56 UTC	config	update
testlab-work5.lau.hel.fi.nblnet.com	83.145.248.44	testlab-iosxr	success	2014-06-06 10:15:35 UTC	config	update

reload list of nodes

RESTful API

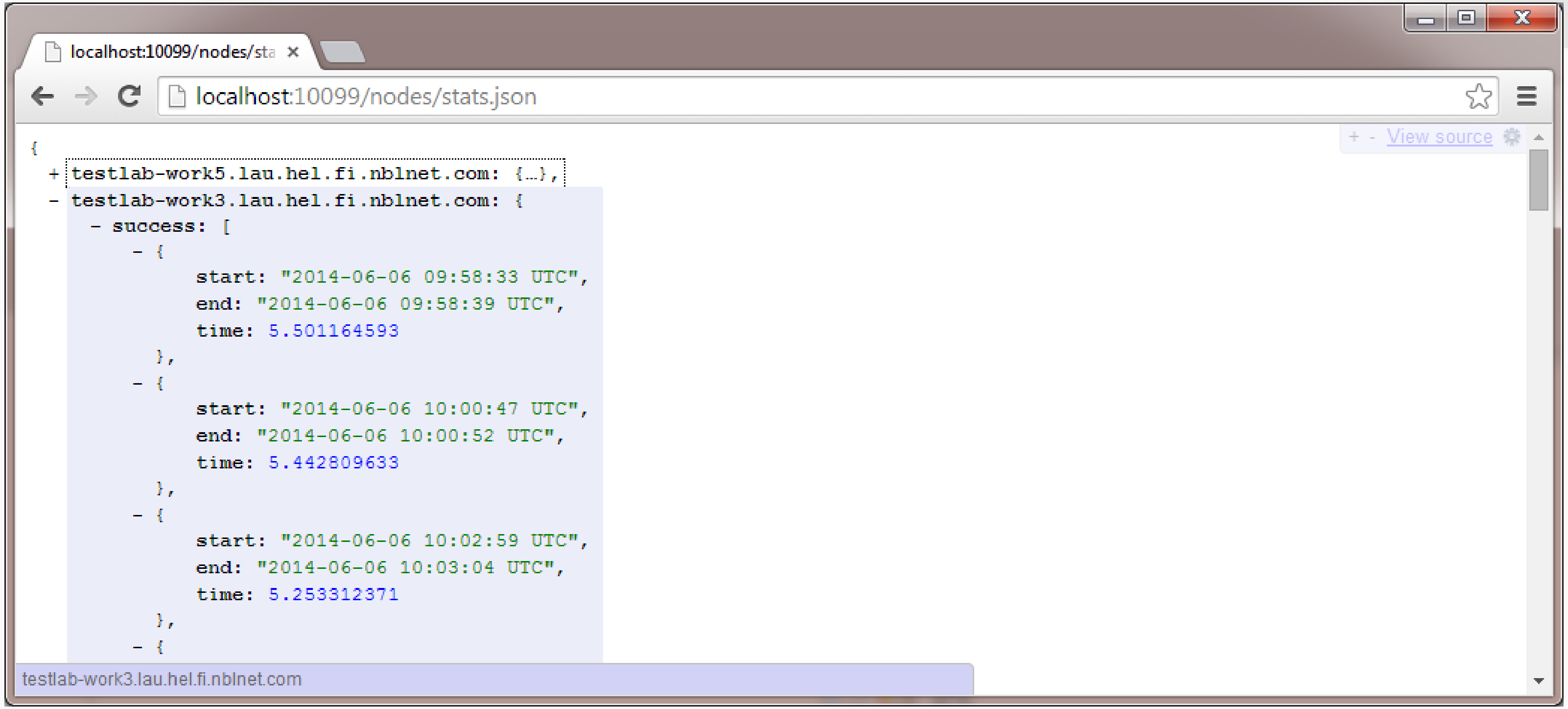
- Node details


localhost:10099/node/sho
localhost:10099/node/show/testlab-work3.lau.hel.fi.nblnet.com.json
+ - View source

```
{
    name: "testlab-work3.lau.hel.fi.nblnet.com",
    full_name: "testlab-timos/testlab-work3.lau.hel.fi.nblnet.com",
    ip: "83.145.248.42",
    group: "testlab-timos",
    model: "TiMOS",
  - last: {
        start: "2014-06-06 10:20:02 UTC",
        end: "2014-06-06 10:20:08 UTC",
        status: "success",
        time: 5.470788407
    },
  - vars: {
        enable: null
    }
}
```

RESTful API

- Node statistics



localhost:10099/nodes/sta ×

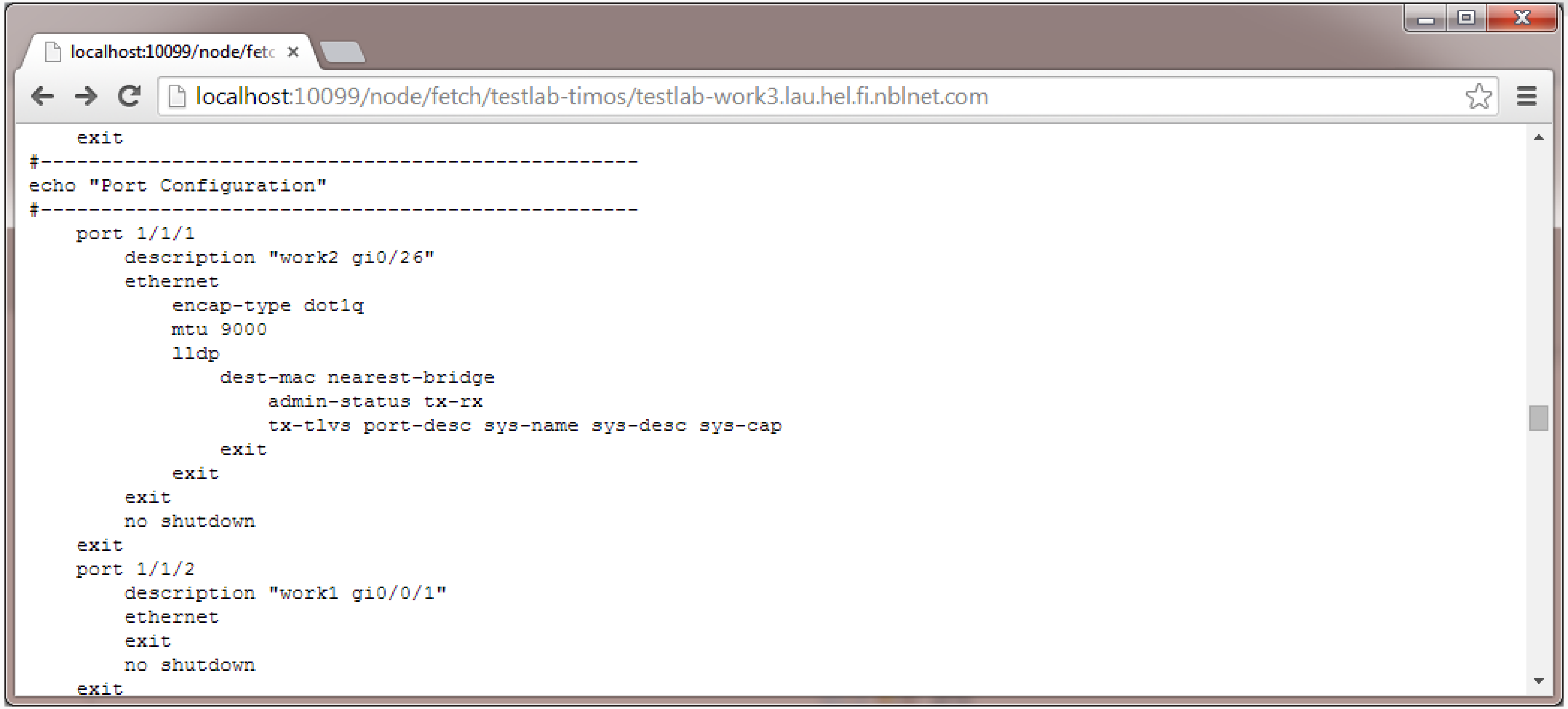
localhost:10099/nodes/stats.json

```
{
  + testlab-work5.lau.hel.fi.nblnet.com: {…},
  - testlab-work3.lau.hel.fi.nblnet.com: {
    - success: [
      - {
             start: "2014-06-06 09:58:33 UTC",
             end: "2014-06-06 09:58:39 UTC",
             time: 5.501164593
        },
      - {
             start: "2014-06-06 10:00:47 UTC",
             end: "2014-06-06 10:00:52 UTC",
             time: 5.442809633
        },
      - {
             start: "2014-06-06 10:02:59 UTC",
             end: "2014-06-06 10:03:04 UTC",
             time: 5.253312371
        },
      - {
```

testlab-work3.lau.hel.fi.nblnet.com

RESTful API

- Node configuration (latest)


localhost:10099/node/fetc ×

localhost:10099/node/fetch/testlab-timos/testlab-work3.lau.hel.fi.nblnet.com

```
    exit
#--------------------------------------------------
echo "Port Configuration"
#--------------------------------------------------
    port 1/1/1
        description "work2 gi0/26"
        ethernet
            encap-type dot1q
            mtu 9000
            lldp
                dest-mac nearest-bridge
                    admin-status tx-rx
                    tx-tlvs port-desc sys-name sys-desc sys-cap
                exit
            exit
        exit
        no shutdown
    exit
    port 1/1/2
        description "work1 gi0/0/1"
        ethernet
        exit
        no shutdown
    exit
```

Model API

- Rationale

- How to make node integration to Oxidized as easy and non-frustrating as possible?
- Should be fast, easy and DRY
    - We really don’t need hundreds of lines of duplicated code
- Should be achievable even for non-programmers

Model API

- Oxidized is written in Ruby
- Models are Ruby classes
- Oxidized implements Domain-Specific Language (DSL) for interaction and convenience
- Model is usually short and expressive
- More complex context-aware models are possible through helper methods and instance variables
    - Your imagination is the only limit although models should still be maintainable

Model API

- Example

```
class TiMOS < Oxidized::Model
  prompt /^([-\w\.:>\*]+\s?[#>]\s?)$/

  cmd 'admin display-config’

  cfg :ssh do
    post_login 'environment no more'
    pre_logout 'logout'
  end
end
```

Model API

- Example – Enhanced to include hardware information

```
class TiMOS < Oxidized::Model
  comment  '# '
  prompt /^([-\w\.:>\*]+\s?[#>]\s?)$/

  cmd 'show card state' do |cfg|
    comment cfg
  end

  cmd 'admin display-config’

  cfg :ssh do
    post_login 'environment no more'
    pre_logout 'logout'
  end
end
```

Model API

- Example – Context awareness

```
class JunOS < Oxidized::Model
  ...snip...

  cmd 'show version' do |cfg|
    @model = $1 if cfg.match /^Model: (\S+)/
    comment cfg
  end

  post do
    out = ''
    case @model
    when 'mx960'
      out << cmd('show chassis fabric reachability')  { |cfg| comment cfg }
    end
    out
  end

  ...snip...
end
```

TODO

- Oxidized is used in production though there is still a lot to do
- Things we need to improve:
    - code
    - testing
    - documentation
    - model support
- Please provide us feedback and tell if you are using Oxidized :)

Thank you

- `# gem install oxidized oxidized-web`
- https://github.com/ytti/oxidized

- Also try oxidized-script for all your shell scripting needs