A Puppet Approach To Application Deployment And Automation In Nokia

Oliver Hookins

Principal Engineer – Services & Developer Experience

NOKIA

Who am I?

- Rockstar SysAdmin

NOKIA

Who am I?

- Puppet User since 0.23
- Working in Nokia's Location-Based Services for 1.5 years
- Previously at Anchor Systems (manages GitHub infrastructure with Puppet)
- Love/Hate Relationship with Ruby

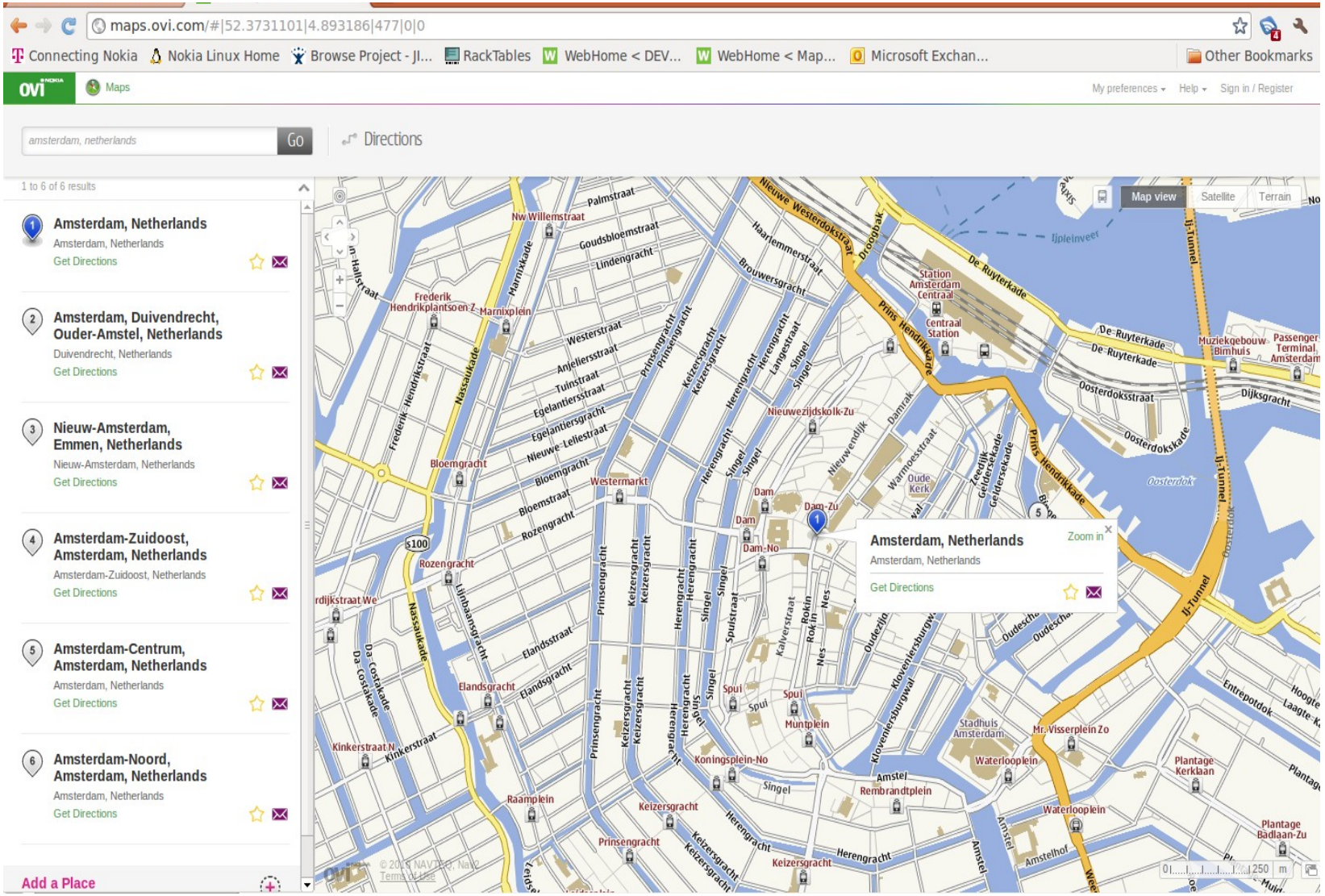
NOKIA

What do we do?

- “Location Based Services”
- Diverse collection of applications
- Rapidly moving environment, competing directly with well-known giants in the services arena
- Large, traditionally-organised company with heavy change control

NOKIA

What do we do?



NOKIA

What do we do?


Amsterdam
Netherlands
577km
Koog Aan De
Zaan
Ilpendam
Den Ilp
Oostzaan
Landsmeer
N516
A10
E22
E35
Amsterdam
Diemen
Duivendrecht
Schiphol
Noord
Schiphol
Oost
Amstelveen
Ouder-Amstel
Amsterdam-Zui
N231
2.5MB
Nes Aan De
Menu
List

The Problem Domain

- Deployment consistent across environments
- Reduce duplication of effort (and as a by-product, human errors)
- Increase testing, feedback of errors, reliability.

NOKIA

The Legacy System

NOKIA

The 1.0 Puppet System

NOKIA

New Problems

- Node definitions
- Lack of trust built-in
- Code constantly evolving
- Not enough testing to counter poor Puppet knowledge
- Victim of our own success

NOKIA

BDD

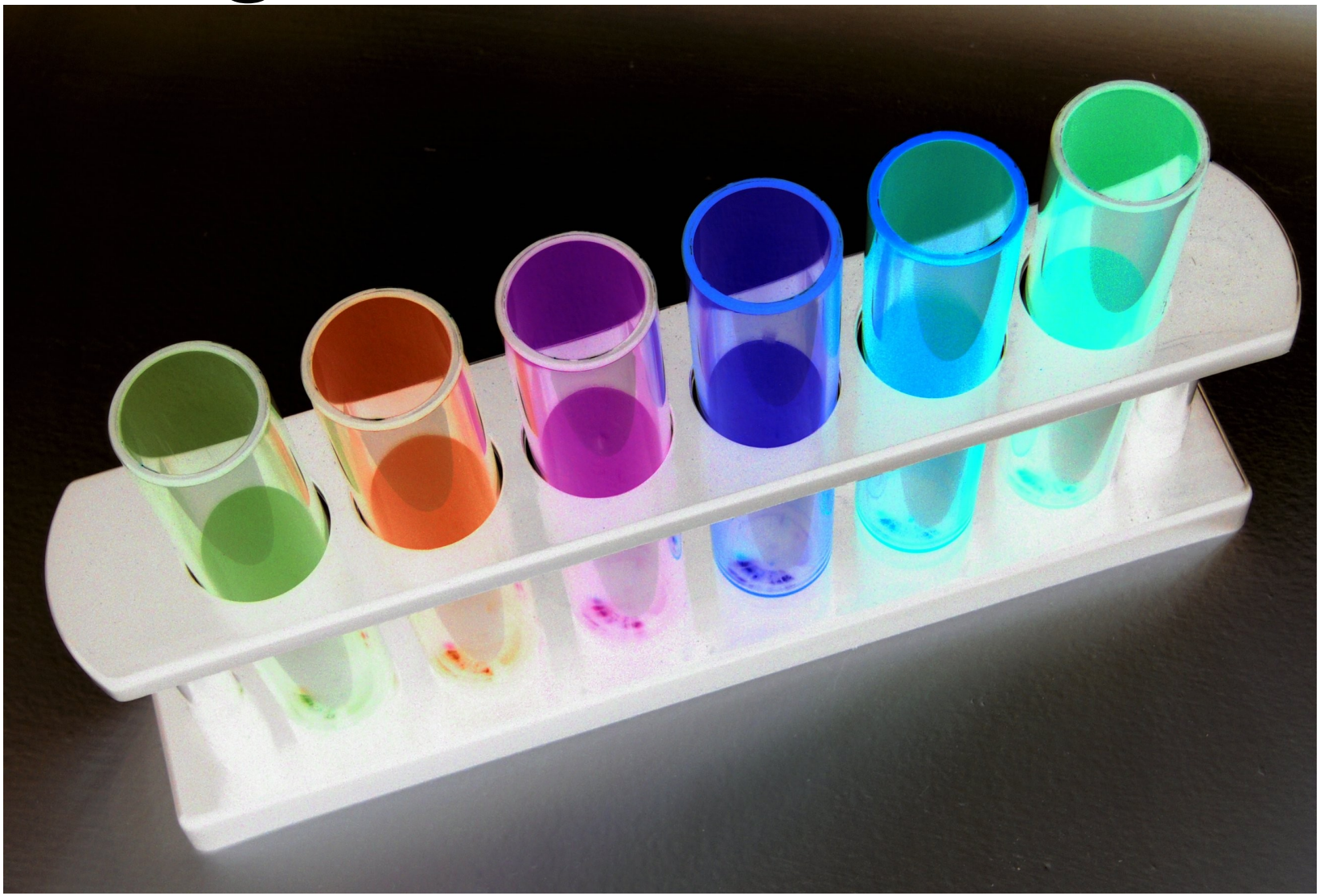
NOKIA

Goals

- Approach more like traditional software development
- Better versioning of components
- Testing, testing, testing!
- Automation using Jenkins
- Easier deployments
- Better developer tools

NOKIA

Testing



NOKIA

Jenkins

- CI/build tool
- Split testing into stages
- simple tests (lint, --parseonly, templates) – take < 2min to run
- compile tests (Puppet Faces) – should take < 10min
- integration (full-deploy) - ??
- Use in production! (canary)
- Eventually: continuous deployment

NOKIA

API Approach

- Expose a well-defined interface
- Remove host-, role-, DC- (etc) specifics from modules
- Drive configuration “through the front door”
- Remove need to look at the code (basically reverse engineering)

NOKIA

ENC

- Write one!
  … or try Dashboard
- Seriously, it unlocks Puppet's potential

NOKIA

Why not use extlookup?

- Equivalent of global variables in programming, ∴ bad!
- No well-defined interface to classes/defines
- Requires that you code-dive to find out how to use a class/define
- Well-defined API and general documentation should be sufficient
- Can't easily test code in isolation

NOKIA

Class Introspection

```
# Prepare a hash to dump out the API
h = {}
h['class'] = {}
h['define'] = {}

# Inspect the PP file
file = 'foo.pp'
Puppet[:manifest] = file
File.open(file, 'r').readlines().each do |l|
  if /^(class|define)\s+([^\({\s]+)/
    tipe = $~[1] # capture the type of object we found
    id = $~[2] # we want the class/define name

    # Grab the arguments and class/define name
    args = Puppet::Resource::Type.find(id).to_pson_data_hash['arguments']
    klass = Puppet::Resource::Type.find(id).to_pson_data_hash['name']
  end
end

h[tipe][klass] = args
```

NOKIA

Class Introspection

```
# Prepare a hash to dump out the API
h = {}
h['class'] = {}
h['define'] = {}

# Inspect the PP file
file = 'foo.pp'
Puppet[:manifest] = file
File.open(file, 'r').readlines().each do |l|
  if /^(class|define)\s+([^\({\s]+)/
    tipe = $~[1] # capture the type of object we found
    id = $~[2] # we want the class/define name

    # Grab the arguments and class/define name
    args = Puppet::Resource::Type.find(id).to_pson_data_hash['arguments']
    klass = Puppet::Resource::Type.find(id).to_pson_data_hash['name']
  end
end

h[tipe][klass] = args
```

NOKIA

Class Introspection

```
# Prepare a hash to dump out the API
h = {}
h['class'] = {}
h['define'] = {}

# Inspect the PP file
file = 'foo.pp'
Puppet[:manifest] = file
File.open(file, 'r').readlines().each do |l|
  if /^(class|define)\s+([^\({\s]+)/
    tipe = $~[1] # capture the type of object we found
    id = $~[2] # we want the class/define name

    # Grab the arguments and class/define name
    args = Puppet::Resource::Type.find(id).to_pson_data_hash['arguments']
    klass = Puppet::Resource::Type.find(id).to_pson_data_hash['name']
  end
end

h[tipe][klass] = args
```

NOKIA

Class Introspection

```
# Prepare a hash to dump out the API
h = {}
h['class'] = {}
h['define'] = {}

# Inspect the PP file
file = 'foo.pp'
Puppet[:manifest] = file
File.open(file, 'r').readlines().each do |l|
  if /^(class|define)\s+([^\({\s]+)/
    tipe = $~[1] # capture the type of object we found
    id = $~[2] # we want the class/define name

    # Grab the arguments and class/define name
    args = Puppet::Resource::Type.find(id).to_pson_data_hash['arguments']
    klass = Puppet::Resource::Type.find(id).to_pson_data_hash['name']
  end
end

h[tipe][klass] = args
```

NOKIA

Class Introspection

```
# Prepare a hash to dump out the API
h = {}
h['class'] = {}
h['define'] = {}

# Inspect the PP file
file = 'foo.pp'
Puppet[:manifest] = file
File.open(file, 'r').readlines().each do |l|
  if /^(class|define)\s+([^\({\s]+)/
    tipe = $~[1] # capture the type of object we found
    id = $~[2] # we want the class/define name

    # Grab the arguments and class/define name
    args = Puppet::Resource::Type.find(id).to_pson_data_hash['arguments']
    klass = Puppet::Resource::Type.find(id).to_pson_data_hash['name']
  end
end

h[tipe][klass] = args
```

NOKIA

Class Introspection

Now you have a YAML interface definition:

```
class:
  postfix:
    root_email: /dev/null
    relayhost:
    inet_interfaces: localhost
    alias_maps: hash:/etc/aliases
    mynetworks: ""
```

NOKIA

Class Introspection

- “Canary” testing
- Interface documentation (have a look at the :doc string as well)
- Generate web forms

NOKIA

Module Organisation

“Dist”

- **Provides “platform” services, as well as APIs for application modules to use**
- **Stable, defined release cycle**
- **Should be of sufficient quality that app teams cannot resist using it**
- **Eventually delegate maintenance to specialist teams**

NOKIA

Module Organisation

“Dist”

- **Pull in things from Moduleforge, or just use Moduleforge (in future)**

NOKIA

Module Organisation

“App”

- Blueprints for how to deploy applications
- Uses Dist APIs rather than reimplement functionality
- Sets up development environments for application – still unsure of correct approach for this

NOKIA

Packaging

NOKIA

Packaging

http://hunnur.com/blog/2010/10/dynamic-git-branch-puppet-environments/

- Modules under /etc/puppet/modules
- Each has metadata files (version of app, version of dist)
- Module path:

$confdir/environments/$environment/app:
$confdir/environments/$environment/dist

- Spec file creates symlinks from actual app and dist to above locations

Environment Change

```
def get_node_previous_environment(fqdn)
  factsdir = File.join(get_node_yamldir(), 'facts')
  File.directory?(factsdir) or raise NoFactsDirectoryError,
factsdir

  # Inspect the client machine's facts
  factsfile = File.join(factsdir, "#{fqdn}.yaml")
  if [nil, ''].include?fqdn or not File.exist?(factsfile)
    raise(NoClientFactsError,factsfile)
  end

  # Create a Puppet::Node object from the stored YAML object
  y = YAML.load_file(factsfile)
  p = Puppet::Node::Facts.new(y.name, y.values)
  p.values['environment']
end
```

NOKIA

Environment Change

```
def get_node_previous_environment(fqdn)
  factsdir = File.join(get_node_yamldir(), 'facts')
  File.directory?(factsdir) or raise NoFactsDirectoryError,
factsdir

  # Inspect the client machine's facts
  factsfile = File.join(factsdir, "#{fqdn}.yaml")
  if [nil, ''].include?fqdn or not File.exist?(factsfile)
    raise(NoClientFactsError,factsfile)
  end

  # Create a Puppet::Node object from the stored YAML object
  y = YAML.load_file(factsfile)
  p = Puppet::Node::Facts.new(y.name, y.values)
  p.values['environment']
end
```

NOKIA

Environment Change

```
def get_node_previous_environment(fqdn)
  factsdir = File.join(get_node_yamldir(), 'facts')
  File.directory?(factsdir) or raise NoFactsDirectoryError,
factsdir

  # Inspect the client machine's facts
  factsfile = File.join(factsdir, "#{fqdn}.yaml")
  if [nil, ''].include?fqdn or not File.exist?(factsfile)
    raise(NoClientFactsError,factsfile)
  end

  # Create a Puppet::Node object from the stored YAML object
  y = YAML.load_file(factsfile)
  p = Puppet::Node::Facts.new(y.name, y.values)
  p.values['environment']
end
```

NOKIA

Environment Change

```
def get_node_previous_environment(fqdn)
  factsdir = File.join(get_node_yamldir(), 'facts')
  File.directory?(factsdir) or raise NoFactsDirectoryError,
factsdir

  # Inspect the client machine's facts
  factsfile = File.join(factsdir, "#{fqdn}.yaml")
  if [nil, ''].include?fqdn or not File.exist?(factsfile)
    raise(NoClientFactsError,factsfile)
  end

  # Create a Puppet::Node object from the stored YAML object
  y = YAML.load_file(factsfile)
  p = Puppet::Node::Facts.new(y.name, y.values)
  p.values['environment']
end
```

NOKIA

Environment Change

```
if node_puppetenvironment != oldenv then
  Puppet.notice("Changing environment from #{oldenv} to
                 #{node_puppetenvironment} for node #{fqdn}")
  y = {}
  y['parameters'] = {}
  y['environment'] = node_puppetenvironment
  y['classes'] = {}
  y['classes']['puppet::client'] = {}
  y['classes']['puppet::client']['puppet_environment'] =
                                  node_puppetenvironment.clone
  # YAML catches duplicate references
  y['classes']['puppet::client']['puppetmaster'] =
                                              node_puppetmaster
end
```

NOKIA

Environment Change

```
if node_puppetenvironment != oldenv then
  Puppet.notice("Changing environment from #{oldenv} to
                #{node_puppetenvironment} for node #{fqdn}")
  y = {}
  y['parameters'] = {}
  y['environment'] = node_puppetenvironment
  y['classes'] = {}
  y['classes']['puppet::client'] = {}
  y['classes']['puppet::client']['puppet_environment'] =
                                    node_puppetenvironment.clone
  # YAML catches duplicate references
  y['classes']['puppet::client']['puppetmaster'] =
                                          node_puppetmaster
end
```

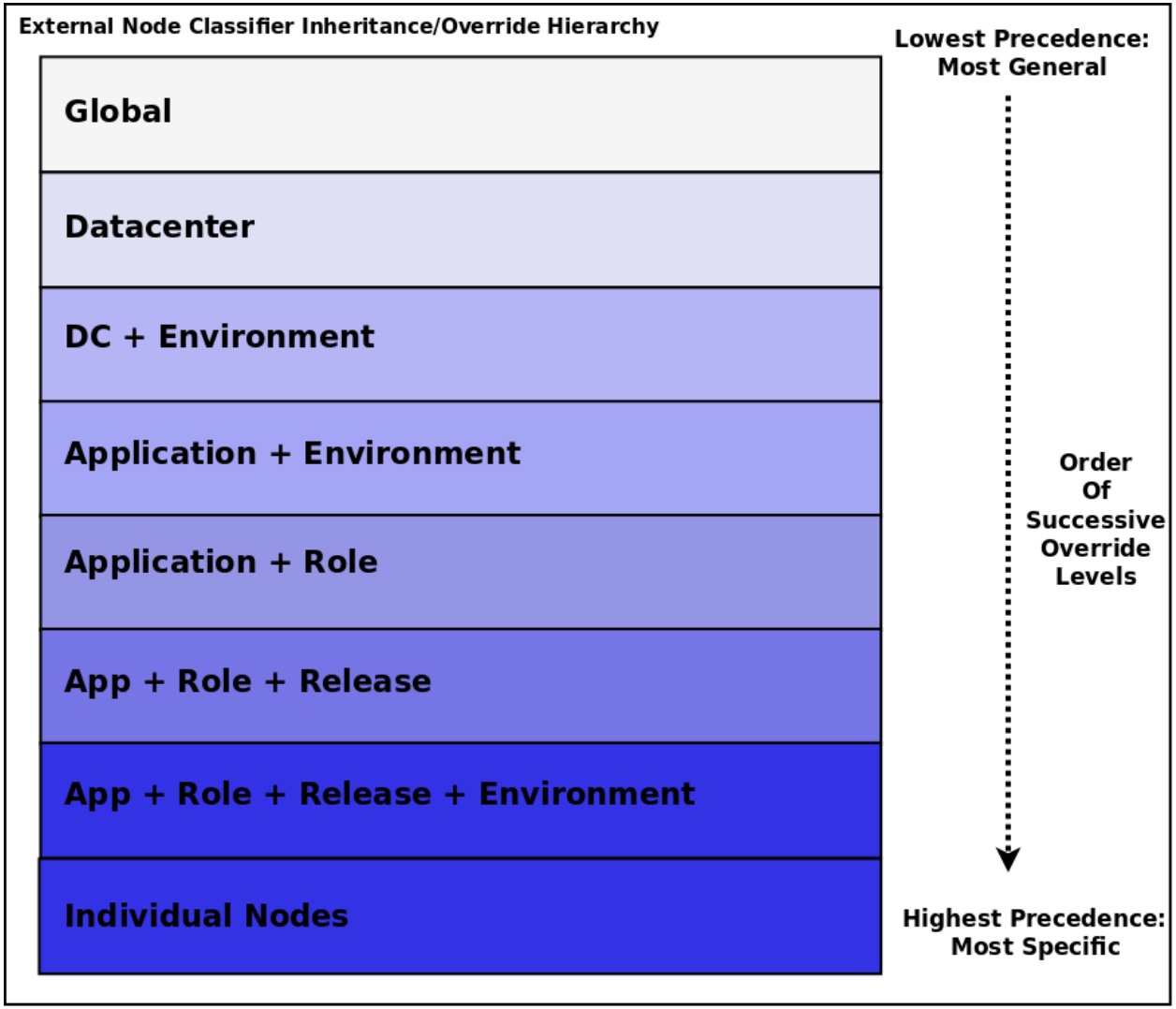
NOKIA

Environment Change

```
if node_puppetenvironment != oldenv then
  Puppet.notice("Changing environment from #{oldenv} to
                 #{node_puppetenvironment} for node #{fqdn}")
  y = {}
  y['parameters'] = {}
  y['environment'] = node_puppetenvironment
  y['classes'] = {}
  y['classes']['puppet::client'] = {}
  y['classes']['puppet::client']['puppet_environment'] =
                                  node_puppetenvironment.clone
  # YAML catches duplicate references
  y['classes']['puppet::client']['puppetmaster'] =
                                          node_puppetmaster
end
```

NOKIA

Our ENC



NOKIA

Our ENC

```
vars:
  puppet_version: 2.6.7
  ntp_servers:
    - 192.0.2.1
    - 192.0.2.2
    - 192.0.2.3
  repo_server: repo.ny.example.com
  accounts:
    - bofh
    - peon
  myapp_version: 3.0.0

classes:
  platform:
    puppet_version: ${puppet_version}
    ntp_servers: ${ntp_servers}
    repo_server: ${repo_server}
    accounts: ${accounts}
  myapp:
    version: ${myapp_version}
```

NOKIA

Future

NOKIA

Future

- Deployment strategy not entirely worked out
- Developer workflows still could be improved (some promising work in this area, e.g. cloudsmith/geppetto (IDE integration))

NOKIA

Thank You!

oliver.hookins@nokia.com
ohookins@gmail.com
http://paperairoplane.net/

P.S. We're Hiring!

NOKIA

Attributions

- Photo on slide 8 by dandeluca, available under a Creative Commons Attribution licence
- Photo on slide 9 by graemenewcomb, available under a Creative Commons Attribution licence
- Photo on slide 11 by seenful, available under a Creative Commons Attribution licence
- Photo on slide 13 by f-oxymoron, available under a Creative Commons Attribution licence
- Photo on slide 28 by oskay, available under a Creative Commons Attribution licence
- Photo on slide 39 by kirrilyrobert, available under a Creative Commons Attribution licence

NOKIA