Nordea

PUPPET AT SCALE AT NORDEA?

Bryan Østergaard, Senior IT Infrasucture Specialist

1.11.2017

Who am I?

- I have a varied background in
  - IT Security
  - Software Development
  - IT Operations

- Worked at very small companies as well as companies up to a few thousand employees

- Nordea is by far the most complex organisation I've been at
  - With unique challenges and opportunities

- Also have a background in Open Source Software
  - Was a very active Gentoo Linux developer
  - Started my own Linux distribution later
  - Part of the freenode IRC network staff for many years and in charge of their infrastructure team

What's Nordea like in numbers?

- About 32000 employees
- Lots of consultants as well
- Lots of older banks merged to form Nordea
- Several datacentre sites
  - In different countries

- Hundreds if not thousands of applications
- Ever changing regulations for financial institutions
- Audits
- Huge server growth
- Multiple OS platforms

So how do we do it?

- Reasonably strict processes and workflows
- Flexibility where it makes sense
- Automation where possible
- A big focus on Separation of Concerns
  - Puppet code implementing functionality to setup and configure things are (mostly) separated from data describing how it should be configured
  - Code and configuration are mostly handled by different teams
- Automated testing
  - Static analysis like syntax checking, linting and unit testing checking expected resources are present
  - Dynamic testing
    - Spinning up a new server and checking that Puppet works
    - Further checks to see services are running, listening on correct ports, ..
    - Cross platform support complicates this greatly

A simple example

```
class timesync (
  String $conf_file,
  String $tmp,
  String $service_name,
  Array[String] $servers,
  Boolean $control,
  $restart = undef,
) {
  if ($control) {
    file { $conf_file:
      ensure  => file,
      owner   => root,
      group   => root,
      mode    => '0644',
      content => epp("timesync/${tmp}"),
    }
```

```
    service { $service_name:
          ensure  => running,
          restart => $restart,
        }

        File[$conf_file]
        ~> Service[$service_name]
  }
}
```

Taking advantage of Puppet APIs

- Not everything is supported by Puppet out of the box
  - So we extend Puppet in various ways
- Lots of custom facts
  - Describing installed versions
  - Bits of configuration like which job server is used, where logs are sent, ..
- Custom resource types
  - Executing external commands directly from Puppet code is error-prone and hard to control
  - Sometimes you need to execute lots of external commands..
  - Resuming after errors can get particularly gnarly
  - Implementing a custom resource type solves this in a very nice way
- “You can’t automate this”
  - Sounds like a challenge :)

Nordea

Thank you!

Bryan Østergaard, Senior IT Infrasucture Specialist
bryan.ostergaard@nordea.com