A Reference Architecture for Distributed Software Deployment

Sander van der Burg

Delft University of Technology, EEMCS, Department of Software Technology

August 2, 2013
A Reference Architecture for Distributed Software Deployment

Sander van der Burg
Software deployment

Sander van der Burg

A Reference Architecture for Distributed Software Deployment
Software deployment

All of the activities that make a software system available for use.
Challenges

A Reference Architecture for Distributed Software Deployment
Challenges

Software deployment

- Time consuming
- Error prone
- Destructive upgrades
- Downtimes
Some history: Early history

Appendix 4, Standard program for class 6 and class 7

<table>
<thead>
<tr>
<th>DA</th>
<th>DO</th>
<th>DI</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>A</td>
<td></td>
</tr>
</tbody>
</table>

Sander van der Burg

A Reference Architecture for Distributed Software Deployment
Some history: High-level languages and operating systems
Some history: High-level languages and operating systems

Software components

Requires compiler or interpreter and a compatible operating system
Some history: Component-based software engineering

The Nix project

Welcome to the homepage of the Nix project, which consists of a number of related open source subprojects:

- **Nix** is a *purely functional* package manager. This means that it can ensure that an upgrade to one package cannot break others, that you can always roll back to previous version, that multiple versions of a package can coexist on the same system, and much more.
- **Nixpkgs** is a large collection of packages that can be installed with the Nix package manager.
- **NixOS** is a Nix-based Linux distribution. Thanks to Nix, it supports atomic upgrades, rollbacks and multi-user package management, and it has a declarative approach to system configuration management that makes it easy to reproduce a configuration on another machine.
- **Hydra** is a Nix-based continuous build system.
- **Disnix** is a Nix-based distributed service deployment system.

**Latest News**

PatchELF 0.6 released **2011/11/7**

PatchELF 0.6 has been released. Apart from some bug fixes, it adds support for executables produced by the C++ Linker and for PE/COFF files.
Some history: Component-based software engineering

Software components

- Components increase programmer productivity
- Components increase quality of software
Some history: Component-based software engineering

Disadvantages:

A Reference Architecture for Distributed Software Deployment
Nowadays: Services on the internet

Challenges:
Challenges:

Software components

Software deployment has become increasingly more complicated
Earlier research: Nix and NixOS

A GNU/Linux distribution using the Nix package manager

NixOS

Sander van der Burg

A Reference Architecture for Distributed Software Deployment
Main idea: store all packages in isolation from each other:

/\texttt{nix/store/rpdqxnib0cg... -firefox-3.5.4}

Paths contain a 160-bit **cryptographic hash** of all inputs used to build the package:

- Sources
- Libraries
- Compilers
- Build scripts
- ...
{ stdenv, fetchurl, openssl, zlib }:

stdenv.mkDerivation {
    name = "openssh-4.6p1";
    src = fetchurl {
        url = http://.../openssh-4.6p1.tar.gz;
        sha256 = "0fpjlr3bfind0y94bk442x2p...";
    };
    buildCommand = ":
        tar xjf $src
        ./configure --prefix=$out --with-openssl=${openssl}
        make; make install
    ";
}
all-packages.nix

```nix
openssl = import ../development/libraries/openssl {
    inherit fetchurl stdenv perl;
};

stdenv = ...;
openssl = ...;
zlib = ...;
perl = ...;
```

- `nix-env -f all-packages.nix -iA openssh`
- Produces a `/nix/store/l9w6773m1msy...-openssl-4.6p1` package in the Nix store.
Users can have different sets of installed applications.
User environments

- Users can have different sets of installed applications.
- `nix-env` operations create new **user environments** in the store.

```bash
(nix-env -u openssh)
```
User environments

- Users can have different sets of installed applications.
- `nix-env` operations create new **user environments** in the store.

```bash
(nix-env -u opensssh)
```
Users can have different sets of installed applications.

- nix-env operations create new user environments in the store.

```bash
(nix-env -u openssh)
```
User environments

- Users can have different sets of installed applications.
- `nix-env` operations create new user environments in the store.
- We can atomically switch between them.

(nix-env -u openssh)
User environments

- Users can have different sets of installed applications.
- `nix-env` operations create new user environments in the store.
- We can atomically switch between them.
- These are roots of the garbage collector.

```bash
(nix-env --remove-generations old)
```
Users can have different sets of installed applications.

- `nix-env` operations create new **user environments** in the store.
- We can atomically switch between them.
- These are roots of the **garbage collector**.
In NixOS, all packages including the Linux kernel and configuration files are managed by Nix.

NixOS does not have directories such as: `/lib` and `/usr`

NixOS has a minimal `/bin` and `/etc`

But NixOS is more than just a distribution managed by Nix.
NixOS configuration

/etc/nixos/configuration.nix

{pkgs, ...}:
{
  boot.loader.grub.device = "/dev/sda";

  fileSystems = [ { mountPoint = "/"; device = "/dev/sda2"; } ];
  swapDevices = [ { device = "/dev/sda1"; } ];

  services = {
    openssh.enable = true;

    xserver = {
      enable = true;
      desktopManager.kde4.enable = true;
    };
  };

  environment.systemPackages = [ pkgs.mc pkgs.firefox ];
}
NixOS configuration

nixos-rebuild switch

- Nix package manager builds a complete system configuration
  - Includes all packages and generates all configuration files, e.g. OpenSSH configuration
- Upgrades are (almost) atomic
  - Components are stored safely next to each other, due to hashes
  - No files are automatically removed or overwritten
- Users can switch to older generations of system configurations not garbage collected yet
GNU GRUB version 0.97 (636K lower / 129984K upper memory)

NixOS - Default

Windows
NixOS - Configuration 269 (2009-08-11 23:21:10 - 2.6.27.29-default)
NixOS - Configuration 268 (2009-08-11 18:24:09 - 2.6.27.29-default)
NixOS - Configuration 267 (2009-08-05 10:47:20 - 2.6.27.29-default)
NixOS - Configuration 266 (2009-08-05 10:35:27 - 2.6.27.29-default)
NixOS - Configuration 265 (2009-08-05 10:35:06 - 2.6.27.29-default)
NixOS - Configuration 264 (2009-08-04 15:27:25 - 2.6.27.29-default)
NixOS - Configuration 263 (2009-08-04 15:07:21 - 2.6.27.29-default)
NixOS - Configuration 262 (2009-08-04 14:11:27 - 2.6.27.29-default)
NixOS - Configuration 261 (2009-08-04 10:42:23 - 2.6.27.29-default)
NixOS - Configuration 260 (2009-08-04 10:29:25 - 2.6.27.29-default)

Use the ↑ and ↓ keys to select which entry is highlighted.
Press enter to boot the selected OS, 'e' to edit the commands before booting, or 'c' for a command-line.

GNU/Linux
Deploying service-oriented systems

Nix and NixOS are not sufficient for deploying service-oriented systems:

Machine 1: x86_64-linux
- GRUB bootloader
- SSH server
- NFS server
- HTTP proxy server
- Command-line utilities
- MySQL
  - database a
  - database b
- Apache Tomcat
  - web application
  - SOAP web service
  - RESTful web service

Machine 2: i686-cygwin
- NFS client
- SSH server
- Command-line utilities
- Microsoft IIS server
  - web service
- Microsoft SQL server
  - database a
  - database b
Nix and NixOS are not sufficient for deploying service-oriented systems:
Nix and NixOS are not sufficient for deploying service-oriented systems:

**Non-functional requirements**

- Is privacy-sensitive data secured?
- Do the analysis components perform well?
- Is the system resilient to machine crashes?
- Are the software licenses governing the off-the-shelf components properly obeyed?
A Reference Architecture for Distributed Software Deployment

A diagram illustrating the architecture with various components and their connections.
Service deployment

A Reference Architecture for Distributed Software Deployment
Disnix

- Distributed deployment extension for the Nix package manager
- Captures deployment specification in models
- Performs complete deployment process from models
- Guarantees complete dependencies
- Component agnostic
- Supports atomic upgrades and rollbacks
SDS2: Distribution

A Reference Architecture for Distributed Software Deployment

[Diagram showing distribution architecture with components and communication protocols]
$ disnix-env -s services.nix -i infrastructure.nix -d distribution.nix
Disnix expressions

MELogService.nix

{javaenv, config, SDS2Util}:
{mobileeventlogs}:

javaenv.createTomcatWebApplication rec {
    name = "MELogService";
    contextXML = "
        <Context>
            <Resource auth="Container" type="javax.sql.DataSource"
                url="jdbc:mysql://${mobileeventlogs.target.hostname}..." />
        </Context>
    ";

    webapp = javaenv.buildWebService {
        inherit name;
        src = ../../../WebServices/MELogService;
        wsdlFile = "MELogService.wsdl";
        libs = [ config SDS2Util ];
    };
}
Service model

{distribution, system}:

let pkgs = import ..../top-level/all-packages.nix {
    inherit distribution system;
}; in

{ mobileeventlogs = {
    name = "mobileeventlogs";
    pkg = pkgs.mobileeventlogs;
    type = "mysql-database";
};

MELogService = {
    name = "MELogService";
    pkg = pkgs.MELogService;
    dependsOn = { inherit mobileeventlogs; };
    type = "tomcat-webapplication";
};

SDS2AssetTracker = {
    name = "SDS2AssetTracker";
    pkg = pkgs.SDS2AssetTracker;
    dependsOn = { inherit MELogService ...; };
    type = "tomcat-webapplication";
};
...
}
Captures machines in the network and their relevant properties and capabilities.
{infrastructure}:
{
    mobileeventlogs = [ infrastructure.test1 ];
    MELogService = [ infrastructure.test2 ];
    SDS2AssetTracker = [ infrastructure.test1 infrastructure.test2 ];
    ...
}
Specifications are used to derive deployment process:

- **Building** services from source code
- **Transferring** services to target machines
- **Deactivating** obsolete services and **activating** new services
Various events may occur in a network of machines:

- Crashing machines
- Adding a new machine
- Change of a capability (e.g. increase of RAM)
- Dynamic Disnix generates infrastructure and distribution models and redeploys a system
Other deployment aspects

A Reference Architecture for Distributed Software Deployment

Sander van der Burg
Distributed NixOS configuration

network.nix

```nix
{ storage = {pkgs, ...}: 
    { 
        services.nfsKernel.server.enable = true; ...
    };

postgresql = {pkgs, ...}: 
    { 
        services.postgresql.enable = true; ...
    };

webserver = {pkgs, ...}: 
    { 
        fileSystems = [ 
            { mountPoint = "/repos"; device = "storage:/repos"; } ]; 
        services.httpd.enable = true; 
        services.httpd.extraSubservices = [ { serviceType = "trac"; } ]; ...
    };

... 
}
```
Virtualization

nixos-build-vms network.nix; ./result/bin/nixos-run-vms

- Builds a network of QEMU-KVM virtual machines closely resembling the network of NixOS configurations
- We don’t create disk images
- The VM mounts the Nix store of the host system using SMB/CIFS
Virtualization

Sander van der Burg

A Reference Architecture for Distributed Software Deployment
testScript = ''
$postgresql→waitForJob("postgresql");
$postgresql→mustSucceed("createdb trac");

$webserver→mustSucceed("mkdir -p /repos/trac");
$webserver→mustSucceed("svnadmin create /repos/trac");

$webserver→waitForFile("/var/trac");
$webserver→mustSucceed("mkdir -p /var/trac/projects/test");
$webserver→mustSucceed("trac-admin /var/trac/projects/test initenv ".
  "Test postgres://root@postgresql/trac svn /repos/trac");

$client→waitForX;
$client→execute("konqueror http://webserver/projects/test &");
$client→waitForWindow(qr/Test.*Konqueror/);

$client→screenshot("screen");
'';
nix-build tests.nix -A trac
License analysis

- We can also trace all files and processes involved in a build process
- And we can determine the licenses of the original source files to say something about the result
We have shown a reference architecture for distributed software deployment

Reference architecture is based on Nix, a purely functional package manager, and NixOS a Linux distribution built around Nix

We have shown tools to automate the deployment of distributed systems

They provide fully automatic, reliable, reproducible, and efficient deployment for the latest generation of systems

Components of the reference architecture can be used to construct a domain-specific deployment tool
NixOS website: http://nixos.org

- **Nix**: A purely functional package manager
- **Nixpkgs**: Nix packages collection
- **NixOS**: Nix based GNU/Linux distribution
- **Hydra**: Nix based continuous build and integration server
- **Disnix**: Nix based distributed service deployment
- **NixOps**: NixOS-based multi-cloud deployment tool

Software available under free and open-source licenses (LGPL/X11)