

# IT841: Mac OS X Laptop Deployments with Puppet

[http://link-to-presentation/IT841\\_JMcCune.pdf](http://link-to-presentation/IT841_JMcCune.pdf)

Nigel Kersten

*Mac Whisperer, Google.*

Jeff McCune

*Senior Systems Administrator, Netsmart Technologies*

# Question and Answer Information

Please visit:

<http://tinyurl.com/633v6e>

# Puppet is ...

- A configuration management system
- Completely Open Source (Free license)
- A very simple syntax
- Supported on most \*nix platforms
- <http://puppet.reductivelabs.com>

# Puppet is (continued)

- Network-able
  - Client - puppetd
  - Server - puppetmasterd
- Declarative
  - You say what, puppet sorts out how
  - Unlike bash / perl / python / ruby
- Idempotent
  - Can be run multiple times
  - Only makes the changes it needs to

# Puppet is (continued)

A way to manage more than just files...

- Manages resources like:
  - Users, Groups
  - Packages
  - Services (via launchd)
  - MCX
  - Attributes and/or content of files

# Nigel's Story

- Large, diverse and mobile environment at Google
- Needed a flexible solution
- Minimal to comprehensive management
- Must be extensible
- Must work offline
- Must work with Source Code Management

# Jeff's Story

- Mac OS X SA Years - OSU Mathematics
  - 130+ Mac OS X Workstations
- Linux / Solaris Servers
  - CentOS, RedHat, Solaris 10, OpenSolaris
  - Agility is a requirement
  - Rapidly growing SAAS environment

# Why Puppet on Laptops?

Laptops pose a unique and difficult challenge

- Diverse
- Must co-exist with user customizations
- Must work offline
- Must be flexible



# Why Puppet on Servers?

- If you do something once, you'll likely have to do it again
- Guaranteed state
- Disaster Recovery
- Agility
- Work smarter, not harder

# A Crash Course in Puppet Syntax

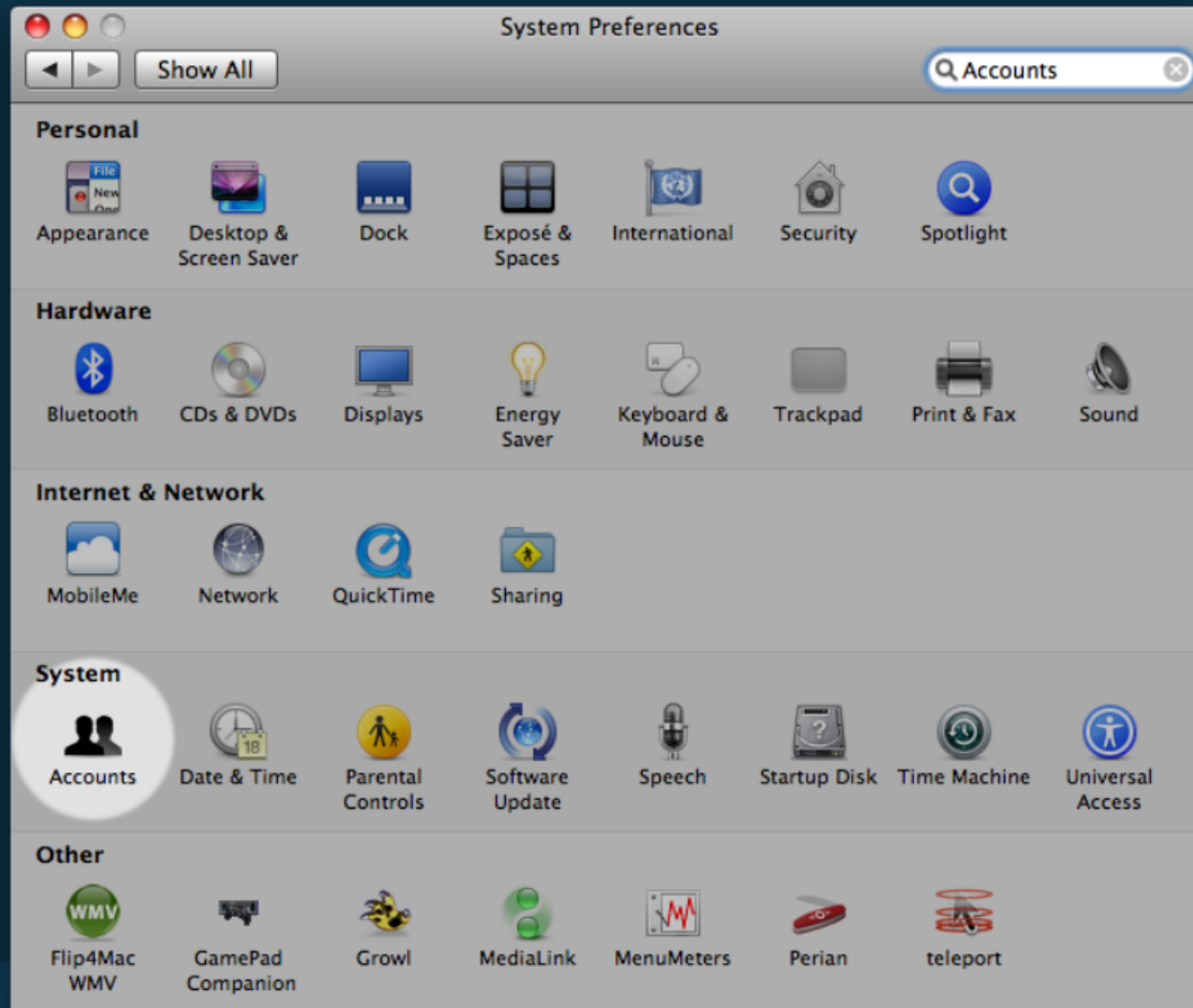
```
file { /etc/motd:  
  content => "Welcome to Macworld",  
}
```

- *file* is the type of the resource
- */etc/motd* is the label of the resource
- *content* is a property of the resource
- *"Welcome..."* is a value of the property

# New Account Demo

- First, create a new account in WGM or System Preferences
- Next, inspect the puppet syntax of this account using `ralsh`
- Finally, use puppet to create the account on any number of other computers

# System Preferences




# New Account

New Account:

Name:


Short Name:

Password:  

Verify:

Password Hint:   
(Recommended)

Turn on FileVault protection



# On ralsh

- Resource Abstraction SHell
- Proof of concept, not guaranteed to produce perfect puppet syntax
- Provides a great starting point
- Reduces the syntax learning curve
- Not a *perfect* description of your resources in puppet syntax

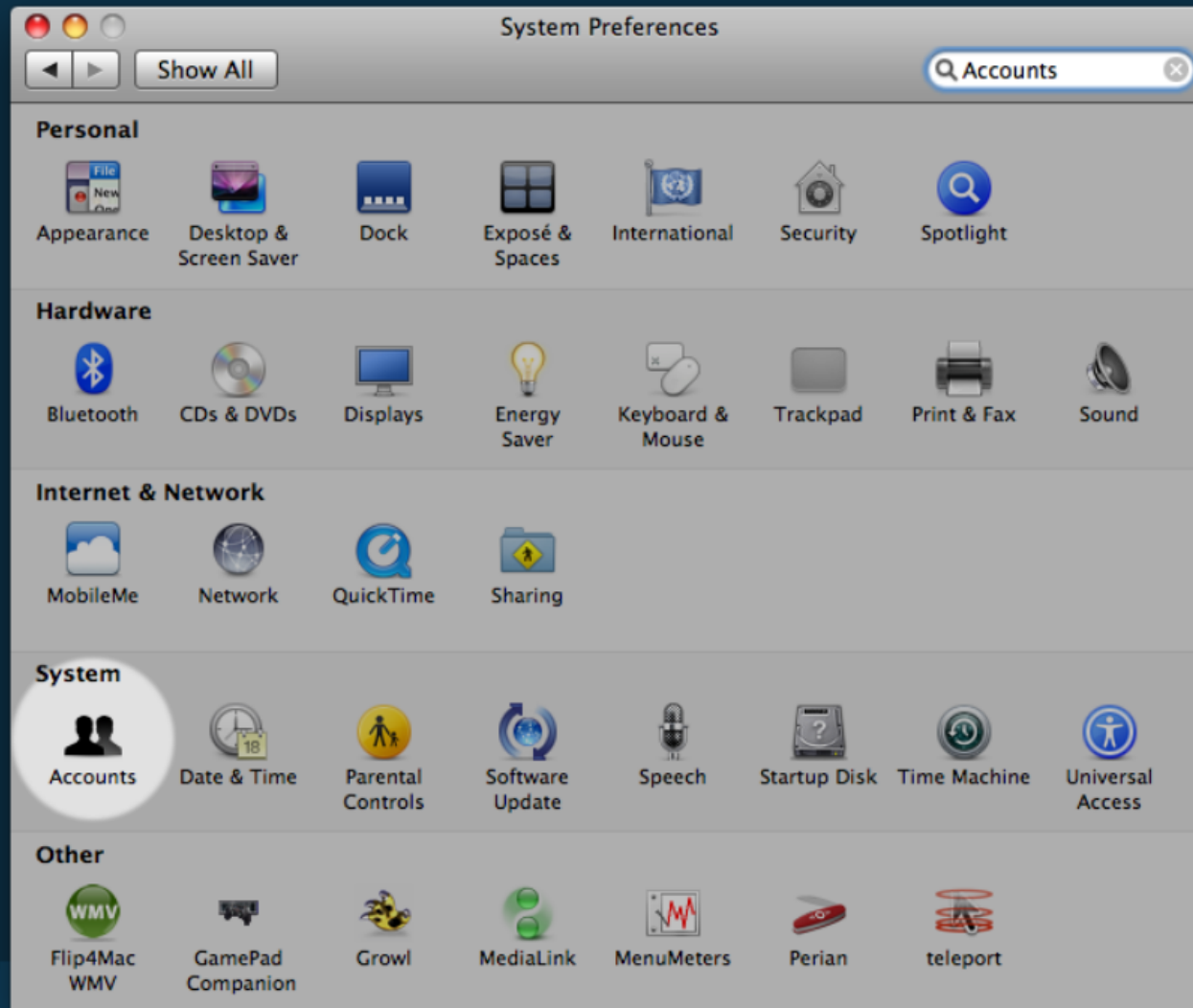


# Group Management Demo

- Create a new group using WGM or System Preferences
- Add account members to the new group
- Derive the puppet resource using `ralsh`
- Run the puppet resource on other machines to create the new group




# System Preferences



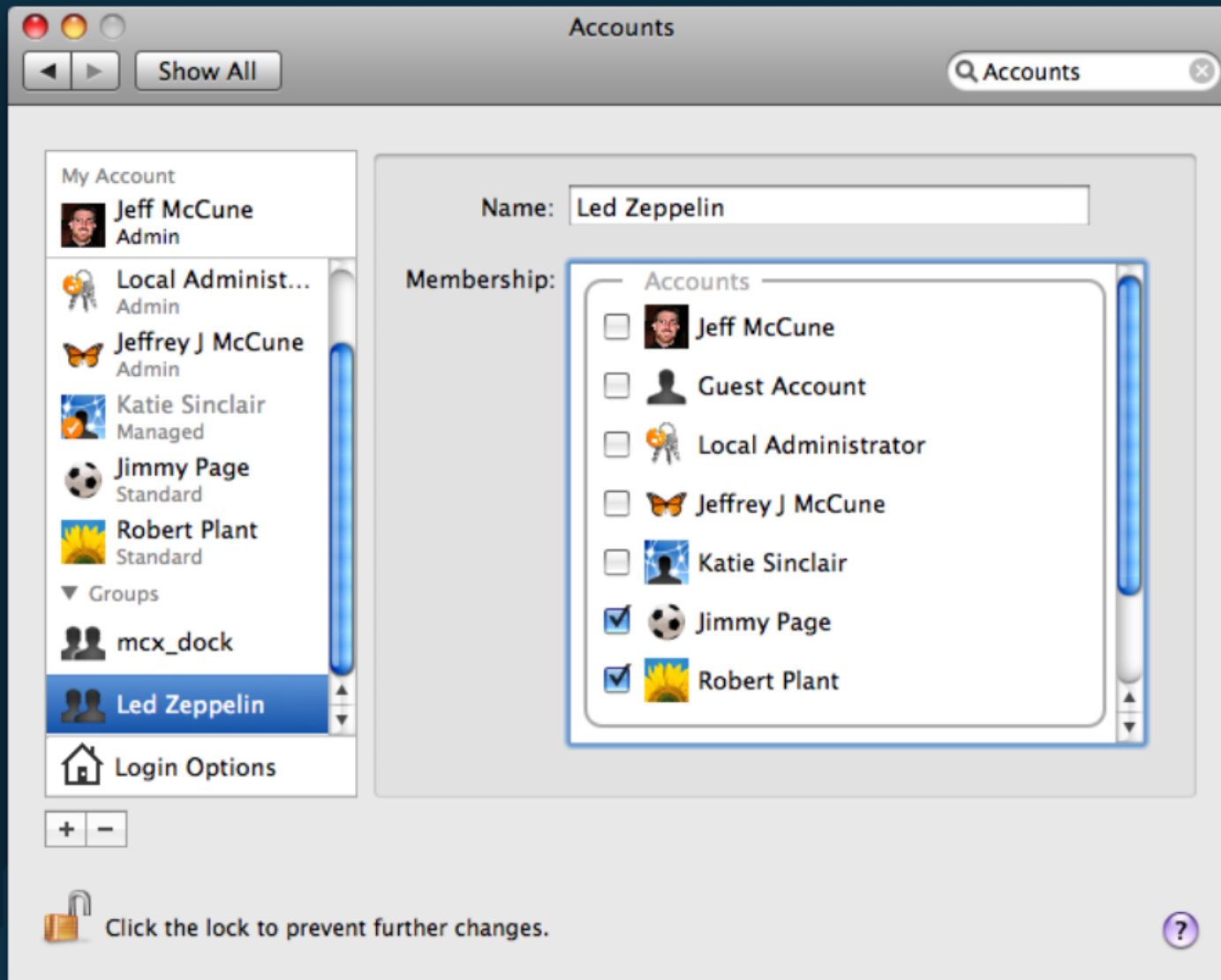
# New Group

New:

Name:



# Group Membership



# ralsh

```
nigelk@demo: ~  
demo: ~ $ sudo ralsh group ledzeppelin  
group { 'ledzeppelin':  
  members => ['jpage', 'rplant'],  
  ensure => 'present',  
  gid => '502'  
}  
demo: ~ $ █
```

# MCX Management Demo

- Use Workgroup Manager to define managed settings on an existing User, Group or Computer
- Inspect the puppet syntax with `ralsh`
- Run the puppet syntax on other machines in order to deploy the same state

# Workgroup Manager

The screenshot shows the 'Workgroup Manager: Local' window. The title bar includes standard macOS window controls and a menu bar with options: Server Admin, Accounts, Preferences, New User, Delete, Refresh, New Window, and Search. Below the menu bar, it indicates the user is authenticated as 'mccune' to the local directory at '/Local/Default'. The main interface is divided into a left sidebar and a right main panel. The sidebar contains a search bar with the text 'Name Contains', a list of users with columns for 'User Name' and 'UID', and a scroll bar at the bottom showing '0 of 7 users selected'. The main panel has tabs for 'Basic', 'Advanced', 'Groups', 'Home', 'Mail', 'Print Quota', 'Info', 'Windows', and 'Inspector'. The 'Basic' tab is active, showing a form for creating a new user with fields for Name, User ID, Short Names, Password, and Verify. There are also checkboxes for 'User can administer this server' and 'User can access account'. An 'Account Summary' section at the bottom of the form lists fields for Location, Home, Primary Group, Mail, Print Quota, and Password. At the very bottom of the window, there is a 'Presets' dropdown menu set to 'No Presets' and 'Revert' and 'Save' buttons.

Workgroup Manager: Local

Server Admin Accounts Preferences New User Delete Refresh New Window Search

Authenticated as mccune to local directory: /Local/Default

Name Contains

User Name	UID
Guest Account	201
Jeff McCune	502
Jeffrey J McCune	503
Jimmy Page	505
Katie Sinclair	504
Local Administrator	501
Robert Plant	507

Basic Advanced Groups Home Mail Print Quota Info Windows Inspector

Name:

User ID:

Short Names:

Password:  Verify:

User can  administer this server  
 access account

Account Summary

Location:

Home:

Primary Group:

Mail:

Print Quota:

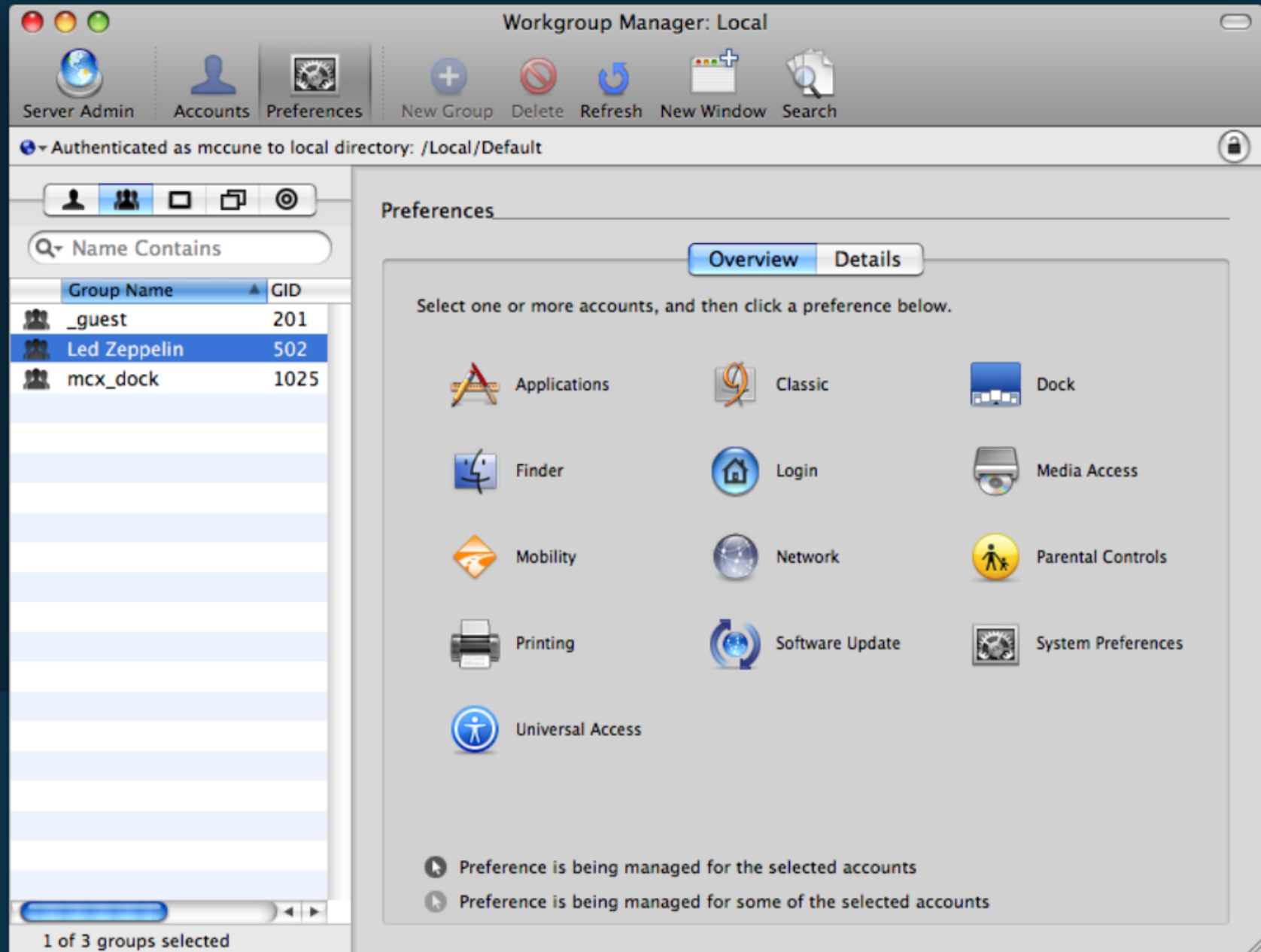
Password:

Presets: No Presets

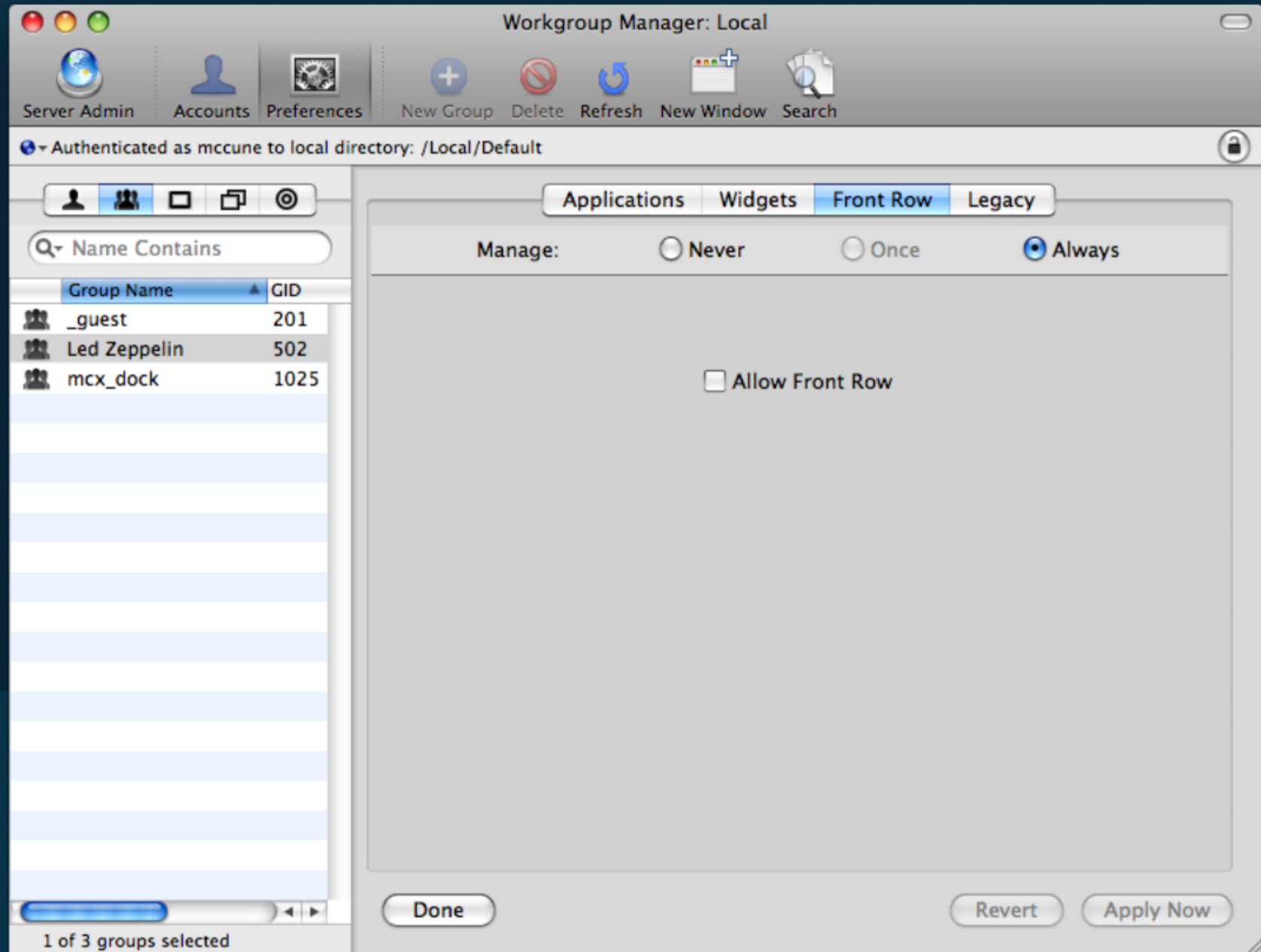
Revert Save

0 of 7 users selected

# Managed Preference



# Manage Front Row





# Review MCX Settings

The screenshot shows the 'Workgroup Manager: Local' window. The title bar includes 'Server Admin', 'Accounts', and 'Preferences' buttons. Below the title bar is a toolbar with icons for 'New Group', 'Delete', 'Refresh', 'New Window', and 'Search'. The main content area is titled 'Authenticated as mccune to local directory: /Local/Default'. On the left, a list of groups is shown with columns for 'Group Name' and 'GID'. The 'mcx\_dock' group is selected. On the right, the 'Preferences' pane is open, showing a grid of preference categories. The 'Overview' tab is active. At the bottom, there are two status indicators: 'Preference is being managed for the selected accounts' and 'Preference is being managed for some of the selected accounts'.

Group Name	GID
_guest	201
Led Zeppelin	502
mcx_dock	1025

Preferences

Overview Details

Select one or more accounts, and then click a preference below.

- Applications
- Classic
- Dock
- Finder
- Login
- Media Access
- Mobility
- Network
- Parental Controls
- Printing
- Software Update
- System Preferences
- Universal Access

1 of 3 groups selected

- Preference is being managed for the selected accounts
- Preference is being managed for some of the selected accounts

# ralsh

nigelk@demo: ~

```
demo: ~ $ sudo ralsh mcx /Groups/ledzeppelin
```

```
mcx { '/Groups/ledzeppelin':
```

```
  content => '<?xml version="1.0" encoding="UTF-8"?>
```

```
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
```

```
<plist version="1.0">
```

```
<dict>
```

```
  <key>com.apple.frontrow</key>
```

```
  <dict>
```

```
    <key>PreventActivation</key>
```

```
    <dict>
```

```
      <key>state</key>
```

```
      <string>always</string>
```

```
      <key>value</key>
```

```
      <string>YES</string>
```

```
    </dict>
```

```
  </dict>
```

```
</dict>
```

```
</plist>
```

```
'
```

```
  ensure => 'present'
```

```
}
```

```
demo: ~ $ █
```

# Putting it all together

```
Terminal — bash — bash — 100x24
[mccune@phi ~]$ sudo puppet -v mw2009.pp
notice: //User[rplant]/ensure: created
notice: //User[jpage]/ensure: created
notice: //Group[ledzeppelin]/ensure: created
notice: //Mcx[/Groups/ledzeppelin]/ensure: created
[mccune@phi ~]$ sudo puppet -v mw2009_clean.pp
notice: //Group[ledzeppelin]/ensure: removed
notice: //User[jpage]/ensure: removed
notice: //User[rplant]/ensure: removed
[mccune@phi ~]$ sudo puppet -v mw2009.pp
notice: //User[rplant]/ensure: created
notice: //User[jpage]/ensure: created
notice: //Group[ledzeppelin]/ensure: created
notice: //Mcx[/Groups/ledzeppelin]/ensure: created
[mccune@phi ~]$ █
```

# Managing launchd services

- Puppet can manage launchd services
- Uses the "job label" as the Puppet label
- Manages *system* LaunchDaemons, LaunchAgents
  - /Library/LaunchDaemons
  - /Library/LaunchAgents
  - /System/Library/LaunchDaemons
  - /System//Library/LaunchAgents

# Managing launchd services

```
root@demo: ~
demo: ~ $ sudo rals service com.openssh.sshd
service { 'com.openssh.sshd':
  enable => 'true',
  ensure => 'running'
}
demo: ~ $
```

- *enable* - Whether the service starts at boot
- *ensure* - The current state - running/stopped

# Managing Relationships

- Client states can be complex
- Resources have required relationships
- Puppet can express these relationships

# Managing Relationships

Relationship parameters:

- *before* - This resource is applied before another resource
- *require* - This resource is applied after another resource
- *subscribe* - Resource changes notify another resource

# Managing Relationships

```
user { "ianmackaye":  
  home => "/Users/ianmackaye",  
  ...  
}
```

```
file { "/Users/ianmackaye/Desktop/README.txt":  
  require => User["ianmackaye"],  
  ...  
}
```



# Managing Relationships

```
service { "com.pretendco.foo":  
  enable => 'true',  
  ensure => 'running',  
  require => File["/etc/foo.conf"],  
}
```

```
file { "/etc/foo.conf":  
  ...  
}
```

# Want to learn more?

- <http://puppet.reductivelabs.com>
- <http://groups.google.com/group/puppet-users>
- “Pulling Strings with Puppet” by James Turnbull
- IRC: #puppet on freenode network
- [jeff@northstarlabs.net](mailto:jeff@northstarlabs.net)
- [nigel@explanatorygap.net](mailto:nigel@explanatorygap.net)

Fin.

