Quick, Easy and Accurate Linux Deployment Under z/VM

SHARE - Baltimore
August 17, 2006
Session 9275

Edmund MacKenty
Rocket Software, Inc.
Agenda

• Manual deployment of Linux under z/VM
• Automated deployment
• Deploying software updates
• Using VSWITCHes and multiple network interfaces
Who is Rocket Software?

- 300+ person software development firm based in Newton, MA
- Development labs in
  - Berkeley, CA
  - Austin, TX
  - Bentonville, AR
  - Warwick, UK
  - Paris, France
  - Chelyabinsk, Russia
  - Beijing, China
- Extensive portfolio of Enterprise Infrastructure Management products
  - Business Intelligence
  - Database
  - Security & Identity
  - Network Discovery & Mgmt
  - zSeries Systems Mgmt
  - Storage
  - Telecom & OSS
  - Application Development
  - Windows & Linux Tools
What Do People Need To Do?

- Create many Linux guests when they are needed
- Destroy Linux guests when they are not needed
- Create copies of systems for testing new software
- Configure guests for different purposes
- Deploy applications
- Apply software updates
- Maintain everything forever
What Do People Not Want To Do?

• Become a Linux expert if you're not one
• Become a z/VM expert if you're not one
• Wade through a 300-page RedBook
• Take months to get it all done
Manual Deployment: Cloning

- Makes exact copies of existing system
- All clones have same filesystem structure
- Manually allocates guests, DASD and network interfaces
- Done ad hoc by z/VM admin using some scripts
- Takes several hours

Diagram:
- Parent Guest
  - Clone
  - Clone
  - Clone
  - Clone
Manual Deployment: Provisioning

- Installs software on clones
- Run by Linux sysadm
- Takes several hours
- Sometimes configures software too
Manual Deployment: Configuration

- Manually configuring each Linux clone
- Copying identical configurations to each clone
- Using various cluster tools
Manual Deployment: Software Updates

- Run installation process on each clone
- Possibly reconfigure new software on each clone
- Some installers require a GUI
How Can I Automate This?

- Any complex task should be automated
- Avoid procedural mistakes
- Capture process knowledge in software
- Save time, avoid repetitive tasks
- Reduce your cognitive load
Automated Deployment: Terminology

- **Instance**: An automatically deployed Linux guest
- **Configuration**: Rules describing how to build an instance
- **Template**: Shareable subsets of configuration rules
- **Base System**: Original installation used to manage deployments
- **Filesystem Snapshot**: Copy of base system shared by instances
- **Groups**: Organize instances into a hierarchy
Sharing DASD between Instances

Base System

0201
0202

Filesystem Snapshot

/ /boot
/opt /usr

Instance1

/ /opt

Instance2

/ /opt
Automated Deployment: Process

- Create a Filesystem Snapshot of the Base System
- Define a Configuration suited to your purposes
- Create and start an Instance using that Configuration
- Copy the Configuration to make Clones

Collect Configuration Data

- Create VM Guest
- Allocate DASD
- Construct Linux Filesystems
- Apply Configuration To Filesystems
Automated Deployment: Components

- **Base System**: A guest with Linux installed into it
  - All Linux software is copied from this guest
  - Provisioning Expert application is installed here
  - Stores all configurations and other data
  - Creates web-based graphical interface

- **VM service machine**: A guest for resource management
  - Manages VM resources
  - Defines guests
  - Allocates DASD from a defined pool
  - Uses DIRMAINT API to alter the directory
  - Uses ESM (e.g., RACF) to control resource access
  - Dynamic linking/detaching of DASD
  - z/VM admin controls all resources
  - Used only by the Base System, not by Instances
Under the Covers: Filesystem Snapshots

- A Filesystem Snapshot is a collection of DASD extents
- Allows for Shared and Instance-specific filesystems
- Defines structure of Linux filesystems for Instances
- Multiple snapshots allows for versioning
- Does not depend on distro's installation process
Under the Covers: Snapshot Layouts

• Describe structure of filesystem tree
• Defines default filesystem types, mount points and options
• Specifies which Base System files are copied into a snapshot, and where they should be placed.
• Says which files are shared (read-only) or not (writeable)
• Can split up directories, making some files writeable and others read-only
• Can arrange for parts of a filesystem to be bind-mounted into other filesystems
Under the Covers: Bind-Mounting

- Allows arbitrary divisions of the Linux directory structure
  - Snap1 contains all system software
  - Snap2 contains all usr software
  - Inst1 contains /dev, /etc, /opt, /tmp and /var; is instance's root
  - Inst2 contains empty space for home directories
  - Inst3 overrides part of /usr, making more space available in /usr/local
Filesystem Snapshot Creation

- Determine sizes of filesystems defined by Layout
- Account for filesystem overhead
- Allocate DASD extents for each filesystem
- Format, partition and make filesystems on new DASD
- Mount new DASD onto base system
- Copy files onto new DASD following Layout rules
- Unmount the new DASD
Automated Deployment: Configuration

- Define how you want the instance to be set up
  - Hardware configuration
  - Software configuration
- Example: an NFS Server
- Explicit Configuration:
  - Specify Instance name
  - Set root password
  - Select a Filesystem Snapshot
  - Export a directory
- Implicit Configuration:
  - Assignment of IP addresses
  - Create a default route
  - Set up the root user account
  - Start NFS daemons
Automated Deployment: Create and Start

Steps to Create an Instance:

- Define a new guest
- Allocate Instance-specific DASD, owned by that guest
- Copy Instance-specific data from Snapshot to new DASD
- Bind-mount DASD onto the base system
- Run configuration scripts on the new DASD
  - Writes /etc/fstab
  - Defines network interface(s)
- Run `mkinitrd` and `zipl` on the new DASD
- Unmount the new DASD

Steps to Start an Instance:

- IPL the new guest
- Wait for Instance to respond on network
Extensibility Features

• Application Configuration Scripts
  - Let you automate configuration of your applications
  - It's just a shell script
  - Built on a rich set of common functions

• Instance Configuration Implementation Scripts
  - Allow for distro-specific configuration
  - Built on a rich set of common functions

• XML Schemas
  - Allows adding to configuration language
  - Configure new kinds of things
  - You had better understand XML

• Snapshot Layout Build Instructions
  - Allows for different layouts of snapshots
  - Can divide filesystems in different ways
  - Not for the faint of heart
Automated Software Updates

• Install new software on the Base System
• Create a new Filesystem Snapshot of the Base System
• Update existing Instances to use the new Snapshot:
  ▪ Stops the Instance, if it is running
  ▪ Detaches old Snapshot's DASD from the instance
  ▪ Links new Snapshot's DASD in its place
  ▪ Mounts Instance-specific DASD on the base system
  ▪ Copies any new files to Instance-specific filesystems
  ▪ Unmounts Instance's DASD
  ▪ Starts the Instance, if it was running
Multiple Network Interfaces

- Uses pre-existing VSWITCHes or CTC
- Data for each network segment is configured by admin
- Assignment of IP addresses is automatic
- Instance configuration is just selecting a network segment
- Default route uses first defined segment
Summary

• Cloning, Provisioning and Configuration is a lot of work
• There is a lot of information to keep track of
• Automation makes it easy to produce different instances
• Uses a high-level configuration language
• Treats virtual hardware and software the same way
• Filesystem Snapshots make software upgrades easier
• And you can back out those upgrades
Quick, Easy and Accurate Linux Deployment Under z/VM

Contact Information:

Presenter: Ed.MacKenty@RocketSoftware.com
Company: http://www.RocketSoftware.com/
Product: http://www.rs.com/portfolio/provisioningexpert/