MIRANTIS

MaaS Overview

Metal as a Service (MaaS) for bare metal provisioning

training.mirantis.com

Objectives

- MaaS concepts, provisioning flow
- MaaS overview, components, architecture
- MaaS open source alternatives
- Hands-on experience with MaaS

What is MaaS

What is MaaS, its limitations and alternatives, use cases, concepts, basic flows

What is MaaS

- Metal as a Service (MaaS) is a bare metal provisioning tool developed by Canonical
  - Serves as a layer underneath Infrastructure-as-a-Service (IaaS)
  - Brings a bare metal server with no operating system installed to a completely working server
- MaaS is an open source (AGPL v3 licence)
- Allows provisioning Windows, Ubuntu, CentOS, RHEL and SUSE

What is MaaS

- Comes with GUI and command line tools
- Can be integrated with configuration management tools e.g. Salt, using its RESTful API
- Scalability and HA

MaaS limitations

- MaaS is not a complete Infrastructure as a Service (IaaS) solution (like OpenStack)

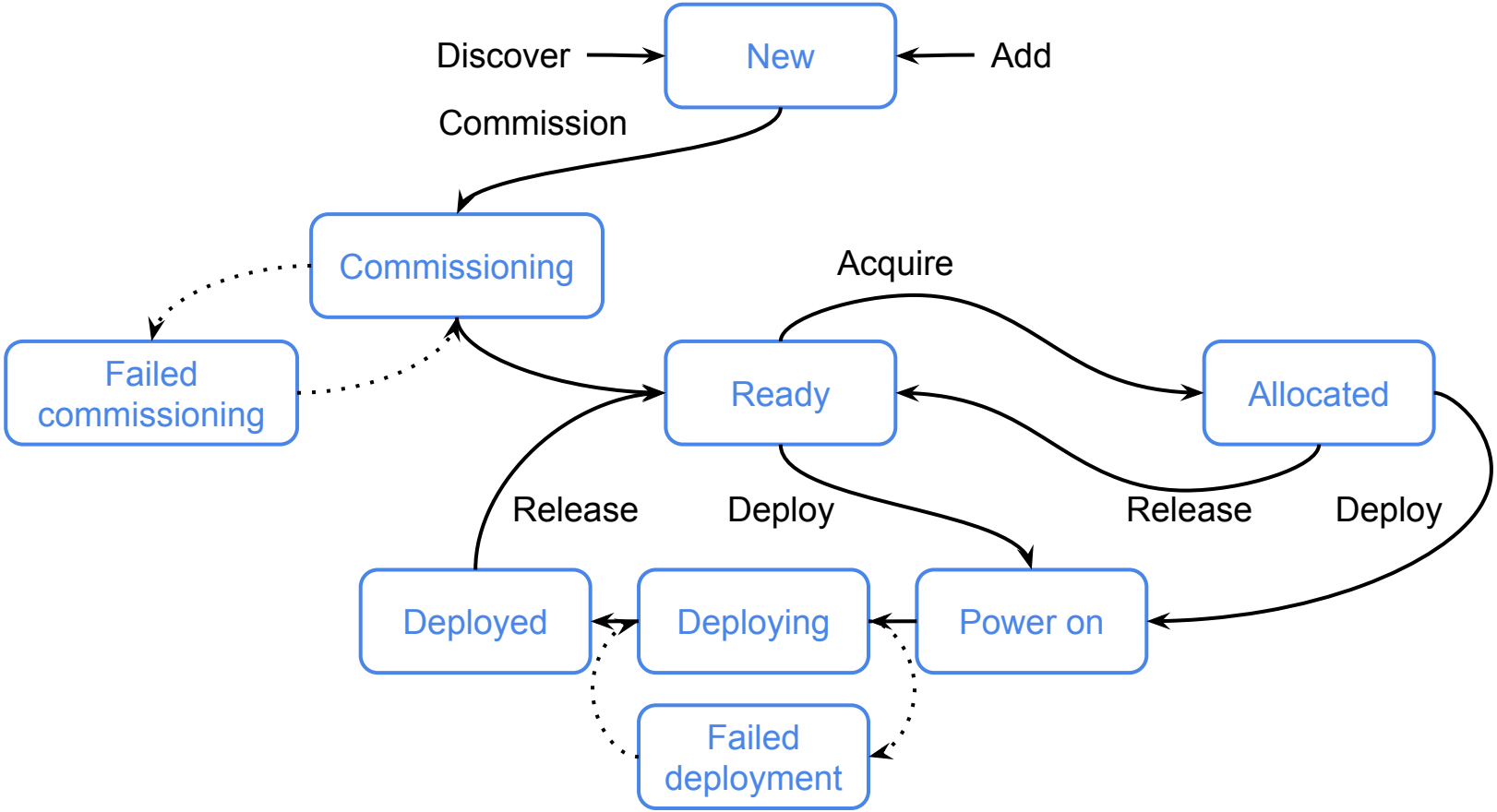
How MaaS works: The simplest flow

- Boot your servers via PXE
  - Or manually add your servers to MaaS (Name, MAC, IP)
  - Or export the existing information about servers from IMS
- MaaS discovers the servers and shows them in UI
  - Edit servers' details, such as name, power management options
- Commission the servers
- Deploy operating systems on the servers

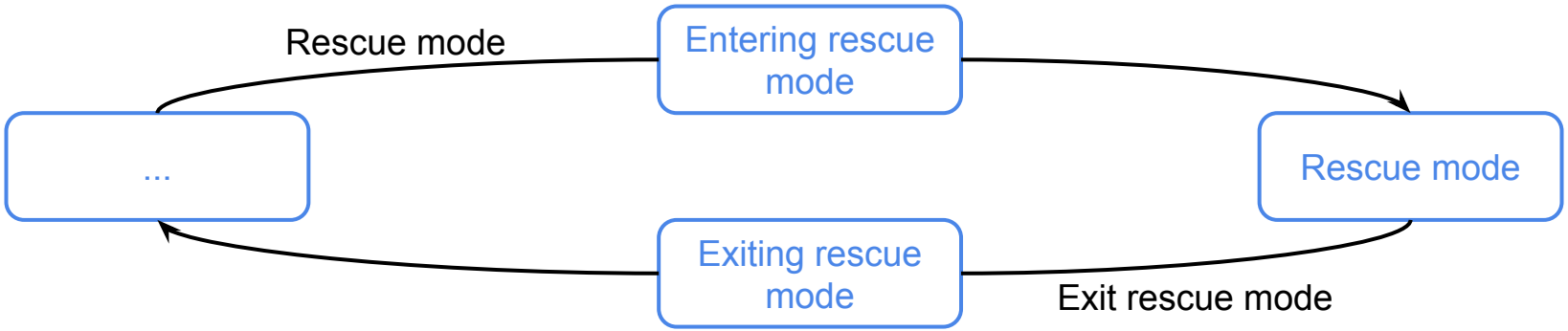
MaaS concepts

- Node - a networked object that is known to MaaS
  - Controller - internal MaaS node
  - Machine - node deployable by MaaS
  - Device - non-deployable node (e.g. router)
- Zone, Region
- Fabric, Subnet
- IP Range
  - Reserved range - an IP range that MAAS will never use
  - Reserved dynamic range - an IP range that MAAS will use for enlisting, commissioning and deploying

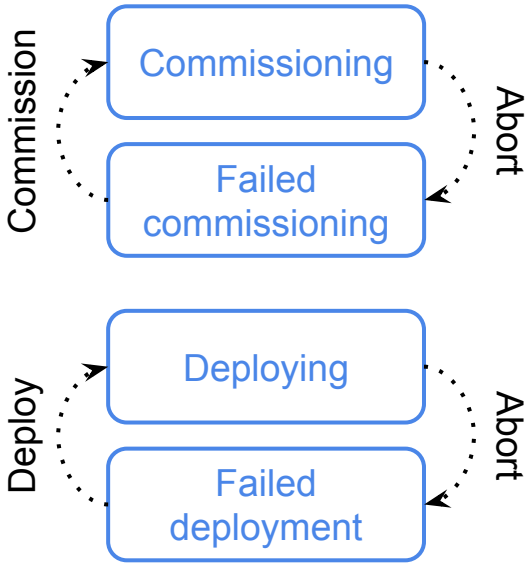
MaaS statuses and actions: Commissioning and Deploying

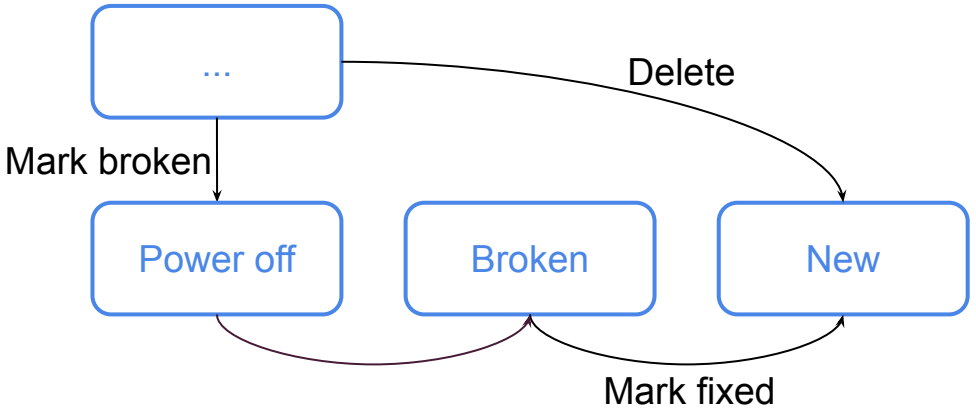


MaaS statuses and actions: Rescue Mode



MaaS statuses and actions: Abort, Delete, Mark Broken





Provisioning types

Classic (or unattended)

- PXE to load installer
- Installer loads selected packages from medium (CDROM, ISO) or via network and installs them to the prepared file system
- Automation via defining answers for installer questions
- Examples:
  - Linux Kickstart
  - Preseed
  - AutoYAST

Image-based

- PXE to load bootstrap image
- Bootstrap loads base image and copies files from the image to the prepared file system
- Automation via post-customization
- Examples:
  - Curtin

MaaS open source alternatives

- Do It Yourself (DIY)
- Cobbler
- Foreman
- OpenStack Ironic

MaaS open source alternatives: DIY

- IPMI tool (potentially vendor specific)
- DHCP server
- TFTP server
- Bootable OS image or bootstrap image
- Installer configuration file(s):
  - Linux Kickstart for Red Hat based OS
  - Preseed for Debian based OS
  - AutoYAST for SUSE Linux
  - Curtin for Ubuntu based OS

MaaS open source alternatives: Ironic

- OpenStack program integrated with the OpenStack Keystone, Nova, Neutron, Glance, Swift
- Integration with Ansible
- Power management

MaaS Architecture

Underlying technologies, components

How MaaS works: Underlying technologies

- Written in Python
- PostgreSQL to store persistent data
- IPMI
- DHCP / TFTP
- PXE / UEFI / Open Firmware
- iSCSI
- initrd
- Squashfs
- Curtin

MaaS controllers

Region Controller

- REST API server (TCP port 5240)
- PostgreSQL database
- DNS
- Caching HTTP proxy
- Web UI

Rack Controller

- DHCP
- TFTP
- HTTP (for images)
- iSCSI
- Power management

Deployment flow

- DHCP server (Rack ctl) is contacted
- Kernel, initrd and parameters are received over TFTP (Rack ctl)
- Machine boots
- Initrd mounts a Squashfs image ephemerally over iSCSI (Rack ctl)
- Cloud-init downloads Curtin script (Region ctl)
- Cloud-init triggers deployment process
  - Cloud-init executes Curtin
  - Squashfs image (same as above) is placed on disk

What is Curtin (Curt installer)

- Ubuntu installer that uses an image-based method
- Replaces Debian installer (preseed)
- Allows specifying provisioning commands and options
  - *Early commands* (load modules, setup hardware) executed on the system, and non-zero exit status will terminate the installation process
  - *Partitioning*
  - *Network Discovery and Setup*
  - *Sources*
  - *Final commands*
  - ...

Curtin file example

```
early_commands:
  driver_01_install: ["sh", "-c", "apt-get update --quiet && apt-get --assume-yes install driver"]
  driver_02_load: ["sh", "-c", "depmod && modprobe driver"]

apt_mirrors:
  ubuntu_archive: http://local.archive/ubuntu
  ubuntu_security: http://local.archive/ubuntu

sources:
  - https://cloud-images.ubuntu.com/xenial/current/xenial-server-cloudimg-amd64-root.tar.gz

power_state:
  mode: poweroff
  delay: 5
  message: Bye Bye
```

MaaS scalability and HA

- Rack controller HA
  - DHCP HA
  - Power management HA
- Region controller HA
  - PostgreSQL HA
  - API HA

MaaS scalability and HA: Rack Controller

- Install multiple rack controllers and register them in region controller
- DHCP is added intelligently when a new rack controller is installed and DHCP HA will become available as an option*
- Power management HA is provided out of the box once a second rack controller is present

MaaS scalability and HA: Region Controller

- PostgreSQL HA is external to MaaS
- Any number of API servers can be present as long as each connects to the same PostgreSQL HA database
- Load balancing for API server
  - Install and configure HAProxy on each API server host
- Virtual IP as the effective IP address of all region API servers
  - Install and configure keepalived on each API server host

MaaS Installation and Configuration

MaaS Installation and Configuration Flow

- Install MaaS
- Create admin user
- Download boot images
  - Create local mirror, if necessary
- Provide ssh key
- Configure networks
  - Enable MaaS DHCP, if necessary, or
  - Configure external DHCP server
- Provide custom Curtin files, if necessary

MaaS installation

There are three ways to install MaaS:

- From an Ubuntu Server ISO
  - Install a complete MAAS environment or a rack controller during the ISO installation of Ubuntu Server.
- From packages
  - Available since Ubuntu 12.04
  - Available in the normal Ubuntu archive
- With LXD
  - Self-contained MAAS environment in LXD containers

MaaS Installation from packages - Single server

- For a version shipped with your Ubuntu
  - do nothing
- For the newest stable version:
  - `apt-add-repository -yu ppa:maas/stable`
- For the current development release:
  - `apt-add-repository -yu ppa:maas/next`

```
apt update
apt install maas
```

MaaS Installation from packages - Distributed environment

- Install the region controller on one machine:

```
apt install maas-region-controller
```

- Install the rack controller on another:

```
apt install maas-rack-controller
maas-rack register
```

Local image mirror

- Images are delivered to MAAS via the SimpleStreams protocol
- Create local image mirror
  - Install `simplestreams` package
  - Choose streams to mirror
  - Use `sstream-mirror` to create a local mirror
- Configure MAAS to use the local mirror
  - UI (Settings, Boot Images, Sync URL and Keyring Path)
  - CLI (`maas boot-sources create`)

External DHCP

- If you are using MaaS in a setup with an existing DHCP, do not set up the MaaS DHCP server
- Use the MaaS reserved IP range
- Configure external DHCP server
  - `bootp` option to point to MaaS rack controller, or
  - `next-server` to point to MaaS rack controller
- Example for dnsmasq

```
dhcp-boot=pxelinux.0,maas_ip,maas_ip
```

Custom Curtin files

- Put custom curtin files to `/etc/maas/preseeds/`
- The files are looked up in the following order:

```
{prefix}_{osystem}_{node_arch}_{node_subarch}_{release}_{node_name}
{prefix}_{osystem}_{node_arch}_{node_subarch}_{release}
{prefix}_{osystem}_{node_arch}_{node_subarch}
{prefix}_{osystem}_{node_arch}
{prefix}_{osystem}
{prefix}
'generic'
```

- {prefix} is either empty (in this case the following underscore is also omitted), or one of the of:

```
enlist, enlist_userdata, commissioning, curtin, curtin_userdata
```

- Examples (Curtin files for OS installation):

```
curtin_userdata_amd64_generic_trusty
curtin_userdata_amd64_generic_trusty_server1
```