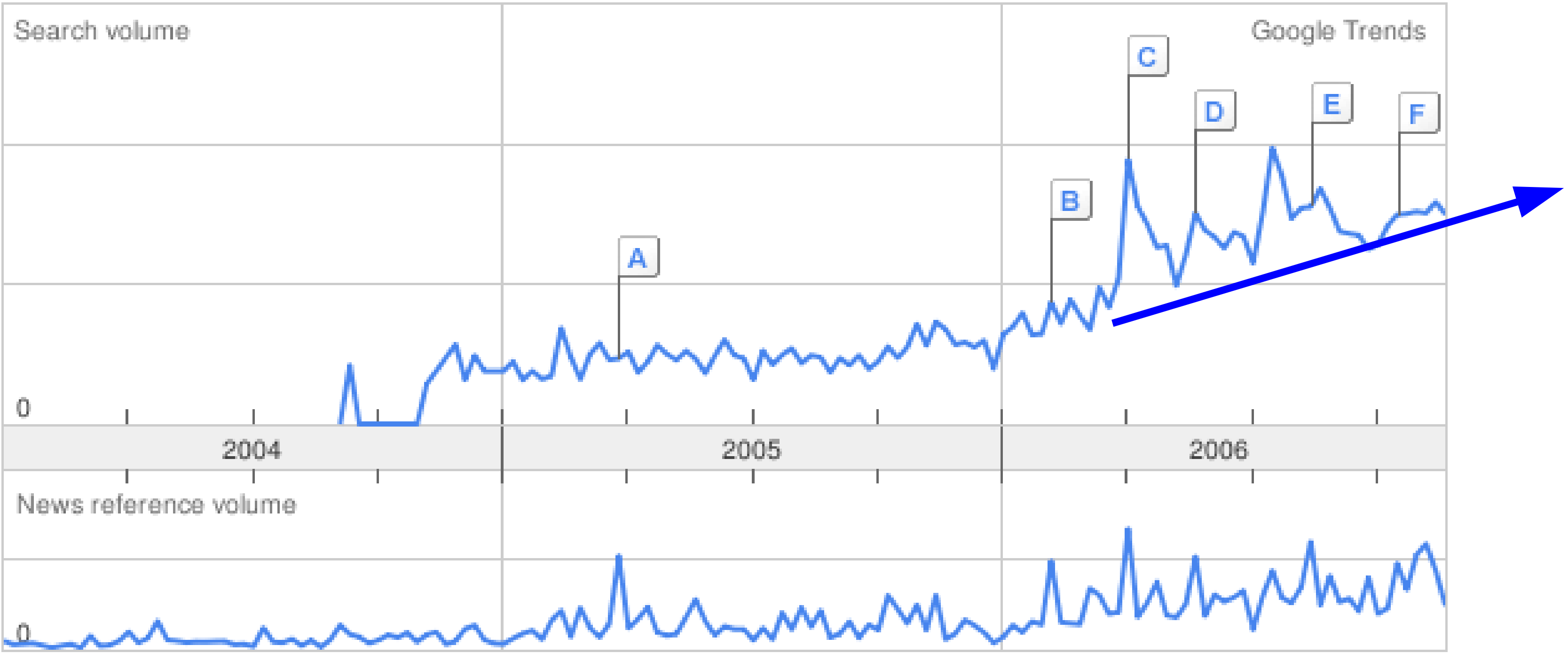
Manage Large Networks of Virtual Machines

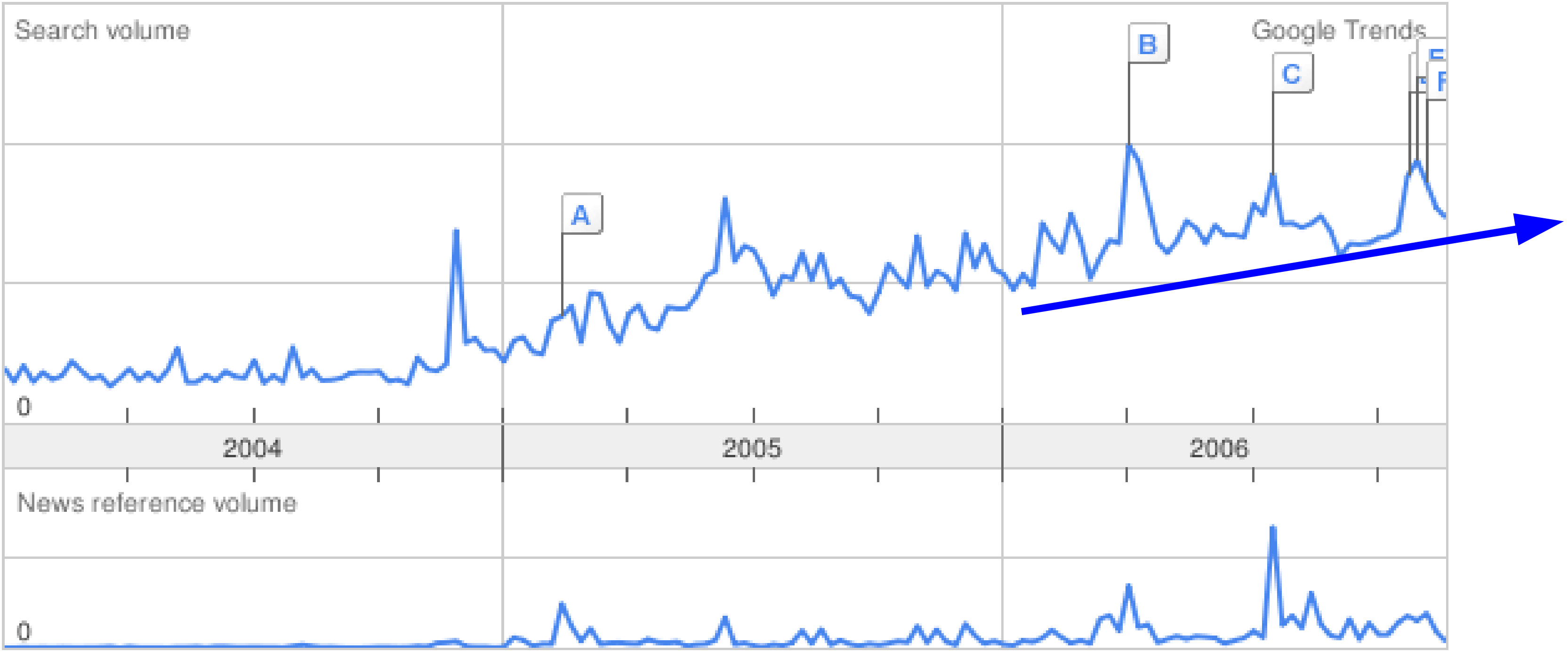
oslo university college
faculty of engineering

Kyrre Begnum - kyrre@iu.hio.no

● virtualization

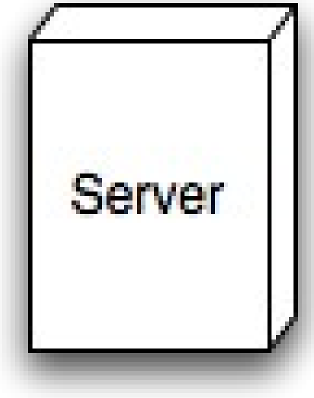


● xen

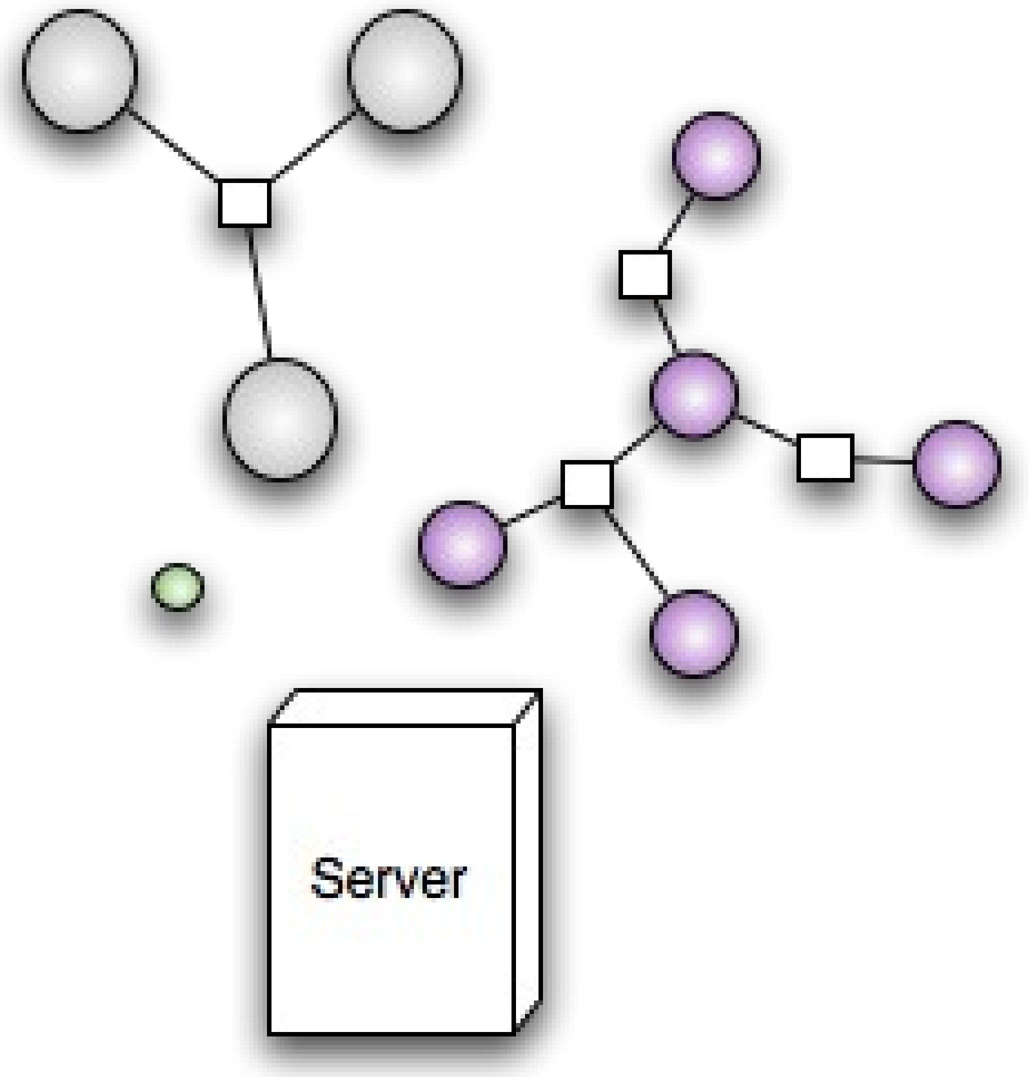


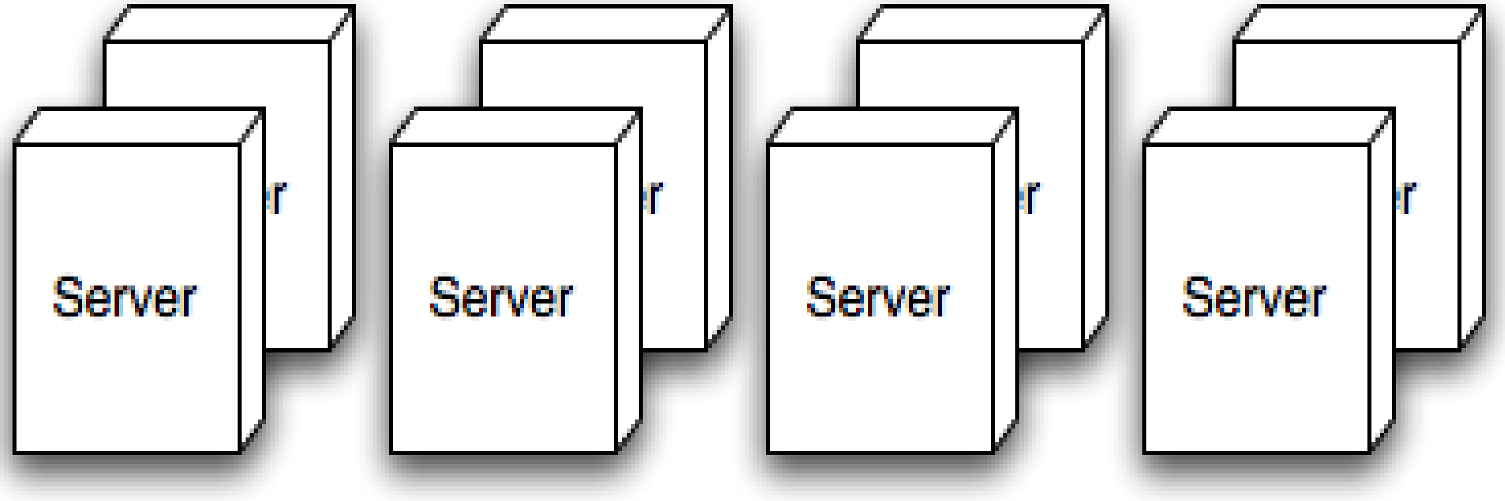
2003

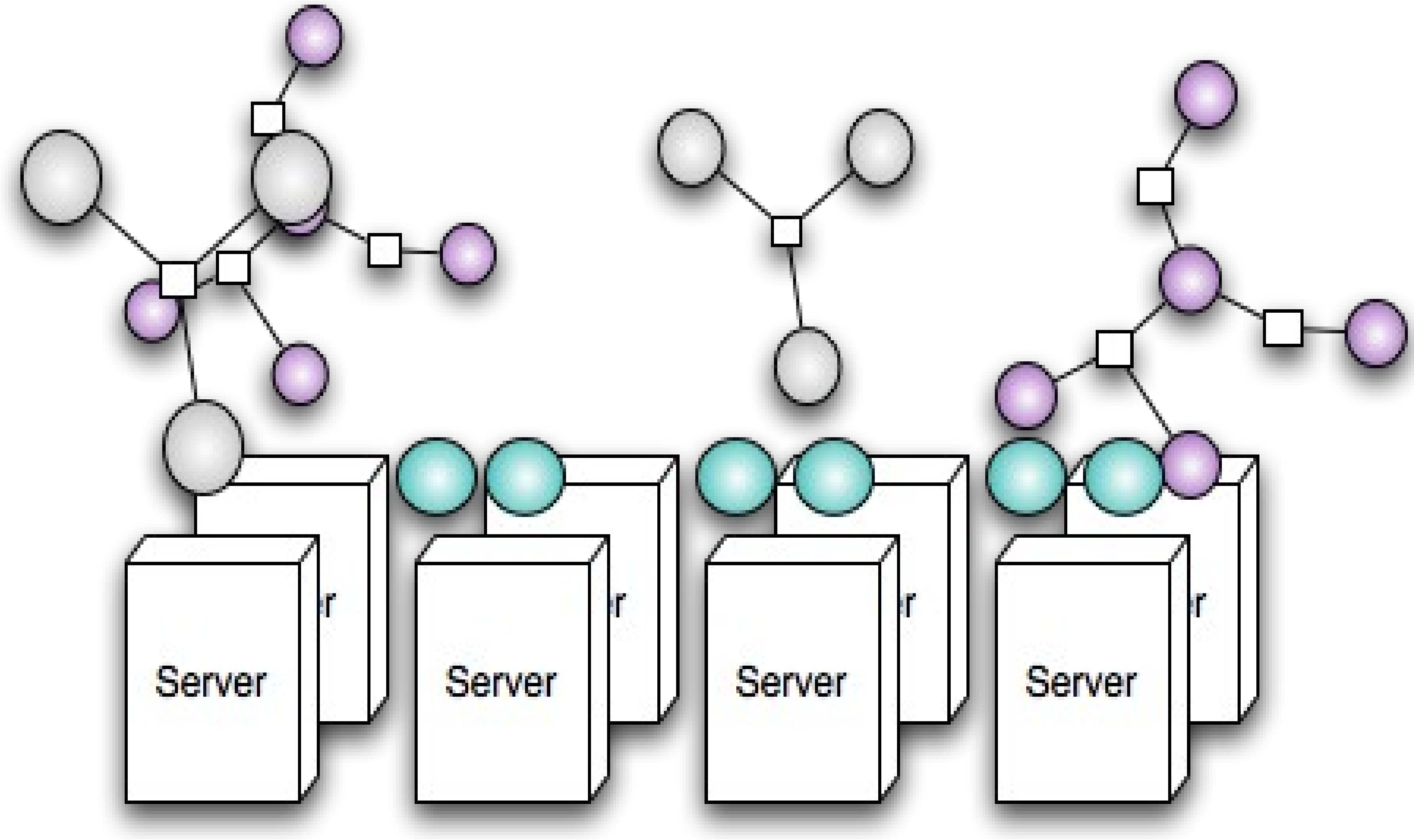
- A group formed at Lisa'03
  - Karst Koymans, University of Amsterdam
  - John Sechrest, Oregon State University
  - David Byers, Universitetet i Linköping
  - Kyrre Begnum, Oslo University College
- Interest in virtualization for services continued with John Sechrest and led to the tool MLN


Server

Server


Server


Server
Server
Server
Server


Server
Server
Server
Server

Problem: How do you cope with complexity in virtualized scenarios?

Goals

- To be able to describe the scenario efficiently
- To go from description to a working system quickly
- Manage the scenario as an atomic unit

MLN's approach

- Virtual machines are grouped into *projects*
- Projects can be distributed among several servers
- Filesystems are copied from templates
- Supported virtualization technologies are Xen and User-Mode Linux
- Expandable architecture that allows for VM specialization
- Written in perl, tested on Ubuntu Linux

How do you create projects?

- MLN projects are written to a file
- Complicated settings can be omitted
- Hosts (VMs) and switches can be connected into networks

```
global {
        project example
}

host one {
        xen
        lvm
        memory 128M
        template ubuntu-server.ext3
        size 2GB
        nameserver 10.0.0.15
        network eth0 {
                address 10.0.0.2
                netmask 255.255.255.0
                gateway 10.0.0.1
        }
        users {
                kyrre   l47/Y.NtB9p7w
        }
}
```

Superclasses

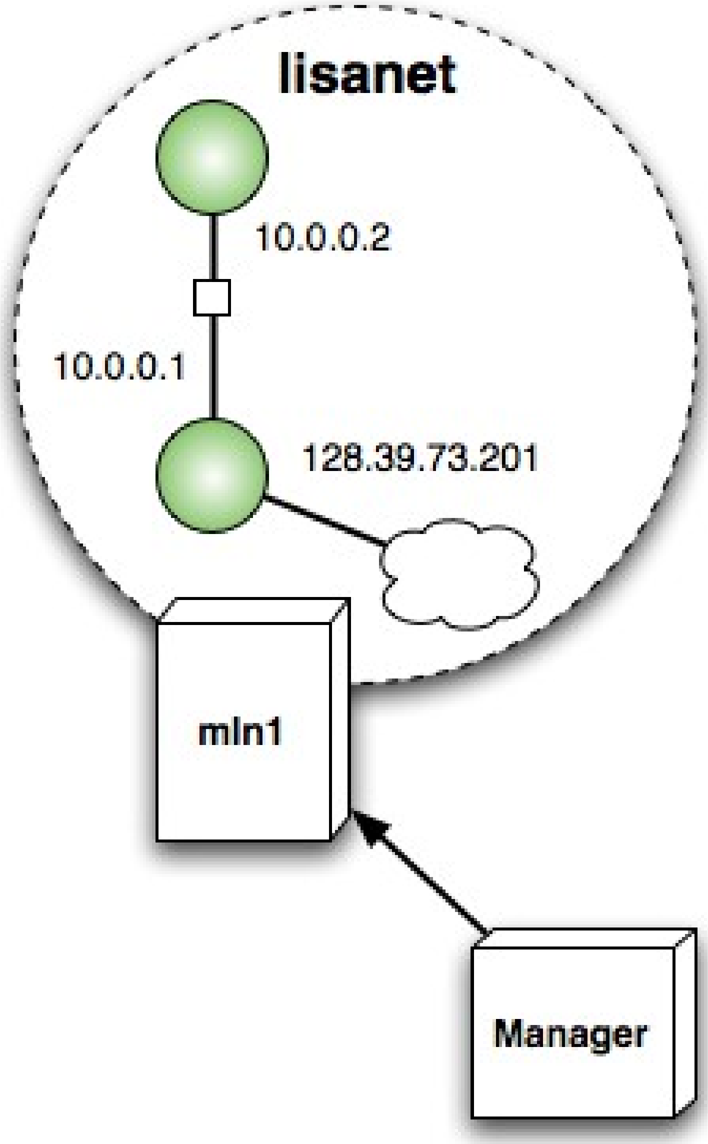
- Group common keywords into superclasses
- Hierarchies of superclasses can be constructed.
- Keywords can be overridden locally

```
global {
        project example2
}
superclass common {
        xen
        free_space 500M
        term screen
        network eth0 {
            switch lan
        }
}
host one {
        superclass common
}
host two {
        superclass common
}
host three {
        superclass common
        free_space 600M
}
switch lan {  }
```

Distributed Projects

- Hosts are assigned a service_host
- Servers run the MLN daemon
- The project remains «as one»

Demo I : Creating a network


lisanet
10.0.0.2
10.0.0.1
128.39.73.201
mln1
Manager

Things you can do to the VM

- Network interfaces and their configuration
- Disk size
- Users and groups
- Copy files into the VM
- Mount extra partitions
- Startup commands

Not enough? Perhaps you want to write your own ...

Plug-ins

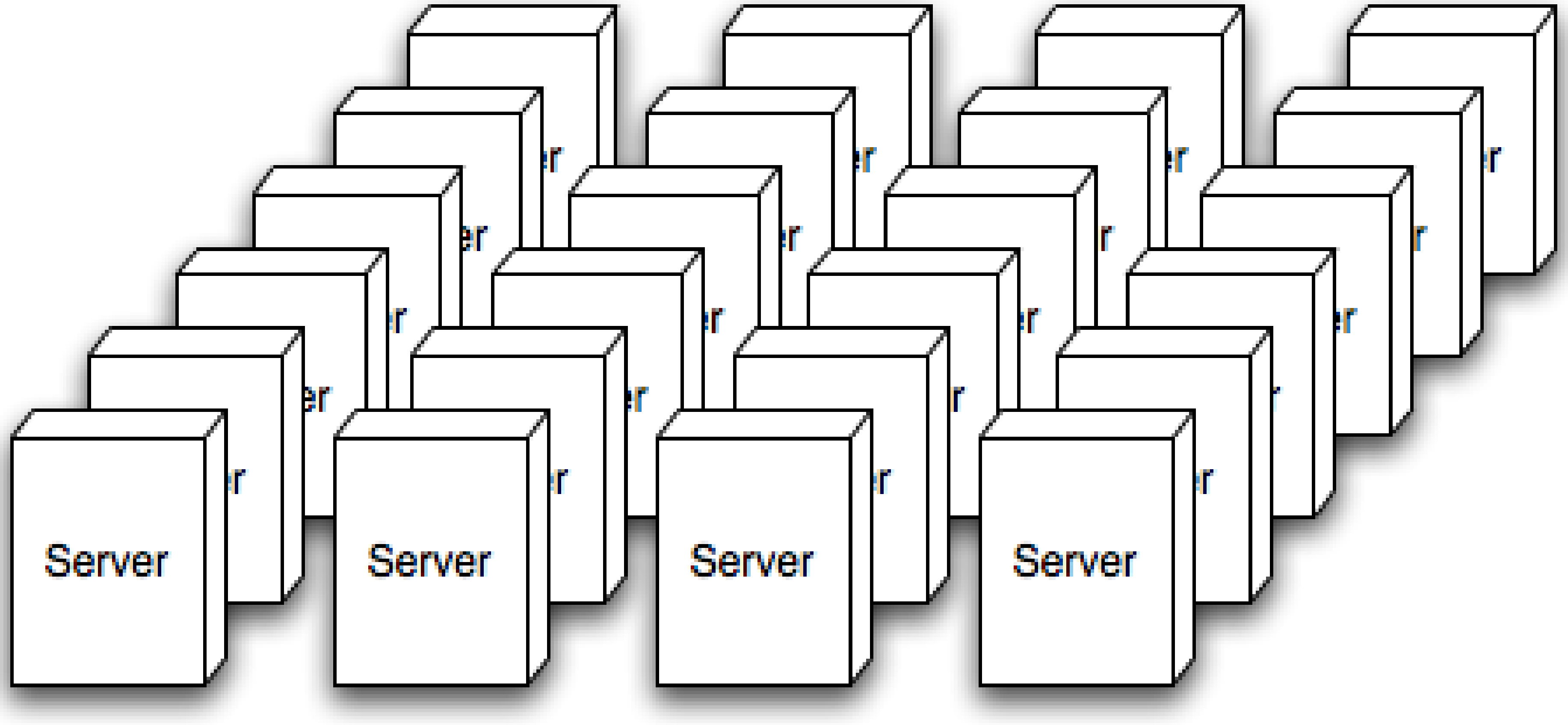
- Can seamlessly extend the MLN syntax
- Utilize variables and superclasses
- Plug-ins may affect a VM directly or the MLN data structure
- Plug-ins are only available using perl at present

```
global {
        project example
}
superclass common {
        apache {
                max_connections 30
        }
}
host one {
        superclass common
        apache {
                doc_root /var/www
        }
}
```

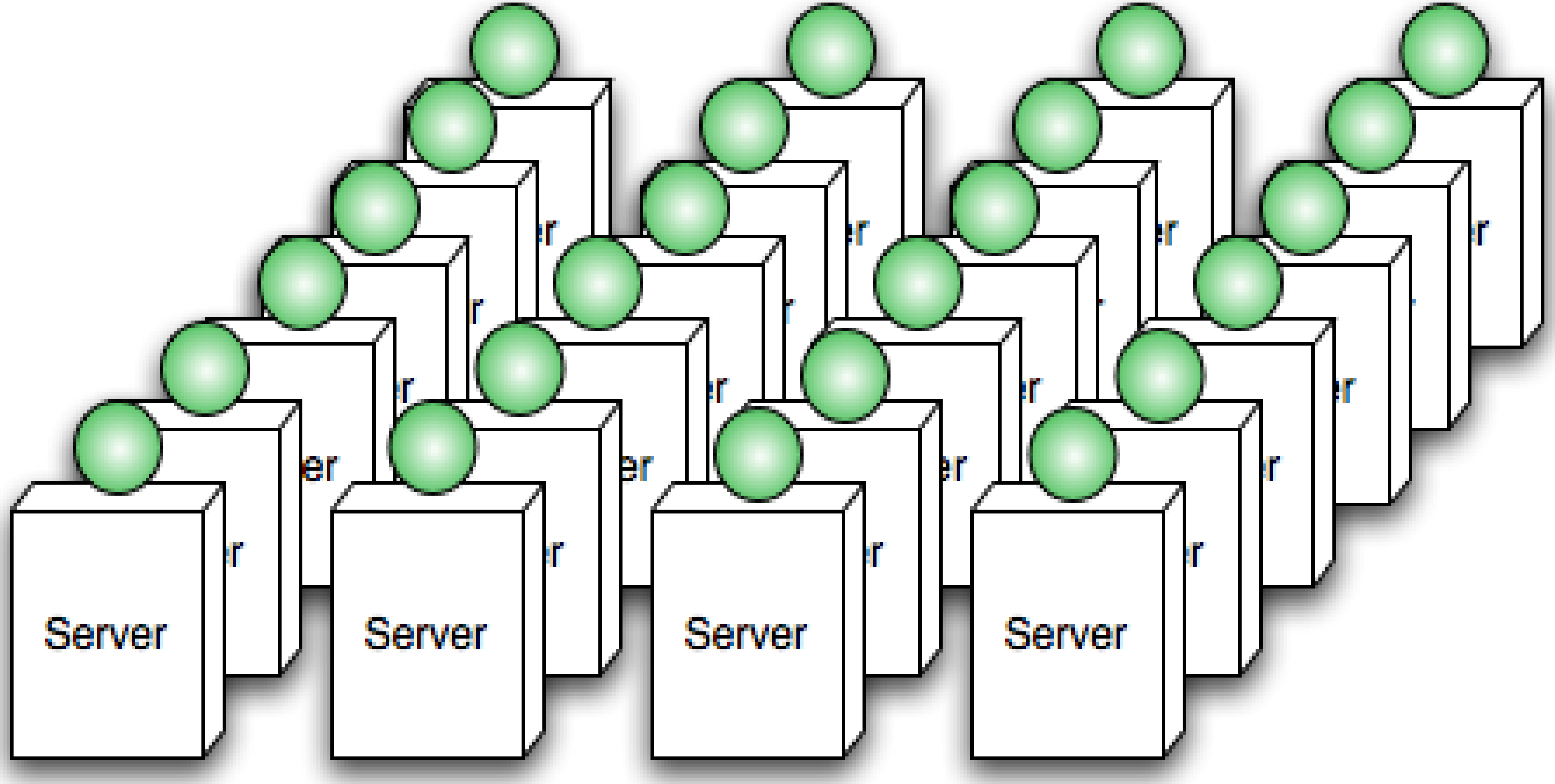
Autoenum – A plug-in for very large projects

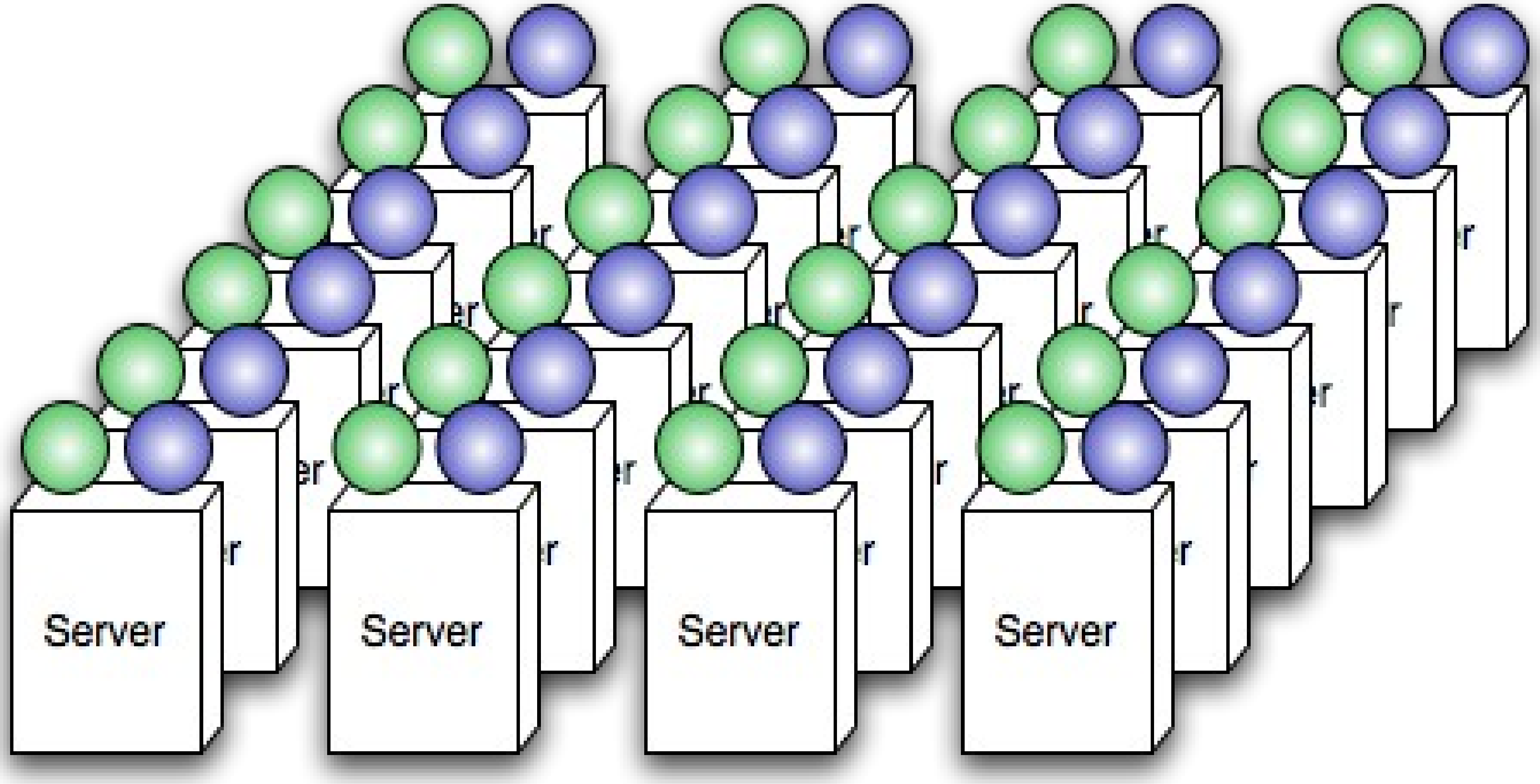
```
global {
    project mycluster
    autoenum {
        superclass cluster_node
        numhosts 36
        address auto
        addresses_begin 150
        net 128.39.73.0
        service_hosts {
            #include /root/servers.txt
        }
    }
    $gateway_ip = 128.39.73.1
    cluster {
            head node1
        }
    }
}
```

```
superclass cluster_node {
    template ubuntu_mpi_tourque.ext3
    memory 312M
    free_space 1G
    network eth0 {
        gateway $gateway_ip
    }
}
```



Server
Server
Server
Server

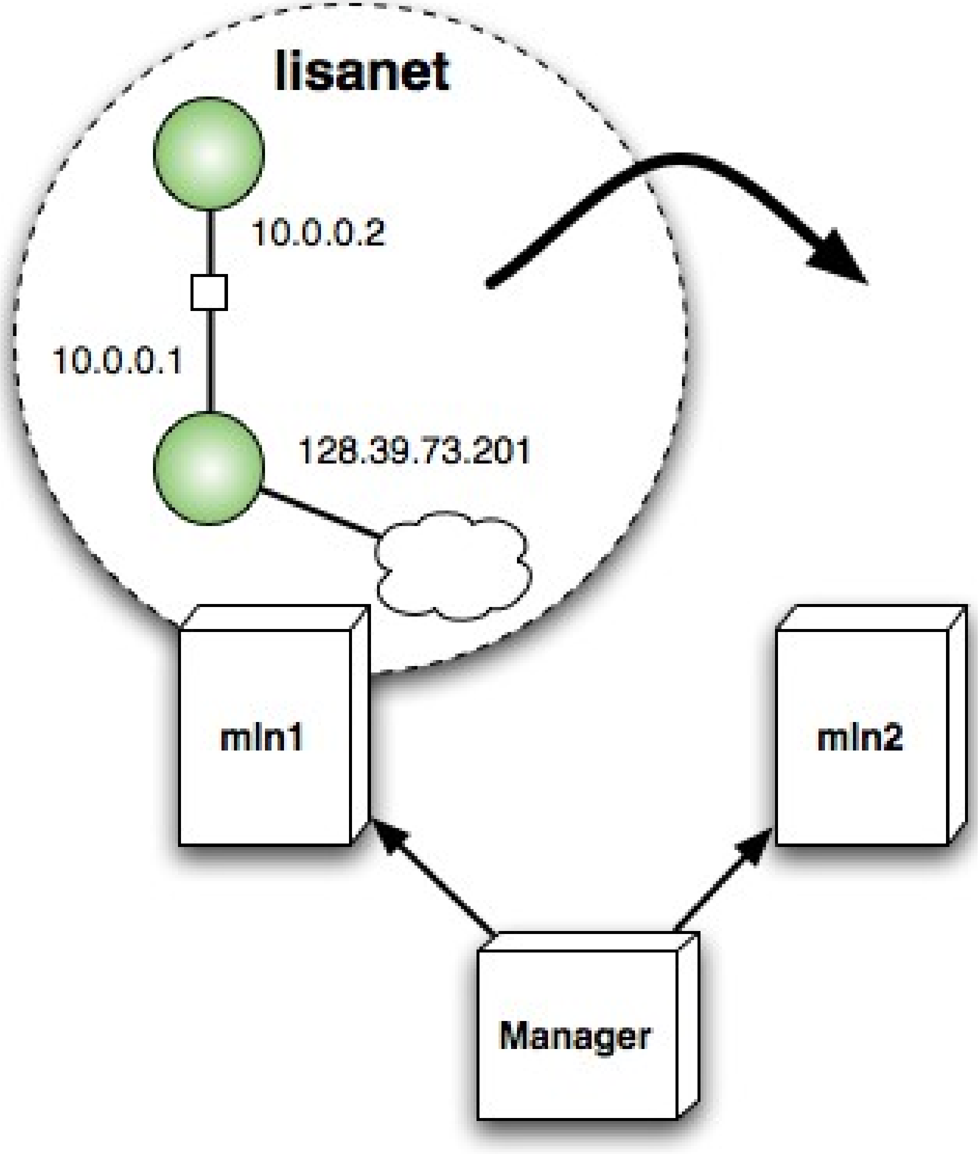

Server
Server
Server
Server

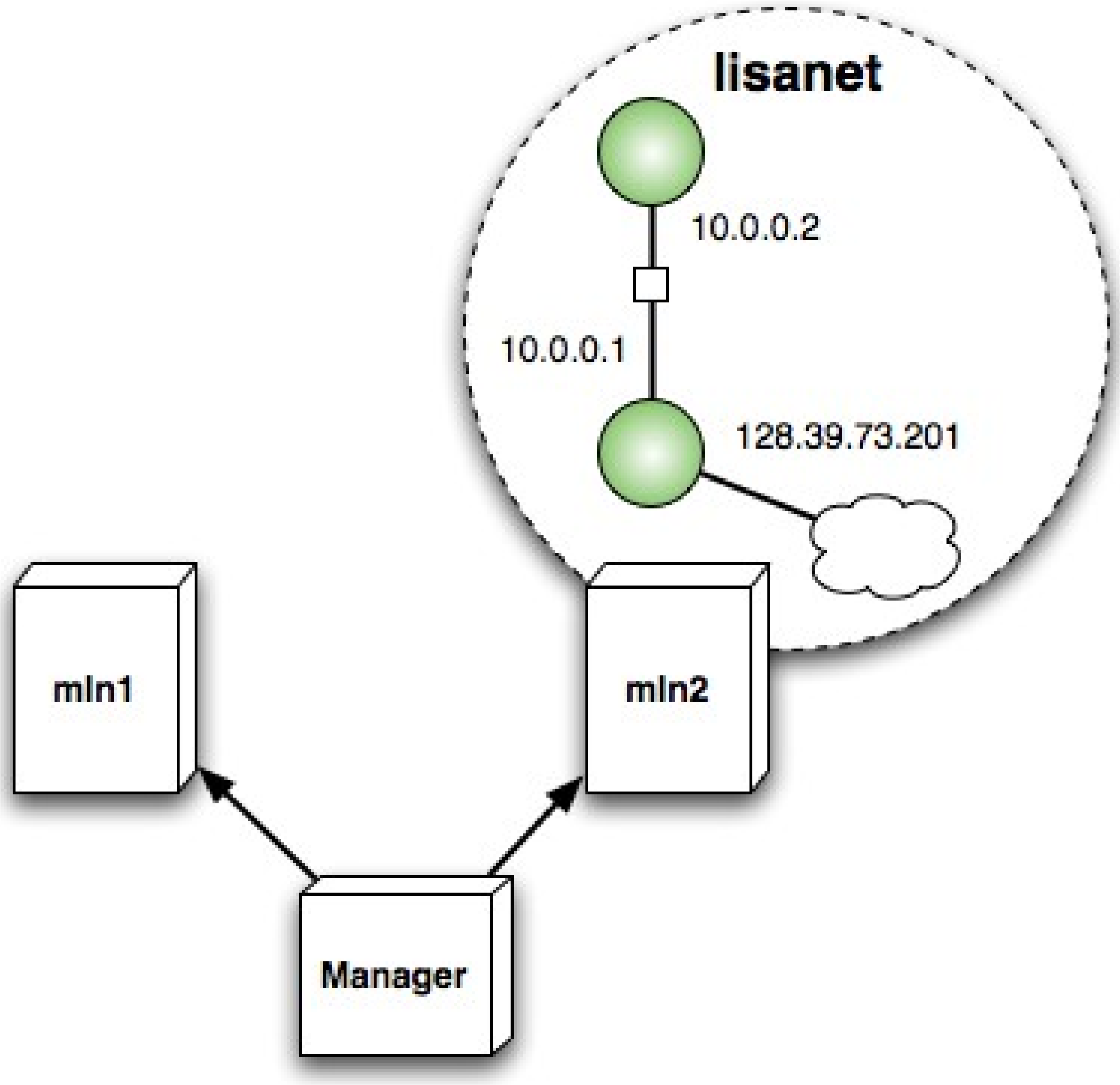

Server
Server
Server
Server

Maintenance

- Small adjustments to a long-running project are likely
- MLN supports an upgrade command that reads a new version of the project file
- VM properties such as memory, size and VM technology can be changed
- Changing the service_host for a VM will result in a migration

Demo II: Moving a project


lisanet
10.0.0.2
10.0.0.1
128.39.73.201
mln1
mln2
Manager


lisanet
10.0.0.2
10.0.0.1
128.39.73.201
mln1
mln2
Manager

Steps of operation

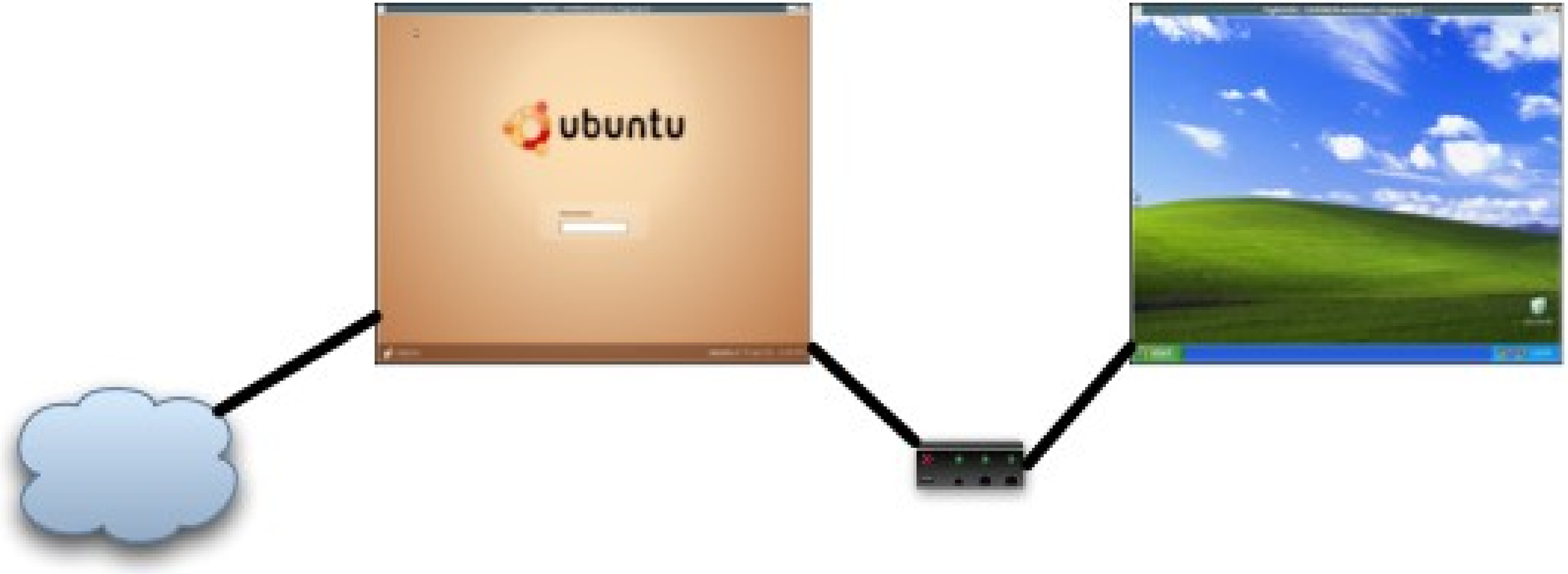
Investment for each type of VM:

1. Create the filesystem template with the desired software
2. Write an MLN plugin for automated software configuration

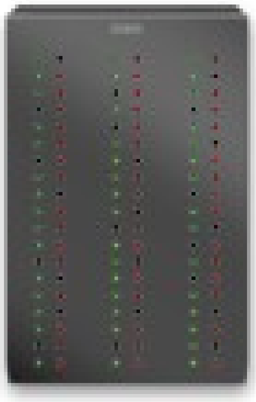
For each project instance:

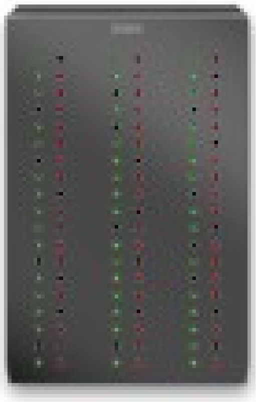
1. Write an MLN project file
2. Build the project: `mln build -f mycluster.mln`
3. Start the project: `mln start -p mycluster`

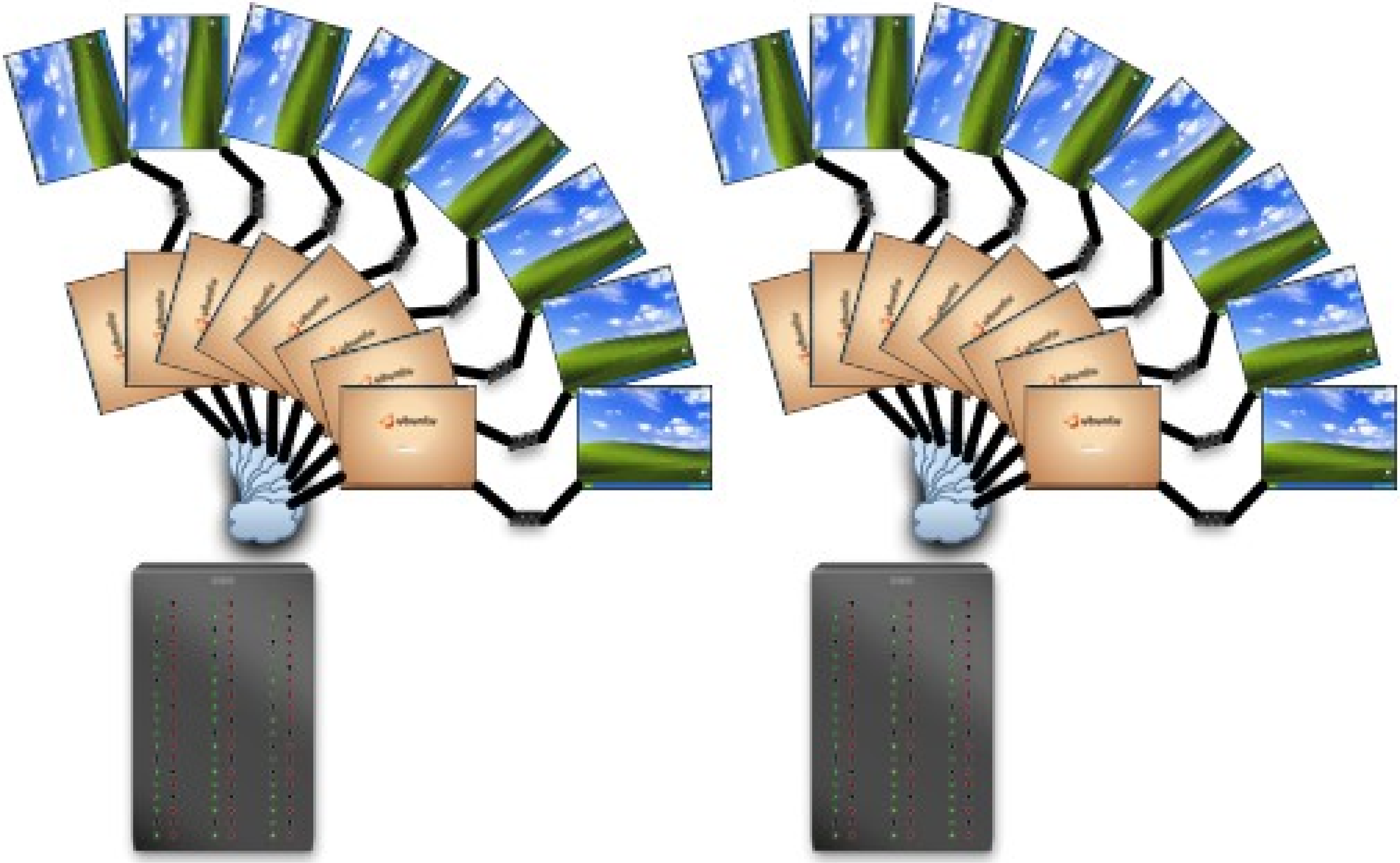
Case: Introductory OS course



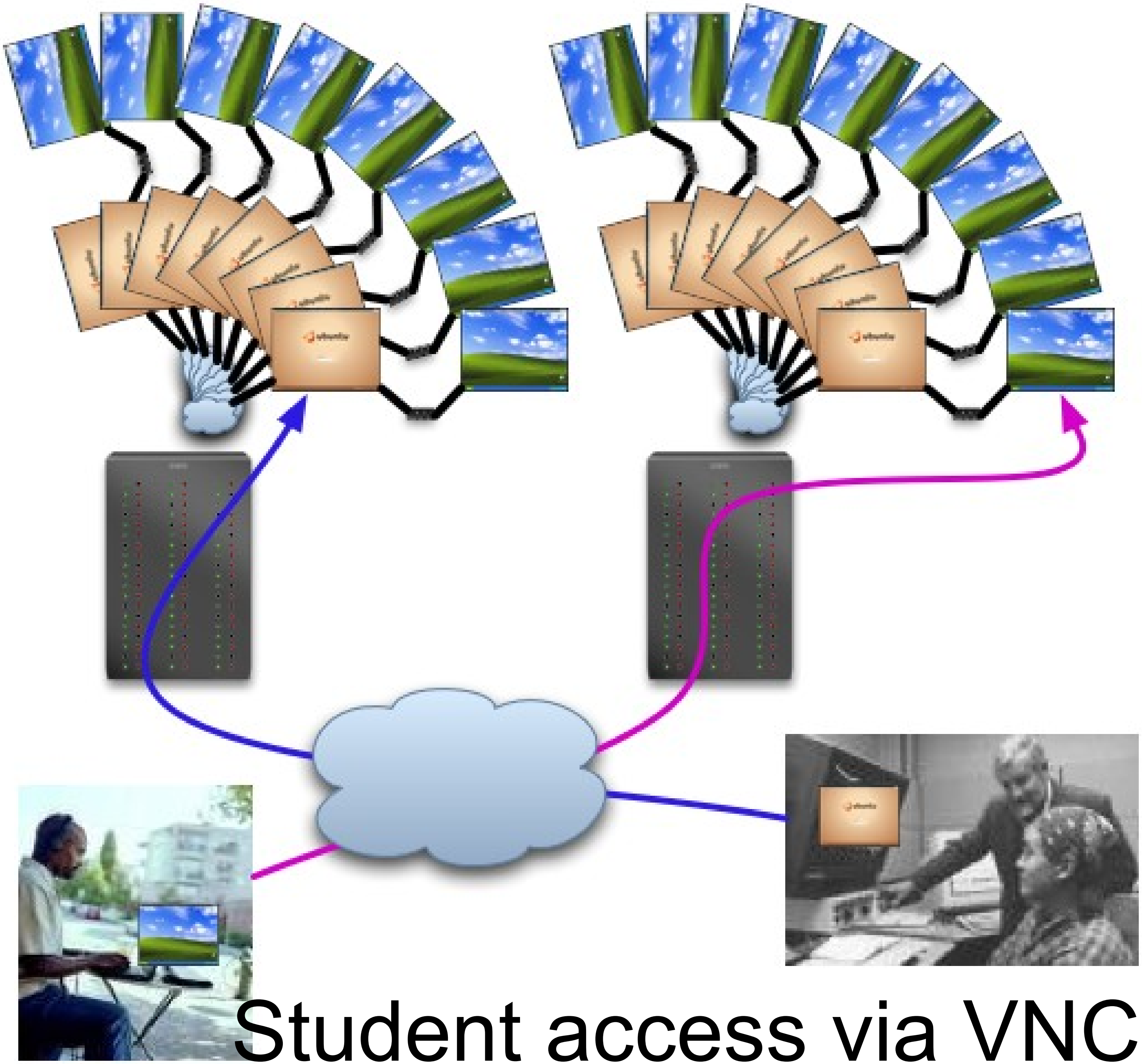
We acquired a few AMD AM2 machines





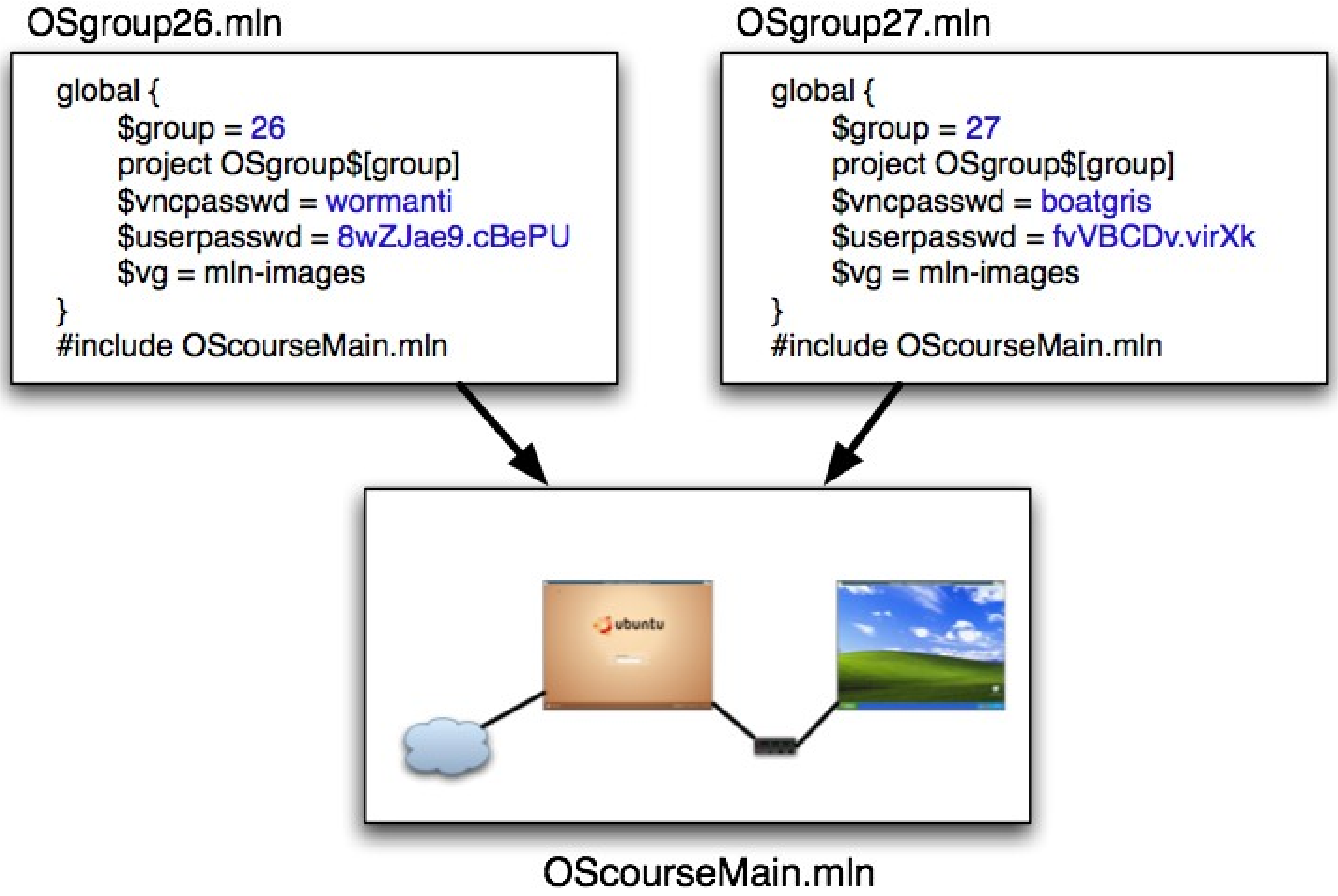


... and voila! :)



Student access via VNC

Organization



OSgroup26.mln

```
global {
      $group = 26
      project OSgroup$[group]
      $vncpasswd = wormanti
      $userpasswd = 8wZJae9.cBePU
      $vg = mln-images
}
#include OScourseMain.mln
```

OSgroup27.mln

```
global {
      $group = 27
      project OSgroup$[group]
      $vncpasswd = boatgris
      $userpasswd = fvVBCDv.virXk
      $vg = mln-images
}
#include OScourseMain.mln
```

OScourseMain.mln

Goals

- To be able to describe the scenario efficiently
  - superclasses
  - plug-ins
- To go from description to a working system quickly
  - templates
  - distributed building
- Manage the scenario as an atomic unit
  - projects

But virtualization in production brings more challenges:

- Design – How can you express the properties of your infrastructure?
- Cost – How expensive infrastructure do you need and with what features?
- Availability – How do you maintain the physical machines and re-provision the VMs?
- Monitoring – What data do you need in order to make sound provisioning decisions?

Future goals for MLN

- Policy-aware analysis
- VM performance monitoring
- Other VM technologies – KVM and VMware?
- Live migration for Xen

Thank You :-)

http://mln.sourceforge.net