Trust VM Creation Recipes, not Images

A Scheme to Create and Distribute Trusted VM to Grid Sites

Yushu Yao
Lawrence Berkeley National Laboratory

Overview

- A Scheme to define and distribute trusted VMs to Sites.
- Some random bullets on VM/Cloud/Security and ATLAS
- Many thanks to the workshop organizers, especially Michel for accommodating the talk.

The Fight - VOs vs. Sites

✦ **VO says to Site:** I must customize the VM image that you run. I need to configure it to fit my needs, I need to install my application stack. So I can run my software.

✦ **Site says to VO:** I don't TRUST you!!! I will not run an image you provide, because no matter what you do, according to my 100-page long security policies, it is insecure.

The Problem

✦ To fully leverage the benefits of Virtualization, VOs need to be able to decide what they run in the VM. However, sites have security policies to follow.

✦ There has to be a way to let Sites **trust** the VMs that VO provide.

The Solution - Recipes

We propose to work with the **Recipe** to Create VM images, instead of the **images** themselves. In this way:

- ✦ **VOs** will have the freedom to customize the VM they want to run.
- ✦ **Sites** can run VO-provided VM without sacrificing security (E.g. not giving out root access on any running instance), and with least added effort

Two Sets of Recipes

✦ Definition: A Recipe is a set of scripts that can configure the target system. E.g. install software, run services, start firewalls, etc.

✦ VO Recipe: VO provided, contains all VM specific functionalities

✦ Site Recipe: Site provided, contains Site-specific configurations and security measures. E.g. mount GPFS, hookup batch scheduler, etc.

In a Nutshell

- Starting from the same well trusted base image
- VO defines VO recipes to achieve desired functionality
- Site defines Site recipes
- At deploy time, Site boots the base image, apply VO recipes, then Site recipes. Now the running instance has VO defined functionalities while at the same time, complies with Site security policies
- Site does not give out root privilege on running instances

Images vs. Recipes

BERKELEY LAB

We need the following features:

- ✦ Easy to Store and Transfer
- ✦ Easy to exam for defects and vulnerabilities
- ✦ Versioning Control for Project Lifecycle Management

	Image	**Recipe**
Size	GB	KB
# of Files	>50k	1 to several
Exam for vulnerabilities	Extremely Hard	Much Easier
Transfer	Expensive	Cheap
Versioning Control	Nightmare	Simple

How to Define a Recipe?

Requirements for the Recipe Tool:

- VOs/Sites are not always willing to learn new technologies: **Easy to Use**
- It must define complex software security configuration details: **Powerful and Reliable**
- Recipes are not applied until deploy time: **Fast Executing**
- Many VMs might be deployed at the same time: **Scalable**

Puppet - The Recipe Language

BERKELEY LAB

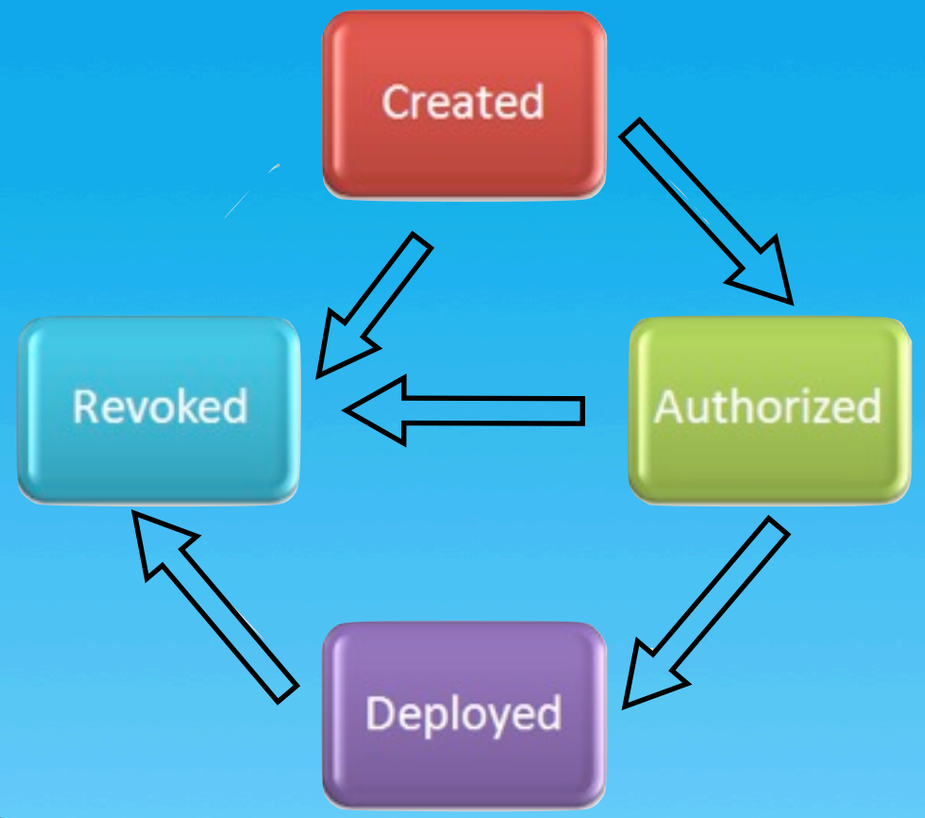
- Puppet is a **Configuration Management System**
- It uses a language describe how you want to set up a Linux system.
- E.g. below is how ssh service is described

```
class ssh {
    package { ssh: ensure => installed }
    file { sshd_config:
        name => "/etc/ssh/sshd_config",
        owner => root,
        group => root,
        source => "puppet://server/apps/ssh/sshd_config",
        after => Package[ssh]
    }
    service { sshd:
        ensure => running,
        subscribe => [Package[ssh], File[sshd_config]]
    }
}
```

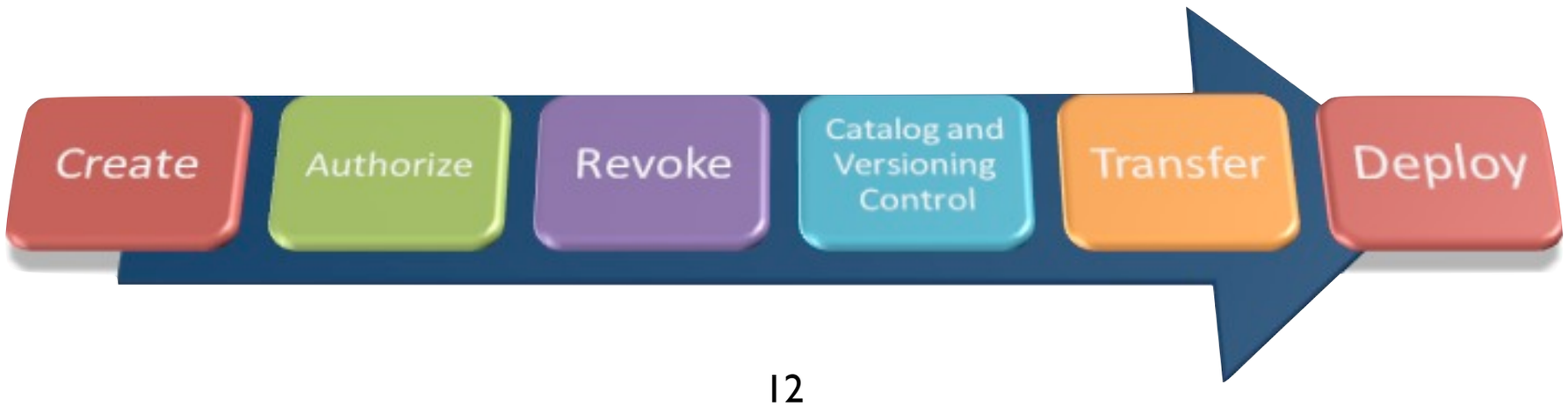
Why Puppet

- Easy To Use
- Module-Based: reusable
- Large User Base
- Already Adopted by Many Sites for admin automation
- Least work to implement auditing

BERKELEY LAB



The Scheme



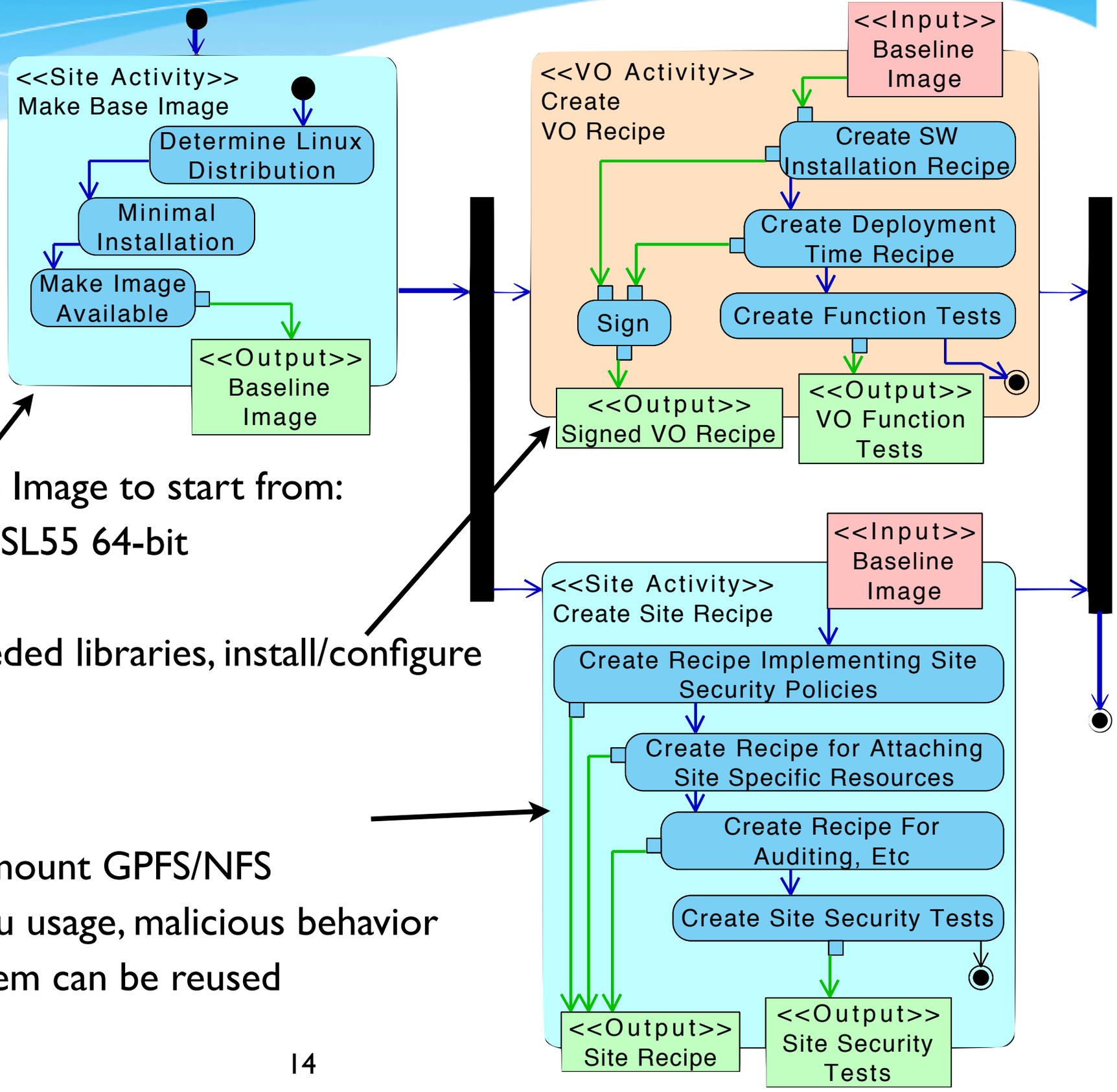
Who does What?

- VO:
  - provides VO recipes
  - provides VO function tests
- Site:
  - specifies baseline image
  - define Site recipes
  - authorize VO recipes
    - scan VO recipes for defects
    - generate behavior patterns
  - deploy VM
  - monitor running VM

Creation

Object Flow

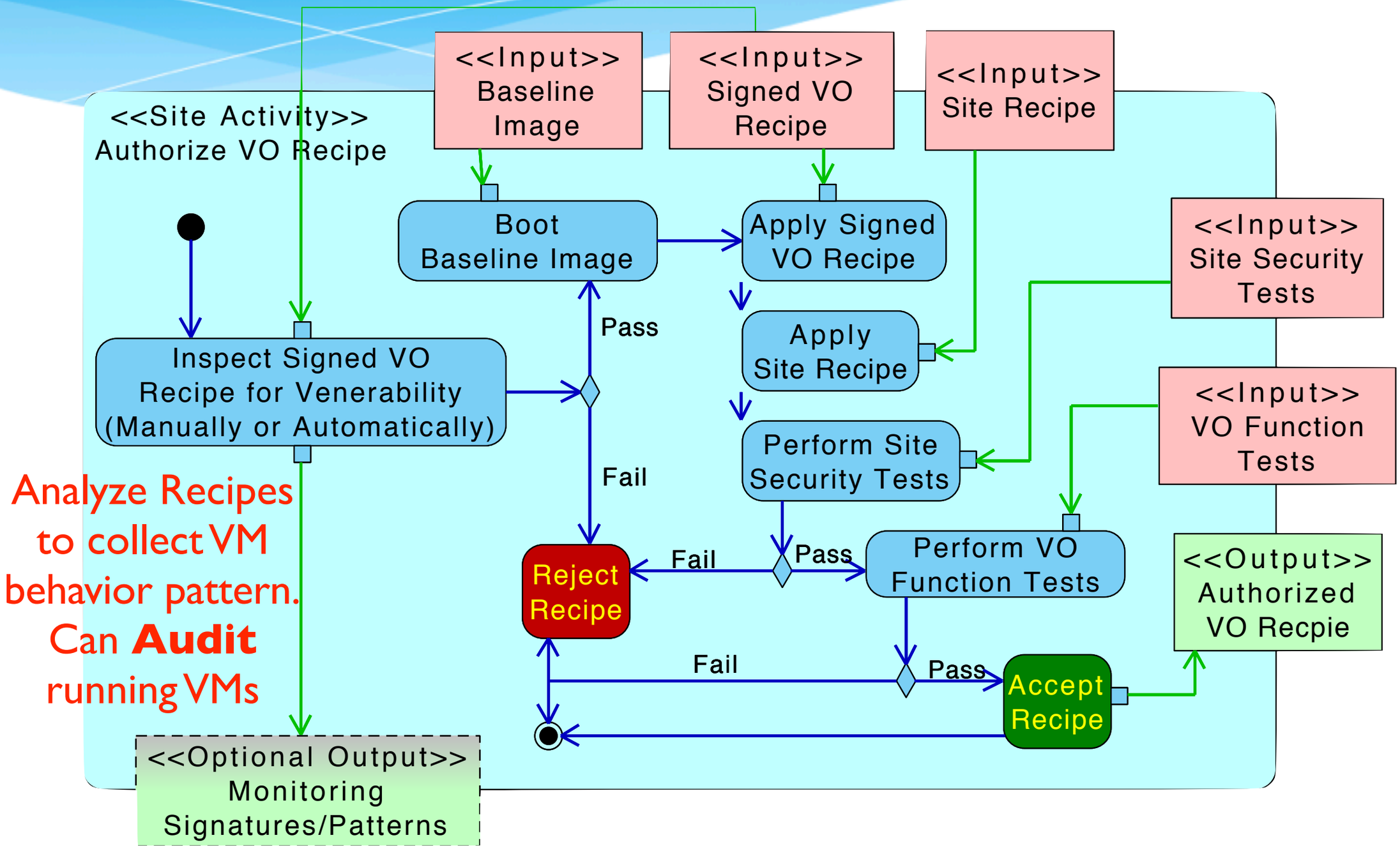
Control Flow



1. Site determines which Base Image to start from:
   - E.g. a basic installation of SL55 64-bit
2. VO creates:
   - VO Recipe: e.g. Install needed libraries, install/configure CVMFS
   - Function Tests: e.g. .
3. Site creates:
   - Site Recipe: e.g. firewall, mount GPFS/NFS
   - Security tests: e.g. idle cpu usage, malicious behavior

Existing Grid Certificate System can be reused

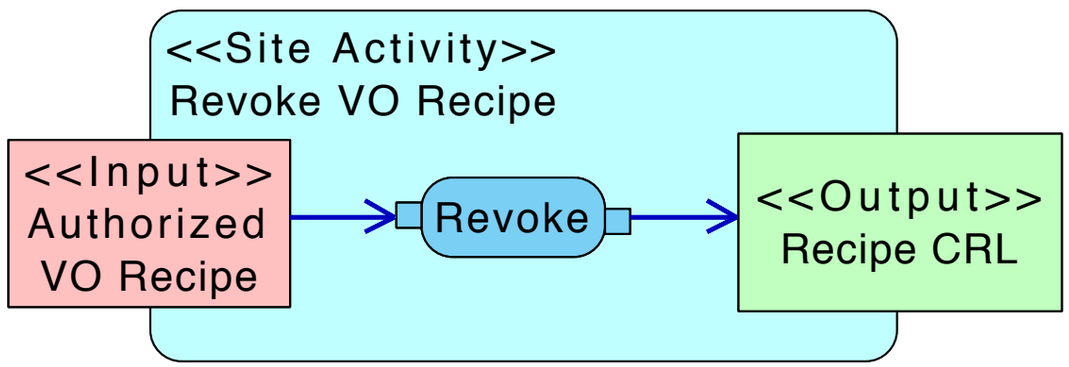
Authorization



Only Site can Authorize a **Signed VO Recipe** and produces **Authorized VO Recipe**.

Basically, the Site: 1. boot up the base image, 2. apply both VO and Site recipes, 3. run site security tests, 4. run VO function tests. If all success then accept it.

Revoke



Here we use a CRL-like mechanism.

Revoked Recipes are pushed into Recipe CRL, which is checked at deployment time.

Catalog/Version Control

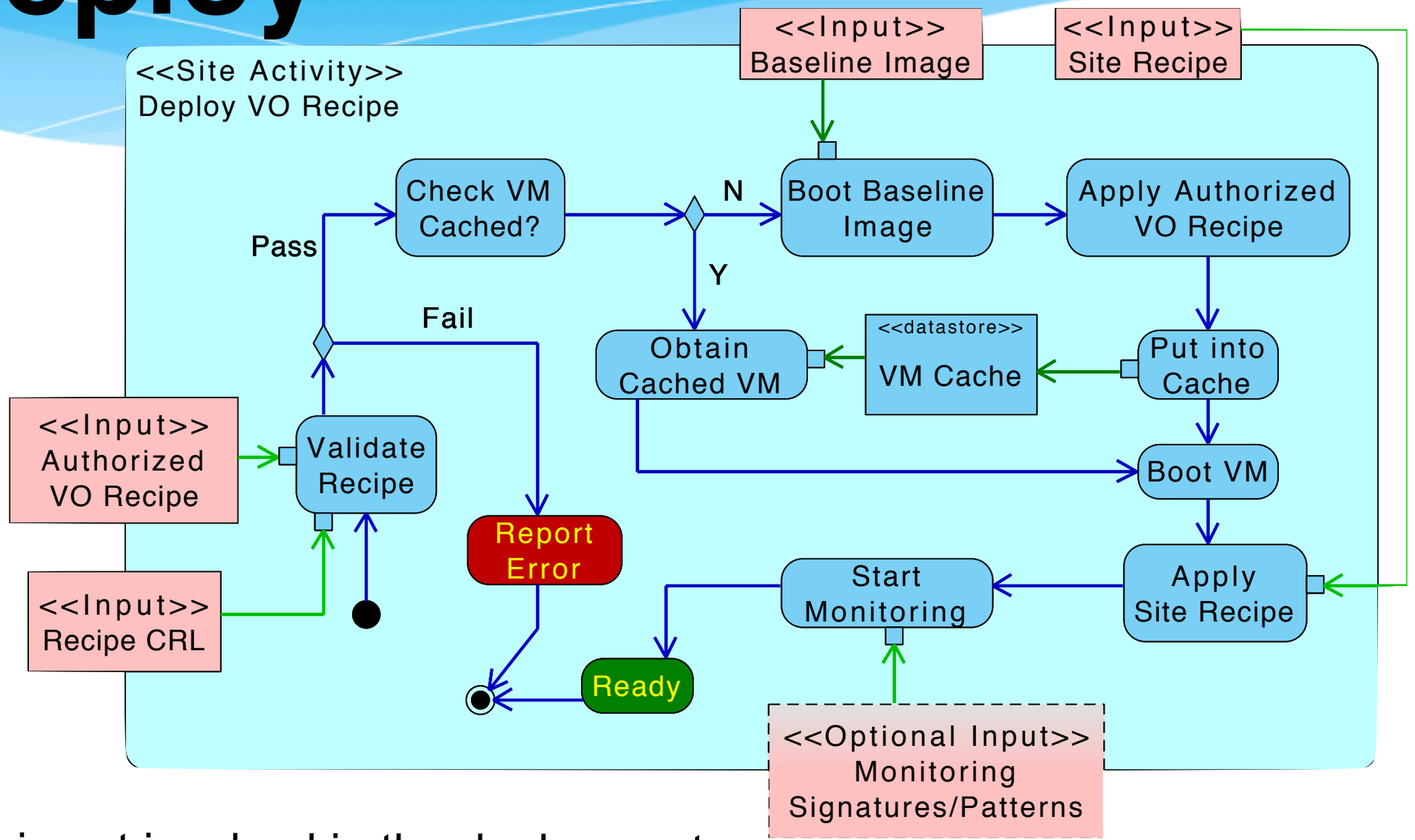
Can do similar things like images, only much faster doing every step

Transfer

No need to transfer any bulky image from VO to Sites.

All recipes are in SVN repositories. Site will pull in the recipes at deployment/authorization time.

Deploy



VO is not involved in the deployment process.

1. Check against CRL; 2. Start base image, apply VO recipes then Site recipes. (Site recipes are applied later to close any “backdoors” VO recipe left open); 3. Since applying VO recipe might take some time, the result of this step can be cached for faster deployment in the future; 4. Optionally, the site can start monitoring the running instance for security or performance oddities.

Security/Auditing

- Manually or Automatically Scan recipes for security issues.
- Site recipes are applied later than VO recipe at deployment time
- Generate expected behavior patterns from recipes. (Still need some engineering but much easier to do than images)
- Monitor running instances and report malicious behavior according to the generated pattern. Or freeze the instance for further inspection.

3-levels of security

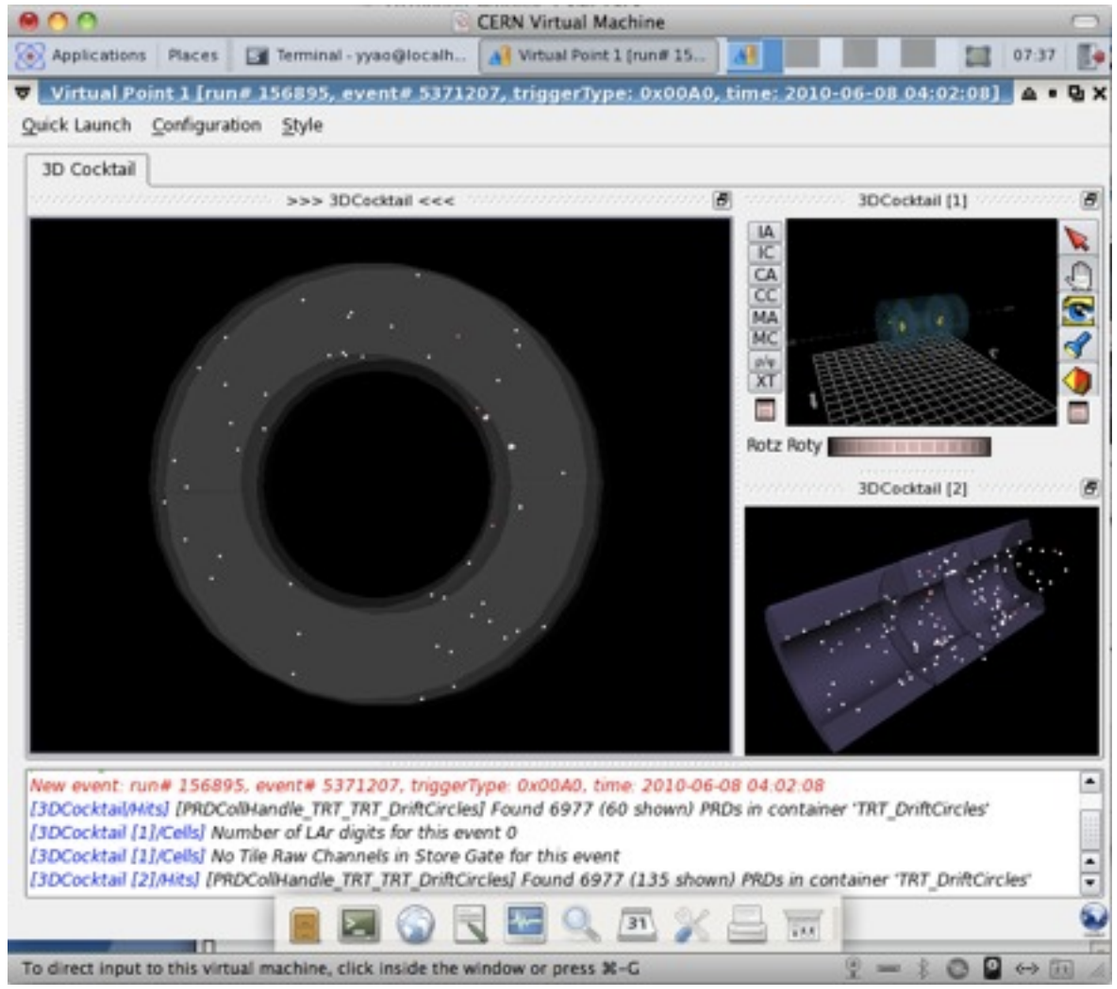
Sites can provide the following 3 trust levels:

- VO provided Image, not examined: treat it like visiter's laptop.
- Recipe Based: have some level of access to local resource, enforce auditing
- Choose from Site's Menu: E.g. existing lxbatch, full access

CernVM

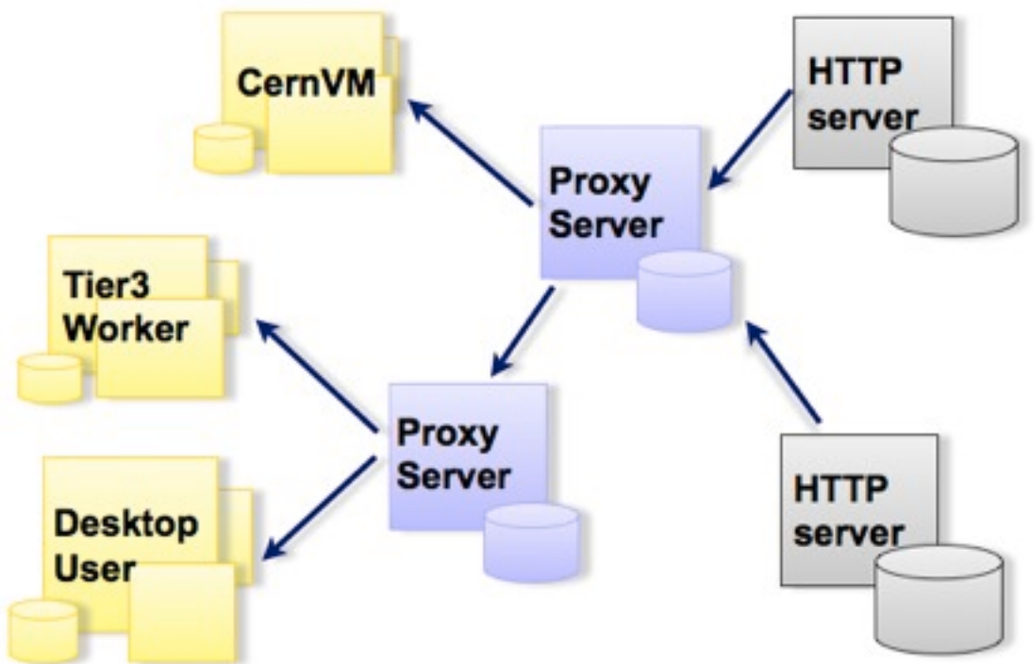
- The simplest yet fully functional Tier3-workstation.
- Great for Code Development/Job Submission
- 1-click VP1 Live
- Tutorial: https://twiki.cern.ch/twiki/bin/view/Atlas/CernVMTutorialHead

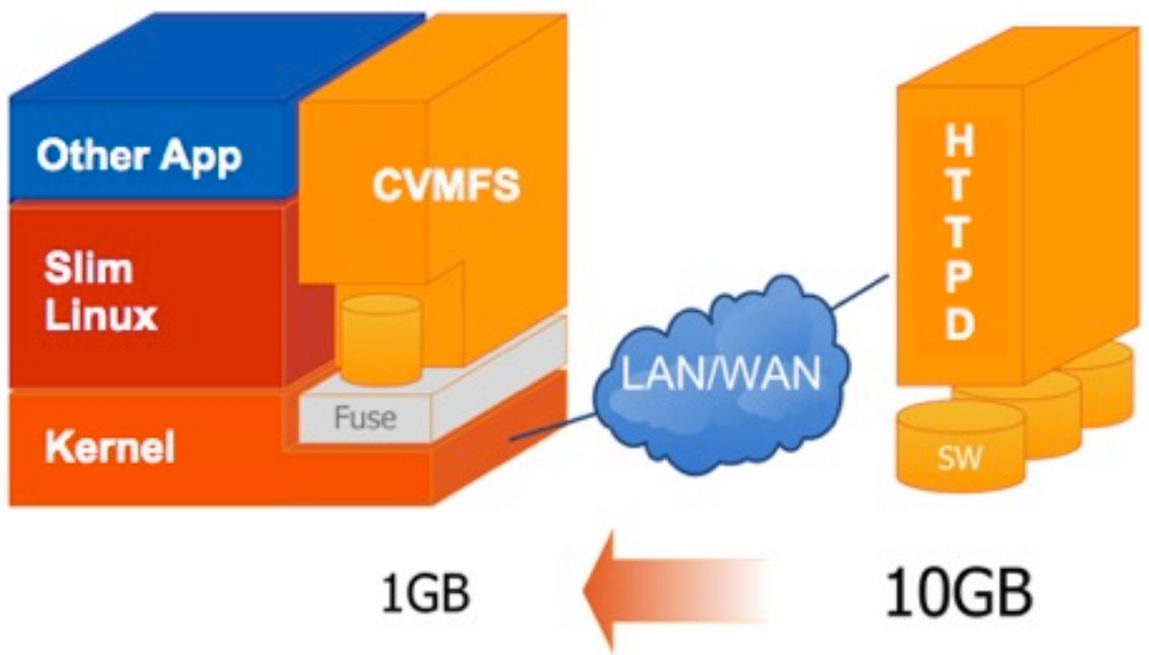
~2000 different IP addresses



CernVM-FS

- CernVM-FS - Officially Supported by ATLAS
- As soon as a new ATLAS release come out, it is on CernVM FS.





VM Performance

- VM is not a friend of HPC. Performance hit expected. Experimenting with the VMs and looking in to the future is good. All VM related work is a wasted of time if the performance penalty is not acceptable.

- Unless one can provision raw hardware like provisioning VM. In this case most of our R&D in VM (creation/authorization/distribution) is still useful.

Attention!!!

ATLAS on the Cloud

- We are building an auto-scaling ATLAS cluster on the Magellan
- Details Wednesday Afternoon's Talk

BERKELEY LAB

Questions?