UWS in action: The CANFAR VM lifecycle web service

Experiences in using the Universal Worker Service as an interface to VM clouds

Brian Major

Canadian Astronomy Data Centre

CANFAR

Canadian Advanced Network for Astronomical Research

“CANFAR aims to provide to its users easy access to very large resources for both storage and processing, using a cloud based framework. CANFAR allows astronomers to run processing jobs on a set of computing clusters, and to store data at a set of data centres.”

More on CANFAR…

ADASS, Wednesday Oct 2, 14:30
Sebastien Fabbro

Services for data intensive astronomy. Science use and experience from CANFAR

UWS, Briefly

- UWS = Universal Worker Service Pattern
- IVOA standard version 1.0
- Framework allowing for the asynchronous execution of jobs.
- For CADC, it's a java jar file that a web service imports (available at OpenCADC
  http://code.google.com/p/opencadc/)

-> Create jobs with parameters

-> Run the jobs

-> Poll to see job results

Jobs in UWS

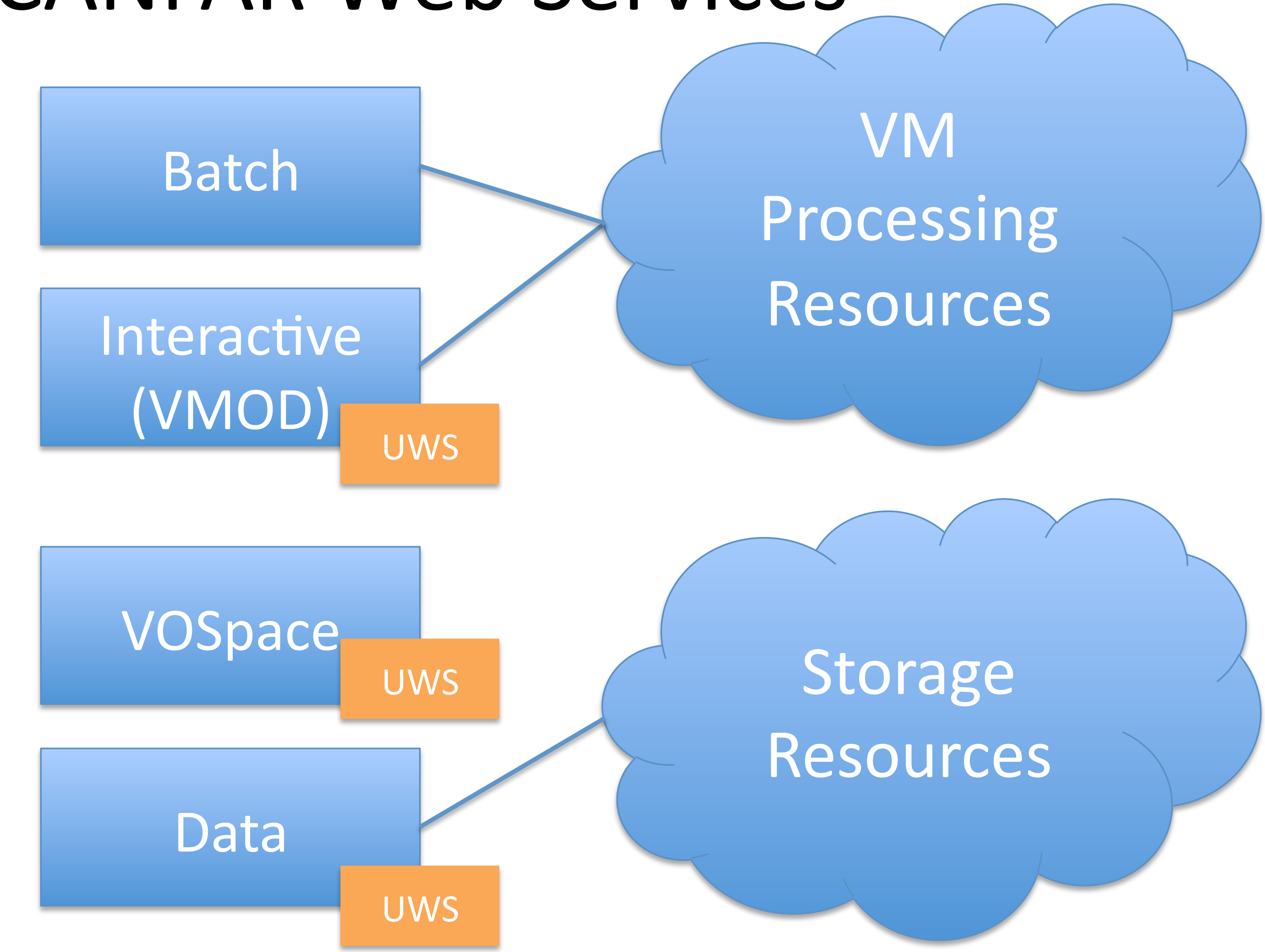
Typically, a job is a piece of software designed for a specific purpose. For example, execute a TAP query.

UWS will run your job and provide mechanisms for:

- Setting job execution phase (it's state)
- Storing the results of the job

In the case of the VMOD web service, a job maps to the lifecycle of a virtual machine…

CANFAR Web Services



VMOD: VMs On-Demand

- VMOD is a virtual machine lifecycle management web service

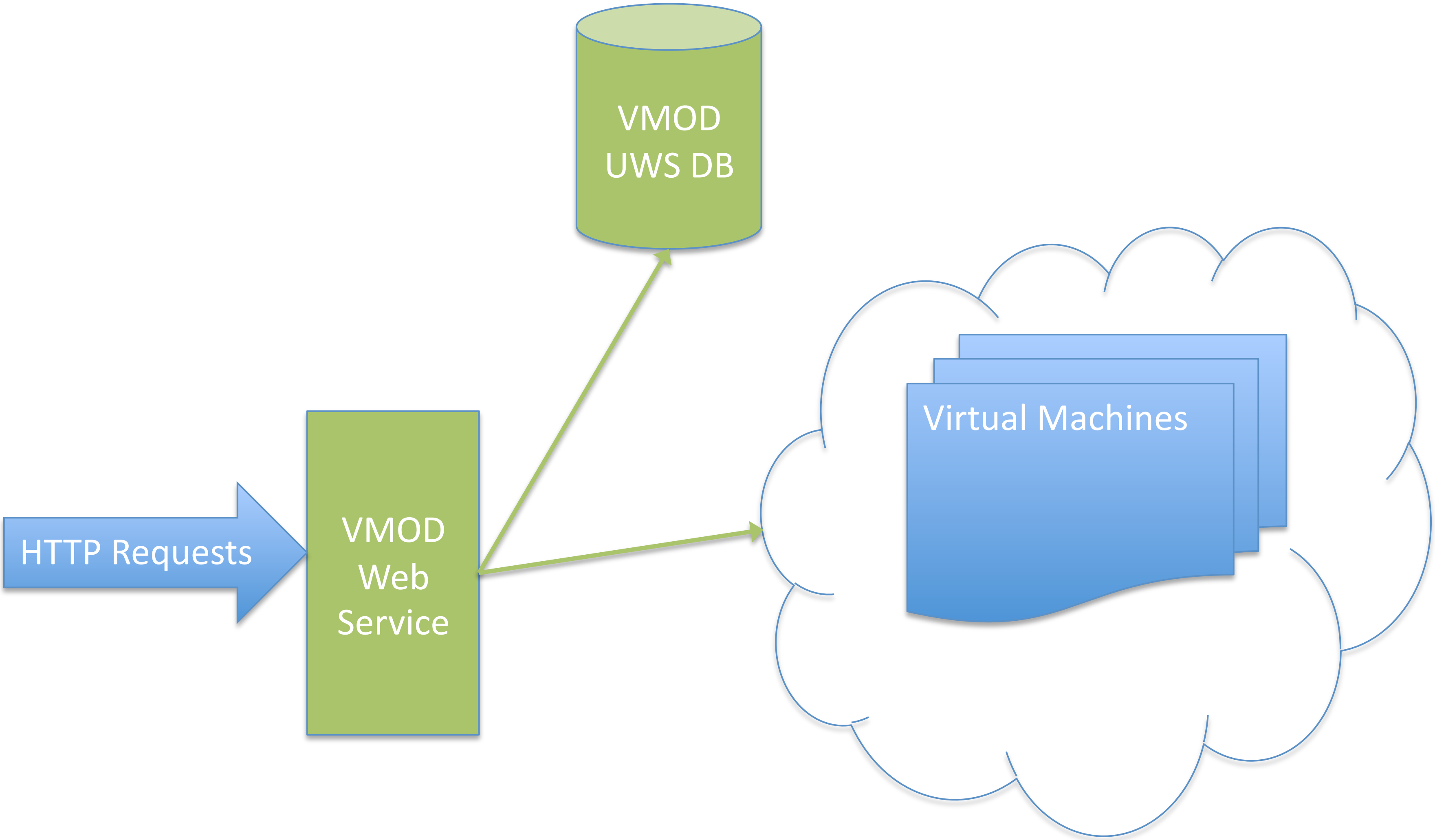
- RESTful service that allows operations like:
    - Start a virtual machine
    - Stop a virtual machine
    - Give me details about a virtual machine
    - List my virtual machines

Why use UWS in VMOD?

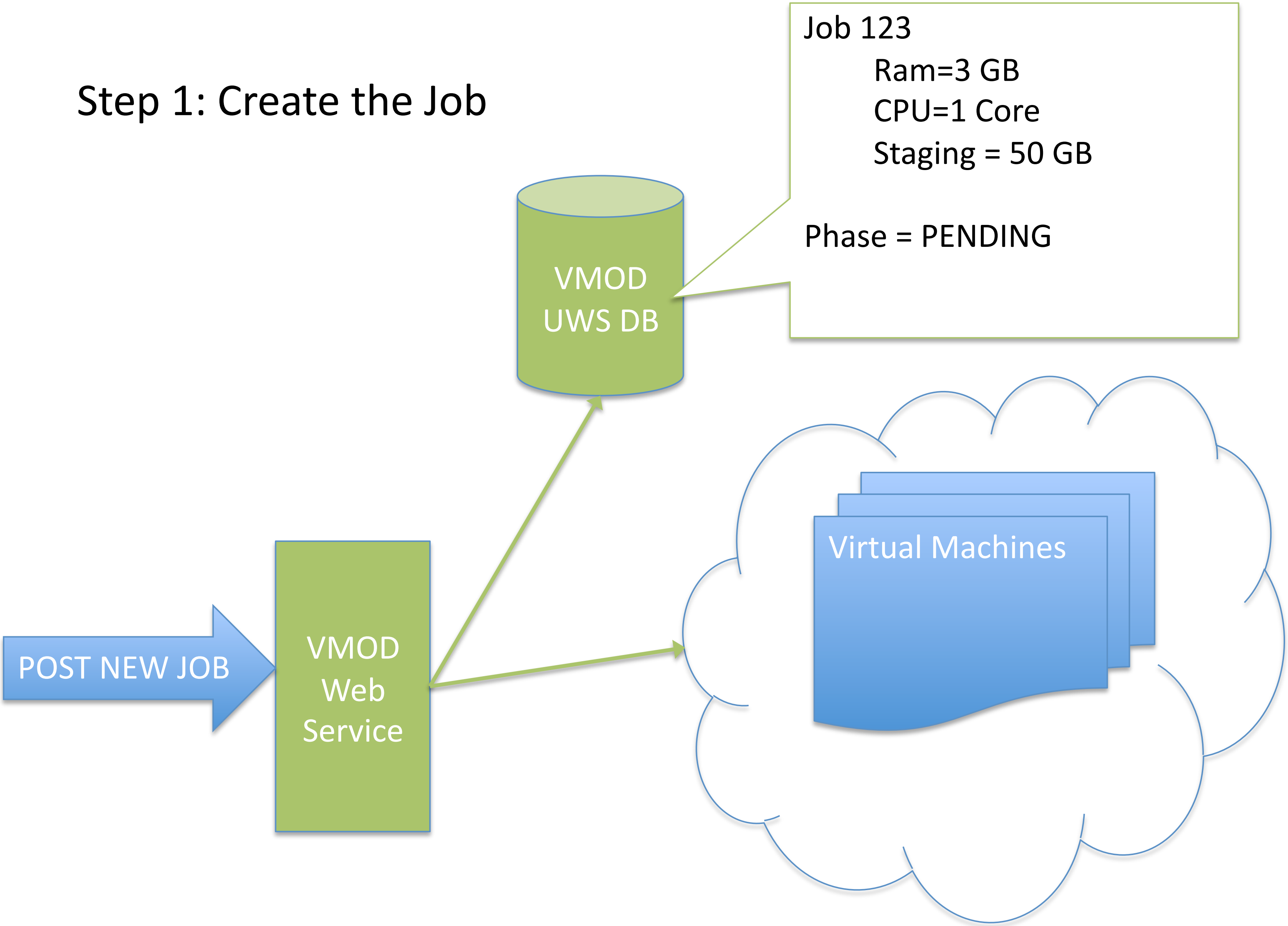
- Needed some sort of persistence layer to capture state, results of current and historical virtual machines.
- UWS job execution phases match the lifecycle stages of a VM quite well (next slide).
- Asynchronous aspects of UWS a good match for VMs because most operations, such as booting a VM, take some time.

Execution Phases -> VM State

UWS Execution Phase	VM State	Notes
PENDING	N/A	
QUEUED	VM is Booting	Usually takes 2 minutes
EXECUTING	VM is Running	For entire time VM is up and running
COMPLETED	VM has been gracefully shutdown by user.	Since VMs do not 'complete', this means users shutdown
ABORTED	VM has been shutdown by system.	Maximum lifetime reached, for example
ERROR	VM Failed to boot	


VMOD
UWS DB
HTTP Requests
VMOD
Web
Service
Virtual Machines

Step 1: Create the Job



Job 123

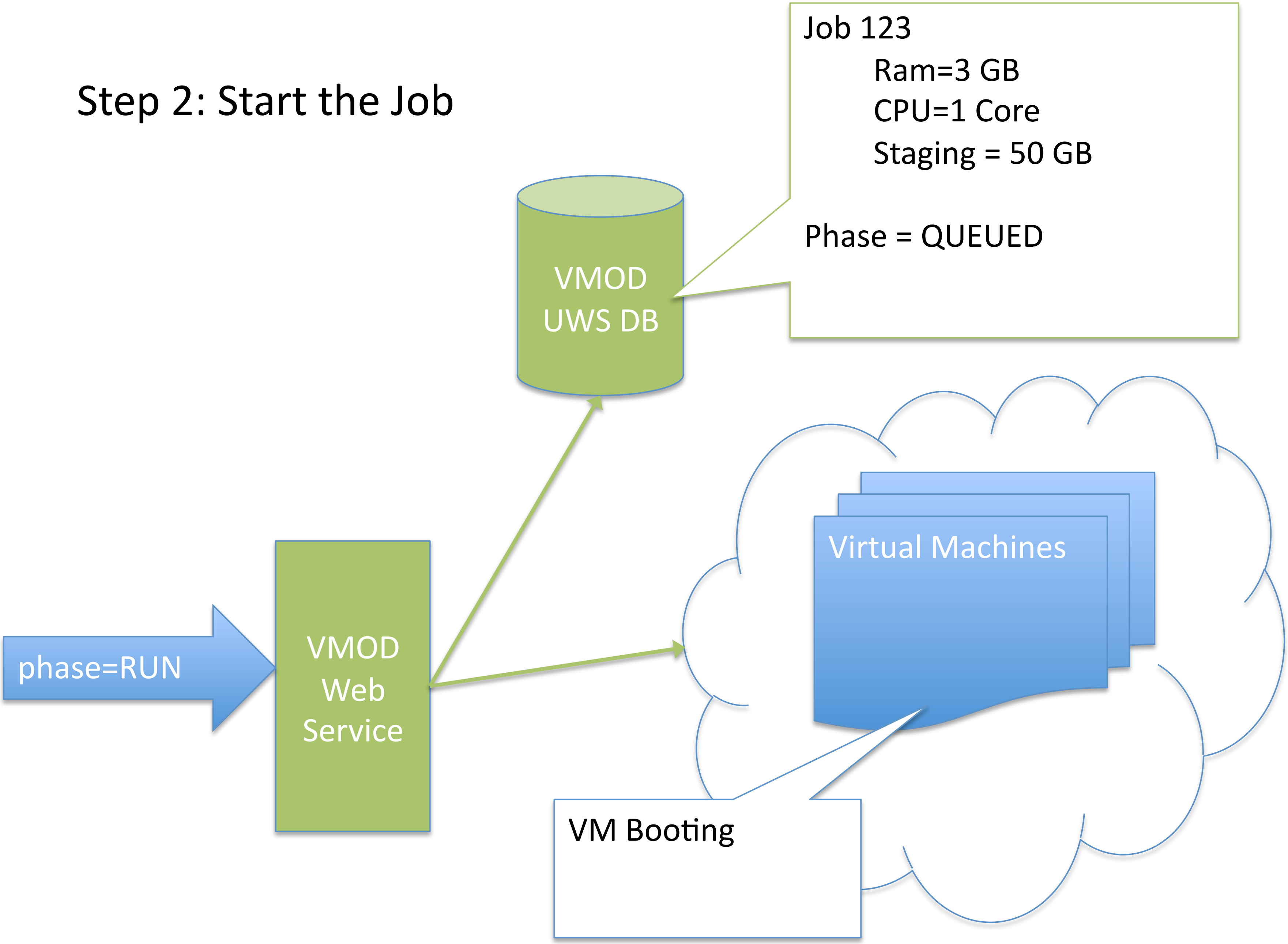
Ram=3 GB

CPU=1 Core

Staging = 50 GB

Phase = PENDING

Step 2: Start the Job



Job 123

Ram=3 GB

CPU=1 Core

Staging = 50 GB

Phase = QUEUED

VMOD UWS DB

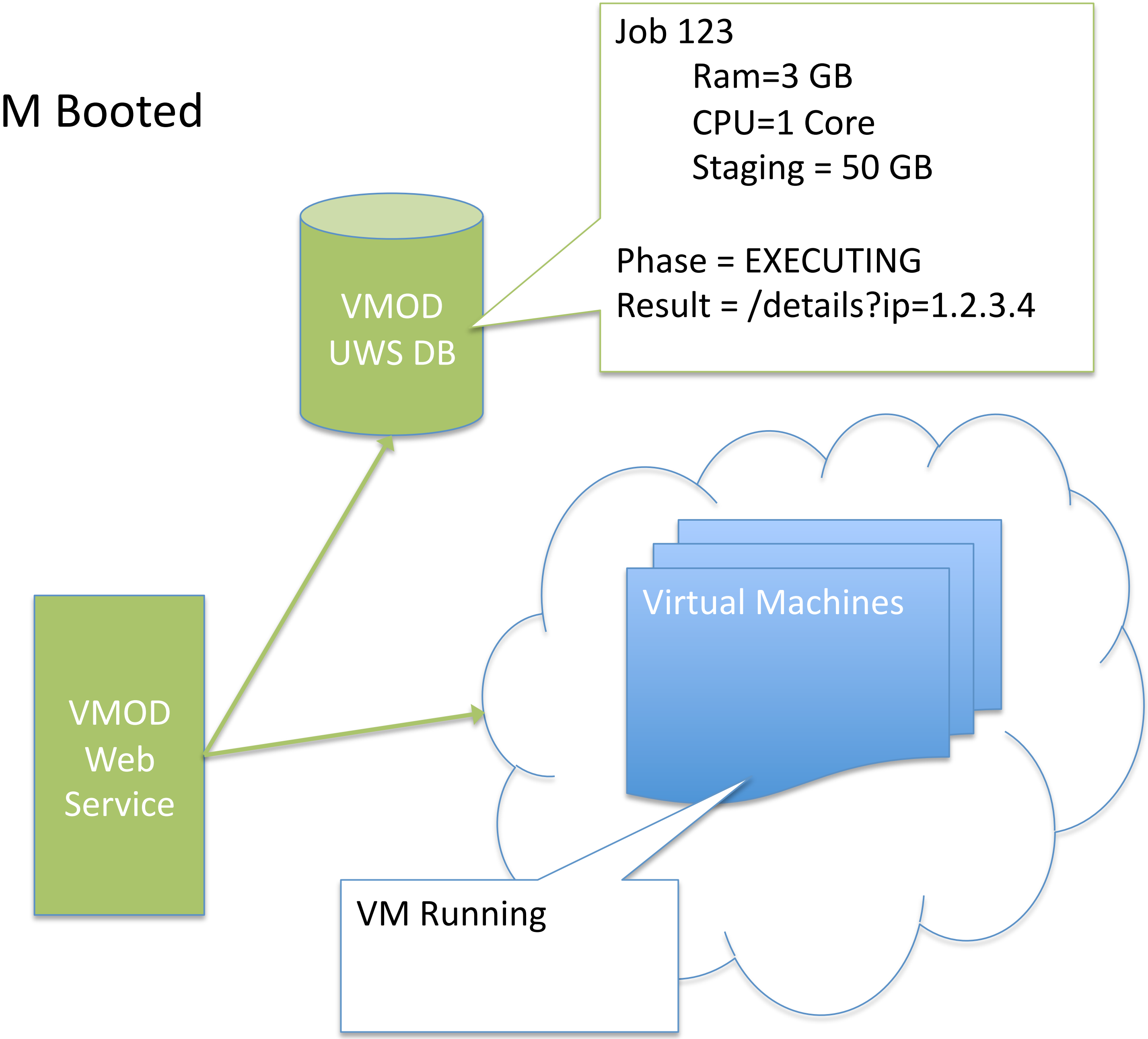
phase=RUN

VMOD Web Service

Virtual Machines

VM Booting

Step 3: VM Booted



Job 123

Ram=3 GB

CPU=1 Core

Staging = 50 GB

Phase = EXECUTING

Result = /details?ip=1.2.3.4

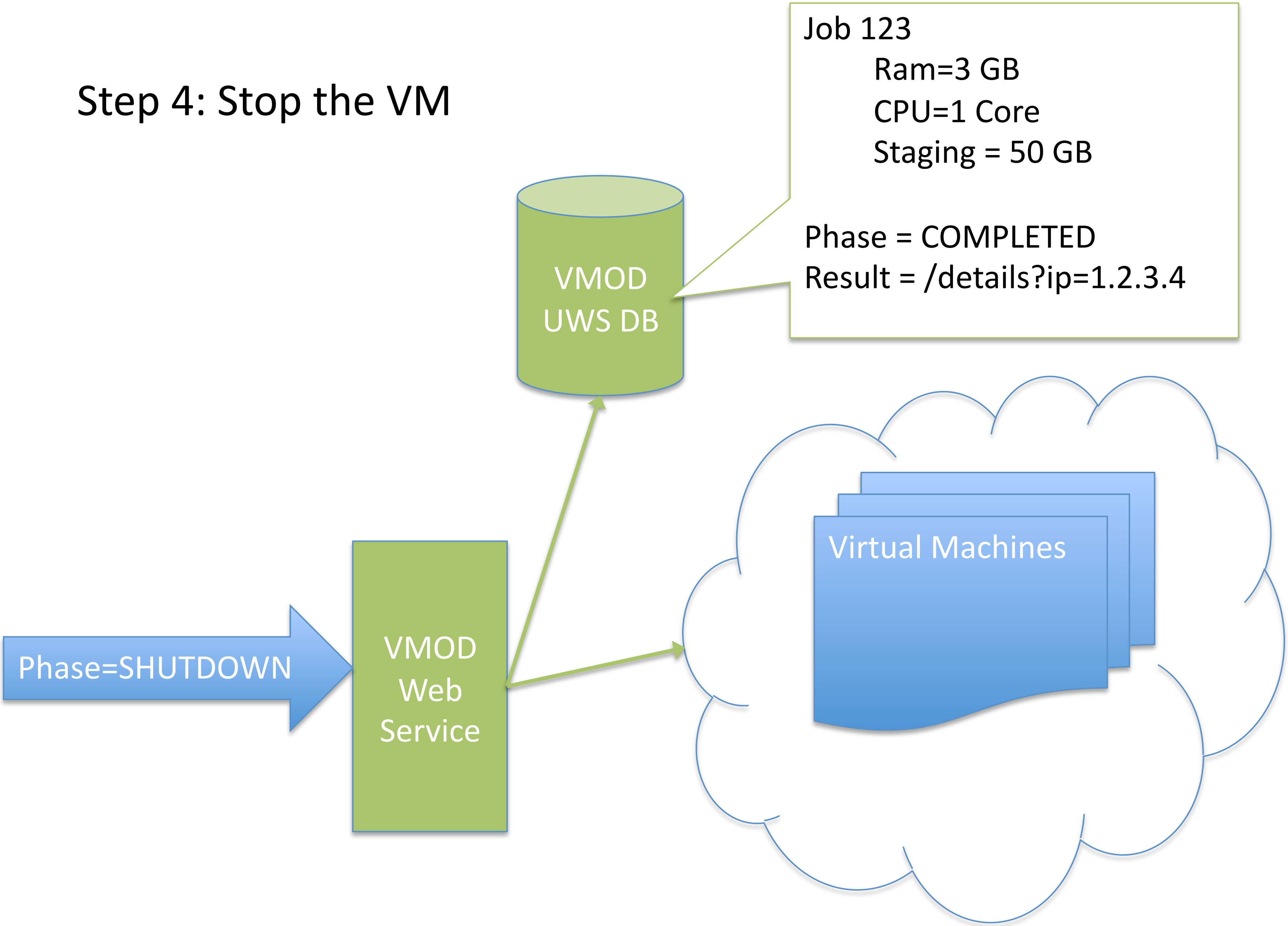
VMOD UWS DB

VMOD Web Service

Virtual Machines

VM Running

Step 4: Stop the VM



Job 123
    Ram=3 GB
    CPU=1 Core
    Staging = 50 GB

Phase = COMPLETED
Result = /details?ip=1.2.3.4

Examples…

Problem 1: Extended ‘EXECUTING’ phase

In VMOD, the job remains in EXECUTING for entire time VM is running (1 week maximum)

This presents some unique UWS problems:

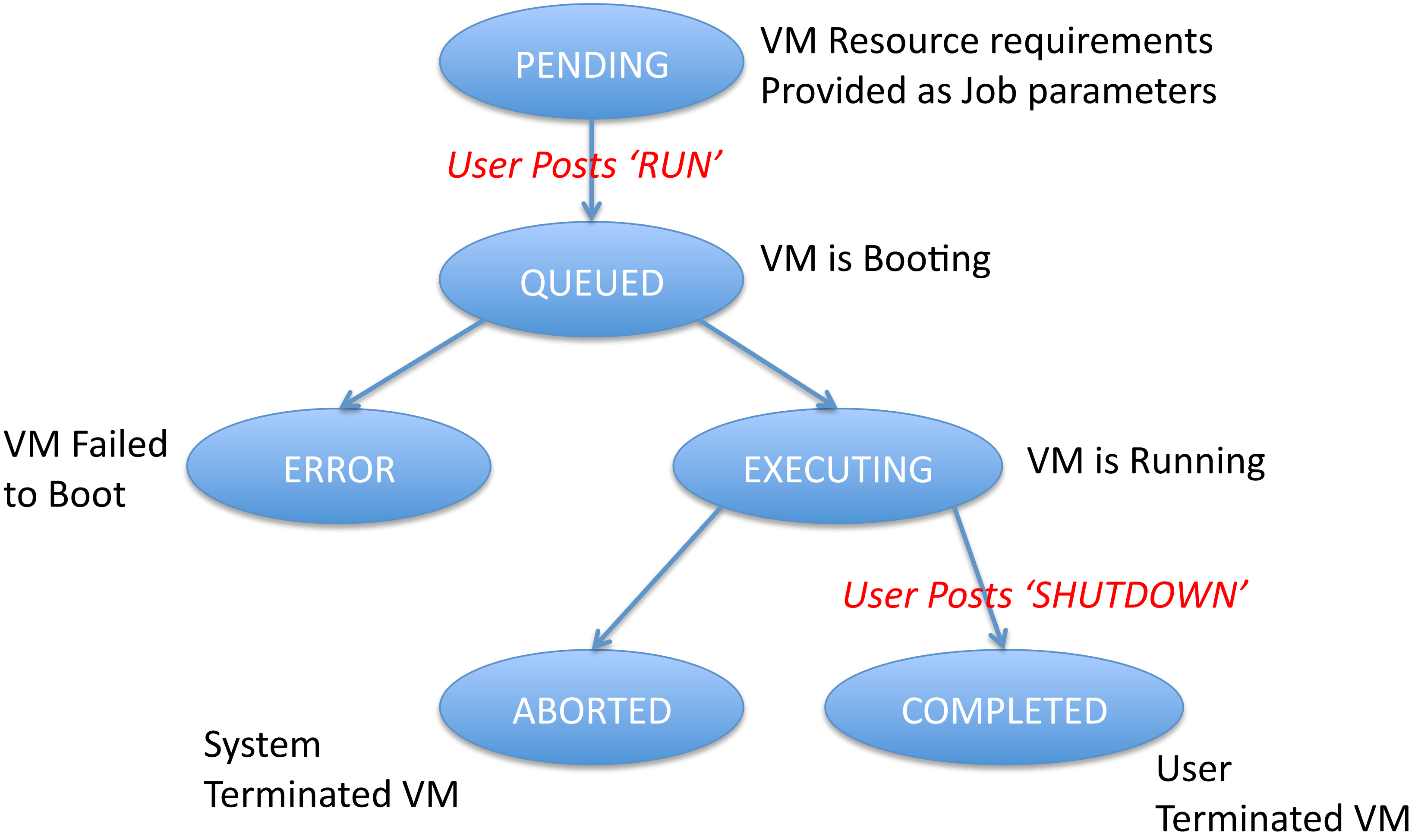
1. User initiated shutdown: Only ABORT available in UWS—a misleading term for the service API.
2. State synchronization: How to update Job phase when system shuts down VM.

Our Solutions…

- ABORTED phase is only used for system initiated shutdowns (e.g. max time exceeded)

- For user initiated shutdown, we made a UWS Vocabulary extension: New SHUTDOWN phase command, which takes the job from EXECUTING to COMPLETED

- If job phase is EXECUTING, ask system if VM still running. If yes, no changes. If no, set job phase to ABORTED



PENDING

VM Resource requirements
Provided as Job parameters

User Posts ‘RUN’

QUEUED

VM is Booting

VM Failed
to Boot

ERROR

EXECUTING

VM is Running

User Posts ‘SHUTDOWN’

ABORTED

System
Terminated VM

COMPLETED

User
Terminated VM

Problem 2: Storing VM Boot Information

After VM boot, users need information about the VM, such as IP Address

Normally, in UWS, results are stored in a file then referenced in the job. However, the VM information small, so using external persistence is overkill.

UWS results only allow for URIs (references)

So…

Solution (ugly)

Save the VM info as query strings in the URI.
Example:

```
/vmod/details?ipAddress=192.168.0.12
```

Where /details is a service that echos the query parameters as an XML doc:

```
<details>
   <ipAddress>192.168.0.12</ipAddress>
</details>
```

UWS Result Flexibility?

For use cases such as this, where results are small, it would be nice to save data directly in the Job as an arbitrary object.

This would save clients from making an extra call to the result store (in this case, our simple echoing service.)

Summary

- UWS a good design fit for VM lifecycle management; and, probably, for any service that run jobs for an arbitrary length of time
- Was nice to have the flexibility of extending the Execution Phase state machine
- In addition to allowing results to be data references, would like the option for results to be stored as actual data (next minor UWS version?)