	<b>Prepared by:</b> BioTeam Inc. Cambridge, MA USA <a href="http://www.bioteam.net">http://www.bioteam.net</a>
	<b>Primary Contact:</b> Chris Dagdigian Email: <a href="mailto:chris@bioteam.net">chris@bioteam.net</a> Tel/Fax: (866) 957-9994

# Howto: UniCluster & Amazon EC2

***BioTeam Lab Summary:***

***This document summarizes our experiences deploying Univa UD UniCluster Express within the Amazon Elastic Compute Cloud (EC2)***

***Prepared by:*** Chris Dagdigian, Principal Consultant

**July 3, 2008**

<b>Project Background</b>	<b>2</b>
History	2
Challenge	2
The Plan	3
The Use Case	3
<b>Getting Started With Amazon Web Services (AWS)</b>	<b>5</b>
<i>Getting Comfortable With AWS</i>	5
<i>Signing Up For An AWS Account</i>	5
<i>Configuring Client Side EC2 Management Environment</i>	6
<b>Deploying the UniCluster 3.2 Master &amp; Monitoring Console</b>	<b>8</b>
<i>Grid Master</i>	8
<i>Univa Monitoring Console</i>	8
<b>Installing UniCluster on Amazon EC2 Systems</b>	<b>9</b>
<i>Base Amazon Machine Image (“AMI”) -- Getting Started</i>	9
<i>EC2 Server Preparation</i>	9
<i>EC2 Network Security Requirements - Opening the required ports</i>	10
<i>Customizing the EC2 image for UniCluster Installation</i>	11
<i>Putting it all together - Towards a completely automated deployment</i>	13
<i>Bundling our custom server and registering a new AMI</i>	15
<b>Amazon EC2 UniCluster Deployment Summary</b>	<b>16</b>
<b>Video Demonstrations</b>	<b>17</b>
<b>Public AMI Image with UniCluster &amp; Deployment Scripts</b>	<b>18</b>

# Project Background

## History

BioTeam first became interested in Univa UD's software efforts after hearing Univa CTO Steve Tuecke speak in Regensburg, Germany at the [2007 Grid Engine workshop](#). Shortly after that event Univa formally became Univa UD after merging with United Devices. At the time, Steve's company seemed to be one of the few companies positioning themselves to offer full support and professional services encompassing commonly used open source products such as Sun Grid Engine that BioTeam often works with in the field. Individually these popular open source resources are relatively easy to acquire but Univa UD seemed to be making an interesting effort to become the one stop shop for a fully supported and integrated set of commonly required tools and technology.

One outcome of the new Univa UD is the [UniCluster Express](#) effort. UniCluster Express is a cluster tool-stack capable of deploying fully configured and integrated set of technologies including [Grid Engine](#) 6.1, [Grid Engine Analysis & Reporting Console](#) ("ARCo"), [Ganglia](#), central monitoring console and (incredibly) a fully operational [Globus](#) grid computing infrastructure. It itself is fully open-sourced.

All of these technologies, of course, are independently available and can be deployed by anyone with sufficient motivation and time. The ease of deployment for these technologies runs the full range: **easy** (Ganglia) -- **moderate** (Grid Engine & ARCo) -- **very complex** (Globus stack) so one very real value of UniCluster is the ability to deploy all of these technologies, simultaneously, in a fully integrated and supported manner.

Of significant note is that Univa UD plans to offer free support via [Grid.org](#) to all users of UniCluster Express. Given the capability and complexity of some of the components (Grid Engine and Globus in particular) this is a significant 'selling point' for potential users and Univa UD customers.

UniCluster also ships with a capable internally-developed Java-based central monitoring console (also fully open source) that uses the secure Globus web services framework to simultaneously collect, distill and present data collected from multiple grid sources including Grid Engine 'qstat', the ARCo job reporting database and the rrd-based system status monitoring databases created by Ganglia. Data from all of these sources is consolidated into a single console for monitoring and reporting.

We are largely a professional services company. To maintain our own technical skills we encourage our consultants to experiment in the lab with new products and technologies whenever there is downtime available. UniCluster express has been on our "to-do" list since late 2007 but evaluation efforts kept getting pushed back by client projects and consulting engagements. We had also spent most of early 2008 getting deeply familiar with [Amazon Web Services](#), including the [EC2](#) compute cloud and [S3](#) storage services.

BioTeam has talked publicly about some of our EC2 cloud computing projects including a [demonstration](#) shown at the 2008 [Open Source Grid & Cluster Conference](#) showcasing the automatic deployment and self-organization of full Grid Engine clusters within the EC2 cloud.

Knowing our work with and interest in EC2 and related technologies, Univa UD made us an offer that we could not refuse ...

## Challenge

Univa came to us with an interesting project proposal -- a challenge to deploy their UniCluster software within the Amazon Web Services framework and then document the experience. To facilitate, Univa UD would fund four days of BioTeam professional services effort so that we could treat this as a official company project and add it to our busy summer consulting schedule.

BioTeam accepted the challenge and this document (and related materials including screencasts and Amazon machine AMIs) is the result.

Just to be clear: Univa UD paid for some of the time necessary to undertake the work but the company did not have any controlling influence over what we choose to write after the fact. BioTeam takes it's independence pretty seriously.

## The Plan

The research plan was pretty simple. Assuming we were successful in deploying UniCluster within EC2 (success was not a given) we would attempt to provide the following resources to the community:

- A summary of the technical issues encountered and workarounds
- Enough detail to allow others to replicate our work
- Screencasts and other visual demonstrations and examples
- (if possible) a public Amazon AMI machine image containing ready-to-deploy UniCluster software for others to use as they see fit

## The Use Case

There were two main deployment options considered before beginning this project -- deploy 100% of the UniCluster installation within the EC2 cloud or go for a 'hybrid' solution where a dedicated server sits outside of the cloud and merely makes use of EC2-based compute nodes as-needed.

We chose to go with the hybrid approach - maintaining our own colocated server as an external, dedicated grid master. This approach matches more closely with how we have seen some of our more advanced clients and partners handle WAN-scale computing needs. Often there are core, company operated systems handling authentication, data staging and workflow while the remote systems operating across Metro or WAN-scale networking distances are merely used as engines for computation.

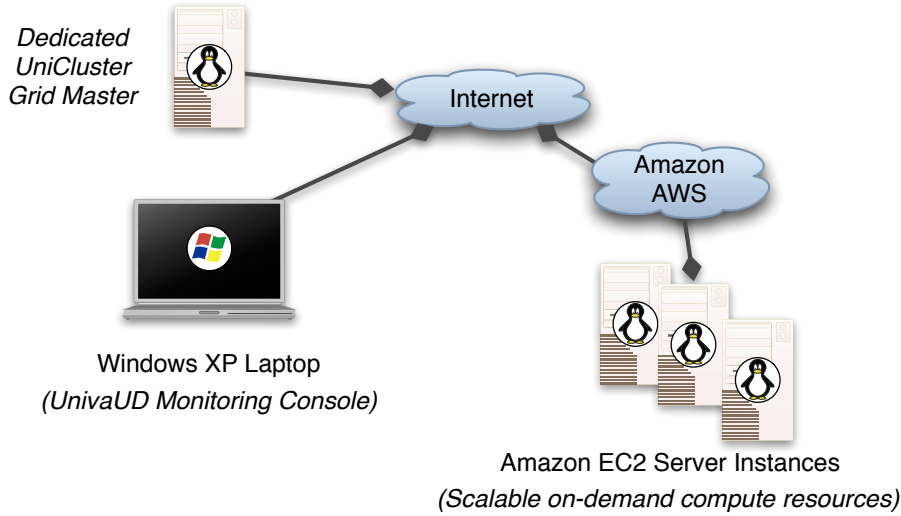
To this end, we deployed "cloudseeder.bioteam.net" - a formally decommissioned server from our colocation cage. Cloudseeder runs [Centos 4](#) Linux on a pair of 32bit Xeon CPUs. Our Amazon EC2 compute nodes run the more modern Centos 5 release. We had planned on upgrading Cloudseeder to Centos 5 but decided not to bother after the UniCluster installer handled the existing Centos 4 OS without issues or problems.

For the Univa UD monitoring console we made the explicit choice to install the Java application on a mobile platform rather than the dedicated grid master so that we could more fully test it's capabilities, especially when authenticating and transmitting data via the Globus web services infrastructure.

Our chosen test architecture would be compatible with an organization that operates a small cluster onsite that would occasionally like to be able to harness a bit of extra external computing power. For interested users without onsite resources, Amazon now offers "Elastic IP Addresses" for EC2 systems -- elastic IPs are static IP addresses that can be permanently associated with a particular server instance. By using an elastic IP associated with an always-running EC2 machine instance, a user can operate a dedicated master node without having to worry about hostname, DNS or IP allocation issues.

---

**cloudseeder.bioteam.net**



---

*Figure 1: The deployment architecture assumes a static and persistent system operating outside of the Amazon service(s) framework. The dedicated grid master is able to launch EC2-resident compute nodes on-demand.*

# Getting Started With Amazon Web Services (AWS)

To be productive with AWS services such as the EC2 cloud, there are three prerequisites:

1. You must be familiar with the available Amazon services (particularly EC2 and S3) - what they are, how they are used and what they are capable of.
2. You must sign up for AWS access in order to receive your access credentials and account number.
3. You must install the appropriate tools and utilities on your local computer that allow you to monitor and control AWS services

## Getting Comfortable With AWS

The best source of information is Amazon itself, particularly two websites:

- <http://aws.amazon.com> - Central portal for all information related to AWS
- <http://developer.amazonwebservices.com> - Central site for support, tutorials and FAQs

With the online tutorials and online guides it is possible to become comfortable with managing EC2-based servers within 1-2 hours of personal experimentation. The “must-read” document is the “Getting Started Guide” that is available as a prominent link off of the Amazon Developer Connection “Resources for Amazon Elastic Compute Cloud (Beta)” page.

## Signing Up For An AWS Account

Instantiating a remote virtual server node requires first signing up for the Amazon EC2 (Elastic Compute Cloud) web service. Signing up for EC2 involves filling out billing information using Amazon's on-line web form and downloading/recording five AWS Access Identifiers (Amazon Web Service).

Signing up for EC2 access needs to be performed once per billing entity. This might be once per institution or perhaps several times per institution. It really depends on how the AWS bills will be paid and how willing users are to share common control. All users of a common AWS account will minimally require a copy of the Private Key File and X.509 Certificate File to instantiate their own systems.

If users plan to create or modify an AMI (Amazon Machine Image) or upload/download data to S3 (Simple Storage Service), they will additionally require Your Access Key ID, Your Secret Access Key, and Account Number.

---

**WARNING:** Multiple users of a common EC2 account have the ability to terminate any other user's instances and delete any other user's AMI(s) or S3 data.

---

Step-by-step:

1. Sign up for AWS access on [Amazon's EC2 Page](#)
2. Once billing information has been entered return to [Amazon's EC2 Page](#) and select *AWS Access Identifiers* under *Your Web Services Account*
3. Record the strings **Your Access Key ID** and **Your Secret Access Key**
4. Under X.509 Certificate click **Create New** and **Yes**
5. Download the files **Private Key File** and **X.509 Certificate File**
6. Select Your AWS Profile and record your **Account Number** (without the hyphens)

These five values can get a bit confusing, so it's worth describing them out-front:

- X.509 Certificate File is a username-like file, named like cert-*<lettersandnumbers>*.pem, can be re-created on [Amazon's EC2 Page](#), and is used with Amazon's command-line tools
- The Private Key File is a password-like file, named like pk-*<lettersandnumbers>*.pem, can be re-created on [Amazon's EC2 Page](#), and is used with Amazon's command-line tools
- Account Number is a string of integers that is permanently associated with an EC2 account and is used in creating AMI(s)
- Your Access Key ID is a username-like string that is permanently associated with an EC2 account, and is used when creating AMI(s) or using S3
- Your Secret Access Key is a password-like string, can be re-created on [Amazon's EC2 Page](#), and is used when creating AMI(s) or using S3

## Configuring Client Side EC2 Management Environment

Configuring the client-side environment involves downloading and installing Amazon's java-based EC2 command-line tools, copying your Private Key File and X.509 Certificate File somewhere safe and accessible, and defining several environmental variables.

The following installation steps should work under a Linux or Mac OS X environment:

### Download and un-archive the [EC2 command-line tools](#)

These tools can be kept anywhere, however in this example we show keeping them within a subdirectory within a user home directory.

```
-bash$ cd $HOME
-bash$ wget http://s3.amazonaws.com/ec2-downloads/ec2-api-tools.zip
-bash$ unzip ec2-api-tools.zip
```

Download and install the Private Key File and X.509 Certificate File. These can be kept anywhere, however in this example I show keeping them within a `~/X.509` directory. This example assumes the files have already been downloaded to your HOME directory.

```
-bash$ mkdir $HOME/.X.509
-bash$ mv $HOME/*.pem ~/.X.509/
```

### Define environmental variables.

Several environmental variables are required to use Amazon's command-line tools. These can be defined in many ways, however in this example we show defining these within a `~/.bash_profile`. Then source `~/.bash_profile`. `$JAVA_HOME` should point to the parent directory of a recent JRE (Java Runtime Environment) containing `bin/java` and `lib/java`. A Java JRE can be downloaded from <http://java.sun.com>.

```
export JAVA_HOME=<path to Java JRE>
export EC2_HOME=~/.ec2-api-tools
export PATH=~/.bin:$EC2_HOME/bin:$S3SYNC:$PATH
export EC2_PRIVATE_KEY=~/.X.509/pk-<lettersandnumbers>.pem
export EC2_CERT=~/.X.509/cert-<lettersandnumbers>.pem
export AWS_ACCOUNT_NUMBER=<numbers>
export AWS_ACCESS_KEY_ID=<lettersandnumbers>
export AWS_SECRET_ACCESS_KEY=<lettersandnumbers>
```

### Testing the EC2 tools environment

The easiest test may just be to see if you can successfully query the EC2 service to see a list of available machine images you are allowed to access. The list returned will be a list of all "public" AMIs in addition to any private AMIs created and associated with your access key.

If this command runs successfully, your local environment is ready to be used:

```
ec2-describe-images --all
```

# Deploying the UniCluster 3.2 Master & Monitoring Console

## Grid Master

Our dedicated master host for this experiment was a recently retired server from the BioTeam colocation cage. "cloudseeder.bioteam.net" is a Dual CPU 1.4Ghz Pentium III with 2GB physical memory and approximately 500GB of RAID5 storage provided by a 3Ware controller.

The system runs a fully up to date version of Centos 4.6, a planned upgrade to Centos 5 was cancelled when the UniCluster software installed without issue directly onto the Centos 4.6-based system.

The UniCluster software is available as a tar.gz distribution, we downloaded and used `unicluster_express-3.2.0-rh4-i386.tar.gz` from [grid.org](http://www.grid.org) for all work described in this document. The UniCluster Installation Guide (available online at <http://www.grid.org>) is very upfront about the various OS dependencies and requirements, none of which are particularly difficult to satisfy.

Recommended Firewall and SELinux settings were implemented as recommended in the Installation Guide.

As a nice touch, the UniCluster installation script `./install_unicluster.sh` also checks for various dependency and configuration problems and will report on errors encountered.

Once all of the OS level dependencies were satisfied, installing the UniCluster Express software on the master node was as simple as running the `./install_unicluster.sh` script and answering the interactive questions. The only decisions made were to deploy a host-type of "master" and to pick a bootstrap password that will be used by future clients.

## Univa Monitoring Console

UniCluster Express ships with a consolidated monitoring and reporting console created by Univa UD. The console is a Java application supported on both Windows and Linux clients that uses the Globus secure web services framework to communicate with the grid.

The use of Globus in this context adds a bit of extra difficulty to the Monitor Console installation as the appropriate credentials and certificates must accompany the Java software in order for everything to function correctly.

Installation on a virtualized Windows XP system was a simple 2-step process:

1. Copy the existing `client-installer.zip` file from the `$UNICLUSTER_ROOT/share/unicluster_monitor_console/` directory onto the Windows system
2. Create a `".globus/"` folder in the home directory of the windows user and populate the folder with the certificates found within the `$UNICLUSTER_ROOT/share/certificates/` directory.

Parties interested in UniCluster Express should take a serious look at the Monitoring Console. Univa UD has created a secure, unified interface that combines real-time and historical data taken from multiple sources including:

- Grid Engine 'qstat' client (for live grid and job status data)
- Grid Engine ARCo database (for historical job and usage reporting)
- Ganglia 'gmetad' (for live host information)
- Ganglia RRD databases (for historical host state and status information)

All of these data sources are consolidated into a unified view presented by the Univa Monitor Console, quite a significant achievement.

## Installing UniCluster on Amazon EC2 Systems

### Base Amazon Machine Image (“AMI”) -- Getting Started

Rather than build out our own Centos server image for use within EC2, we started with a publicly available Centos-5 server image created by the brilliant folks at RightScale ([www.rightscale.com](http://www.rightscale.com)). The folks at RightScale not only make public server images available they also fully publish and document their build and update scripts -- an incredibly useful resource for people learning how to follow in their footsteps.

The RightScale Centos 5 images (available in both 32-bit and 64-bit versions) can be launched by anyone and come with all of the latest Amazon EC2 utilities and management software pre-installed. It is an excellent springboard for organizations and individuals starting out with Linux based EC2 systems.

The RightScale AMI build philosophy is also something that BioTeam strongly supports and has emulated in our own private AMI server images. Rather than build and bundle monolithic server images packed with custom software, a RightScale image is fairly lightweight. After initial boot, the RightScale system will automatically connect to the S3 storage cloud to download the latest provisioning, configuration and software installation resources. This method is a significant time saver - rather than waste time bundling and re-bundling server images to push out changes, modifications are instead made to the scripts and software sitting in the S3 storage “buckets”. The next time the EC2 server reboots, the fresh changes will be pulled down from S3.

---

For EC2-based UniCluster systems, BioTeam started with the RightScale public server image **AMI ID ami-08f41161** - A server image for 32-bit systems containing Centos 5 Linux and EC2 utilities.

---

### EC2 Server Preparation

After launching 1 instance of EC2 ami-08f41161 we uploaded our personal Amazon AWS/EC2 certificates and key files and also created a new user account (“griddag”) with the same UID/GID as “griddag” has on cloudseeder.bioteam.net.

### Installing Net::Amazon::EC2

One limitation of the **ami-08f41161** server image is that it does not include the perl module [Net::Amazon::EC2](#) which we find essential for scripting EC2-based post-install client actions. The Net::Amazon::EC2 module provides a scriptable Perl interface useful for querying information about our reservation, instances and public/private host names.

The Centos “yum” tool allowed us to install some Net::Amazon::EC2 dependencies via simple RPMs:

```
# yum -y install perl-XML-Simple perl-Params-Validate perl-Digest-HMAC \  
perl-Sub-Name perl-Sub-Exporter perl-Test-Exception
```

There were, however, a few additional Perl module dependencies that needed to be installed. Rather than an interactive install via CPAN.pm, scriptable actions were used to complete the installation:

```
wget http://cpan.org/modules/by-module/Test/Test-LongString-0.10.tar.gz  
tar zxvf Test-LongString-0.10.tar.gz  
cd Test-LongString-0.10  
perl Makefile.PL
```

```

make
make install
cd ..

wget http://cpan.org/modules/by-module/Class/Class-MOP-0.53.tar.gz
tar zxvf Class-MOP-0.53.tar.gz
cd Class-MOP-0.53
perl Makefile.PL
make
make install
cd ..

wget http://cpan.org/modules/by-module/XML/XML-Simple-2.18.tar.gz
tar zxvf XML-Simple-2.18.tar.gz
cd XML-Simple-2.18
perl Makefile.PL
make install
cd ..

wget http://search.cpan.org/CPAN/authors/id/S/ST/STEVAN/Moose-0.40.tar.gz
tar zxvf Moose-0.40.tar.gz
cd Moose-0.40
perl Makefile.PL
make
make install
cd ..

wget http://cpan.org/modules/by-module/Net/Net-Amazon-EC2-0.06.tar.gz
tar zxvf Net-Amazon-EC2-0.06.tar.gz
cd Net-Amazon-EC2-0.06
perl Makefile.PL
make
make install
cd ..

rm -rf Test* Class* Moose* XML* Net*

```

## Installing UniCluster dependencies

There were only two additional RPMs that needed to be installed in the AMI image in order to match UniCluster requirements: Xinetd and LibArt:

```
# yum install xinetd libart_lgpl
```

## EC2 Network Security Requirements - Opening the required ports

Amazon EC2 systems live behind a restrictive firewall, modifications need to be made to the security configuration data associated with the AWS user account. For this experiment we made the changes to our “default” security profile, others may wish to create a specific profile just for UniCluster/EC2 usage.

The UniCluster Administration Guide has a complete chapter on “**Configuring UniCluster Express Behind a Firewall**” which contains explicit information about what ports and services need to be able to communicate with each other. The only deviation we made from the suggestions was to constrain the Globus GridFTP ports to the range 4000-5000 rather than open up the entire port range above 1024.

With the list of ports and services in-hand, it is easy to use Amazon tools to configure the access rules.

From the command line, enabling TCP port 2222 for GSI-SSH access from any external client would be:

```
# ec2-authorize default -P tcp -p 2222 -s 0.0.0.0/0
```

... and enabling the TCP port range from 4000-5000 for GridFTP is handled via:

```
# ec2-authorize default -P tcp -p 4000-5000 -s 0.0.0.0/0
```

Without explicitly listing every command, we used 'ec2-authorize' to make sure the required UniCluster ports as described in the Administration Guide were open to access from the outside world.

---

**Note:** For those who prefer a browser-based GUI, the excellent Firefox plugin - "[Elasticfox Firefox Extension for Amazon EC2](#)" provides an excellent GUI interface to EC2 management and control actions.

---

### Special ICMP 'ping' Note

The UniCluster installation script will first attempt to ping the public interface of the node it is running on.

By default the Amazon EC2 network access profile blocks the ICMP traffic sent by the 'ping' command. When the install\_unicluster.sh script fails to ping the public IP address it will fail with the following error:

```
Unable to ping short hostname: ec2-72-44-38-173
This will prevent SGE setup from succeeding.
Please rerun the installation after you have configured your system to resolve
its short hostname.

at ./lib_install_unicluster.pl line 2086
    main::ValidateBaseOptions() called at ./lib_install_unicluster.pl line
3497
    main::ValidateOptions() called at ./lib_install_unicluster.pl line 5158
```

The solution to this problem is to make one final "ec2-authorize" change to allow ICMP message traffic:

```
ec2-authorize default -P icmp -t -1:-1
```

## Customizing the EC2 image for UniCluster Installation

There are a number of server changes that need to be made to EC2-based server images both **before** and **after** the UniCluster installation in order for all components to function.

### (Pre-Install) - Fixing hostname and /etc/hosts

---

There are three issues with the default setup of the EC2 server image that will break the UniCluster installation process. All three issues need to be fixed prior to running the UniCluster installation.

#### Issue 1 - /etc/hosts

The /etc/hosts file created after the EC2 image is provisioned contains hostnames and IP addresses only for the internal/private EC2 network. This file needs to be completely replaced with a file that references the external public hostname and IP addresses. The public IP addresses and hostnames are used by Grid Engine and the Globus infrastructure.

#### Issue 2 - System hostname

Both Grid Engine and Globus key in on the system hostname. By default this is set to the private/internal EC2 hostname. This needs to be changed to the public hostname before installing UniCluster.

#### Issue 3 - Learning the EC2 public IP address

The default server has no idea of its own public IP address as it is hidden behind a NAT gateway on a private network. To learn it, we must first query our reservation to learn our public hostname and then query that hostname against a public DNS server to resolve it back to an IP address. This IP address is

used in 2 places - the first is in the /etc/hosts file and we also write it to a file location: /etc/sysconfig/EC2-public-IPaddr so it can be used later by the post-install script that fixes Grid FTP.

### Automated script to make these changes

To facilitate these fixes for unattended and automatic EC2 node provisioning, a rough perl script was created: “\_pre-01-fix-hosts-and-dns\_pl\_.txt”. The script can be viewed or downloaded at the following URL:

Preflight Script #1 - [http://bioteam.net/dag/UniCluster/\\_pre-01-fix-hosts-and-dns\\_pl\\_.txt](http://bioteam.net/dag/UniCluster/_pre-01-fix-hosts-and-dns_pl_.txt)

---

**Note:** These scripts are “proof-of-concept” quality and are not optimized or seriously reviewed in any way. Use at your own risk.

---

### (Install) - Automatic Installation of UniCluster Express “Login” Host Type

---

Once the required hostname and /etc/hosts changes (and ICMP traffic authorization via ec2-authorize) described in previous sections are completed, the node is ready to run the Univa UD “install\_unicluster.sh” script without any problems.

To auto-deploy UniCluster on an EC2 node without requiring user input on the command line:

```
./install_unicluster.sh --batch --host-type login --accept-eula
--bootstrap-server-hostname cloudseeder.bioteam.net \
--bootstrap-client-user ucluster \
--bootstrap-client-password <bootstrap password>
```

The only user required information is the hostname of the public grid master and the bootstrap password. The script used to auto-deploy UniCluster can be found here:

Install Script #1 - <http://bioteam.net/dag/UniCluster/install-UniCluster-login-node.sh.txt>

### (Post Install) - Fixing the UniCluster installation for EC2-based operation

---

There are three issues with the default setup of the EC2 server image that will break UniCluster operation after installation has occurred. All can be fixed via “post install” scripts that run after UniCluster has been deployed.

#### Issue 1 - Tell unicluster-gridftp about the constricted TCP port range it must use

Rather than open all ports higher than 1024 for GridFTP access (as the installed software assumes will be available) we need to tell the Globus infrastructure that it must operate within a constrained TCP port range. Previously we had used the Amazon EC2 ec2-authorize script to enable access for the TCO port range [4000-5000].

To do this involves setting 2 environment variables (\$GLOBUS\_TCP\_PORT\_RANGE=4000,5000 and \$GLOBUS\_TCP\_PORT\_RANGE\_STATE\_FILE=/tmp/port\_state).

These extra environment variables must be added to the /etc/xinetd.d/unicluster-gridftp file so they become visible to the GridFTP server when it launches.

An automatic script to create this change after the UniCluster install has been created. It can be viewed here:

Post-Install Script #1 -- [http://bioteam.net/dag/UniCluster/\\_post-01-Fix-Globus-ENV.sh.txt](http://bioteam.net/dag/UniCluster/_post-01-Fix-Globus-ENV.sh.txt)

## Issue 2 - Fix Ganglia data reporting

By default, Ganglia uses multicast broadcasts to report system status and state information. This fails utterly in the Amazon EC2 environment where nodes may be on entirely different networks and subnets which certainly do not propagate multicast traffic.

The fix is to explicitly tell Ganglia to use unicast methods to communicate directly with the known grid master.

A script that automates this change (by pulling the public hostname of the grid master from the Grid Engine configuration) can be found here:

Post-Install Script #2 - [http://bioteam.net/dag/UniCluster/\\_post-02-Modify-Ganglia\\_pl\\_.txt](http://bioteam.net/dag/UniCluster/_post-02-Modify-Ganglia_pl_.txt)

## Issue 3 - Fix GridFTP environment for operation behind NAT gateway

A 2nd change is needed in the /etc/xinetd.d/unicluster-gridftp file. Because the EC2 server node is hidden from the internet via a network address translation (NAT) gateway, we must explicitly pass a new configuration directive to the GridFTP server to make it aware of its public-facing IP address.

The new configuration directive is "-data-interface <IP>" and to edit this into the server startup script we pull our public IP address from the /etc/sysconfig/EC2-public-IPaddr file that was created by our initial pre-install hostname and /etc/hosts fix.

A script that automates this change can be found here:

Post-Install Script #3 - [http://bioteam.net/dag/UniCluster/\\_post-03-Fix-GridFTP.sh.txt](http://bioteam.net/dag/UniCluster/_post-03-Fix-GridFTP.sh.txt)

## Putting it all together - Towards a completely automated deployment

The following "deploy-this-node.sh" script will automatically run all of the pre- and post-install scripts in the proper order:

```
#!/bin/sh

##-----
## Must set these to working values
export AWS_ACCESS_KEY_ID=<YOUR KEY HERE>
export EC2_PRIVATE_KEY=<YOUR KEY HERE>
##-----

# Fix hostname and DNS issues and build new /etc/hosts file
echo "## Running _pre-01-fix-hosts-and-dns.pl "
/opt/univa-prep-scripts/_pre-01-fix-hosts-and-dns.pl

# Autoinstall ClusterExpress
echo "## Running _install-UniCluster-login-node.sh"
. /opt/univa-prep-scripts/_install-UniCluster-login-node.sh

# Add TCP port ranges for Globus GridFTP to xinetd.d/unicluster-gridftp
echo "## Running _post-01-Fix-Globus-ENV.sh"
. /opt/univa-prep-scripts/_post-01-Fix-Globus-ENV.sh
```

```

# Reconfigure the Univa Ganglia install to unicast status
# data back to our remote grid master instead of using broadcast
# based method which don't span subnets
perl /opt/univa-prep-scripts/_post-02-Modify-Ganglia.pl

# Restart ganglia
/etc/rc.d/init.d/gmond restart

## Add a new -data-interface line to gridFTP configuration so
## it will work behind the NAT used by Amazon AWS EC2
. /opt/univa-prep-scripts/_post_03-Fix-GridFTP.sh

```

The script can be found here: <http://bioteam.net/dag/UniCluster/deploy-this-node.sh.txt>

## The final step

Once we have a working set of “pre install” and “post install” scripts that are called in the correct order by “deploy\_this\_node.sh” there is only one more step to complete: Ensure that this script gets invoked as the EC2 server image starts up for the first time.

There are a number of ways to get “on first boot” behavior. We chose the simple process of simply appending some extra lines to the /etc/rc.local file in which the listed commands are executed by the root user as one of the final system init steps.

Our installation logic depends on the presence or absence of a simple text lock file called “/var/db/.UniClusterConfigDone”. If that file exists, the auto-installation process is skipped. If the lock file is not found, the auto-installation process is triggered.

This allows us to re-deploy unicluster on a running EC2 node simply by wiping out the Unicluster install directory, removing the lockfile and rebooting the AMI instance.

Appended to /etc/rc.local file:

```

## Unicluster auto deployment ...

if [ -e /var/db/.UniClusterConfigDone ]; then
  echo "#####"
  echo "/var/db/.UniClusterConfigDone exists."
  echo "UniCluster Auto-deployment skipped."
  echo "#####"
else
  echo "#####"
  echo "/var/db/.UniClusterConfigDone does not exist."
  echo "UniCluster Auto-deployment starting."

  echo "Hostname before:" >> /root/node-deploy-log.txt
  /bin/hostname >> /root/node-deploy-log.txt

  /opt/univa-prep-scripts/deploy-this-node.sh >> /root/node-deploy-log.txt

  echo "Hostname after:" >> /root/node-deploy-log.txt
  /bin/hostname >> /root/node-deploy-log.txt

  touch /var/db/.UniClusterConfigDone
  echo "    /var/db/.UniClusterConfigDone created"
  echo "UniCluster auto-deployment complete."
  echo "#####"
fi

```

## Bundling our custom server and registering a new AMI

Now that a functional server exists that is capable of automatically deploying UniCluster it is time to “bundle” our system up so that it can be published and used as an Amazon EC2 server image. Once bundled, we “register” the bundle in order to be assigned a new machine instance ID (AMI ID).

Three Amazon EC2 utility programs are used in this process:

- ec2-bundle-vol
- ec2-upload-bundle
- ec2-register

The commands to bundle and register a system were placed into a script. This script must be run manually because occasionally there are upload issues getting data into the S3 cloud:

```
#!/bin/sh

echo "" > /var/log/messages

rm -rf ~/.ssh/known_hosts /var/db/.UniClusterConfigDone

ec2-bundle-vol -d /mnt -e /usr/local/unicluster/ \
-k $EC2_PRIVATE_KEY -c $EC2_CERT -u $AWS_ACCOUNT_NUMBER \
-r i386 -p 32bit_clusterExpress-loginNode

ec2-upload-bundle -b Univa UD_demo_images \
-m /mnt/32bit_clusterExpress-loginNode.manifest.xml \
-a $AWS_ACCESS_KEY_ID -s $AWS_SECRET_ACCESS_KEY \

ec2-register Univa UD_demo_images/32bit_clusterExpress-loginNode.manifest.xml
```

When the above script runs successfully, we are presented with our new AMI identifier. This AMI can now be used throughout the Amazon EC2 system.

---

**Note:** Any AMI images created are set as “private” by default - usable only by the person who created and registered the image. The exact image used by BioTeam for this experiment can not be made public as the image contains our EC2 credentials as well as the UniCluster bootstrap password specific to cloudseeder.bioteam.net.

---

# Amazon EC2 UniCluster Deployment Summary

To recap the steps involved in creating an EC2 AMI server image that can successfully and automatically bootstrap itself into a running UniCluster environment we did the following:

1. Started with the public RightScale Centos 5 server image (AMI-08f41161)
2. Uploaded our AWS keys and certificates and created a 'griddag' test user account
3. Installed a few UniCluster dependencies (xinetd & libart\_lgpl)
4. Installed the perl module Net::Amazon::EC2 and all related dependencies
5. Made public network access policy changes via 'ec2-authorize'
6. Developed a Pre-UniCluster script to automatically recreate /etc/hosts and change the system host-name
7. Developed a script to automatically install UniCluster Express (host type 'login node')
8. Developed a Post-UniCluster script to add Globus TCP\_RANGE environment variables
9. Developed a Post-UniCluster script to modify the Ganglia communication configuration
10. Developed a Post-UniCluster script to fix NAT issues with GridFTP
11. Developed a simple shell script that runs all of the pre and post-install commands in proper order
12. Developed logic in /etc/rc.local to trigger the auto-deployment script when needed.
13. Developed a script to "bundle" our server and register it as a new (private) Amazon EC2 AMI

All of the created scripts can be found online here:

<http://bioteam.net/dag/UniCluster/>

## Video Demonstrations

A series of recorded flash videos have been created to compliment this document. The recorded videos demonstrate various capabilities of the EC2/UniCluster grid.

All videos are available online at the following URL:

<http://www.screencast.com/users/BioTeam/folders/UniCluster-in-Amazon-EC2>

The most interesting video is likely “Deploying UniCluster live into the Amazon EC2 Cloud” which shows the process of starting up EC2 nodes that automatically configure themselves as new UniCluster grid clients.

That video is available at: <http://www.screencast.com/t/RUFRVKfSvgr>

Other recorded sessions:

- [“ClusterExpress build on an EC2 node”](#) (*long*)
- [“Demo: Globus GSISSE Usage”](#)
- [“Demo: GridFTP with globus-url-copy”](#)
- [“Demo: UniCluster Monitor Console”](#)

# Public AMI Image with UniCluster & Deployment Scripts

The exact AMI used by BioTeam in these experiments can not be shared because it contains our built-in AWS access and billing credentials as well as a UniCluster deployment script that points directly at master node "cloudseeder.bioteam.net".

To make our work easier to replicate, we've taken the private AMI image and stripped out BioTeam specific AWS access identifiers. We've also edited the auto-deployment scripts to remove our AWS keys and any hard coded pointers to "cloudseeder.bioteam.net"

---

This new public AMI is available as: **ami-aeaf4bc7**.

---

AMI description:

```
dag$ ec2-describe-images ami-aeaf4bc7
IMAGE    ami-aeaf4bc7
PUBLIC   UniCluster-EC2/32bit-UniCluster-3.2-loginNode.manifest.xml
609971441117   available   public           i386   machine
```

This AMI closely matches the server image used in creating this document.

Of note:

- The UniCluster software distribution can be found in /opt/unicluster-express/
- All deployment helper scripts discussed here can be found in /opt/univa-prep-scripts/ - they will need editing to match specific deployment requirements.
- If you make changes and want to create a new private bundle for your organization to use there is an example bundling, upload and register script located at /root/make-clusterExpress-bundle.sh
- It is not expected that this AMI would be used for serious work or production purposes. It should be used as an experimental testbed and possible jumping off point for creating private custom AMIs

Comments and feedback can be addressed to [chris@bioteam.net](mailto:chris@bioteam.net).

## Acknowledgements

We would like to thank the following Univa UD staff for providing excellent technical assistance:

**Siniša Vesel** -- Siniša was singlehandedly responsible for correctly diagnosing GridFTP problems caused by EC2 compute nodes being hidden behind a NAT gateway, the ("-data-interface") parameter fix solved a globus-url-copy problem that had been vexing us for a significant amount of time.

**Cameron Brunner** -- Cameron knows the UniCluster product inside and out and gamely answered all of our technical and not-so-technical questions. Although the details did not end up in this paper, Cameron provided fascinating technical information on the ARCo integration and how the Monitoring Console can query across multiple reporting sources and unify the results.



Howto: UniCluster & Amazon EC2 by Chris Dagdigan is licensed under a Creative Commons Attribution-Share Alike 3.0 United States License.  
Based on a work at [grid.org](http://grid.org).  
Permissions beyond the scope of this license may be available at <http://grid.org>