



# **Magnet™ Mobile App Server Deployment Guide**

## **2.0**

**Revision A**

© 2013 Magnet Systems, Inc. All rights reserved. This manual, in whole or in part, may not be reproduced, translated, or reduced to any machine-readable form without the prior written approval of Magnet Systems, Inc. Magnet Systems, Inc. reserves the right to make any modification to this manual or the information contained herein at any time without notice.

MAGNET SYSTEMS PROVIDES NO WARRANTY WITH REGARD TO THIS MANUAL, THE SOFTWARE, OR OTHER INFORMATION CONTAINED HEREIN AND HEREBY EXPRESSLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE WITH REGARD TO THIS MANUAL, THE SOFTWARE, OR SUCH OTHER INFORMATION. IN NO EVENT SHALL MAGNET SYSTEMS, INC. BE LIABLE FOR ANY INCIDENTAL, CONSEQUENTIAL, OR SPECIAL DAMAGES, WHETHER BASED ON TORT, CONTRACT, OR OTHERWISE, ARISING OUT OF OR IN CONNECTION WITH THIS MANUAL, THE SOFTWARE, OR OTHER INFORMATION CONTAINED HEREIN OR THE USE THEREOF.

All other trademarks are trademarks of their respective owners. Any Magnet Systems patents granted or pending in the United States and other countries are protected by law.

**Magnet Systems, Inc.**

435 Tasso Street  
Suite 100  
Palo Alto, CA  
94301

650-329-5904

[info@magnet.com](mailto:info@magnet.com)

<b>1. About This Guide .....</b>	<b>1</b>
Intended Audience .....	1
Document Organization .....	1
Related Documents .....	2
<b>2. Introducing Magnet Mobile App Server.....</b>	<b>3</b>
Magnet Mobile Enterprise Server .....	4
Custom Controllers .....	4
Caching and Off-line Message Delivery .....	4
Persistence .....	4
Transactions .....	5
Service Integration .....	5
Third-Party Service Controllers .....	5
Magnet Mobile App Manager .....	6
Mobile App Management Console .....	6
Mobile App Store .....	6
Deployment .....	6
Developer Factory .....	7
<b>3. Getting Started .....</b>	<b>9</b>
Checking your Environment .....	9
Understanding the aws-server Directory Structure .....	9
Gathering Resources .....	11
Creating Relational Database Store .....	11
<b>4. Assembling the AMI .....</b>	<b>13</b>
Creating a New Base AMI from Scratch .....	13
Running a Shell Script .....	16
Launching a New Instance with AMI .....	17
Installing the Application Bits .....	18

---

<b>5. Using a Cloud Formation Template .....</b>	<b>19</b>
Using CloudFormationBasic.template .....	19
Using AutoScalingGroupsWithRDS.template .....	19
Creating a Stack .....	20
Creating a Relational Database Service (RDS) .....	23



The Magnet™ Mobile App Server Deployment Guide provides information for deploying your Magnet Mobile App Server projects onto the Amazon cloud.

## Intended Audience

This guide is intended for administrators and developers who deploy the server-side projects to the Amazon cloud. This document assumes that you understand the data model and control flow used by the Java objects. It also assumes that you have a understanding of the following:

- The Amazon Web Service Console
- Basic Linux System Administration
- Basic Shell Programming
- Use of SSH and SCP
- Use of Maven to perform a build

## Document Organization

This guide includes these chapters:

- *Introducing Magnet Mobile App Server*, provides a high-level description of each component, its relationship, and interaction between the components.
- *Getting Started*, provides information that you need to gather before deploying your project.
- *Assembling the AMI*, provides instructions for building a complete AMI that includes the operating system, Java, your application binaries, and the scripts used to start them
- *Using a Cloud Formation Template*, provides information and instructions for using two templates and creating a stack as well as creating a Rational Database Service.

## Related Documents

The following table lists and describes other documents that are related to the Magnet products.

Document Title	Description
Magnet™ Android Developer Guide	Intended for Android app developers, this guide provides information and code samples for building enterprise native Android apps. It also provides instructions to upload the completed Android app to your private sandbox for testing.
Magnet™ iOS Developer Guide	Intended for iOS app developers, this guide provides information and code samples for building enterprise native iOS apps. It also provides instructions to upload the completed iOS app to your private sandbox for testing.
Magnet™ Mobile App Server Application Developer Guide	Intended for developers who write Java-based server-side application, this guide provides information and code samples for building Magnet Enterprise Server applications.
Magnet™ Mobile App Management Console Administrator Guide	Intended for IT administrators, this guide provides information and instructions for managing apps using the Magnet Mobile App Management Console.
Magnet™ Mobile for Android User Guide	Intended for users of Android devices, this guide provides information and instructions for installing the Magnet Mobile Server and using Apps@Work.
Magnet™ Mobile for iOS User Guide	Intended for users of iOS devices, this guide provides information and instructions for installing Setup@Work and using Apps@Work.

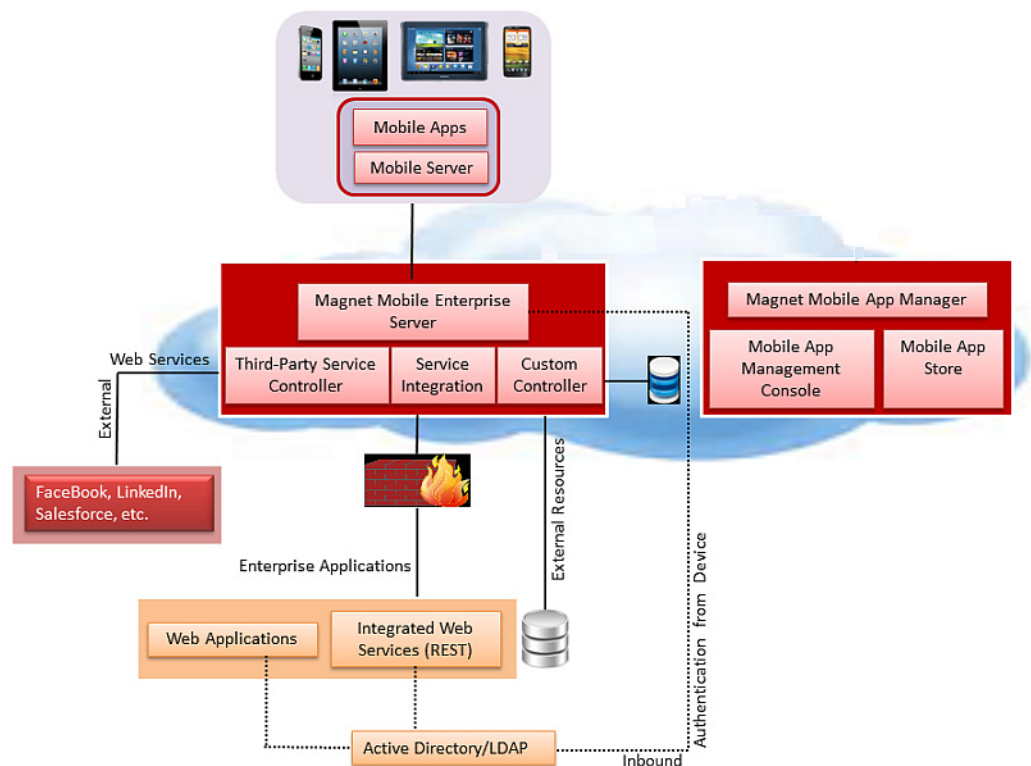


## 2. Introducing Magnet Mobile App Server

The Magnet™ Mobile App Server provides basic constructs for creating and manipulating app objects spanning the server and mobile app contexts. The Magnet Mobile App Server:

- Enables data flow between users' devices and enterprise back-end systems, as well as monitoring usage of mobile apps.
- Supports performance, availability, scalability, and reporting.
- Ensures that security-sensitive data is encrypted locally on device storage.

The following diagram shows the Magnet Mobile App Server components and their relationships.



## Magnet Mobile Enterprise Server

Magnet Mobile Enterprise Server (MES) is a server that contains all business logic components in a single deployable JAR file. Each encapsulation of business logic is herein referred to as a Controller. Instances of the Magnet Mobile Enterprise Server are deployed in the cloud, and may contain a blend of off-the-shelf (OTS) and custom controllers that you write for your app.

### Custom Controllers

Should you want to create your own custom controllers, you can download the source code for a reference server controller implementation. You can then modify this reference controller implementation to add your own business logic, then upload the modified project for your custom controller back into the Magnet Developer Factory so that this custom controller can be included into a build of the MES instance along with other controllers defined in your project. Custom controllers leverage services such as caching and off-line mode and persistence.

### Caching and Off-line Message Delivery

To deal with unreliable connections between mobile devices and MES and to ensure reliable message delivery, Magnet Mobile App Server supports off-line operations that store and cache messages, and forwards information when the network connection is reestablished.

You can invoke these asynchronous services locally on the server or remotely from a mobile client. The invocations from the mobile client can occur asynchronously and reliably (reliability is an option). They are asynchronous in that the invocation is recorded for later playback to the server. They are reliable in that after an invocation has been submitted, it is guaranteed to be run once—and only once from the server perspective.

After entries are processed and results arrive back from the server, a listener that you optionally provide will be notified. Listeners are either transient or durable. A transient listener will be forgotten if the app JVM instance is cycled for any reason. Durable (non-transient) listeners will survive JVM restarts.

### Persistence

Persistence is about the storage and retrieval of structured data. Magnet Mobile App Server provides support for entities and mapping entities to the data store. This creates, in effect, a “virtual entity data store” over any combination of MySQL, LDAP, and so forth, that can be used from within the programming interface Magnet provides.





Magnet Mobile App Server uses Java interfaces to define functions, methods, classes, and requests. Magnet Mobile App Server generates implementation from these classes built on annotations declared on interfaces, which allows for relationship, attributes, and so forth. Using user-friendly interfaces on entities, the Magnet Mobile App Server provides these features:

- Queries on entity with query composition
- CRUD (Create, Read, Update, Delete) operations on entities
- Complex queries such as paging and ordering

A developer can choose any technology (JDBC, Hibernate, etc.) to access a data store. However, implementing with Magnet persistence APIs is easier to use, and powerful over time. Every time an entity is used, unstructured data is collected. Over time, recommendations can be derived from that unstructured data. For example, when something is changed on an employee record, the implementation will track when that change is made and by whom. Of course, the recommendations will only be provided if you choose to incorporate those modules into your app. The entire methodology of building apps is modular - allowing you to blend the right combination of modules that make sense for your app.

## Transactions

Magnet Mobile App Server can optionally use transactions to maintain the integrity of data.

## Service Integration

Service integration can automatically transform Web applications from legacy enterprise application servers to the Magnet Mobile Enterprise Server. It exposes traditional SOAP-based Web Services hosted in the enterprise application servers to REST-based services for mobile access. Using the free on-line Developer Factory, you create a project with specific requirements, and download the generated APIs that abstract SOAP/REST services as controllers. Once generated, you can customize them as you see fit.

## Third-Party Service Controllers

The third-party service controllers are sample controllers that allow you to connect to well-known social services, such as Facebook, LinkedIn, or Salesforce. These controllers have built-in support for OAuth, so you can securely retrieve contacts information, and share an update. The source code for these controllers is included in the project generated by the Developer Factory if you choose to include them in your project.



## Magnet Mobile App Manager

The Magnet™ Mobile App Manager (MAM) is a run-time server that monitors the service status of Magnet MES and provides app management functionality by way of the Magnet Mobile App Management Console and Mobile App Store.

### Mobile App Management Console

The Magnet MAM Console is a Web interface of the Magnet Mobile App Manager that enables IT administrators to centrally manage and administer apps on client devices. Access to the MAM Console is defined by role-based and group association, which is defined in LDAP. Using the MAM Console, IT administrators can:

- Enable employee BYOD.
- Remotely install, remove, upgrade, and track apps.
- Customize an enterprise private-label app store.
- Remotely activate / deactivate user accounts.
- Push required apps to authorized users.
- View logs to monitor events history.

### Mobile App Store

The Mobile App Store is an app for installing and managing apps on a mobile device and is installed on iOS and Android devices. The Mobile App Store includes a suite of sample apps that are organized in categories. Via the MAM Console, IT administrators can brand the Mobile App Store with the enterprise private label. The App Store branding kit can be obtained on request.

## Deployment

An executable JAR file contains all controllers required for a specific deployment. Instances of the Magnet Mobile Enterprise Server are deployed in the Amazon cloud—a virtual private cloud that is managed by customers.



## Developer Factory

The Developer Factory is a free on-line service that provides a secure and private place for you to dynamically construct controller interfaces specific to the back-end services required for use in your mobile apps. The Developer Factory empowers you to

- Create and manage your app projects.
- Generate customized controllers and runtime binaries for your projects.
- Obtain source code for all your customized project assets.
- Upload modified controller and entities source codes for custom app controllers to regenerate executable JAR files.
- Obtain binaries for additional components required to deploy your projects in the Amazon cloud. Automatically create other language packs of your controllers (for example, iOS/XCode).
- Test-drive your app projects by getting access to your own private sandbox environment in the cloud.





Now that you have successfully built and tested your Magnet Mobile App Server application, it is time to go live and deploy it to the Amazon cloud. Magnet provides a set of tools and instructions on how to create an Amazon Web Services(t) deployment. When you use the Magnet Developer Factory to create your server application, you are provided with a separate server-aws Maven sub project that includes a set of scripts and templates. You can use these to build an environment that leverages AWS services such as EC2, RDS and VPC to achieve a highly reliable and scalable deployment of your Magnet based application.

### Checking your Environment

The following items are supported for deploying the server to Amazon:

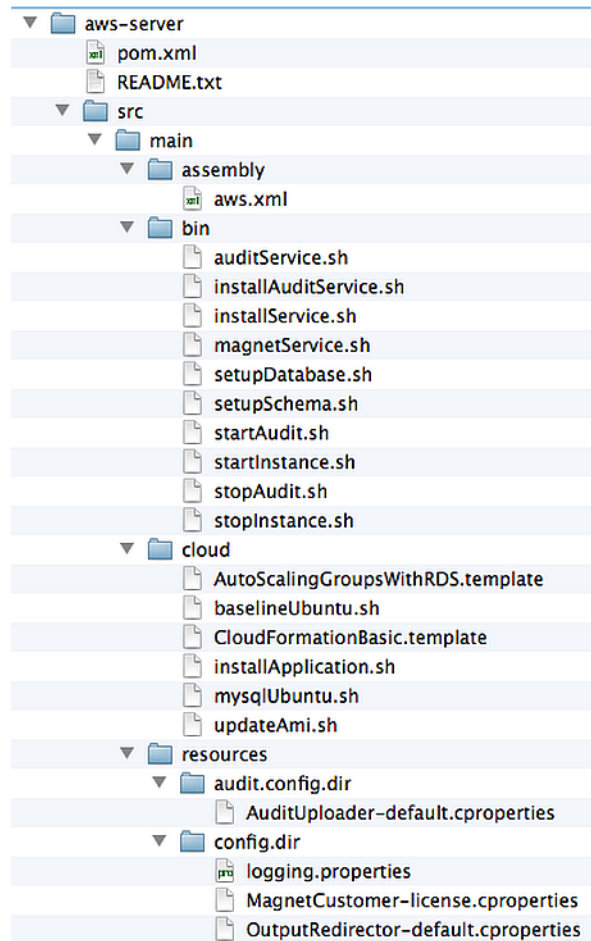
- Ubuntu/Linux 12.04 LTS (64bit)
- Oracle JDK 7

### Understanding the aws-server Directory Structure

The *aws-server* includes directories containing files that you can use to fine-tune your deployment (see the following diagram).

- The *aws.xml* file uses the Maven plugin to build a zip file that is used to integrate your application with an existing Amazon AMI. You can modify data in this file for your own environment.
- The *bin* directory includes standard shell scripts that you can use to start and stop your server instance.
  - *installService.sh* integrates your server process with the Host OS so that it is started and stopped at the same time.
  - *magnetService.sh* starts an instance of the process.
  - *startAudit.sh* starts a process that uploads audit data to an s3 bucket for purposes of licensing.
  - *startInstance.sh* starts an instance of your server project with a customizable set of options.

- *stopInstance.sh* stops your instance by sending it a SIGINT and waiting for it to gracefully shutdown.



- The *cloud* directory includes standard shell scripts and templates you can use to quickly jump start your production environment.
- The *audit.config.dir* includes the *AuditUploader-default.cproperties* file that contains the location and customer keys where the audit files are delivered.
- The *config.dir* directory includes the following cproperties files:
  - *logging.properties* contains “production” logging configuration options to managed a file rotation.
  - *MagnetCustomer-license.cproperties* contains your user key and signed license file.
  - *OutputRedirector.cproperties* causes the stdout and stderr to be redirected to the server log files.



## Gathering Resources

Assemble other required production configuration parameters, such as LDAP server locations.

## Creating Relational Database Store

Many Magnet application requires access to a relational database store (RDS). Magnet recommends using the example template as a starting point. You need to create an RDS to host your database. For instructions and to download the RDS\_VPC.template, visit [https://s3.amazonaws.com/cloudformation-templates-us-east-1/RDS\\_VPC.template](https://s3.amazonaws.com/cloudformation-templates-us-east-1/RDS_VPC.template).

When the process is complete, make note of the URL that is provided.







## 4. Assembling the AMI

For purposes of building a highly scalable and reliable environment, Magnet recommends building a complete AMI that includes the operating system, Java, your application binaries, and the scripts used to start them. The image that we end up creating will result in your application being up and ready to accept requests as soon as it is started.

### Creating a New Base AMI from Scratch

- 1 Launch a new EC2 instance that you will provision with all the required software.

**Create a New Instance** [Cancel]

Select an option below:

- ☐ **Classic Wizard**  
Launch an On-Demand or Spot instance using the classic wizard with fine-grained control over how it is launched.
- ☒ **Quick Launch Wizard**  
Launch an On-Demand instance using an editable, default configuration so that you can get started in the cloud as quickly as possible.
- ☐ **AWS Marketplace**  
AWS Marketplace is an online store where you can find and buy software that runs on AWS. Launch with 1-Click and pay by the hour.

**Name Your Instance:** MagnetBaseline

Pick a meaningful name, e.g. Web Server

**Choose a Key Pair:**  
Public/private key pairs allow you to securely connect to your instance after it launches.

☒ **Select Existing** ☐ **Create New** ☐ **None**

TestKeyPair

**Choose a Launch Configuration:**

Linux 3.4, AWS tools, and repository access to multiple versions of MySQL, PostgreSQL, Python, Ruby, and Tomcat.	Free tier eligible
<b>Red Hat Enterprise Linux 6.4</b> Red Hat Enterprise Linux version 6.4, EBS-boot.	64 bit <input checked="" type="radio"/> 32 bit <input type="radio"/> Free tier eligible
<b>SUSE Linux Enterprise Server 11</b> SUSE Linux Enterprise Server 11 Service Pack 2 basic install, EBS boot with Amazon EC2 AMI Tools preinstalled; Apache 2.2, MySQL 5.0, PHP 5.3, and Ruby 1.8.7	64 bit <input checked="" type="radio"/> 32 bit <input type="radio"/> Free tier eligible
<b>Ubuntu Server 12.04.2 LTS</b> Ubuntu Server 12.04.2 LTS with support available from Canonical ( <a href="http://www.ubuntu.com/cloud/services">http://www.ubuntu.com/cloud/services</a> ).	64 bit <input checked="" type="radio"/> 32 bit <input type="radio"/> Free tier eligible
<b>Ubuntu Server 13.04</b> Ubuntu Server 13.04 with support available from Canonical ( <a href="http://www.ubuntu.com/cloud/services">http://www.ubuntu.com/cloud/services</a> ).	64 bit <input checked="" type="radio"/> 32 bit <input type="radio"/> Free tier eligible
<b>Cluster Compute Amazon Linux AMI 2013.03.1</b> The Amazon Linux AMI is an EBS-backed, HVM image. It includes Linux 3.4, AWS tools, and repository access to multiple versions of MySQL, PostgreSQL	64 bit <input checked="" type="radio"/> 32 bit <input type="radio"/> Free tier eligible

Note: You can customize your settings in the next step.

[Continue]

[Submit Feedback](#) [Getting Started Guide](#)

- a. Sign in to your AWS account, and select **Launch Instance** from the EC2 Management Console.
- b. Choose **Quick Launch Wizard**.
- c. Name your instance, for example, *MyAppBaseline*.

- d. Choose **Select Existing** under Choose a key pair, and select the KeyPair (for example, MyKeyPair) that you created earlier.



You will be prompted to create a new key pair if you did not set it up earlier. Repeat step 1.

- e. Select **Ubuntu 12.04.2 LTS** from the Choose a Launch Configuration list.
- f. Click **Continue**.

- 2 Click **Edit data** to customize your settings.

**Create a New Instance** [Cancel]

**Ubuntu Server 12.04.2 LTS (ami-fe002cbb)**  
Platform: Ubuntu Architecture: x86\_64  
Ubuntu Server 12.04.2 LTS with support available from Canonical (<http://www.ubuntu.com/cloud/services>).

Please review your settings and click **Launch** to finish or **Edit details** to make changes.

**Instance Details**

Name: MagnetBaseline	Type: t1.micro
Detailed Monitoring: No	Availability Zone: No preference
Shutdown Behaviour: Stop	Termination Protection: No
Launch into a VPC: No	

**Security Details**

Key Pair: TestKeyPair	Security Group: ssh_access
-----------------------	----------------------------

**Advanced Details**

Kernel ID: Default	Ramdisk ID: Default
User Data:	IAM Role: ?
Network Interfaces:	

[Go Back] [Edit details] [Launch]

- a. Select **t1.Micro** as the instance *Type* (you will be using this for configuration).
- b. Select the name of instance that you created, for example, *MyAppBaseline*.
- c. Select the KeyPair that you created, for example, *MyKeyPair*.
- d. Create a new Security Group, for example, *myapp\_access*, and add the following security rules:



You can search your public IP address by entering “what is my ip” in a search engine. You can use this address when creating the security rule.

- SSH if you want to configure this to be only for a subset of remote IPs.





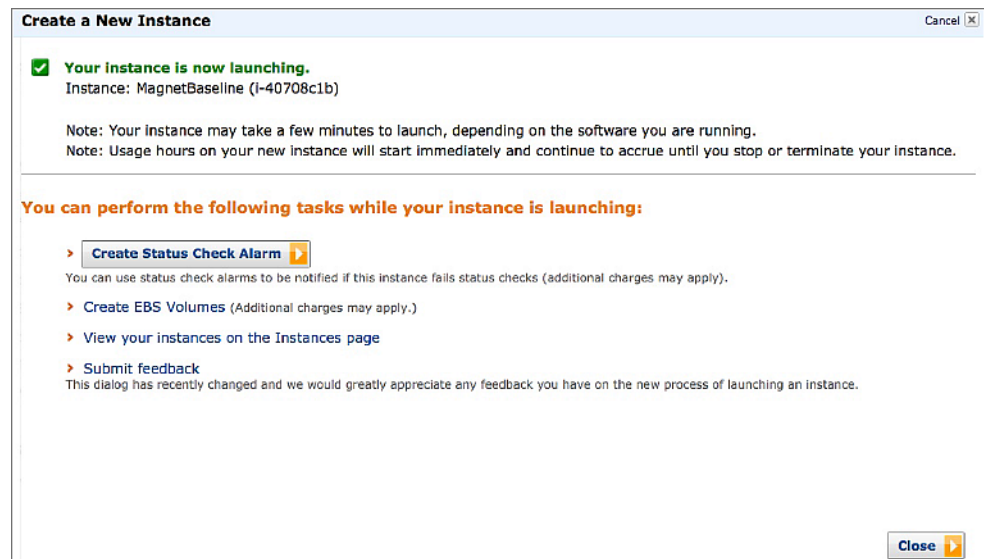
You can do this later by editing the security group.

- Custom: Enter a port for a Magnet server configuration if you want to change the default (Port Range 8080).

e. Click **Create**.

3 Click **Save** to store it as an AMI snapshot.

4 Click **Launch**.

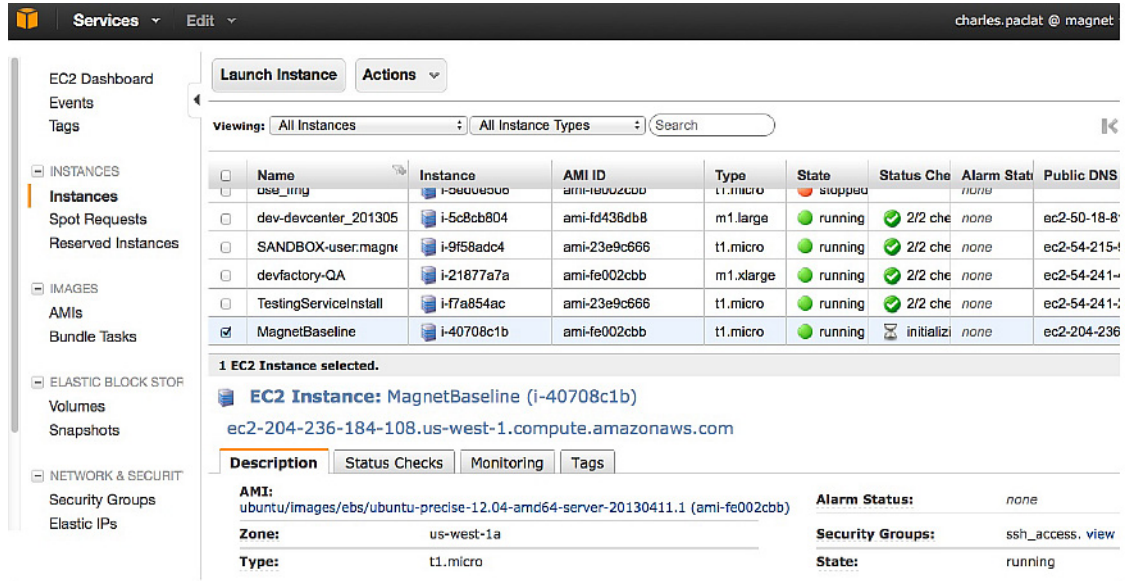


5 Navigate to the Instances page (Services) after the instance is launched.

6 Select your instance.

Your public DNS address are displayed, as shown in the following example.





Name	Instance	AMI ID	Type	State	Status Checks	Alarm Status	Public DNS
use_img	i-3e00e300	ami-18002cbb	t1.micro	stopped	2/2 checks passed	none	
dev-devcenter_201305	i-5c8cb804	ami-fd436db8	m1.large	running	2/2 checks passed	none	ec2-50-18-8
SANDBOX-user.magnet	i-9f58adc4	ami-23e9c666	t1.micro	running	2/2 checks passed	none	ec2-54-215-1
devfactory-QA	i-21877a7a	ami-fe002cbb	m1.xlarge	running	2/2 checks passed	none	ec2-54-241-1
TestingServiceInstall	i-f7a854ac	ami-23e9c666	t1.micro	running	2/2 checks passed	none	ec2-54-241-1
<b>MagnetBaseline</b>	<b>i-40708c1b</b>	<b>ami-fe002cbb</b>	<b>t1.micro</b>	<b>running</b>	<b>initializing</b>	<b>none</b>	<b>ec2-204-236</b>

**1 EC2 Instance selected.**

**EC2 Instance: MagnetBaseline (i-40708c1b)**  
[ec2-204-236-184-108.us-west-1.compute.amazonaws.com](https://ec2-204-236-184-108.us-west-1.compute.amazonaws.com)

**Description** | **Status Checks** | **Monitoring** | **Tags**

**AMI:** ubuntu/images/ebs/ubuntu-precise-12.04-amd64-server-20130411.1 (ami-fe002cbb) | **Alarm Status:** none

**Zone:** us-west-1a | **Security Groups:** ssh\_access, view

**Type:** t1.micro | **State:** running

7 Copy this value; you will need it to execute the script examples that follow.



Do not log out the account; you will need to come back to this page after running a shell script.

## Running a Shell Script

Now that you have a running instance, you can prepare it for deploying your application. To do this, you need access to a shell and to the key pair .pem file you created earlier, for example, **MyAppKeyPair.pem**.

```
export INSTANCE_NAME= The URL of my instance
export IDENTITY=MyKeyPair.pem
src/main/cloud/updateAmi.sh
src/main/cloud/installApplication.sh
```

The following example is created in Bourne Shell. You can cut and paste the following commands and run it from the root of your server project.



```
INSTANCE_NAME=ec2-204-236-184-108.us-west-1.compute.amazonaws.com
IDENTITY=MyAppKeyPair.pem

SCP_OPTS=-oStrictHostKeyChecking=no -i $IDENTITY
SSH_OPTS=-oStrictHostKeyChecking=no -i $IDENTITY ubuntu@$INSTANCE_NAME

ssh -i $IDENTITY ubuntu@$INSTANCE_NAME uname -a

# configure java, zip, and other useful utilities.
scp $SCP_OPTS src/main/cloud/baselineUbuntu.sh
ubuntu@$INSTANCE_NAME:baselineUbuntu.sh
ssh $SSH_OPTS chmod +x baselineUbuntu.sh
ssh $SSH_OPTS ./baselineUbuntu.sh

# for single instance deployments with a local mysql database.
scp $SCP_OPTS src/main/cloud/mysqlUbuntu.sh ubuntu@$INSTANCE_NAME:mysqlUbuntu.sh
ssh $SSH_OPTS chmod +x mysqlUbuntu.sh
ssh $SSH_OPTS ./mysqlUbuntu.sh
```

## Launching a New Instance with AMI

- 1 Go back to the Instances page.
- 2 Select the running instance.
- 3 Select Actions->Create Image (EBS AMI).
- 4 Name the AMI MyAppBaseline, and launch a new instance with that AMI when you need to install the new applications bits.



## Installing the Application Bits

After you have completed this step, Create another Snapshot of the “fully baked” server. This is the last Magnet image that contains all the bits needed as a reference image. If you need to spin up a new production server, you can refer to this image as the AMI-Base.

```
INSTANCE_NAME=_ENTER_YOUR_RUNNING_INSTANCE_PUBLIC_DNS
IDENTITY=ENTER_YOUR_RUNNING_INSTANCE_PUBLIC_DNS.pem

ARTIFACT_NAME='test-server-aws-1.0.0-deploy.zip'
TARGET_USER=magnet
TARGET_DIR=/usr/local/magnet

SCP_OPTS="-oStrictHostKeyChecking=no -i $IDENTITY"
SSH_OPTS="-oStrictHostKeyChecking=no -i $IDENTITY ubuntu@$INSTANCE_NAME"

# set up trust
ssh $SSH_OPTS uname -a
# copy the bits to the server
scp $SCP_OPTS target/$ARTIFACT_NAME ubuntu@$INSTANCE_NAME:$ARTIFACT_NAME

# unzip the bits and make the scripts executable and chown to the target user
ssh $SSH_OPTS sudo -u $TARGET_USER unzip -o -d $TARGET_DIR $ARTIFACT_NAME
ssh $SSH_OPTS sudo -u $TARGET_USER chmod +x $TARGET_DIR/bin/*

# install the start script as a service instance.
ssh $SSH_OPTS $TARGET_DIR/bin/setupDatabase.sh
ssh $SSH_OPTS sudo -u $TARGET_USER -i ./bin/setupSchema.sh

# install the start script as a service instance.
ssh $SSH_OPTS $TARGET_DIR/bin/installService.sh

# install the start auditing as a service.
ssh $SSH_OPTS $TARGET_DIR/bin/installAuditService.sh
```



Magnet provides two cloud formation templates that you can use to quickly jump start your production environment. This chapter provides information and instructions on the following topics:

- Using CloudFormationBasic.template (page **19**)
- Using AutoScalingGroupsWithRDS.template (page **19**)
- Creating a stack (page **20**)
- Creating a Rational Database Service (page **23**)

### Using CloudFormationBasic.template

This template is used in the Amazon—one Elastic Load Balancer (ELB) deployment. When this template is provided to the CloudFormation Tool, ELB acts as the entry point, and two servers being served by this Load Balancer are located in two different zones.

### Using AutoScalingGroupsWithRDS.template

This template is suited for a High-Availability environment with no single-point-of-failure (within the cut-over times provided by the Amazon environment). When the setup is complete, this template automatically creates a RDS MySQL with Multi-AZ deployment that will provide a hot database backup in a different AZ.

This template creates the following deployment in Amazon:

- One ELB with two different Auto Scaling Groups in two different regions.
- A Relational Database Service (RDS) based MySQL instance with Multi-AZ configuration On.
- Each AutoScaling Group has a minimum of 1 and a maximum of 3 instances (by default).

- The AutoScaling Groups have the following runtime characteristics:
  - Scale-up if CPU Utilization metric is > 90% for 60 seconds or, if data is missing for a continuous run of 3 evaluations of the entire group (up to a maximum of three instances per ASG).
  - Scale-down if CPU Utilization metric is < 15% for 60 seconds for a continuous run of 5 evaluations of the entire group (down to a minimum of 1 instance per ASG).
  - Cool down period of 300 secs between any two auto-scaling events to prevent trashing.

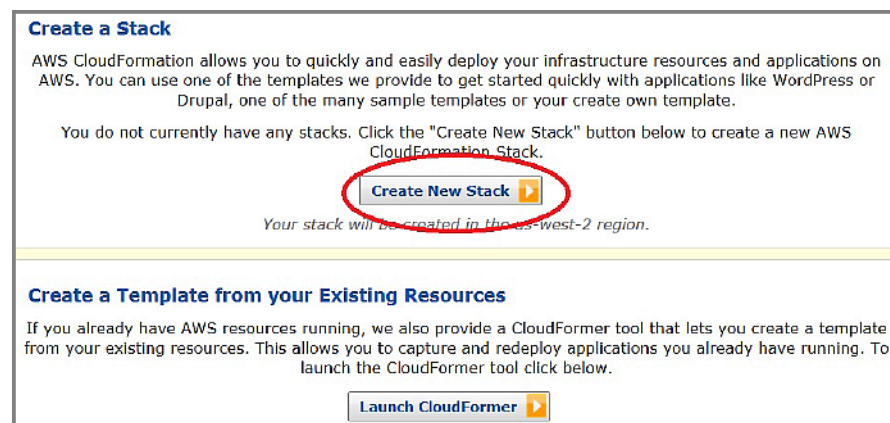
As detailed in the Amazon\_CloudFormer\_Createstack document, use Amazon Cloudformation tools and fill in the values according to your production needs and then trigger stack creation.

If the process completes successfully, you will have a Magnet production environment as described above. Once completed, ensure that the DB URL of the newly created MySQL database is put into the appropriate configuration file on each of the servers.

## Creating a Stack

Use the Amazon Cloud formation tools to create a stack that contains values specific to your production environment.

- 1 Create the Final Magnet Enterprise Server base AMI as described in *Installing the Application Bits* on page 18.
- 2 Select **Create New Stack** to start the stack wizard.





- 3 Enter the name of the stack, and upload the **CloudFormationBasic.template** file.

**Create Stack** [Cancel]

SELECT TEMPLATE | SPECIFY PARAMETERS | ADD TAGS | REVIEW

AWS CloudFormation gives you an easier way to create a collection of related AWS resources (a stack) by describing your requirements in a template. To create a stack, fill in the name for your stack and select a template. You may choose one of the sample templates to get started quickly, or one of your own templates stored in S3 or on your local hard drive.

**Stack Name:**  
MagnetProduction-CloudFormer Basic

**Template:**

☐ Use a sample template

☒ Upload a Template File

☐ Provide a Template URL

CloudFormationBasic.template

☐ Show Advanced Options

➔

- 4 Click **Continue**.

**Create Stack** [Cancel]

SELECT TEMPLATE | SPECIFY PARAMETERS | ADD TAGS | REVIEW

**Stack Description:** Magnet Enterprise Server Basic with - Elastic Load Balancer - Two instances should be in two different AZs in the same Region

**Specify Parameters**  
Below are the parameters associated with your CloudFormation template. You may review and proceed with the default parameters or make customizations as needed below.

**AvailabilityZone1** us-west-1a  
Name of availability zone in which to put Magnet server 1; Put 1 & 2 in different zones within the same region

**KeyName** MyKeyPair  
Name of an existing EC2 KeyPair to enable SSH access to the instances

**AMIID** ami-d8537a9d  
Name of an existing AMI from which to launch instances

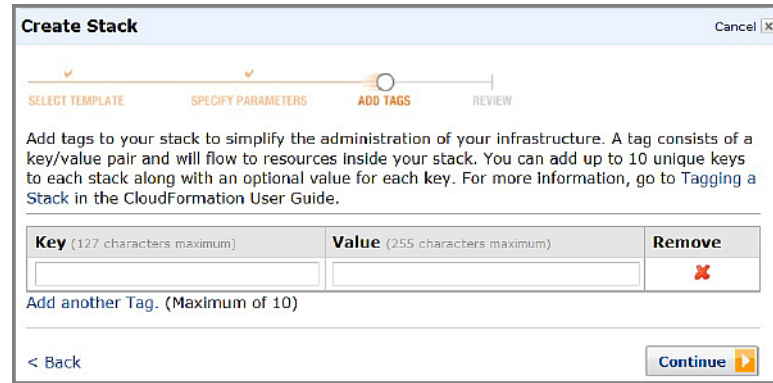
**AvailabilityZone2** us-west-1b  
Name of availability zone in which to put Magnet server 2; Put 1 & 2 in different zones within the same region

**InstanceType** m1.medium  
Magnet Server EC2 instance type

➔

- 5 Enter each parameter in its respective fields.
- 6 Click **Continue**.





**Create Stack** [Cancel] [X]

SELECT TEMPLATE | SPECIFY PARAMETERS | **ADD TAGS** | REVIEW

Add tags to your stack to simplify the administration of your infrastructure. A tag consists of a key/value pair and will flow to resources inside your stack. You can add up to 10 unique keys to each stack along with an optional value for each key. For more information, go to [Tagging a Stack](#) in the CloudFormation User Guide.

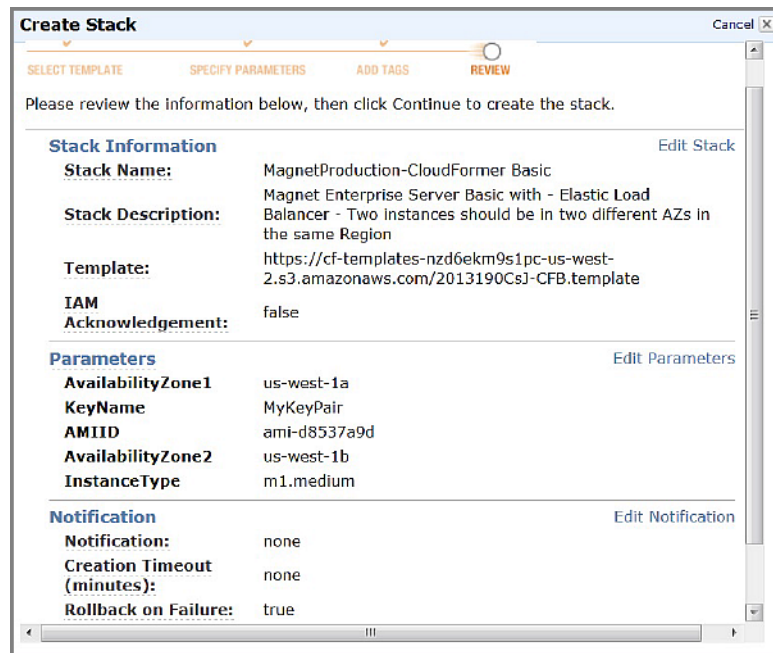
Key (127 characters maximum)	Value (255 characters maximum)	Remove
<input type="text"/>	<input type="text"/>	

[Add another Tag.](#) (Maximum of 10)

< Back Continue >

7 Add the key/value pair to the stack.

8 Click **Continue**.



**Create Stack** [Cancel] [X]

SELECT TEMPLATE | SPECIFY PARAMETERS | ADD TAGS | **REVIEW**

Please review the information below, then click Continue to create the stack.

**Stack Information** [Edit Stack](#)

**Stack Name:** MagnetProduction-CloudFormer Basic

**Stack Description:** Magnet Enterprise Server Basic with - Elastic Load Balancer - Two instances should be in two different AZs in the same Region

**Template:** <https://cf-templates-nzd6ekm9s1pc-us-west-2.s3.amazonaws.com/2013190CsJ-CFB.template>

**IAM Acknowledgement:** false

**Parameters** [Edit Parameters](#)

**AvailabilityZone1:** us-west-1a

**KeyName:** MyKeyPair

**AMIID:** ami-d8537a9d

**AvailabilityZone2:** us-west-1b

**InstanceType:** m1.medium

**Notification** [Edit Notification](#)

**Notification:** none

**Creation Timeout (minutes):** none

**Rollback on Failure:** true

9 Review the information.

- If all information are correct, click to create the deployment
- If you need to modify any information, click the appropriate **Edit**.

10 Continue to Create a Relational Database Service (RDS) based MySQL instance to host the Data that these two Servers need.

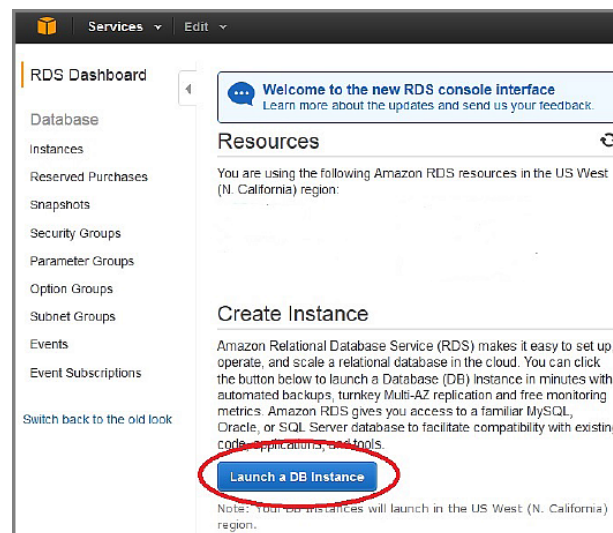


## Creating a Relational Database Service (RDS)

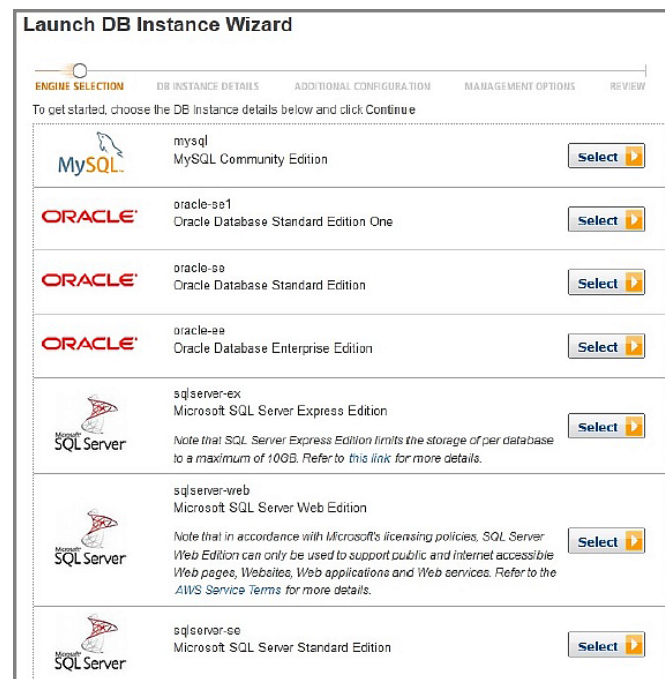
Once the set up has been created, Create a Relational Database Service (RDS) based MySQL instance to host the Data that these two Servers need.

Once you complete this MySQL setup, copy the URL pointing to the RDS instance and add it to the appropriate configuration files in the two servers created above (usually in MySQLJDBCPersist.cproperties). This ensures that the servers use the RDS MySQL as their Datastore.

- 1 Click **Launch a DB Instance** to start the DB Instance Wizard.



- 2 Select the MySQL instance that you will be using.



### 3 Click **Continue**.

**Launch DB Instance Wizard**

ENGINE SELECTION DB INSTANCE DETAILS ADDITIONAL CONFIGURATION MANAGEMENT OPTIONS REVIEW

To get started, choose a DB engine below and click **Continue**

DB Engine:   
License Model:   
DB Engine Version:   
DB Instance Class:   
Multi-AZ Deployment:   
Auto Minor Version Upgrade: ☒ Yes ☐ No

Provide the details for your RDS Database Instance.

Allocated Storage:  GB (Minimum: 5 GB, Maximum: 3072 GB) Higher allocated storage may improve IOPS performance.  
Use Provisioned IOPS: ☐  
DB Instance Identifier:  (e.g. mydbinstance)  
Master Username:  (e.g. awsuser)  
Master Password:  (e.g. mypassword)

[< Back](#) [Continue >](#)

### 4 Enter information in the appropriate fields, click **Continue**.

**Launch DB Instance Wizard**

ENGINE SELECTION DB INSTANCE DETAILS ADDITIONAL CONFIGURATION MANAGEMENT OPTIONS REVIEW

Provide the optional additional configuration details below.

Database Name:  (e.g. mydb)  
Note: If no database name is specified then no initial MySQL database will be created on the DB Instance.  
Database Port:   
Choose a VPC:  Only VPCs with a DB Subnet Group(s) are allowed  
Availability Zone:  Multi-AZ deployment selected  
Option Group:   
If you have custom DB Parameter Groups or DB Security Groups you would like to associate with this DB Instance, select them below, otherwise proceed with default settings.  
Parameter Group:   
DB Security Group(s):

[< Back](#) [Continue >](#)

### 5 Enter the name of the database and the database port in the respective fields, and select additional configuration.



6 Click **Continue**.

**Launch DB Instance Wizard**

ENGINE SELECTION DB INSTANCE DETAILS ADDITIONAL CONFIGURATION **MANAGEMENT OPTIONS** REVIEW

Enabled Automatic Backups: ☒ Yes ☐ No

The number of days for which automated backups are retained.

Please note that automated backups are currently supported for InnoDB storage engine only. If you are using MyISAM, refer to details [here](#).

Backup Retention Period: 1 days

The daily time range during which automated backups are created if automated backups are enabled

Backup Window: ☐ Select Window ☒ No Preference

The weekly time range (in UTC) during which system maintenance can occur.

Maintenance Window: ☐ Select Window ☒ No Preference

[Back](#) **Continue**

7 Select other options, click **Continue**.

**Launch DB Instance Wizard**

ENGINE SELECTION DB INSTANCE DETAILS ADDITIONAL CONFIGURATION **MANAGEMENT OPTIONS** REVIEW

Please review the information below, then click Launch DB Instance.

Engine: mysql  
 Engine Version: MySQL 5.6.31  
 License Model: general-public-license  
 Auto Minor Ver. Upgrade: Yes  
 DB Instance Class: db.t1.micro  
 Multi-AZ Deployment: Yes  
 Allocated Storage: 5  
 Provisioned IOPS: default  
 DB Instance Identifier: Test-DBInstance  
 Master User Name: management\_admin  
 Master User Password: \*\*\*\*\*

---

Database Name: management\_db  
 Database Port: 3306  
 Availability Zone: Using a Multi-AZ Deployment disables this preference.  
 Option Group: default:mysql-5-5  
 DB Parameter Group: default:mysql5.5  
 DB Security Group(s): default  
 DB Subnet Group:  
 Publicly Accessible: Only applicable with VPC

---

Backup Retention Period: 1  
 Backup Window: No Preference  
 Maintenance Window: No Preference

[Back](#) **Launch DB Instance**

8 Click **Launch DB Instance**.

