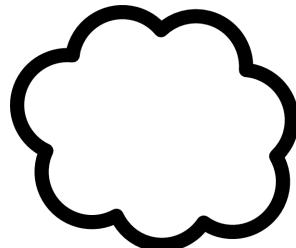
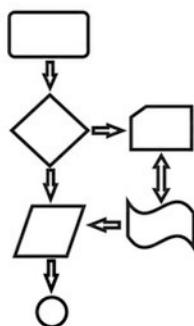


# San Jose State University

CMPE 281  
Cloud Technologies

AWS & Grails Lab



# Setup

# Step #0. Create Team Key Pairs

The screenshot shows the AWS EC2 Key Pairs management interface. On the left, a sidebar lists various services: EC2 Dashboard, Events, Tags, Reports, Limits, Instances, AMIs, Elastic Block Store, Network & Security, Auto Scaling, and Key Pairs (which is currently selected). The main content area has tabs for 'Create Key Pair' (selected), 'Import Key Pair', and 'Delete'. A search bar at the top says 'Filter by attributes or search by keyword' with a placeholder 'None found'. Below it, a dropdown menu allows filtering by 'Key pair name' or 'Fingerprint'. A message at the top right states 'An error occurred fetching key pair data: Request limit exceeded.' A modal window titled 'Create Key Pair' is open in the center, prompting for a 'Key pair name' which is set to 'team-0-keypair'. It includes 'Cancel' and 'Create' buttons. The background shows a table with columns for 'Name', 'Fingerprint', and 'Actions'.

# Step #1. Setup VPC

VPC with a Single Public Subnet,

Screenshot of the AWS VPC setup wizard Step 1: Select a VPC Configuration.

The screenshot shows the AWS navigation bar at the top with options like AWS, Services, Edit, and user information (Paul Nguyen, Oregon, Support). The main content area is titled "Step 1: Select a VPC Configuration".

A sidebar on the left lists four configuration options:

- VPC with a Single Public Subnet** (selected)
- VPC with Public and Private Subnets
- VPC with Public and Private Subnets and Hardware VPN Access
- VPC with a Private Subnet Only and Hardware VPN Access

The selected configuration ("VPC with a Single Public Subnet") is described as follows:

Your instances run in a private, isolated section of the AWS cloud with direct access to the Internet. Network access control lists and security groups can be used to provide strict control over inbound and outbound network traffic to your instances.

**Creates:**

A /16 network with a /24 subnet. Public subnet instances use Elastic IPs or Public IPs to access the Internet.

A "Select" button is located next to the description.

To the right, there is a diagram illustrating the setup:

The diagram shows a central box labeled "Amazon Virtual Private Cloud" containing several overlapping squares. Below this box is the label "Public Subnet". Above the VPC box is a cloud-like shape containing the text "Internet, S3, DynamoDB, SNS, SQS, etc.". A line connects the VPC box to the cloud shape.

**Cancel and Exit** and **Feedback** buttons are located at the bottom of the page.

# Step #1. Setup VPC

VPC with a Single Public Subnet,

Screenshot of the AWS VPC setup wizard Step 2: VPC with a Single Public Subnet.

The form fields are as follows:

- IP CIDR block\*: 10.0.0.0/16 (65531 IP addresses available)
- VPC name: team-0-vpc
- Public subnet\*: 10.0.0.0/24 (251 IP addresses available)
- Availability Zone\*: No Preference
- Subnet name: Public subnet
- You can add more subnets after AWS creates the VPC.
- Enable DNS hostnames\*:  Yes  No
- Hardware tenancy\*: Default

Buttons at the bottom:

- Cancel and Exit
- Back
- Create VPC (highlighted in blue)

Page footer:

© 2008 - 2015, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy Policy](#) [Terms of Use](#)

[Feedback](#)

# Step #1. Setup VPC

VPC with a Single Public Subnet,

The screenshot shows the AWS VPC Dashboard. At the top, there's a navigation bar with icons for AWS, Services, and Edit, along with user information for Paul Nguyen, Oregon, and Support. The main area has a sidebar on the left with links like VPC Dashboard, Filter by VPC (set to None), Virtual Private Cloud (Your VPCs, Subnets, Route Tables, Internet Gateways, DHCP Options Sets, Elastic IPs, Peering Connections), Security (Network ACLs, Security Groups), and VPN Connections (Customer Gateways, Virtual Private Gateways, VPN Connections). A central modal window titled "VPC Successfully Created" contains the message: "Your VPC has been successfully created. You can launch instances into the subnets of your VPC. For more information, see [Launching an Instance into Your Subnet](#)". An "OK" button is at the bottom right of the modal. At the bottom of the page, there's a footer with copyright information: "© 2008 - 2015, Amazon Web Services, Inc. or its affiliates. All rights reserved." and links to Privacy Policy and Terms of Use, along with a Feedback button.

<http://docs.aws.amazon.com/AmazonVPC/latest/GettingStartedGuide/ExerciseOverview.html>

# Step #2. Deploy Tomcat VM

AWS Marketplace

Amazon Web Services Home

Your Account | Help | Sell on AWS Marketplace

Shop All Categories ▾ Search AWS Marketplace GO Your Software

## Tomcat powered by Bitnami

Sold by: BitRock Inc.



Bitnami Tomcat Stack greatly simplifies the development and deployment of applications based on Tomcat. It includes ready-to-run versions of Apache, MySQL, Tomcat, Java and all of the other software required to run each of those components. The Bitnami Tomcat Stack is completely integrated and configured, so you'll be ready to start developing your application as soon as the AMI is launched onto Amazon EC2. Bitnami Tomcat Stack is regularly updated to make sure that you always have access to the latest stable releases of each of the bundled components. Note that this image does not support HVM ... [Read more](#)

**Customer Rating** ★★★★★ (5 Customer Reviews)

**Latest Version** 7.0.59-0 on Ubuntu 14.04.1 (Other available versions)

**Base Operating System** Linux/Unix, Ubuntu 14.04.1

**Delivery Method** 64-bit Amazon Machine Image (AMI) ([Learn more](#))

**Support** See details below

**AWS Services Required** Amazon EC2, Amazon EBS

**Highlights**

- Ready to run Tomcat development environment
- Includes Apache, MySQL, and Java, as well as all other required dependencies
- Regularly updated with the latest stable release of each component

**Product Description**

Bitnami Tomcat Stack greatly simplifies the development and deployment of applications based on Tomcat. It includes ready-to-run versions of Apache, MySQL, Tomcat, Java and all of the other software required to run each of those components. The Bitnami Tomcat Stack is completely integrated and configured, so you'll be ready to start developing your application as soon as the AMI is launched onto Amazon EC2. Bitnami Tomcat Stack is regularly updated to make sure that you always have access to the latest stable releases of each of the bundled components. Note that this image does not support HVM instances. Bitnami also publishes images that support the HVM format. Please look for the same application from Bitnami with '(HVM)' in the listing title to use those images

**Continue**

You will have an opportunity to review your order before launching or being charged.

**Pricing Details**

For region: US East (N. Virginia)

**Free Tier Eligible**

EC2 charges for Micro instances are free for up to **750 hours** a month if you qualify for the [AWS Free Tier](#). See [details](#).

**Hourly Fees**

Total hourly fees will vary by instance type and EC2 region.

EC2 Instance Type	EC2 Usage	Software	Total
t1.micro	\$0.02/hr	\$0.00/hr	<b>\$0.02/hr</b>
m1.small	\$0.044/hr	\$0.00/hr	<b>\$0.044/hr</b>
m1.medium	\$0.087/hr	\$0.00/hr	<b>\$0.087/hr</b>
m1.large	\$0.175/hr	\$0.00/hr	<b>\$0.175/hr</b>
m1.xlarge	\$0.35/hr	\$0.00/hr	<b>\$0.35/hr</b>
m2.xlarge	\$0.245/hr	\$0.00/hr	<b>\$0.245/hr</b>
m2.2xlarge	\$0.49/hr	\$0.00/hr	<b>\$0.49/hr</b>
m2.4xlarge	\$0.98/hr	\$0.00/hr	<b>\$0.98/hr</b>
c1.medium	\$0.13/hr	\$0.00/hr	<b>\$0.13/hr</b>
c1.xlarge	\$0.52/hr	\$0.00/hr	<b>\$0.52/hr</b>
hi1.4xlarge	\$3.10/hr	\$0.00/hr	<b>\$3.10/hr</b>
hs1.8xlarge	\$4.60/hr	\$0.00/hr	<b>\$4.60/hr</b>
m3.medium	\$0.07/hr	\$0.00/hr	<b>\$0.07/hr</b>
m3.large	\$0.14/hr	\$0.00/hr	<b>\$0.14/hr</b>

# Step #2. Deploy Tomcat VM

The screenshot shows the AWS Marketplace interface for deploying a Tomcat VM. At the top, there's a navigation bar with the AWS Marketplace logo, user information (Hello, Paul Nguyen. (Sign out)), and links to Amazon Web Services Home, Your Account, Help, and Sell on AWS Marketplace. Below the navigation is a search bar with 'Search AWS Marketplace' and a 'GO' button. A message box indicates a new version (7.0.59-0) is available on Ubuntu 14.04.1.

**Launch on EC2:**

**Tomcat powered by Bitnami**

**1-Click Launch**  
Review, modify, and launch

**Manual Launch**  
With EC2 Console, APIs or CLI

**Click "Launch with 1-Click" to launch this software with the settings below**

The default settings are provided by the software seller and AWS Marketplace.

**Version**  
7.0.59-0 on Ubuntu 14.04.1, released 02/25/2015

**Region**  
US West (Oregon)

**EC2 Instance Type**

m1.small  
m1.medium  
m1.large  
m1.xlarge  
m2.xlarge  
m2.2xlarge  
m2.4xlarge  
c1.medium  
c1.xlarge

	Memory	1.7 GiB
CPU	1 EC2 Compute Unit (1 virtual core with 1 EC2 Compute Unit)	
Storage	1 x 160 GB	
Platform	64-bit	
Network performance	Low	
API Name	m1.small	

**Price for your selections:**

**\$0.04 / hour**  
\$0.04 m1.small EC2 Instance usage fees +  
\$0.00 hourly software fee

**\$0.10 / GB / month**  
EBS General Purpose (SSD)

**Launch with 1-Click**

**Cost Estimator**

**\$31.68 / month**  
m1.small EC2 Instance usage fees  
Assumes 24 hour use over 30 days

**Software Charges**

**\$0.00 / month**  
\$0.00 hourly software fees for m1.small

**AWS Infrastructure Charges**

**\$31.68 / month**

# Step #2. Deploy Tomcat VM

**VPC Settings**

VPC  
vpc-277acf42 (10.0.0.0/16)

Or Create a VPC

Subnet  
subnet-fdbc7ca4 (10.0.0.0/24) us-west-2c Name

Or Create a subnet

\* indicates a default vpc or subnet

**Security Group**

**Updated:** Due to a change in other settings, security group settings is updated.

A security group acts as a firewall that controls the traffic allowed to reach one or more instances. Learn more about [Security Groups](#).

You can create a new security group based on seller-recommended settings or choose one of your existing groups.

Create new based on seller settings

Description:  
A new security group will be generated by AWS Marketplace. It is based on recommended settings for Tomcat powered by Bitnami version 7.0.59-0 on Ubuntu 14.04.1 provided by BitRock Inc..

Connection Method	Protocol	Port Range	Source (IP or Group)
SSH	tcp	22 - 22	Anywhere <input type="button" value=""/> 0.0.0.0/0
HTTP	tcp	80 - 80	Anywhere <input type="button" value=""/> 0.0.0.0/0
HTTPS	tcp	443 - 443	Anywhere <input type="button" value=""/> 0.0.0.0/0

**Warning**  
Rules with source of 0.0.0.0/0 allows all IP addresses to access your instance. We recommend limiting access to only known IP addresses.

**Key Pair**

team-0-keypair

To ensure that no other person has access to your software, the software installs on an EC2 instance with an EC2 key pair that you created. Choose an existing EC2 key pair in the list.

Cost varies for storage fees  
\$31.68 hourly EC2 Instance fees for m1.small  
Varied EBS Storage and data transfer fees

# Step #2. Deploy Tomcat VM

An instance of this software is now deploying on EC2.

- If you would like to check the progress of this deployment, go to the [AWS Management Console](#).
- The software will be ready in 2-3 minutes.

**Usage Instructions**

Once the instance is running, you can connect to your using your Amazon private key and the 'bitnami' login via SSH. You can find all the Stack components like Apache, MySQL, etc and the required dependencies in the '/opt/bitnami' folder. Please check our documentation at... [Show more](#)

**Software Installation Details**

<b>Product</b>	Tomcat powered by Bitnami
<b>Version</b>	7.0.59-0 on Ubuntu 14.04.1, released 02/25/2015
<b>Region</b>	US West (Oregon)
<b>EC2 Instance Type</b>	m1.small
<b>VPC</b>	vpc-277acf42
<b>Subnet</b>	subnet-fdbc7ca4
<b>Security Group</b>	Tomcat powered by Bitnami-7-0-59-0 on Ubuntu 14-04-1-AutogenByAWSMP-
<b>Key Pair</b>	team-0-keypair

# Step #2. Deploy Tomcat VM

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with navigation links like EC2 Dashboard, Events, Tags, Reports, Limits, Instances, Spot Requests, Reserved Instances, AMIs, Bundle Tasks, Volumes, Snapshots, Security Groups, Elastic IPs, Placement Groups, Load Balancers, Key Pairs, Network Interfaces, Launch Configurations, and Auto Scaling Groups. The 'Instances' link is currently selected.

The main area displays a table of instances. One instance, 'team-0-tomcat' (Instance ID: i-55ab399c), is highlighted with a blue selection bar. The table columns include Name, Instance ID, Instance Type, Availability Zone, Instance State, Status Checks, and Alarm Status. The instance state for 'team-0-tomcat' is 'running'. Below the table, a detailed view for the selected instance is shown. It includes fields for Description, Status Checks, Monitoring, Tags, Usage Instructions, Instance ID, Instance state, Instance type, Private DNS, Private IPs, Public DNS, Public IP, Elastic IP, Availability zone, and Security groups. The security group description mentions 'Tomcat powered by Bitnami-7-0-59-0 on Ubuntu 14-04-1- AutogenByAWSMP- . view rules'.

At the bottom, there are sections for Secondary private IPs and Scheduled events, both of which show 'No scheduled events'.

Page footer: © 2008 - 2015, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Feedback

# Step #3. Setup Elastic IP

The screenshot shows the AWS EC2 Dashboard with the 'Allocate New Address' button highlighted. A modal dialog box is open, asking 'Are you sure you want to allocate a new IP address?'. The 'Yes, Allocate' button is also highlighted.

AWS Services Edit Paul Nguyen Oregon Support

EC2 Dashboard Events Tags Reports Limits

INSTANCES Instances Spot Requests Reserved Instances

IMAGES AMIs Bundle Tasks

ELASTIC BLOCK STORE Volumes Snapshots

NETWORK & SECURITY Security Groups Elastic IPs Placement Groups Load Balancers Key Pairs Network Interfaces

AUTO SCALING Launch Configurations Auto Scaling Groups

Allocate New Address Release Addresses Associate Address Disassociate Address

Filter by attributes or search by keyword None found

You do not have any elastic IPs in this region.

Click on the "Allocate New Address" button to allocate your first elastic IP.

Allocate New Address

Allocate New Address

Are you sure you want to allocate a new IP address?

Select an address above

Cancel Yes, Allocate

Feedback

# Step #3. Setup Elastic IP

The screenshot shows the AWS EC2 Dashboard with the 'Elastic IPs' section selected. The left sidebar lists various EC2 services. The main pane displays a table of elastic IP addresses, with the first address, 52.11.45.67, selected. A detailed view panel on the right shows the following information for the selected address:

Elastic IP	Instance	Private IP Address	Scope
52.11.45.67			vpc
52.11.16.45			vpc
52.11.37.149			vpc
52.11.37.230			vpc

**Address: 52.11.45.67**

Elastic IP	52.11.45.67	Network interface ID	-
Instance		Private IP address	-
Scope	vpc	Network interface owner	-
Public DNS	-	Allocation ID	eipalloc-6907d70c

At the bottom, there is a footer with copyright information and a feedback link.

# Step #3. Setup Elastic IP

The screenshot shows the AWS EC2 Dashboard with the 'Associate Address' tab selected. A modal dialog box titled 'Associate Address' is open, prompting the user to select an instance or network interface to associate the IP address with. The 'Instance' section is active, showing a search bar and a list of running instances:

- i-12ab39db (running)
- i-985a5494 (team-12-tomcat) (running)
- i-d15856dd (running)
- i-0c585600 (running)
- i-e9a93b20 (running)
- i-55ab399c (team-0-tomcat) (running)
- i-6cab39a5 (team-17-tomcat) (running)
- i-81c4288c (running)
- i-a3c428ae (running)
- i-5f565853 (running)

An orange warning box states: "Warning: If you associate an Elastic IP address with multiple instances, traffic will be distributed across them. Learn more about public IP addresses." At the bottom right of the dialog are 'Cancel' and 'Associate' buttons. The background shows the EC2 dashboard with various navigation links like EC2 Dashboard, Events, Tags, Reports, Limits, and Auto Scaling.

# Step #3. Setup Elastic IP

The screenshot shows the AWS EC2 service interface. In the top navigation bar, 'Services' is selected. Below it, the 'Associate Address' tab is active. A search bar and filter options for 'Elastic IP', 'Instance', 'Private IP Address', and 'Scope' are visible. A modal window titled 'Associate Address' is open, prompting the user to select an instance or network interface to associate the IP address 52.11.45.67. The 'Instance' field contains 'i-55ab399c'. An 'Or' option allows selecting a 'Network Interface' via a search bar. A 'Private IP Address' dropdown is set to '10.0.0.54\*'. A checkbox for 'Reassociation' is present. A warning message states: 'If you associate an Elastic IP address with your instance, your current public IP address is released. Learn more about [public IP addresses](#)'. At the bottom right of the modal are 'Cancel' and 'Associate' buttons. The main EC2 dashboard shows sections for EC2 Dashboard, Events, Tags, Reports, Limits, Instances, Images, AMIs, Bundles, Elastic IP, Volumes, Snapshots, Networks, Security Groups, Elastic Load Balancing, Places, Load Balancers, Key Pairs, and Networks. The 'Associations' section shows one entry: 'Allocation ID: eipalloc-6907d70c'. The bottom of the page includes a footer with copyright information and links to Privacy Policy and Terms of Use, along with a 'Feedback' button.

# Step #3. Setup Elastic IP

The screenshot shows the AWS EC2 Dashboard with the 'Elastic IPs' section selected. The left sidebar lists various EC2 services: EC2 Dashboard, Events, Tags, Reports, Limits, Instances, Images, AMIs, Bundle Tasks, Elastic Block Store, Volumes, Snapshots, Network & Security (Security Groups, Elastic IPs, Placement Groups, Load Balancers, Key Pairs, Network Interfaces), and Auto Scaling (Launch Configurations, Auto Scaling Groups). The 'Elastic IPs' link is highlighted with an orange vertical bar.

The main content area displays a table of elastic IP addresses. The columns are: Select (checkbox), Elastic IP, Instance, Private IP Address, and Scope. The table contains the following data:

Select	Elastic IP	Instance	Private IP Address	Scope
<input type="checkbox"/>	52.11.68.222			vpc
<input type="checkbox"/>	52.11.45.67	i-55ab399c (team-0-tomcat)	10.0.0.54	vpc-277acf42
<input type="checkbox"/>	52.11.59.153			vpc
<input type="checkbox"/>	52.11.16.45			vpc
<input type="checkbox"/>	52.11.37.230			vpc
<input type="checkbox"/>	52.11.37.149			vpc
<input type="checkbox"/>	52.11.67.96			vpc
<input type="checkbox"/>	52.11.67.98			vpc

Below the table, a message says "Select an address above". There are three small icons at the bottom right of the table area.

# Step #3. Setup Elastic IP / Test DNS

ec2-52-11-45-67.us-west-2.compute.amazonaws.com

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with navigation links like EC2 Dashboard, Events, Tags, Reports, Limits, Instances, Images, and Auto Scaling. The 'Instances' link is highlighted. The main area displays a table of instances. One instance, 'team-0-tomcat' (i-55ab399c), is selected and highlighted with a blue border. Below the table, detailed information for this instance is shown:

Description	Value
Instance ID	i-55ab399c
Instance state	running
Instance type	m1.small
Private DNS	ip-10-0-0-54.us-west-2.compute.internal
Private IPs	10.0.0.54
Secondary private IPs	
VPC ID	vpc-277acf42
Public DNS	ec2-52-11-45-67.us-west-2.compute.amazonaws.com
Public IP	52.11.45.67
Elastic IP	52.11.45.67
Availability zone	us-west-2c
Security groups	Tomcat powered by Bitnami-7-0-59-0 on Ubuntu 14-04-1-AutogenByAWSMP-. view rules
Scheduled events	No scheduled events
AMI ID	bitnami-tomcatstack-7.0.59-0-linux-ubuntu-14.04.1-x86_64-ebs-mp-5e86feb4-015f-4f67-b6dc-52bef2088612-ami-c485cfac.2 (ami-3b53720b)

# Step #3. Setup Elastic IP / Test DNS

The screenshot shows a web browser window with the URL `ec2-52-11-45-67.us-west-2.compute.amazonaws.com` in the address bar. The page content is as follows:

**Congratulations!**

You are now running Bitnami [Tomcat 7.0.59-0](#) in the Cloud.

**Access my application**

**Access phpMyAdmin**

**Bitnami Wiki**

**Bitnami Forums**

The default application administrator is "manager". Please check [our documentation](#) to learn how to get your password.

Want more from the cloud?

Bitnami Cloud Hosting simplifies the process of **deployment** and **managing** web applications on the **Amazon Cloud**. It provides:

- Automatic backups
- Multi-app deployments
- One click server resizing
- Control over the disk size
- Apache and MySQL Monitoring
- and more!

**Get Started**

# Step #4. SSH into VM

ec2-52-11-45-67.us-west-2.compute.amazonaws.com

The screenshot shows the AWS Management Console with the EC2 service selected. A modal dialog box titled "Connect To Your Instance" is open in the center. The dialog contains instructions for connecting to an instance using an SSH client or Java SSH Client. It provides steps for finding the private key file and connecting via IP address. An example command is shown, along with a note about the default AMI username. A "Close" button is at the bottom right. The background shows a list of instances on the right and navigation menus on the left.

EC2 Dashboard

Events

Tags

Reports

Limits

INSTANCES

Instances

Spot Requests

Reserved Instances

IMAGES

AMIs

Bundle Tasks

ELASTIC BLOCK STORE

Volumes

Snapshots

NETWORK & SECURITY

Security Groups

Elastic IPs

Placement Groups

Load Balancers

Key Pairs

Network Interfaces

AUTO SCALING

Launch Configurations

Auto Scaling Groups

Paul Nguyen | Oregon | Support

Launch Instance Connect Actions

1 to 24 of 24

Connect To Your Instance

I would like to connect with:

A standalone SSH client

A Java SSH Client directly from my browser (Java required)

To access your instance:

1. Open an SSH client. (find out how to [connect using PuTTY](#))
2. Locate your private key file (`team-0-keypair.pem`). The wizard automatically detects the key you used to launch the instance.
3. Your key must not be publicly viewable for SSH to work. Use this command if needed:  
`chmod 400 team-0-keypair.pem`
4. Connect to your instance using its Elastic IP:  
**52.11.45.67**

Example:

```
ssh -i team-0-keypair.pem ubuntu@52.11.45.67
```

Please note that in most cases the username above will be correct, however please ensure that you read your AMI usage instructions to ensure that the AMI owner has not changed the default AMI username.

If you need any assistance connecting to your instance, please see our [connection documentation](#).

Close

Feedback

# Step #4. SSH into VM

ec2-52-11-45-67.us-west-2.compute.amazonaws.com

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with navigation links like EC2 Dashboard, Events, Tags, Reports, Limits, Instances, Images, and Auto Scaling. The 'Instances' link is highlighted. The main area displays a table of instances. One instance, 'team-0-tomcat' (ID: i-55ab399c), is selected and highlighted with a blue border. A context menu is open over this instance, with 'Instance Settings' selected. The menu includes options like Connect, Get Windows Password, Launch More Like This, Instance State, Instance Settings (selected), Image, Networking, CloudWatch Monitoring, Add/Edit Tags, Attach to Auto Scaling Group, Change Instance Type, Change Termination Protection, View/Change User Data, Change Shutdown Behavior, and Get System Log.

Description	Status Checks	Monitoring	Tags	Usage Instructions
Instance ID: i-55ab399c				Public DNS: ec2-52-11-45-67.us-west-2.compute.amazonaws.com
Instance state: running				Public IP: 52.11.45.67
Instance type: m1.small				Elastic IP: 52.11.45.67
Private DNS: ip-10-0-0-54.us-west-2.compute.internal				Availability zone: us-west-2c
Private IPs: 10.0.0.54				Security groups: Tomcat powered by Bitnami-7-0-59-0 on Ubuntu 14-04-1-AutogenByAWSMP-. view rules
Secondary private IPs				Scheduled events: No scheduled events
VPC ID: vpc-277acf42				AMI ID: bitnami-tomcatstack-7.0.59-0-linux-ubuntu-14.04.1-x86_64-ebs-mp-5e86feb4-015f-4f67-b6dc-52bef2088612-ami-

© 2008 - 2015, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy Policy](#) [Terms of Use](#) [Feedback](#)

# Step #4. SSH into VM

ec2-52-11-45-67.us-west-2.compute.amazonaws.com

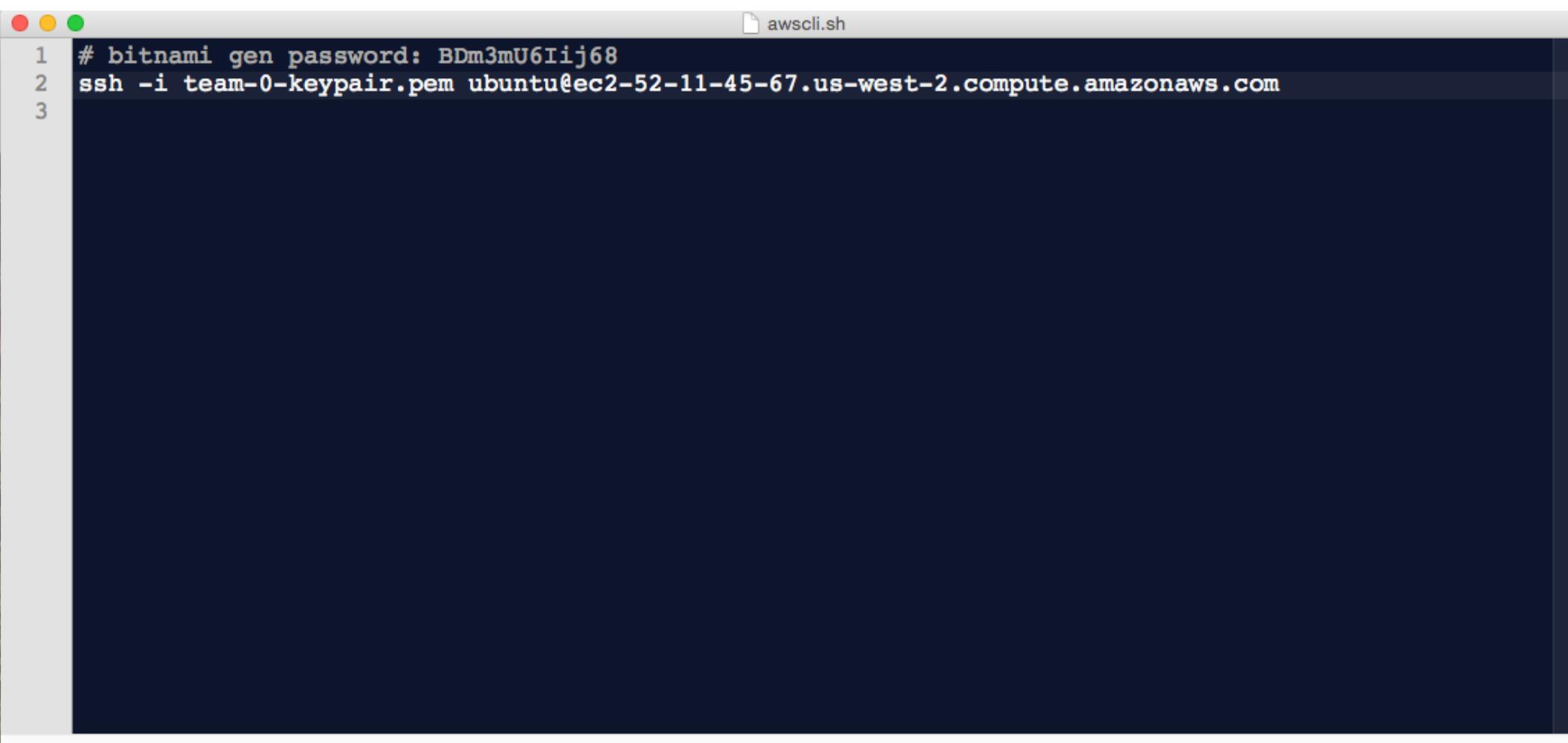
The screenshot shows the AWS EC2 Dashboard with the 'Instances' section selected. A modal window titled 'System Log: i-55ab399c (team-0-tomcat)' is open, displaying a terminal log. The log output is as follows:

```
The filesystem is already 2621440 blocks long. Nothing to do!
Generating locales...
  en_US.UTF-8... up-to-date
Generation complete.
small
[Sat Feb 28 03:22:33 UTC 2015] Regenerating keys for *
/opt/bitnami/var/init/pre-start/070_regenerate_keys: 22: /opt/bitnami/var/init/pre-start/070_regen
[Sat Feb 28 03:22:33 UTC 2015] Regenerating keys for * finished
[Sat Feb 28 03:22:33 UTC 2015] Finished regenerating keys
#####
#      Setting Bitnami application password to 'BDm3mU6Iij68'
#
150228 03:22:35 mysqld_safe Logging to '/opt/bitnami/mysql/data/mysqld.log',
150228 03:22:35 mysqld_safe Starting mysqld bin daemon with databases from /opt/bitnami/mysql/d
* Stopping Handle applying cloud-config[74G[ OK ]
/opt/bitnami/mysql/scripts/ctl.sh : mysql started at port 3306
[Sat Feb 28 03:22:42 UTC 2015] Setting up password for apache-tomcat service
[Sat Feb 28 03:22:56 UTC 2015] Setting up password for apache-tomcat service finished
[Sat Feb 28 03:22:56 UTC 2015] Setting up password for mysql service
150228 03:23:09 mysqld_safe mysqld from pid file /opt/bitnami/mysql/data/mysqld.pid ended
[Sat Feb 28 03:23:24 UTC 2015] Setting up password for mysql service finished
/opt/bitnami/mysql/scripts/ctl.sh : mysql (pid 2326) already running
[Sat Feb 28 03:23:24 UTC 2015] Setting up password for * application
/opt/bitnami/var/init/pre-start/100_set_default_passwords: 55: /opt/bitnami/var/init/pre-start/100_
[Sat Feb 28 03:23:24 UTC 2015] Setting up password for * application finished
[Sat Feb 28 03:23:24 UTC 2015] Finished setting password
/opt/bitnami/mysql/scripts/ctl.sh : mysql (pid 2326) already running
/opt/bitnami/apache-tomcat/scripts/ctl.sh : tomcat started
Syntax OK
/opt/bitnami/apache2/scripts/ctl.sh : httpd started at port 80
curl: (28) Connection timed out after 5000 milliseconds
curl: (28) Connection timed out after 10000 milliseconds
```

The modal has a 'Close' button at the bottom right. The background shows a list of instances with names like 'ec2-52-11-45-67.us-west-2.compute.amazonaws.com' and their status.

# Step #4. SSH into VM

ec2-52-11-45-67.us-west-2.compute.amazonaws.com



A screenshot of a terminal window titled "awscli.sh". The window has three colored window control buttons (red, yellow, green) at the top left. The title bar also shows the file name "awscli.sh". The terminal content is a history of three commands:

```
1 # bitnami gen password: BDm3mU6Iij68
2 ssh -i team-0-keypair.pem ubuntu@ec2-52-11-45-67.us-west-2.compute.amazonaws.com
3
```

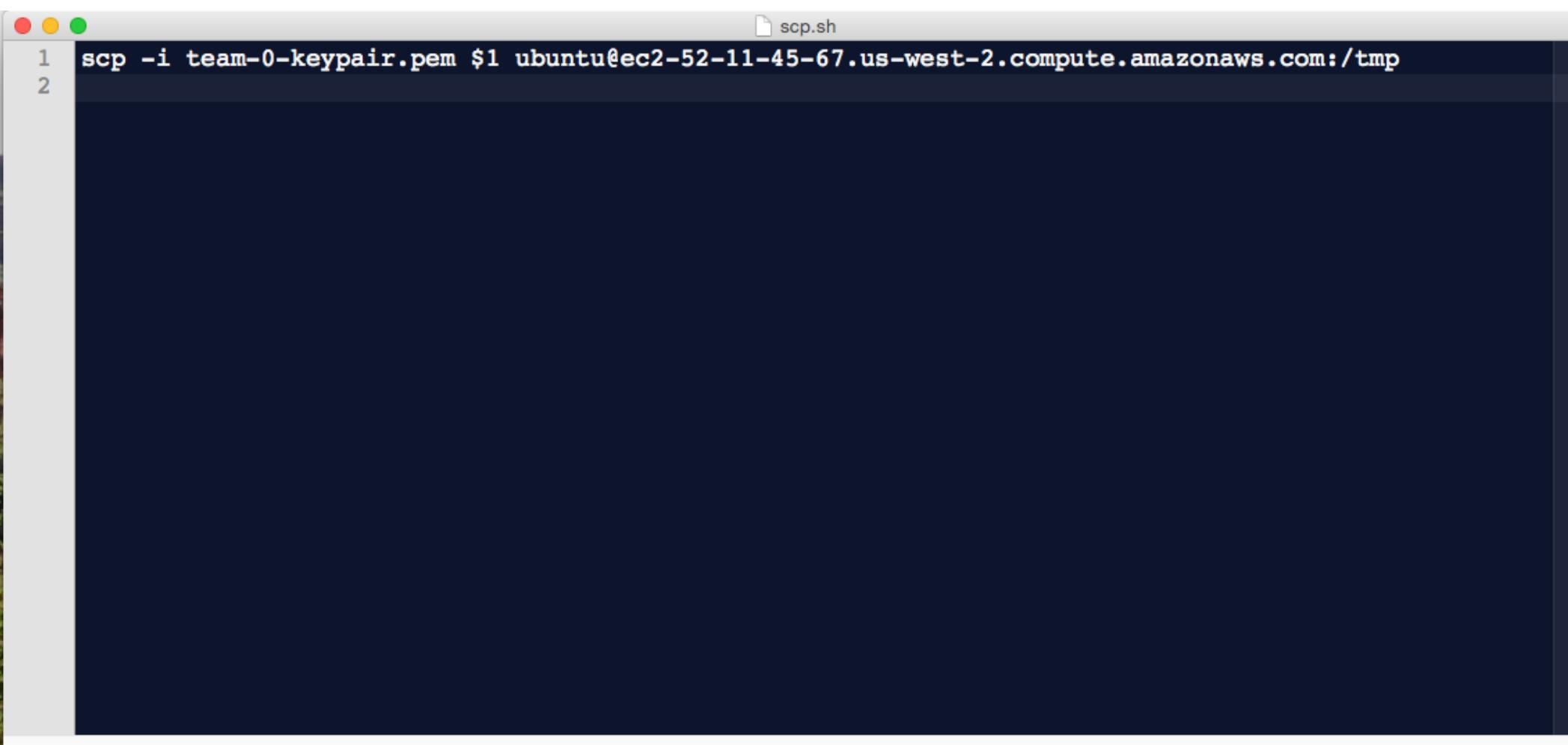
Line: 2 Column: 81 | Shell Script (Bash) | Tab Size: 4 | -

## Step #4. SSH into VM

ec2-52-11-45-67.us-west-2.compute.amazonaws.com

# Step #4. SSH into VM

ec2-52-11-45-67.us-west-2.compute.amazonaws.com



A screenshot of a terminal window titled "scp.sh". The window shows two lines of code:

```
1 scp -i team-0-keypair.pem $1 ubuntu@ec2-52-11-45-67.us-west-2.compute.amazonaws.com:/tmp  
2
```

Line: 2 Column: 1

L Shell Script (Bash)

Tab Size: 4

# Step #4. SSH into VM

ec2-52-11-45-67.us-west-2.compute.amazonaws.com



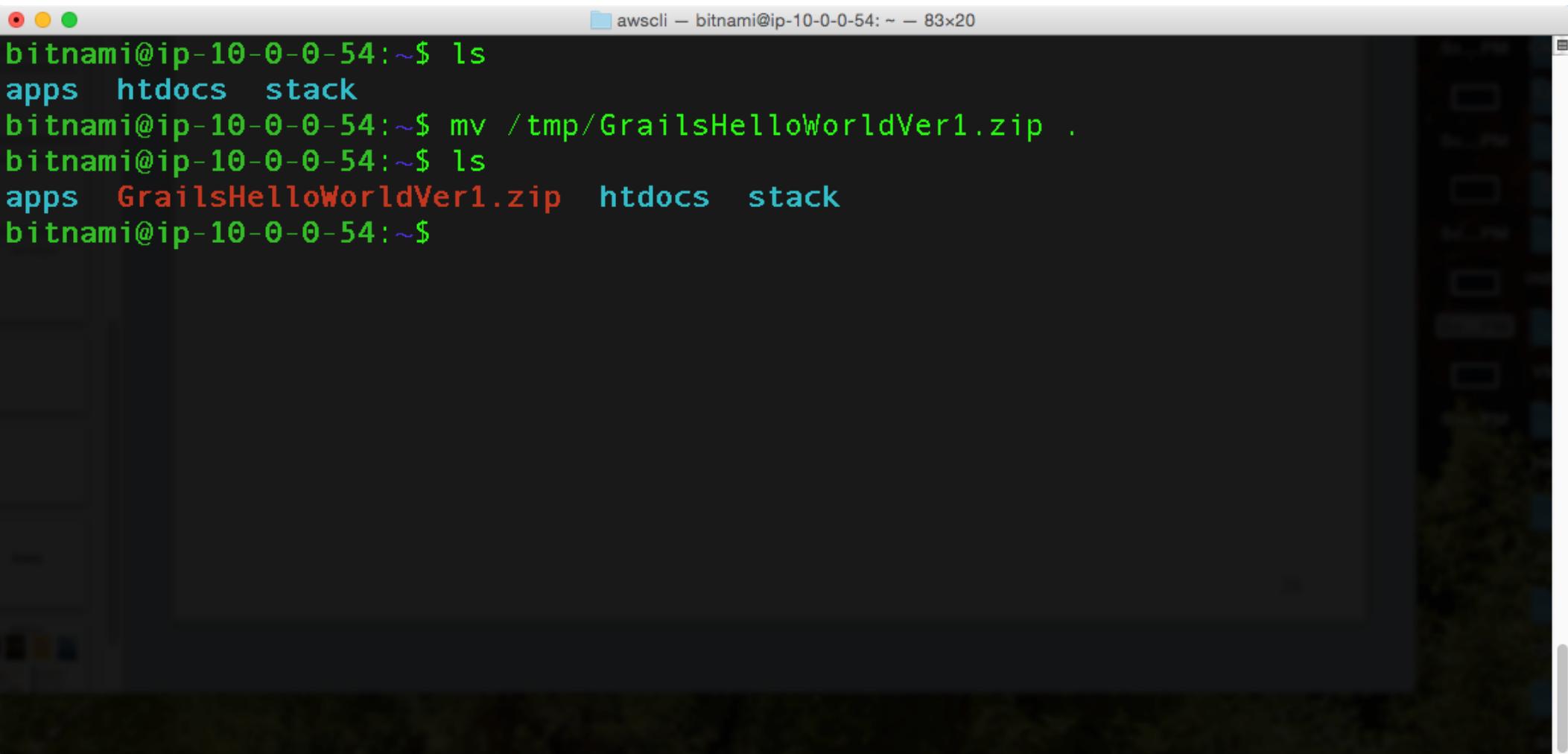
The screenshot shows a terminal window on a Mac OS X desktop. The window title is "awscli – bitnami@ip-10-0-0-54: ~ – 75x20". The terminal content is as follows:

```
dragon:awscli pnguyen$ ./scp.sh GrailsHelloWorldVer1.zip
GrailsHelloWorldVer1.zip                                100% 4254KB   4.2MB/s  00:01
dragon:awscli pnguyen$
```

The terminal window has a dark background with light-colored text. The window frame includes standard OS X window controls (red, yellow, green) and a scroll bar on the right side.

# Step #4. SSH into VM

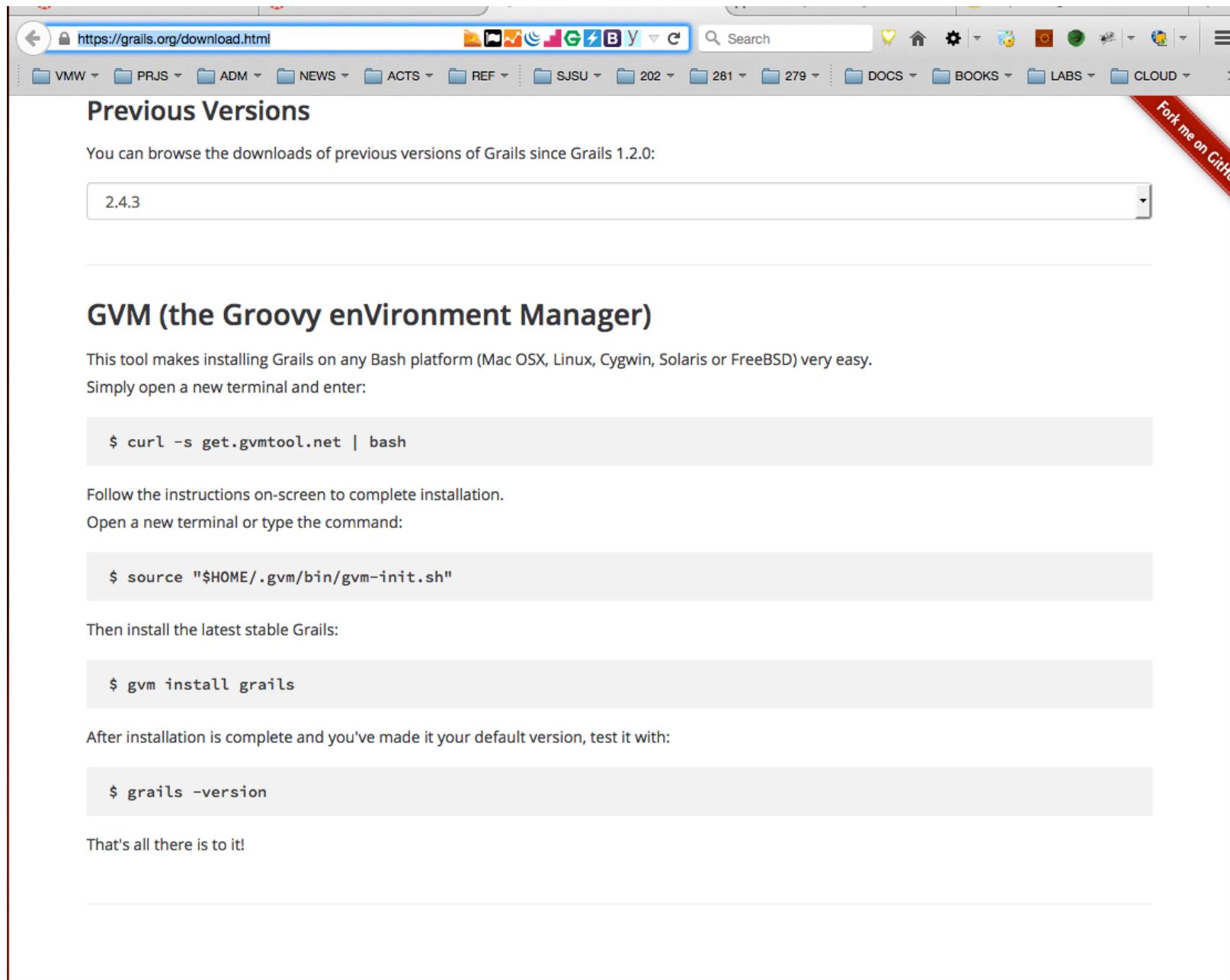
ec2-52-11-45-67.us-west-2.compute.amazonaws.com



```
bitnami@ip-10-0-0-54:~$ ls
apps  htdocs  stack
bitnami@ip-10-0-0-54:~$ mv /tmp/GrailsHelloWorldVer1.zip .
bitnami@ip-10-0-0-54:~$ ls
apps  GrailsHelloWorldVer1.zip  htdocs  stack
bitnami@ip-10-0-0-54:~$
```

# Step #5. Setup Grails on VM

<https://grails.org/download.html>



The screenshot shows a web browser window with the URL <https://grails.org/download.html> in the address bar. The page displays a "Previous Versions" section where users can browse downloads for previous versions of Grails since version 1.2.0. A dropdown menu is open, showing the option "2.4.3". On the right side of the page, there is a red ribbon banner with the text "Fork me on GitHub". The browser's toolbar and menu bar are visible at the top.

## Previous Versions

You can browse the downloads of previous versions of Grails since Grails 1.2.0:

2.4.3

## GVM (the Groovy enVironment Manager)

This tool makes installing Grails on any Bash platform (Mac OSX, Linux, Cygwin, Solaris or FreeBSD) very easy. Simply open a new terminal and enter:

```
$ curl -s get.gvmtool.net | bash
```

Follow the instructions on-screen to complete installation. Open a new terminal or type the command:

```
$ source "$HOME/.gvm/bin/gvm-init.sh"
```

Then install the latest stable Grails:

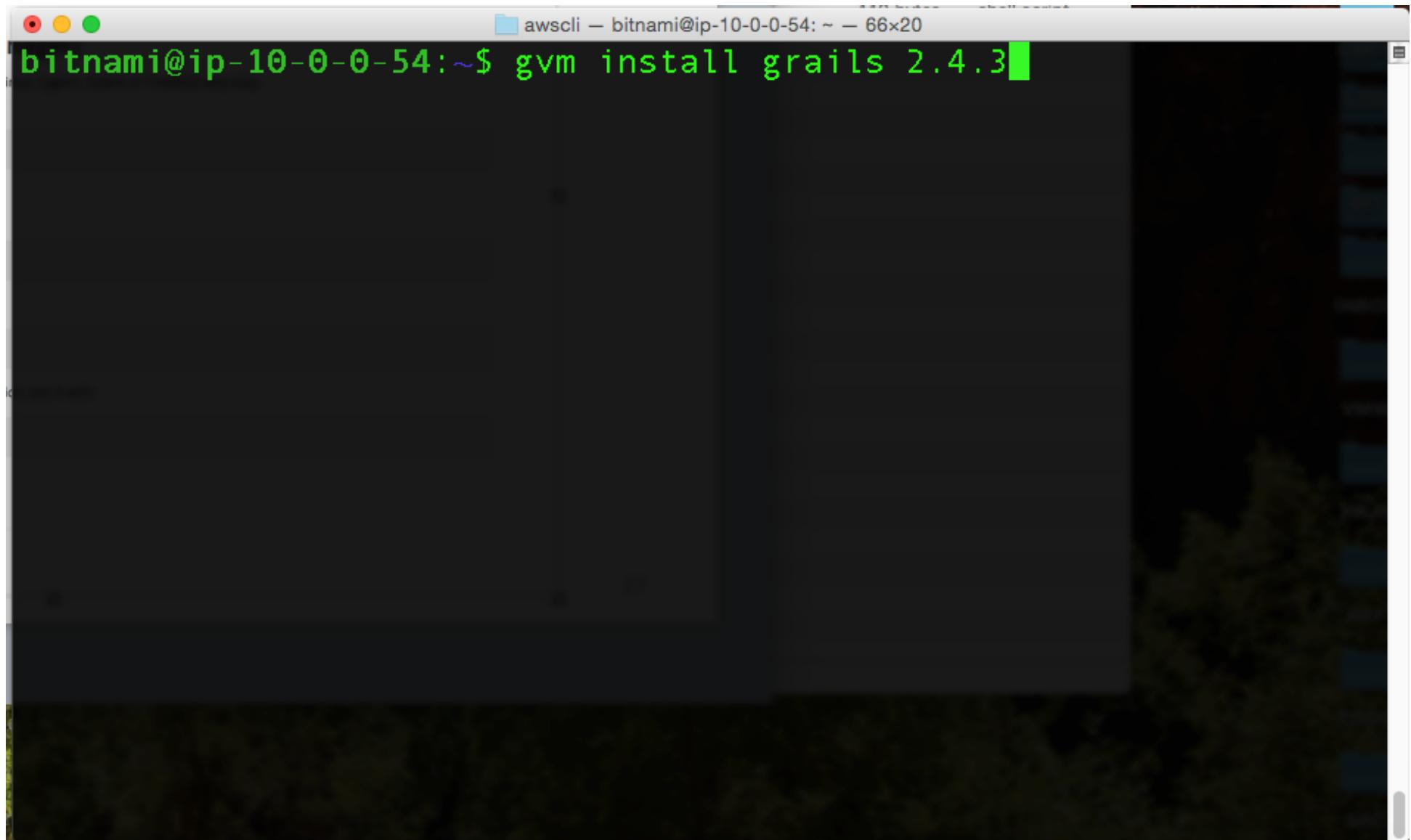
```
$ gvm install grails
```

After installation is complete and you've made it your default version, test it with:

```
$ grails -version
```

That's all there is to it!

# Step #5. Setup Grails on VM



A screenshot of a terminal window titled "awscli – bitnami@ip-10-0-0-54: ~ – 66x20". The window shows a command being typed: "bitnami@ip-10-0-0-54:~\$ gvm install grails 2.4.3". The background of the terminal is dark, and the text is white.

# Step #5. Setup Grails on VM

```
awscli — bitnami@ip-10-0-0-54: ~ — 66x20
76 126M 76 96.3M 0 0 3106k 0 0:00:41 0:00:31 0
77 126M 77 98.2M 0 0 3070k 0 0:00:42 0:00:32 0
80 126M 80 101M 0 0 3079k 0 0:00:41 0:00:33 0
83 126M 83 105M 0 0 3091k 0 0:00:41 0:00:34 0
85 126M 85 108M 0 0 3102k 0 0:00:41 0:00:35 0
88 126M 88 111M 0 0 3112k 0 0:00:41 0:00:36 0
91 126M 91 115M 0 0 3119k 0 0:00:41 0:00:37 0
93 126M 93 118M 0 0 3128k 0 0:00:41 0:00:38 0
96 126M 96 121M 0 0 3136k 0 0:00:41 0:00:39 0
99 126M 99 125M 0 0 3144k 0 0:00:41 0:00:40 0
100 126M 100 126M 0 0 3146k 0 0:00:41 0:00:41 --
:--:-- 3441k

Installing: grails 2.4.3
Done installing!

Do you want grails 2.4.3 to be set as default? (Y/n): Y
Setting grails 2.4.3 as default.
bitnami@ip-10-0-0-54:~$
```

# Step 6. Build and Deploy Grails Hello World

```
awscli - bitnami@ip-10-0-0-54: ~ - 72x24
0 3146k      0  0:00:41 0:00:41 --:--:-- 3441k

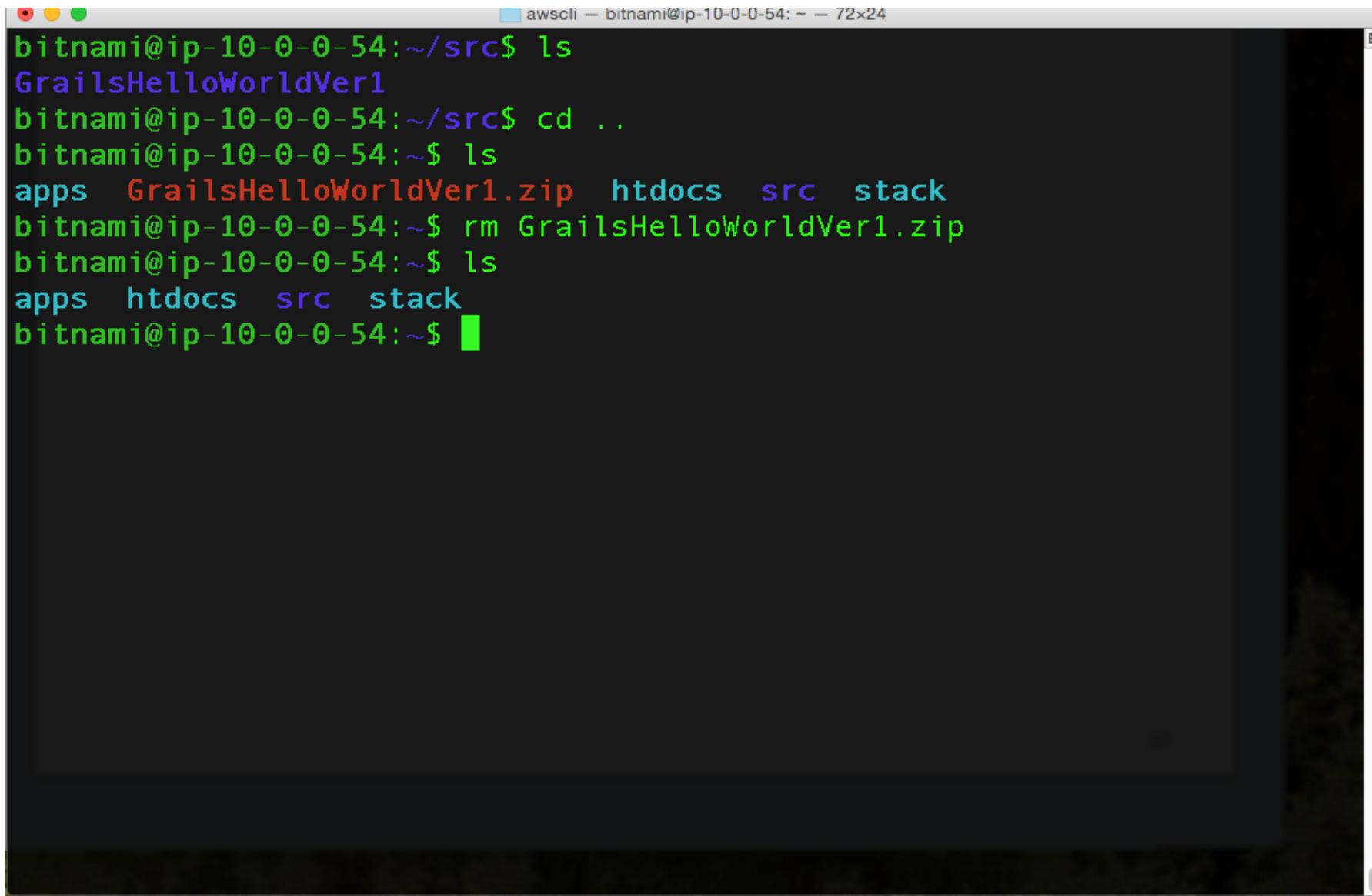
Installing: grails 2.4.3
Done installing!

Do you want grails 2.4.3 to be set as default? (Y/n): Y

Setting grails 2.4.3 as default.
bitnami@ip-10-0-0-54:~$ clear

bitnami@ip-10-0-0-54:~$ ls
apps  GrailsHelloWorldVer1.zip  htdocs  stack
bitnami@ip-10-0-0-54:~$ mkdir src
bitnami@ip-10-0-0-54:~$ ls
apps  GrailsHelloWorldVer1.zip  htdocs  src  stack
bitnami@ip-10-0-0-54:~$ cd src/
bitnami@ip-10-0-0-54:~/src$ unzip ../GrailsHelloWorldVer1.zip
Archive: ../GrailsHelloWorldVer1.zip
  creating: GrailsHelloWorldVer1/
  inflating: GrailsHelloWorldVer1/.classpath
  inflating: GrailsHelloWorldVer1/.DS_Store
  creating: GrailsHelloWorldVer1/.link_to_grails_plugins/
  creating: GrailsHelloWorldVer1/.link_to_grails_plugins/asset-pipeline
-1.9.6/
```

# Step 6. Build and Deploy Grails Hello World



A screenshot of a terminal window titled "awscli" running on a Bitnami instance. The terminal shows the following command-line session:

```
bitnami@ip-10-0-0-54:~/src$ ls
GrailsHelloWorldVer1
bitnami@ip-10-0-0-54:~/src$ cd ..
bitnami@ip-10-0-0-54:~$ ls
apps  GrailsHelloWorldVer1.zip  htdocs  src  stack
bitnami@ip-10-0-0-54:~$ rm GrailsHelloWorldVer1.zip
bitnami@ip-10-0-0-54:~$ ls
apps  htdocs  src  stack
bitnami@ip-10-0-0-54:~$ █
```

# Step 6. Build and Deploy Grails Hello World



A screenshot of a terminal window titled "awscli" running on a Linux system. The title bar shows the session details: "awscli — bitnami@ip-10-0-0-54: ~/src/GrailsHelloWorldVer1 — 72x24". The command "grails compile" is typed at the prompt, which is colored blue. The rest of the terminal window is black, indicating no output has been displayed.

# Step 6. Build and Deploy Grails Hello World



A terminal window titled "awscli" with the command "bitnami@ip-10-0-0-54:~/src/GrailsHelloWorldVer1\$ grails war" entered.

```
awscli - bitnami@ip-10-0-0-54: ~/src/GrailsHelloWorldVer1 - 72x24
bitnami@ip-10-0-0-54:~/src/GrailsHelloWorldVer1$ grails war
```

# Step 6. Build and Deploy Grails Hello World

# Step 6. Build and Deploy Grails Hello World

```
awscli – bitnami@ip-10-0-0-54: ~/stack/apache-tomcat/webapps – 72x24
bitnami@ip-10-0-0-54:~/stack/apache-tomcat/webapps$ ls
docs      GrailsHelloWorldVer1-1.0      host-manager  ROOT
examples  GrailsHelloWorldVer1-1.0.war  manager
bitnami@ip-10-0-0-54:~/stack/apache-tomcat/webapps$
```

# Step 6. Build and Deploy Grails Hello World

Screenshot of the Tomcat Web Application Manager interface showing the deployed GrailsHelloWorldVer1-1.0 application.

The Apache Software Foundation logo is displayed on the left, and the JBoss logo is on the right.

**Tomcat Web Application Manager**

**Message:** OK

**Manager**

List Applications    HTML Manager Help    Manager Help    Server Status

**Applications**

Path	Version	Display Name	Running	Sessions	Commands
/	None specified		true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/GrailsHelloWorldVer1-1.0	None specified	/GrailsHelloWorldVer1-production-1.0	false	0	Start Stop Reload Undeploy
/docs	None specified	Tomcat Documentation	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/examples	None specified	Servlet and JSP Examples	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/host-manager	None specified	Tomcat Host Manager Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/manager	None specified	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

**Deploy**

Deploy directory or WAR file located on server

Context Path (required):

XML Configuration file URL:

WAR or Directory URL:

ec2-52-11-45-67.us-west-2.compute.amazonaws.com/manager/html

# Step 6. Build and Deploy Grails Hello World

ec2-52-11-45-67.us-west-2.compute.amazonaws.com/manager/html/start?path=/GrailsHelloWorldVer1-1%2E0&org.apache.catalina.fil... JB »

The Apache Software Foundation  

## Tomcat Web Application Manager

**Message:** OK - Started application at context path /GrailsHelloWorldVer1-1.0

**Manager**

<a href="#">List Applications</a>	<a href="#">HTML Manager Help</a>	<a href="#">Manager Help</a>	<a href="#">Server Status</a>
-----------------------------------	-----------------------------------	------------------------------	-------------------------------

**Applications**

Path	Version	Display Name	Running	Sessions	Commands
/	None specified		true	0	<a href="#">Start</a> <a href="#">Stop</a> <a href="#">Reload</a> <a href="#">Undeploy</a> <a href="#">Expire sessions</a> with idle ≥ 30 minutes
/GrailsHelloWorldVer1-1.0	None specified	/GrailsHelloWorldVer1-production-1.0	true	0	<a href="#">Start</a> <a href="#">Stop</a> <a href="#">Reload</a> <a href="#">Undeploy</a> <a href="#">Expire sessions</a> with idle ≥ 30 minutes
/docs	None specified	Tomcat Documentation	true	0	<a href="#">Start</a> <a href="#">Stop</a> <a href="#">Reload</a> <a href="#">Undeploy</a> <a href="#">Expire sessions</a> with idle ≥ 30 minutes
/examples	None specified	Servlet and JSP Examples	true	0	<a href="#">Start</a> <a href="#">Stop</a> <a href="#">Reload</a> <a href="#">Undeploy</a> <a href="#">Expire sessions</a> with idle ≥ 30 minutes
/host-manager	None specified	Tomcat Host Manager Application	true	0	<a href="#">Start</a> <a href="#">Stop</a> <a href="#">Reload</a> <a href="#">Undeploy</a> <a href="#">Expire sessions</a> with idle ≥ 30 minutes
/manager	None specified	Tomcat Manager Application	true	1	<a href="#">Start</a> <a href="#">Stop</a> <a href="#">Reload</a> <a href="#">Undeploy</a> <a href="#">Expire sessions</a> with idle ≥ 30 minutes

**Deploy**

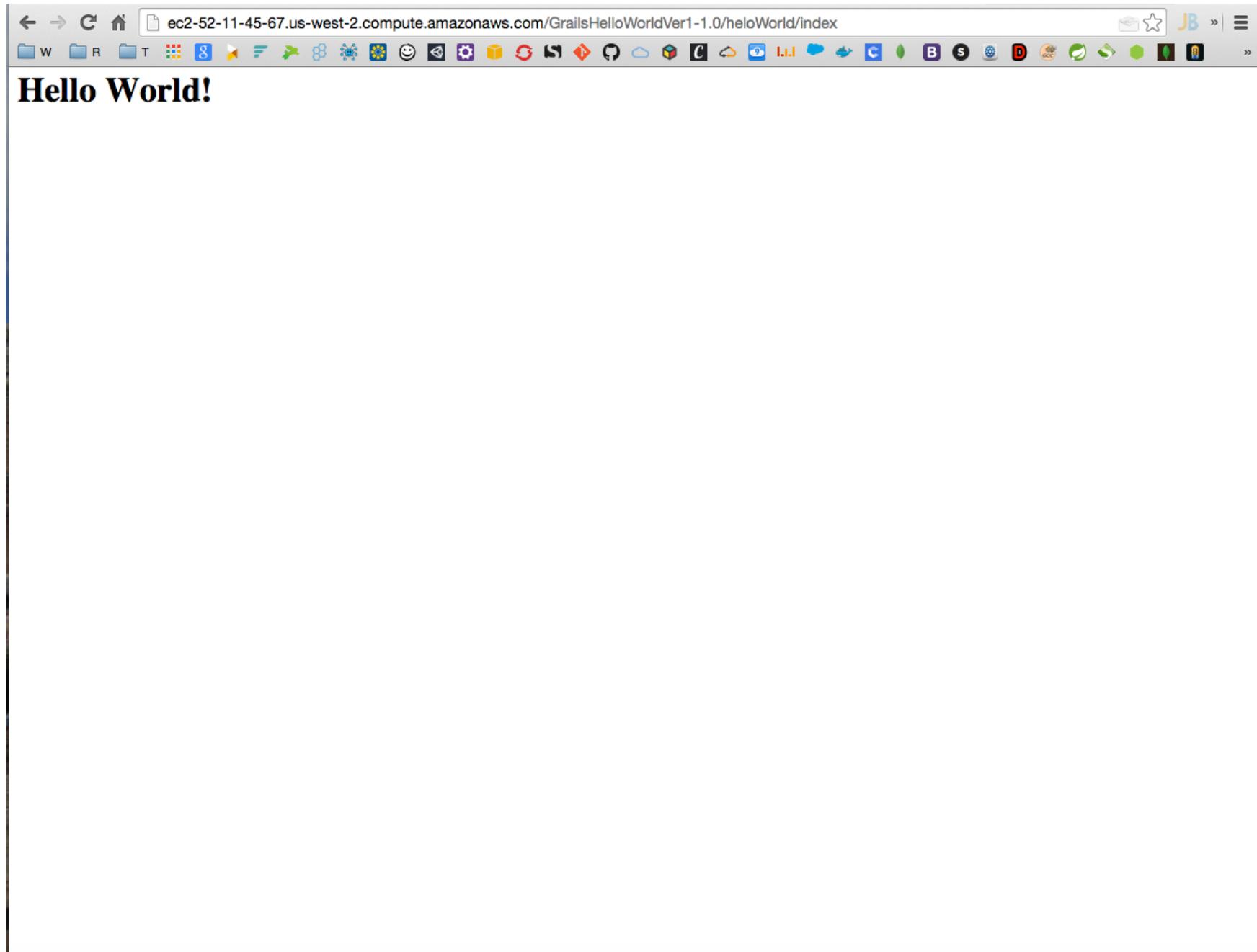
Deploy directory or WAR file located on server

Context Path (required):

XML Configuration file URL:

WAR or Directory URL:

# Step 6. Build and Deploy Grails Hello World



# Step 7. Access to Local MySQL (use your PWD)

```
awscli — bitnami@ip-10-0-0-54: ~ — 72x24
apps htdocs src stack
bitnami@ip-10-0-0-54:~$ mysql -uroot -pBDm3mU6Iij68
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.5.42 MySQL Community Server (GPL)

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

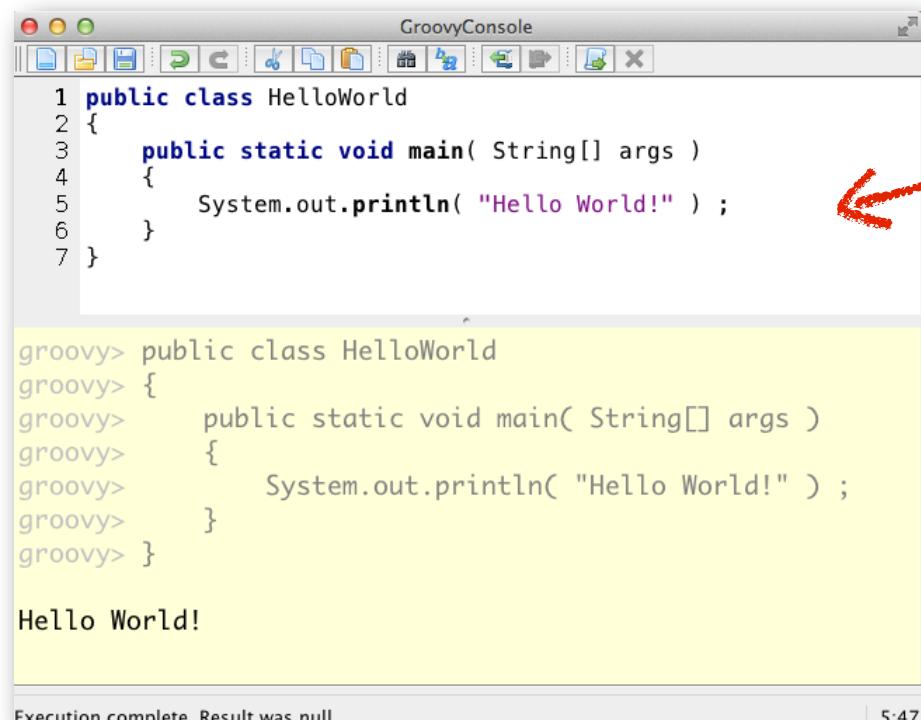
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases
-> ;
+-----+
| Database      |
+-----+
| information_schema |
| mysql          |
| performance_schema |
```

**Groovy**

# Hello World



A screenshot of a GroovyConsole window titled "GroovyConsole". The window has a toolbar at the top with various icons. The code area contains the following Java code:

```
1 public class HelloWorld
2 {
3     public static void main( String[] args )
4     {
5         System.out.println( "Hello World!" ) ;
6     }
7 }
```

The output area below shows the execution of the code:

```
groovy> public class HelloWorld
groovy> {
groovy>     public static void main( String[] args )
groovy>     {
groovy>         System.out.println( "Hello World!" ) ;
groovy>     }
groovy> }
```

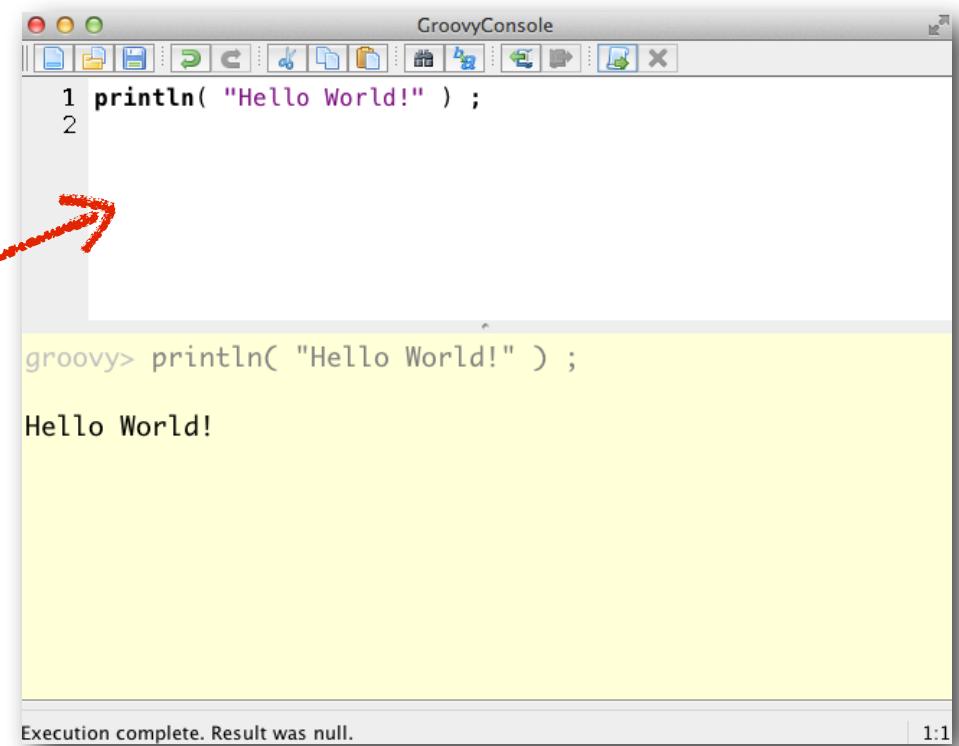
Followed by the output:

```
Hello World!
```

At the bottom, it says "Execution complete. Result was null." and the timestamp is 5:47.

Yes, Groovy is "Java", but  
without the ceremony!

Groovy is "Java"



A screenshot of a GroovyConsole window titled "GroovyConsole". The window has a toolbar at the top with various icons. The code area contains the following Groovy code:

```
1 println( "Hello World!" ) ;
```

The output area below shows the execution of the code:

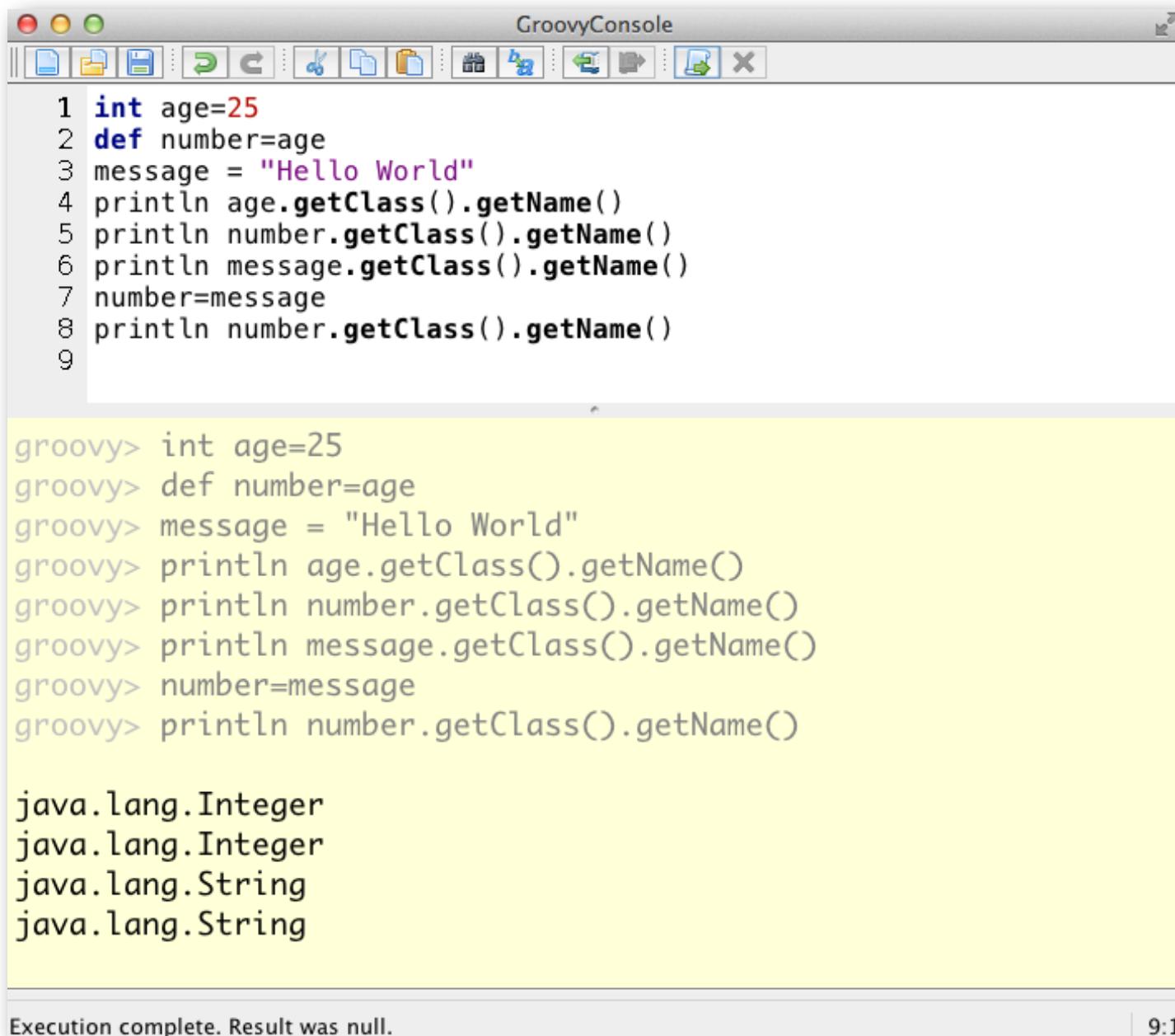
```
groovy> println( "Hello World!" ) ;
```

Followed by the output:

```
Hello World!
```

At the bottom, it says "Execution complete. Result was null." and the timestamp is 1:1.

# Dynamic Types



The screenshot shows a GroovyConsole window with the title "GroovyConsole". The console interface includes a toolbar with various icons for file operations and code navigation, and a status bar at the bottom.

```
1 int age=25
2 def number=age
3 message = "Hello World"
4 println age.getClass().getName()
5 println number.getClass().getName()
6 println message.getClass().getName()
7 number=message
8 println number.getClass().getName()
9
```

```
groovy> int age=25
groovy> def number=age
groovy> message = "Hello World"
groovy> println age.getClass().getName()
groovy> println number.getClass().getName()
groovy> println message.getClass().getName()
groovy> number=message
groovy> println number.getClass().getName()

java.lang.Integer
java.lang.Integer
java.lang.String
java.lang.String
```

Execution complete. Result was null. | 9:1

# Operators

TABLE 2.5 Relational operators

Expression	Method call	Result
5 < 3	5.compareTo(3) < 0	false
5 <= 3	5.compareTo(3) <= 0	false
5 > 3	5.compareTo(3) > 0	true
5 >= 3	5.compareTo(3) >= 0	true

TABLE 2.6 Equality operators

Expression	Method call	Result
5 == 3	5.equals(3)	false
5 != 3	! 5.equals(3) // see Appendix C	true
5 <=> 3	5.compareTo(3)	+1

The screenshot shows a GroovyConsole window with the title 'GroovyConsole'. The console area contains the following code:

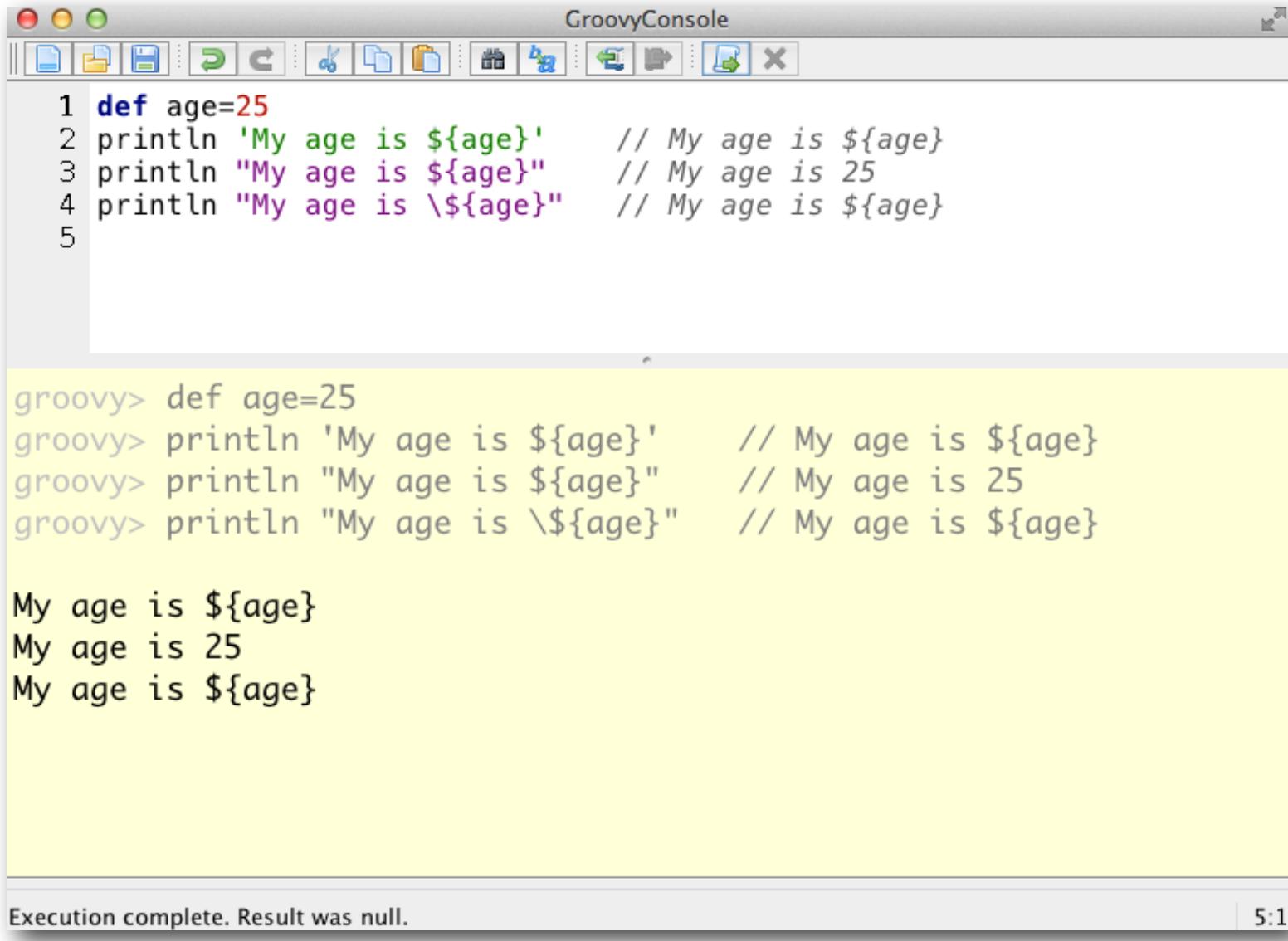
```
1 println 5<3
2 println 5.compareTo(3) < 0
3 println 5>3
4 println 5.compareTo(3) > 0
5
6 println 5==3
7 println 5.equals(3)
8 println 5!=3
9 println !(5.equals(3))
```

Below the code, the console output is displayed:

```
groovy> println 5<3
false
groovy> println 5.compareTo(3) < 0
false
groovy> println 5>3
true
groovy> println 5.compareTo(3) > 0
true
groovy> println 5==3
false
groovy> println 5.equals(3)
false
groovy> println 5!=3
true
groovy> println !(5.equals(3))
true
```

At the bottom of the console window, a status bar shows 'Execution complete. Result was null.' and the time '9:23'.

# String Literals



The screenshot shows a GroovyConsole window with the title "GroovyConsole". The top bar includes standard OS X-style window controls and a toolbar with various icons. The code editor area contains the following Groovy script:

```
1 def age=25
2 println 'My age is ${age}'      // My age is ${age}
3 println "My age is ${age}"      // My age is 25
4 println "My age is \$${age}"    // My age is ${age}
5
```

The console output area displays the results of running the script:

```
groovy> def age=25
groovy> println 'My age is ${age}'      // My age is ${age}
groovy> println "My age is ${age}"      // My age is 25
groovy> println "My age is \$${age}"    // My age is ${age}
```

Below the output, the results are shown as text:

```
My age is ${age}
My age is 25
My age is ${age}
```

At the bottom of the window, a status bar indicates "Execution complete. Result was null." and the time "5:1".

# Some Strings Examples

```

'Hello'.compareToIgnoreCase('hello')           // 0
'Hello'.concat('world')                      // Hello world
'Hello'.endsWith('lo')                       // true
'Hello'.equalsIgnoreCase('hello')             // true
'Hello'.indexOf('lo')                        // 3
'Hello world'.indexOf('o', 6)                // 7
'Hello'.matches('Hello')                     // true
'Hello'.matches('He')                        // false
'Hello'.replaceAll('l', 'L')                 // HELLo
'Hello world'.split('l')                     // 'He', 'o wor', 'd'
'Hello'.substring(1)                         // ello
'Hello'.substring(1, 4)                      // ell
'Hello'.toUpperCase()                        // HELLO
def message = 'Hello'
message.center(11)                           //□□□Hello□□□
message.center(3)                           // Hello
message.center(11, '#')                     // ###Hello###
message.eachMatch('.') { ch ->
    println ch }
message.getValueAt(0)                       // H
message.getValueAt(0..<3)                   // Hel
message.getValueAt([0, 2, 4])                // Hlo
message.leftShift('world')                 // Hello world
message << 'world'                        // Hello world
message.minus('ell')                       // Ho
message-'ell'                            // Ho
message.padLeft(4)                         // Hello
message.padLeft(11)                        // □□□□□Hello
message.padLeft(11, '#')                  // #####Hello
message.padRight(4)                        // Hello
message.padRight(11)                       // Hello□□□□□
message.padRight(11, '#')                  // Hello######
message.plus('world')                     // Hello world
message+'world'                           // Hello world
message.replaceAll('[a-z]') { ch ->
    ch.toUpperCase() }
message.reverse()                          // olleH
message.toList()                           // ['H', 'e', 'l', 'l', 'o']
def message='Hello world'
message.tokenize()                         // ['Hello', 'world']
message.tokenize('l')                     // ['He', 'o wor', 'd']

def greeting='Hello world'
'Hello'+ 'world'                         // Hello world      concatenate
'Hello'*3                                // HelloHelloHello repeat
greeting-'o world'                        // Hell
greeting.size()                           // 11
greeting.length()                         // 11
greeting.count('o')                       // 2
greeting.contains('ell')                 // true

```

In a regular expression, two special *positional characters* are used to denote the beginning and end of a line: caret (^) and dollar sign (\$):

```

def rhyme='Humpty Dumpty sat on a wall'
rhyme=~ '^Humpty'                         // true
rhyme=~ 'wall$'                           // true

```

Regular expressions can also include *quantifiers*. The plus sign (+) represents one or more times, applied to the preceding element of the expression. The asterisk (\*) is used to represent zero or more occurrences. The question mark (?) denotes zero or once. The metacharacter { and } is used to match a specific number of instances of the preceding character. The following all yield true:

```

'aaaaab' =~ 'a+b'
'b' =~ 'a*b'
'aaacd' =~ 'a*c?d'
'aaad' =~ 'a*c?d'
'aaaaab' =~ 'a{5}b'
!('aab' =~ 'a{5}b')

```

# Collections

Example	Description
[11, 12, 13, 14]	A list of integer values
['Ken', 'John', 'Andrew']	A list of Strings
[1, 2, [3, 4], 5]	A nested list
['Ken', 21, 1.69]	A heterogeneous list of object references
[ ]	An empty list

```
def numbers = [11, 12, 13, 14]      // list with four items
numbers [0]                         // 11
numbers [3]                         // 14

numbers [-1]                        // 14
numbers [-2]                        // 13
```

```
[11, 12, 13, 14][2]    // 13

numbers [0..2]      // [11, 12, 13]
numbers [1..<3]     // [12, 13]

numbers [1]=22       // [11, 22, 13, 14]
numbers [1]=[33, 44]  // [11, [33, 44], 13, 14]
```

```
numbers<<15      // [11, [33, 44], 13, 14, 15]

numbers=[11, 12, 13, 14]    // list with four items
numbers+[15, 16]           // [11, 12, 13, 14, 15, 16]
```

```
numbers=[11, 12, 13, 14]      // list with four items
numbers-[13]                  // [11, 12, 14]
```

TABLE 4.2 List methods

Name	Signature/description
add	boolean add(Object value) Append the new value to the end of this List.
add	void add(int index, Object value) Inserts a new value into this List at the given index position.
addAll	boolean addAll(Collection values) Append the new values on to the end of this List.
contains	boolean contains(Object value) Returns true if this List contains the specified value.
flatten *	List flatten() Flattens this List and returns a new List.
get	Object get(int index) Returns the element at the specified position in this List.

[11, 12, 13, 14].add(15)	// [11, 12, 13, 14, 15]
[11, 12, 13, 14].add(2, 15)	// [11, 12, 15, 13, 14]
[11, 12, 13, 14].add([15, 16])	// [11, 12, 13, 14, 15, 16]
[11, 12, 13, 14].get(1)	// 12
[11, 12, 13, 14].isEmpty()	// false
[14, 13, 12, 11].size()	// 4
[11, 12, [13, 14]].flatten()	// [11, 12, 13, 14]
[11, 12, 13, 14].getAt(1)	// 12
[11, 12, 13, 14].getAt(1..2)	// [12, 13]
[11, 12, 13, 14].getAt([2, 3])	// [13, 14]
[11, 12, 13, 14].intersect([13, 14, 15])	// [13, 14]
[11, 12, 13, 14].pop()	// 14
[11, 12, 13, 14].reverse()	// [14, 13, 12, 11]
[14, 13, 12, 11].sort()	// [11, 12, 13, 14]

# Maps

Example	Description
['Ken' : 'Barclay', 'John' : 'Savage']	Forename/surname collection
[4 : [2], 6 : [3, 2], 12 : [6, 4, 3, 2]]	Integer keys and their list of divisors
[ : ]	Empty map

```
def names=['Ken' : 'Barclay', 'John' : 'Savage']
def divisors=[4 : [2], 6 : [3, 2], 12 : [6, 4, 3, 2]]
names['Ken']                                // 'Barclay'
names.Ken                                    // 'Barclay'
names['Jessie']                             // null
divisors[6]                                  // [3, 2]
```

```
divisors[6]=[6, 3, 2, 1]                    // [4 : [2], 6 : [6, 3, 2, 1],
                                            // 12 : [6, 4, 3, 2]]
```

```
def careful=[ 1 : 'Ken', '1' : 'Barclay']
careful[1]                                    // Ken
careful['1']                                 // Barclay
```

TABLE 4.4 Map methods

Name	Signature/description
containsKey	boolean containsKey(Object key) Does this Map contain this key?
get	Object get(Object key) Look up the key in this Map and return the corresponding value. If there is no entry in this Map for the key, then return null.
get *	Object get(Object key, Object defaultValue) Look up the key in this Map and return the corresponding value. If there is no entry in this Map for the key, then return the defaultValue.
getAt *	Object getAt(Object key) Support method for the subscript operator.
keySet	Set keySet() Obtain a Set of the keys in this Map.
put	Object put(Object key, Object value) Associates the specified value with the specified key in this Map. If this Map previously contained a mapping for this key, the old value is replaced by the specified value.
putAt *	Object putAt(Object key, Object value) Support method to allow Maps to operate with subscript assignment.
size	int size() Returns the number of key-value mappings in this Map.
values	Collection values() Returns a collection view of the values contained in this Map.

```
def mp=['Ken' : 2745, 'John' : 2746, 'Sally' : 2742]
mp.put('Bob', 2713)                         // [Bob:2713, Ken:2745, Sally:2742, John:2746]
mp.containsKey('Ken')                        // true
mp.get('David', 9999)                        // 9999
mp.get('Sally')                            // 2742
mp.get('Billy')                            // null
mp.keySet()                               // [David, Bob, Ken, Sally, John]
mp.size()                                  // 4
mp['Ken']                                   // 2745
```

# Methods

```
def methodName() {  
    // Method code goes here  
}
```

```
def greetings() {  
    print 'Hello'  
    print ' and '  
    println 'welcome'  
}
```

```
greetings()
```

```
def methodName(para1, para2, para3) {  
    // Method code goes here  
}
```

```
def greetings(name) {  
    println "Hello and welcome, ${name}"  
}  
  
greetings('John')
```

Hello and welcome

Hello and welcome, John

```
def someMethod(para1, para2=0, para3=0) {  
    // Method code goes here  
}
```

```
def greetings(salutation, name='Ken') {  
    println "${salutation} ${name}"  
}  
  
greetings('Hello', 'John')          // Hello John  
greetings('Welcome')               // Welcome Ken
```

Hello John  
Welcome Ken

# Closures - Invocation & Parameters

## 9.1 CLOSURES

The syntax for defining a closure is:

```
{comma-separated-formal-parameter-list -> statement-list}
```

If no formal parameters are required, then the parameter List and the `->` separator are omitted. Here is a simple example of a closure with no parameters.

```
def clos={println 'Hello world'}
clos.call()
```

```
Hello world
```

```
def clos={param -> println "Hello ${param}"}
clos.call('world')           // actual argument is 'world'
clos.call('again')          // actual argument is 'again'
clos('shortcut')            // abbreviated form
```

```
Hello world
Hello again
Hello shortcut
```

Observe the third invocation in which the `call` has been omitted.

The next illustration repeats the previous example and produces the same result, but shows that an implicit single parameter referred to as `it` can be used.

```
def clos={println "Hello ${it}"}
clos.call('world')
clos.call('again')
clos('shortcut')
```



Default Single Parameter  
"it"

# Closures - Enclosing Scope

```
def greeting='Hello' ← "greeting" in defined scope
def clos={param -> println "${greeting} ${param}"}
clos.call('world')

// Now show that changes to this variable change the closure.
greeting='Welcome'
clos.call('world')

def demo(clo) {
    def greeting='Bonjour'           // does not affect closure
    clo.call('Ken')
}

demo(clos)                         "greeting" in call scope
```

The resulting output is:

```
Hello world
Welcome world
Welcome Ken
```

# Groovy Quick Start

## Overview

Groovy classes compile down to Java bytecode and so there's a 1-1 mapping between a Groovy class and a Java class.

Indeed each Groovy class can be used inside normal Java code - since it is a Java class too.

Probably the easiest way to get groovy is to try working with collections. In Groovy List (`java.util.List`) and Map (`java.util.Map`) are both first class objects in the syntax. So to create a List of objects you can do the following...

```
def list = [1, 2, 'hello', new java.util.Date()]
assert list.size() == 4
assert list.get(2) == 'hello'
assert list[2] == 'hello'
```

Notice that everything is an object (or that auto-boxing takes place when working with numbers). To create maps...

```
def map = ['name':'James', 'location':'London']
assert map.size() == 2
assert map.get('name') == 'James'
assert map['name'] == 'James'
```

Iterating over collections is easy...

```
def list = [1, 2, 3]
for (i in list) { println i }
```

# Groovy Quick Start

## Working with closures

Closures are similar to Java's inner classes, except they are a single method which is invokable, with arbitrary parameters. A closure can have as many parameters as you wish...

```
def closure = { param -> println("hello ${param}") }
closure.call("world!")

closure = { greeting, name -> println(greeting + name) }
closure.call("hello ", "world!")
```

If no parameter(s) is(are) specified before -> symbol then a default named parameter, called 'it' can be used. e.g.

```
def closure = { println "hello " + it }
closure.call("world!")
```

Using closures allows us to process collections (arrays, maps, strings, files, SQL connections and so forth) in a clean way. e.g

```
[1, 2, 3].each ({ item -> print "${item}-" })
["k1":"v1", "k2":"v2"].each {key, value -> println key + "=" + value}
```

# Closures & Collections

Several `List`, `Map`, and `String` methods accept a closure as an argument (see also Appendix H). It is this combination of closures and collections that provides Groovy with some elegant solutions to common programming problems. For example, the `each` method with the signature:

```
void each(Closure closure)
```

is used to iterate through a `List`, `Map`, or `String` and apply the closure on every element. Example 08 presents a number of simple examples of the `each` method and a closure.

```
[1, 2, 3, 4].each {println it}  
  
['Ken' : 21, 'John' : 22, 'Sally' : 25].each {println it}  
['Ken' : 21, 'John' : 22, 'Sally' : 25].each {println "${it.key} maps to: ${it.value}"}  
  
'Ken'.each {println it}
```

```
1  
2  
3  
4  
Sally=25  
John=22  
Ken=21  
Sally maps to: 25  
John maps to: 22  
Ken maps to: 21  
K  
e  
n
```

# Groovy Quick Start

**Note:** If a given closure is the last parameter of a method, its definition can reside outside of the parentheses. Thus the following code is valid:

```
def fun(int i, Closure c) {
    c.call(i)
}

// put Closure out of ()
[1, 2, 3].each() { item -> print "${item}-" } // 1-2-3-
fun(123) { i -> println i } // 123

// omit ()
[1, 2, 3].each ({ item -> print "${item}-" }) // 1-2-3-

// omit enclosing ()
[1, 2, 3].each { item -> print "${item}-" } // 1-2-3-

// normal
[1, 2, 3].each(({ item -> print "${item}-" })) // 1-2-3-

// using the fun function to do the same thing
[1,2,3].each {fun(it,{item -> print "${item}-"})} // 1-2-3-

def closure = { i -> println i}

//[1, 2, 3].each() closure // error. closure has been previously defined
```

# Groovy Quick Start

## each

iterate via a closure

```
[1, 2, 3].each { item -> print "${item}-" }
```

## collect

collect the return value of calling a closure on each item in a collection

```
def value = [1, 2, 3].collect { it * 2 }
assert value == [2, 4, 6]
```

## find

finds first item matching closure predicate

```
def value = [1, 2, 3].find { it > 1 }
assert value == 2
```

## findAll

finds all items matching closure predicate

```
def value = [1, 2, 3].findAll { it > 1 }
assert value == [2, 3]
```

# Groovy Quick Start

## inject

allows you to pass a value into the first iteration and then pass the result of that iteration into the next iteration and so on. This is ideal for counting and other forms of processing

```
def value = [1, 2, 3].inject('counting: ') { str, item -> str + item }
assert value == "counting: 123"

value = [1, 2, 3].inject(0) { count, item -> count + item }
assert value == 6
```

In addition there's 2 new methods for doing boolean logic on some collection...

## every

returns true if all items match the closure predicate

```
def value = [1, 2, 3].every { it < 5 }
assert value

value = [1, 2, 3].every { item -> item < 3 }
assert ! value
```

## any

returns true if any item match the closure predicate

```
def value = [1, 2, 3].any { it > 2 }
assert value

value = [1, 2, 3].any { item -> item > 3 }
assert value == false
```

# Groovy Quick Start

## max / min

returns the max/min values of the collection - for Comparable objects

```
value = [9, 4, 2, 10, 5].max()  
assert value == 10  
value = [9, 4, 2, 10, 5].min()  
assert value == 2  
value = ['x', 'y', 'a', 'z'].min()  
assert value == 'a'
```

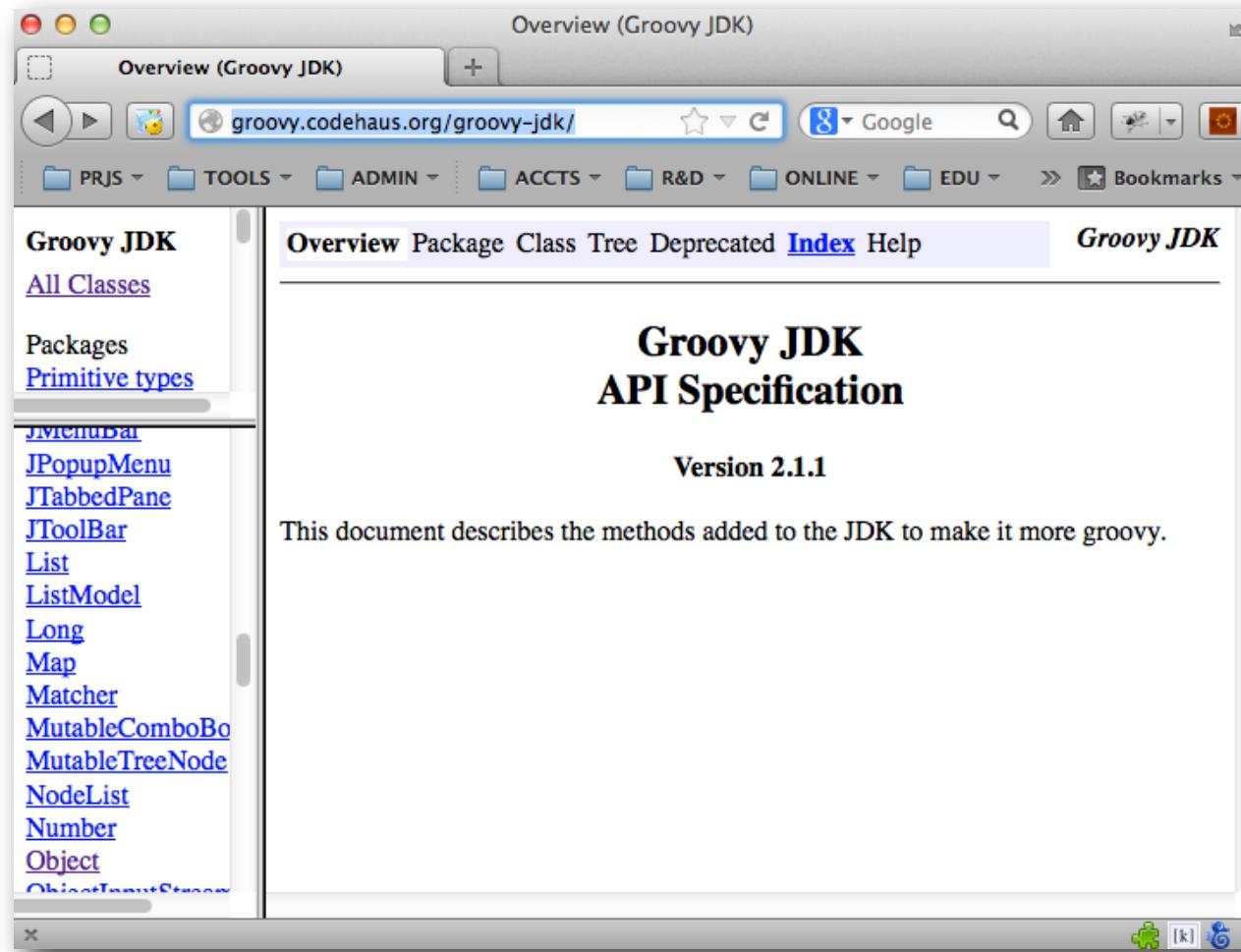
## join

concatenates the values of the collection together with a string value

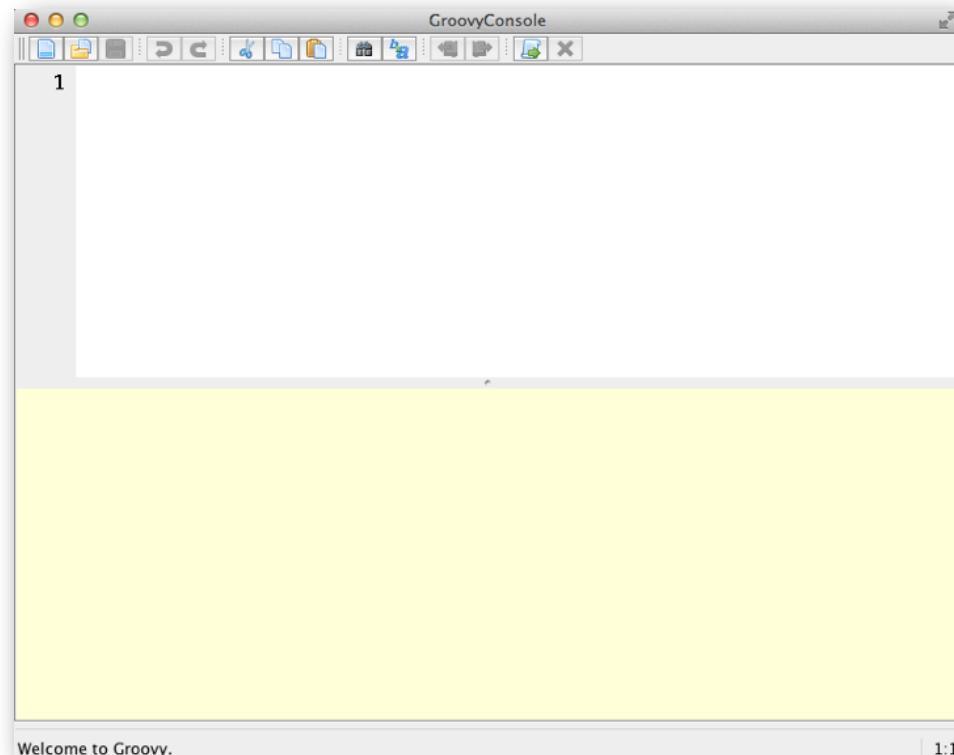
```
def value = [1, 2, 3].join("-")  
assert value == "1-2-3"
```

# GDK

<http://groovy.codehaus.org/groovy-jdk/>



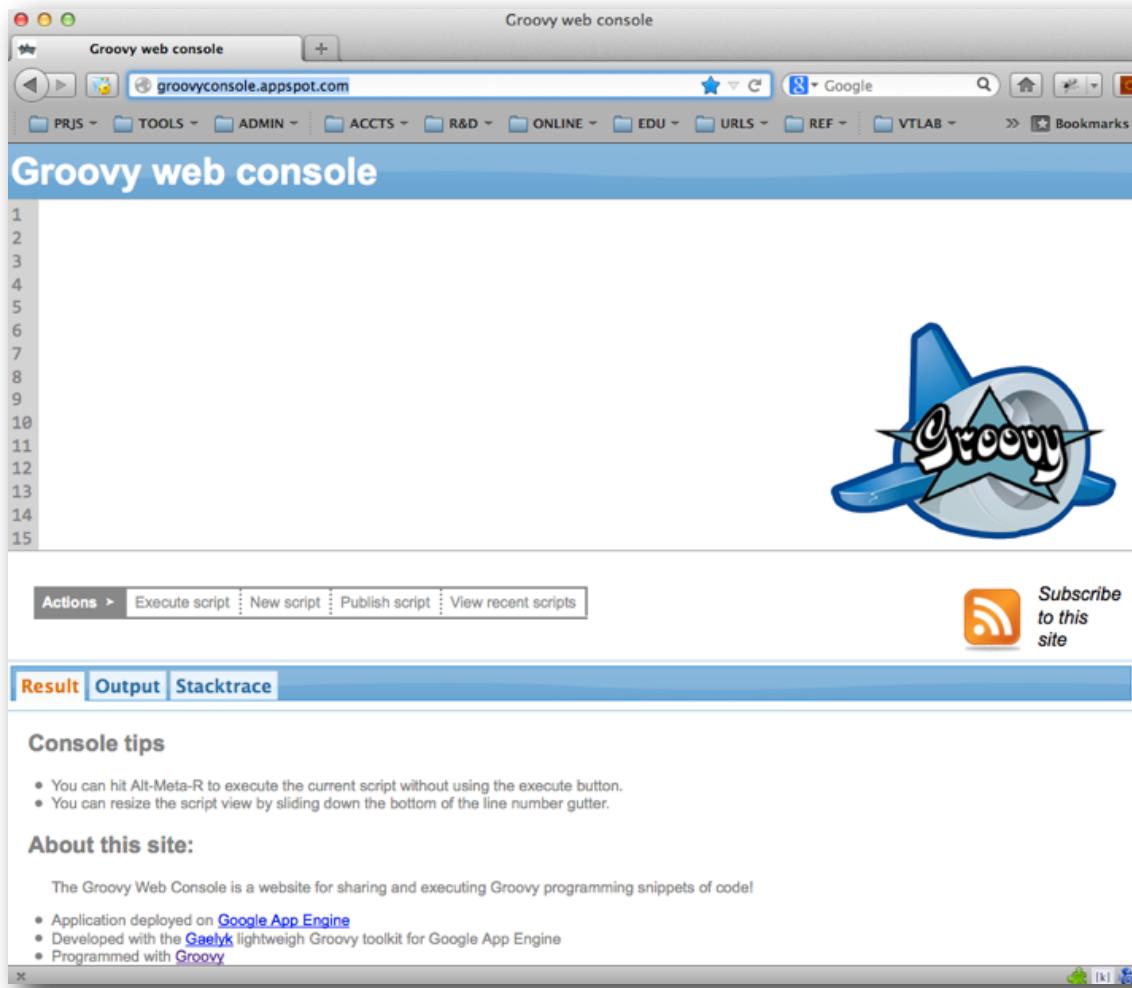
# Groovy Console



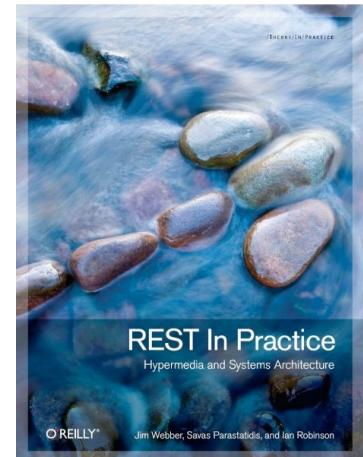
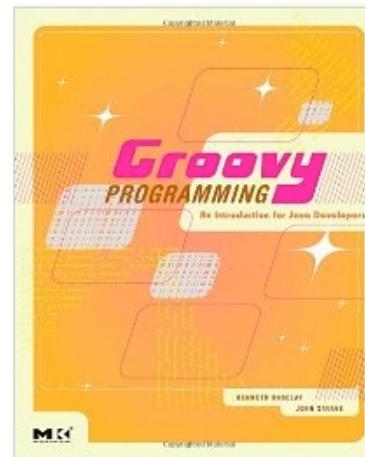
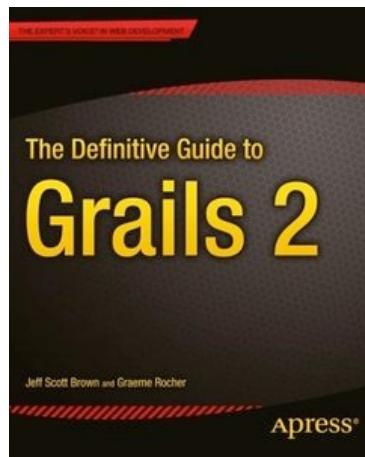
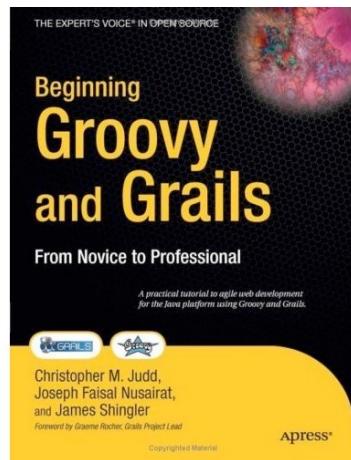
```
pnguyen — 83x25
dragon:~ pnguyen$ which groovyConsole
Volumes/Data/GROOVY/bin/groovyConsole
dragon:~ pnguyen$ echo $GROOVY_HOME
Volumes/Data/GROOVY
dragon:~ pnguyen$ groovyConsole
|
```

# Groovy Web Console

<http://groovyconsole.appspot.com/>



# References



#15

## DZone Refcardz

Get More Refcardz Visit refcardz.com

**CONTENTS INCLUDE:**

- Groovy/Java Integration
- Language Elements
- Operators
- Collective Duties
- Meta Programming
- Hot Tips and more...

### Groovy

By Dierk König

**ABOUT GROOVY**

Groovy is a dynamic language for the Java™ Virtual Machine (JVM). It shines with full object-orientation, scriptability optional typing, operator customization, lexical declarations for the most common data types, advanced semantics like closures and ranges, compact parsing syntax and seamless Java™ integration. Groovy also provides exactly the kind of information you are likely to look up when programming Groovy.

**STARTING GROOVY**

Install Groovy from <http://groovy.codehaus.org> and you will have the following commands available:

Command	Purpose
groovy	Execute Groovy code
groovysh	Compile Groovy code
grind	Open Groovy file
grindConsole	Open Groovy IDE console
groovyc	Migration helper

The groovy command comes with -h and --help options to show all options and required arguments. Typical usages are:

```
Eval groovy <file>.groovy
groovy MyScript.groovy

Evaluate (e) on the command line
groovy <file> 12.5Math.PI

Print (p) for each line of input
echo 12.5 | groovy -pe
"line.toDouble() * Math.PI"

Inline edit (d) file data.txt by reversing each line and save a backup
groovy -lba -pe
"line.reverse()" data.txt
```

**GROOVY/JAVA INTEGRATION**

From Groovy, you can call any Java code like you would do from Java. It's identical!

From Java, you can call Groovy code in the following ways. Note that you need to have the groovy-all.jar in your classpath.

**Cross-compilation**

Use either the groovyc-ant task or your IDE integration to compile your groovy code together with your Java code. This enables you to use your Groovy code as if it was written in Java.

**Eval**

Use class groovy.util.Eval for evaluating simple code that is captured in a Java String: (int) Eval.xy(1,2,3,"x+y\*2");

**Get More Refcardz**  
(They're free!)

- Authoritative content
- Designed for developers
- Written by top experts
- Latest tools & technologies
- Hot tips & examples
- Bonus content online
- New issue every 1-2 weeks

Subscribe Now for FREE!  
Refcardz.com

www.dzone.com

Groovy

#60

## DZone Refcardz

Get More Refcardz Visit refcardz.com

**CONTENTS INCLUDE:**

- Getting Started with Grails
- Plugins
- Domain Class Mosaic
- The Three R's of Controllers
- Services
- Hot Tips and more...

### Getting Started with Grails

By Dave Klein

**GETTING STARTED WITH GRAILS**

Grails is a full-stack web application framework built on top of such tried and true open source frameworks as Spring, Hibernate, and Grails-specific conventions. By combining conventions as Configuration over Configuration and Convention over Configuration, Grails makes it incredibly easy to use these powerful tools. Grails doesn't reinvent the wheel; Grails makes a wheel that inflates itself and rolls where you want it to!

In case you are new to Grails, we'll start with a brief introduction to what Grails is, how to get it up and running, and turn you into a Grails developer. Since this Refcard will come in handy, it is a cheat sheet for Grails developers, a quick source for those things you keep having to go back to the docs to look up. Controllers, Services and Views with a detailed GSP taglist reference.

**Installing Grails**

Download the Grails archive from <http://grails.org/download> and extract it to a local directory. Set a GRAILS\_HOME environment variable to that directory and add GRAILS\_HOME/bin to your path. (You also need a valid JAVA\_HOME environment variable.) Now you're ready to go!

**A Web App in the Blink of an Eye**

To create a new Grails application, type:

```
$ grails create-app AutoMart
```

Now change to the AutoMart directory and create a domain class:

```
$ grails create-domain-class Car
```

Open AutoMart/grails-app/domain/Car.groovy and edit it, like so:

```
class Car {
    String make
    String model
    Integer year
}
```

Save this file, and run:

```
$ grails generate-all car
```

Create app and create-domain-class are Grails scripts. To see what other scripts are provided by Grails, run grails help from the command line.

You now have a complete working web application, with pages for creating, displaying, editing and listing Car instances. You can launch it with:

```
$ grails run-app
```

Grails runs on port 8080 by default. You can easily run on a different port like this:

**Get More Refcardz**  
(They're free!)

- Authoritative content
- Designed for developers
- Written by top experts
- Latest tools & technologies
- Hot tips & examples
- Bonus content online
- New issue every 1-2 weeks

Subscribe Now for FREE!  
Refcardz.com

www.dzone.com

Grails

# Free Training Videos

<http://www.skillbuilders.com/Groovy-and-Grails/Groovy-and-Grails-Training.cfm?tab=groovy-and-grails-training>

The screenshot shows the SkillBuilders website with a dark blue header. The logo 'SKILLBUILDERS' is on the left, followed by navigation links for Education, Oracle, OS, and APEX. Below the header is a search bar and a 'Join the SkillBuilders Community' button. On the right are social sharing icons for Facebook, Twitter, LinkedIn, and Google+. The main content area features a large image of a person's hands typing on a keyboard, the title 'Groovy and Grails Training' in orange, and an average student rating of '4.84 out of 5'. A brief description states: 'Whether you need customized classroom training at your organization's site or instructor-led online training for yourself, SkillBuilders can help!' To the right is a portrait photo of a smiling man with glasses and a beard. Below this is a navigation menu with tabs for Class Calendar, Free Tutorials, Groovy and Grails Training (which is selected), and Groovy and Grails for Ellucian Banner Developers. The 'Groovy and Grails Training' section contains text about starting with free tutorials and then moving to an instructor-led class. It lists several course offerings with their titles, lengths, review counts, and ratings:

Class Title	Length (Days)	Student Reviews	Student Rating (Out of 5)
<b>Java Classes</b>			
<a href="#">Gentle Java and OO Development</a>	4	<a href="#">6 Reviews</a>	★★★★★
<a href="#">Fast Track to Java for OO Developers</a>	3	<a href="#">0 Reviews</a>	★★★★★
<b>Groovy &amp; Grails Classes</b>			
<a href="#">Getting Started with Groovy and Grails</a>	4	<a href="#">5 Reviews</a>	★★★★★
<a href="#">Groovy for Java Developers</a>	4	<a href="#">0 Reviews</a>	★★★★★

At the bottom, a note says: 'We schedule online classes as frequently as possible, based on demand. If you're interested in a class that isn't currently scheduled, please call or [email us](#), or complete our online [class request form](#)'.

U.S.: 1.888.803.5607 International: 001-401-783-6172      About      Contact      Privacy      © 2013