brief contents

PART 1 GETTING STARTED 1

1 ■ What is Amazon Web Services? 3

2 ■ A simple example: WordPress in five minutes 34

PART 2 BUILDING VIRTUAL INFRASTRUCTURE WITH SERVERS AND NETWORKING..51

3 ■ Using virtual servers: EC2 53

4 ■ Programming your infrastructure: the command line, SDKs, and CloudFormation 91

5 ■ Automating deployment: CloudFormation, Elastic Beanstalk, and OpsWorks 124

6 ■ Securing your system: IAM, security groups, and VPC 152

PART 3 STORING DATA IN THE CLOUD 183

7 ■ Storing your objects: S3 and Glacier 185

8 ■ Storing your data on hard drives: EBS and instance store 204

9 ■ Using a relational database service: RDS 225

10 ■ Programming for the NoSQL database service: DynamoDB 253

PART 4 ARCHITECTING ON AWS 279

11 ■ Achieving high availability: availability zones, auto-scaling, and CloudWatch 281

12 ■ Decoupling your infrastructure: ELB and SQS 310

13 ■ Designing for fault-tolerance 331

14 ■ Scaling up and down: auto-scaling and CloudWatch 363

contents

foreword xv
preface xvii
acknowledgments xix
about this book xxi
about the authors xxiv
about the cover illustration xxv

PART 1 GETTING STARTED 1

1 ***What is Amazon Web Services? 3***

1.1 What is cloud computing? 4

1.2 What can you do with AWS? 5

Hosting a web shop 5 ▪ Running a Java EE application in your private network 6 ▪ Meeting legal and business data archival requirements 7 ▪ Implementing a fault-tolerant system architecture 9

1.3 How you can benefit from using AWS 10

Innovative and fast-growing platform 10 ▪ Services solve common problems 11 ▪ Enabling automation 11 ▪ Flexible capacity (scalability) 11 ▪ Built for failure (reliability) 12 ▪ Reducing time to market 12 ▪ Benefiting from economies of scale 12 Worldwide 12 ▪ Professional partner 12

1.4 How much does it cost? 13
Free Tier 13 ▪ Billing example 13 ▪ Pay-per-use opportunities 15

1.5 Comparing alternatives 15

1.6 Exploring AWS services 17

1.7 Interacting with AWS 20
Management Console 20 ▪ Command-line interface 20
SDKs 22 ▪ Blueprints 22

1.8 Creating an AWS account 23
Signing up 23 ▪ Signing In 27 ▪ Creating a key pair 29
Creating a billing alarm 32

1.9 Summary 32

2 ***A simple example: WordPress in five minutes 34***

2.1 Creating your infrastructure 35

2.2 Exploring your infrastructure 41
Resource groups 41 ▪ Web servers 42 ▪ Load balancer 44
MySQL database 45

2.3 How much does it cost? 46

2.4 Deleting your infrastructure 48

2.5 Summary 49

PART 2 BUILDING VIRTUAL INFRASTRUCTURE WITH SERVERS AND NETWORKING..51

3 ***Using virtual servers: EC2 53***

3.1 Exploring a virtual server 53
Launching a virtual server 54 ▪ Connecting to a virtual server 65 ▪ Installing and running software manually 68

3.2 Monitoring and debugging a virtual server 69
Showing logs from a virtual server 69 ▪ Monitoring the load of a virtual server 70

3.3 Shutting down a virtual server 71

3.4 Changing the size of a virtual server 72

3.5 Starting a virtual server in another data center 74

3.6 Allocating a public IP address 78

3.7 Adding an additional network interface to a virtual server 80

3.8 Optimizing costs for virtual servers 83

Reserve virtual servers 84 ▪ Bidding on unused virtual servers 84

3.9 Summary 90

4 *Programming your infrastructure: the command line, SDKs, and CloudFormation 91*

4.1 Infrastructure as code 93

Automation and the DevOps movement 93 ▪ Inventing an infrastructure language: JIML 94

4.2 Using the command-line interface 97

Installing the CLI 97 ▪ Configuring the CLI 98 ▪ Using the CLI 103

4.3 Programming with the SDK 107

Controlling virtual servers with SDK: nodecc 108 ▪ How nodecc creates a server 109 ▪ How nodecc lists servers and shows server details 110 ▪ How nodecc terminates a server 111

4.4 Using a blueprint to start a virtual server 112

Anatomy of a CloudFormation template 113 ▪ Creating your first template 117

4.5 Summary 123

5 *Automating deployment: CloudFormation, Elastic Beanstalk, and OpsWorks 124*

5.1 Deploying applications in a flexible cloud environment 126

5.2 Running a script on server startup using CloudFormation 126

Using user data to run a script on server startup 127 ▪ Deploying OpenSwan as a VPN server to a virtual server 127 ▪ Starting from scratch instead of updating 132

5.3 Deploying a simple web application with Elastic Beanstalk 132

Components of Elastic Beanstalk 132 ▪ Using Elastic Beanstalk to deploy Etherpad, a Node.js application 133

5.4 Deploying a multilayer application with OpsWorks 138

Components of OpsWorks 138 ▪ Using OpsWorks to deploy an IRC chat application 140

5.5 Comparing deployment tools 149

Classifying the deployment tools 149 ▪ Comparing the deployment services 150

5.6 Summary 150

6 *Securing your system: IAM, security groups, and VPC* 152

6.1 Who's responsible for security? 153

6.2 Keeping your software up to date 154

Checking for security updates 154 ▪ Installing security updates on server startup 155 ▪ Installing security updates on running servers 157

6.3 Securing your AWS account 157

Securing your AWS account's root user 158 ▪ Identity and Access Management service 159 ▪ Policies for authorization 160
Users for authentication, and groups to organize users 161
Roles for authentication of AWS resources 163

6.4 Controlling network traffic to and from your virtual server 164

Controlling traffic to virtual servers with security groups 166 ▪ Allowing ICMP traffic 167 ▪ Allowing SSH traffic 168 ▪ Allowing SSH traffic from a source IP address 168 ▪ Allowing SSH traffic from a source security group 170 ▪ Agent forwarding with PuTTY 172

6.5 Creating a private network in the cloud: Virtual Private Cloud (VPC) 173

Creating the VPC and an internet gateway (IGW) 175 ▪ Defining the public bastion host subnet 175 ▪ Adding the private Apache web server subnet 178 ▪ Launching servers in the subnets 178
Accessing the internet from private subnets via a NAT server 179

6.6 Summary 181

PART 3 STORING DATA IN THE CLOUD 183

7 *Storing your objects: S3 and Glacier* 185

7.1 Concept of an object store 186

7.2 Amazon S3 186

7.3 Backing up your data 187

7.4 Archiving objects to optimize costs 190

Creating an S3 bucket for use with Glacier 190 ▪ Adding a lifecycle rule to a bucket 191 ▪ Experimenting with Glacier and your lifecycle rule 193

7.5 Storing objects programmatically 195

Setting up an S3 bucket 195 ▪ Installing a web application that uses S3 196 ▪ Reviewing code access: S3 with SDK 196

7.6 Using S3 for static web hosting 198

Creating a bucket and uploading a static website 199 ▪ Configuring a bucket for static web hosting 199 ▪ Accessing a website hosted on S3 200

7.7 Internals of the object store 201

Ensuring data consistency 201 ▪ Choosing the right keys 202

7.8 Summary 203

8 ***Storing your data on hard drives: EBS and instance store 204***

8.1 Network-attached storage 205

Creating an EBS volume and attaching it to your server 206 ▪ Using Elastic Block Store 206 ▪ Tweaking performance 208 ▪ Backing up your data 210

8.2 Instance stores 212

Using an instance store 214 ▪ Testing performance 215
Backing up your data 216

8.3 Comparing block-level storage solutions 216

8.4 Hosting a shared file system backed by an instance store and EBS 217

Security groups for NFS 218 ▪ NFS server and volume 220
NFS server installation and configuration script 221 ▪ NFS clients 223 ▪ Sharing files via NFS 223

8.5 Summary 224

9 ***Using a relational database service: RDS 225***

9.1 Starting a MySQL database 228

Launching a WordPress platform with an Amazon RDS database 228 ▪ Exploring an RDS database instance with a MySQL engine 231 ▪ Pricing for Amazon RDS 233

9.2 Importing data into a database 234

9.3 Backing up and restoring your database 236

Configuring automated snapshots 236 ▪ Creating snapshots manually 237 ▪ Restoring a database 238 ▪ Copying a database to another region 240 ▪ Calculating the cost of snapshots 240

9.4 Controlling access to a database 241

Controlling access to the configuration of an RDS database 241
Controlling network access to an RDS database 243 ▪ Controlling data access 243

9.5 Relying on a highly available database 244

Enabling high-availability deployment for an RDS database 245

9.6 Tweaking database performance 246

Increasing database resources 246 ▪ Using read replication to increase read performance 248

9.7 Monitoring a database 250

9.8 Summary 251

10 Programming for the NoSQL database service: DynamoDB 253

10.1 Operating DynamoDB 255

Administration 255 ▪ Pricing 255 ▪ RDS comparison 255

10.2 DynamoDB for developers 256

Tables, items, and attributes 256 ▪ Primary keys 257 ▪ NoSQL comparison 257 ▪ DynamoDB Local 258

10.3 Programming a to-do application 258

10.4 Creating tables 260

Users with hash keys 260 ▪ Tasks with hash and range keys 262

10.5 Adding data 263

Adding a user 265 ▪ Adding a task 265

10.6 Retrieving data 266

Getting by key 267 ▪ Querying by key and filter 268 ▪ Using secondary indexes for more flexible queries 270 ▪ Scanning and filtering all of your table's data 272 ▪ Eventually consistent data retrieval 273

10.7 Removing data 273

10.8 Modifying data 274

10.9 Scaling capacity 275

10.10 Summary 277

PART 4 ARCHITECTING ON AWS 279

11 Achieving high availability: availability zones, auto-scaling, and CloudWatch 281

11.1 Recovering from server failure with CloudWatch 283

Creating a CloudWatch alarm 285 ▪ Monitoring and recovering a virtual server based on a CloudWatch alarm 286

11.2 Recovering from a data center outage 289

Availability zones: multiple data centers per region 290
Using auto-scaling to ensure that a virtual server is always running 294 ▪ Recovering a failed virtual server to another availability zone with the help of auto-scaling 296 ▪ Pitfall: network-attached storage recovery 299 ▪ Pitfall: network interface recovery 303

11.3 Analyzing disaster-recovery requirements 307

RTO and RPO comparison for a single virtual server 308

11.4 Summary 309

12 ***Decoupling your infrastructure: ELB and SQS 310***

12.1 Synchronous decoupling with load balancers 312

Setting up a load balancer with virtual servers 313 ▪ Pitfall: connecting a server too early 315

More use cases 316

12.2 Asynchronous decoupling with message queues 322

Turning a synchronous process into an asynchronous one 323
Architecture of the URL2PNG application 324 ▪ Setting up a message queue 324 ▪ Producing messages programmatically 324
Consuming messages programmatically 326 ▪ Limitations of messaging with SQS 329

12.3 Summary 330

13 ***Designing for fault-tolerance 331***

13.1 Using redundant EC2 instances to increase availability 333

Redundancy can remove a single point of failure 334
Redundancy requires decoupling 336

13.2 Considerations for making your code fault-tolerant 337

Let it crash, but also retry 337 ▪ Idempotent retry makes fault-tolerance possible 337

13.3 Architecting a fault-tolerant web application: Imagery 340

The idempotent image-state machine 343 ▪ Implementing a fault-tolerant web service 345 ▪ Implementing a fault-tolerant worker to consume SQS messages 351 ▪ Deploying the application 354

13.4 Summary 362

14 *Scaling up and down: auto-scaling and CloudWatch 363*

14.1 Managing a dynamic server pool 365

14.2 Using metrics and schedules to trigger scaling 370

Scaling based on a schedule 371 ▪ Scaling based on CloudWatch metrics 372

14.3 Decoupling your dynamic server pool 375

Scaling a dynamic server pool synchronously decoupled by a load balancer 377 ▪ Scaling a dynamic server pool asynchronously decoupled by a queue 382

14.4 Summary 385

index 387