Swift Web Applications on the AWS Cloud

Quick Start Reference Deployment

November 2016

Asif Khan, Tom Horton, and Tony Vattathil
Solutions Architects, Amazon Web Services

Contents

Overview

This Quick Start reference deployment guide provides step-by-step instructions for deploying web applications written with the Swift programming language on the Amazon Web Services (AWS) Cloud. Quick Starts are automated reference deployments that use AWS CloudFormation templates to launch, configure, and run the AWS compute, network, storage, and other services required to deploy a specific workload on AWS.

Swift is a popular programming language used to write applications for the Apple iOS, OS X, watchOS, and tvOS platforms. By using Swift with Amazon EC2 Container Service (Amazon ECS), you can create a homogeneous, scalable application stack.

This Quick Start uses Vapor to deploy a Swift web application. Vapor is a web framework for server-side applications written in Swift. This Quick Start uses Vapor as the web framework for running server-side applications written in Swift.

This Quick Start is for users who would like to deploy a Swift web application using Vapor on Amazon ECS. This Quick Start sets up a Swift development environment for demo purposes, but you can also use this environment for your production work.

Costs and Licenses

You are responsible for the cost of the AWS services used while running this Quick Start reference deployment. There is no additional cost for using the Quick Start.

The AWS CloudFormation template for this Quick Start includes configuration parameters that you can customize. Some of these settings, such as instance type, will affect the cost of deployment. See the pricing pages for each AWS service you will be using or the AWS Simple Monthly Calculator for cost estimates.

This Quick Start doesn't require a license for the Swift programming language or for Vapor, which are both open source.

Architecture

Deploying this Quick Start with the **default parameters** builds the following Swift web application environment in the AWS Cloud.

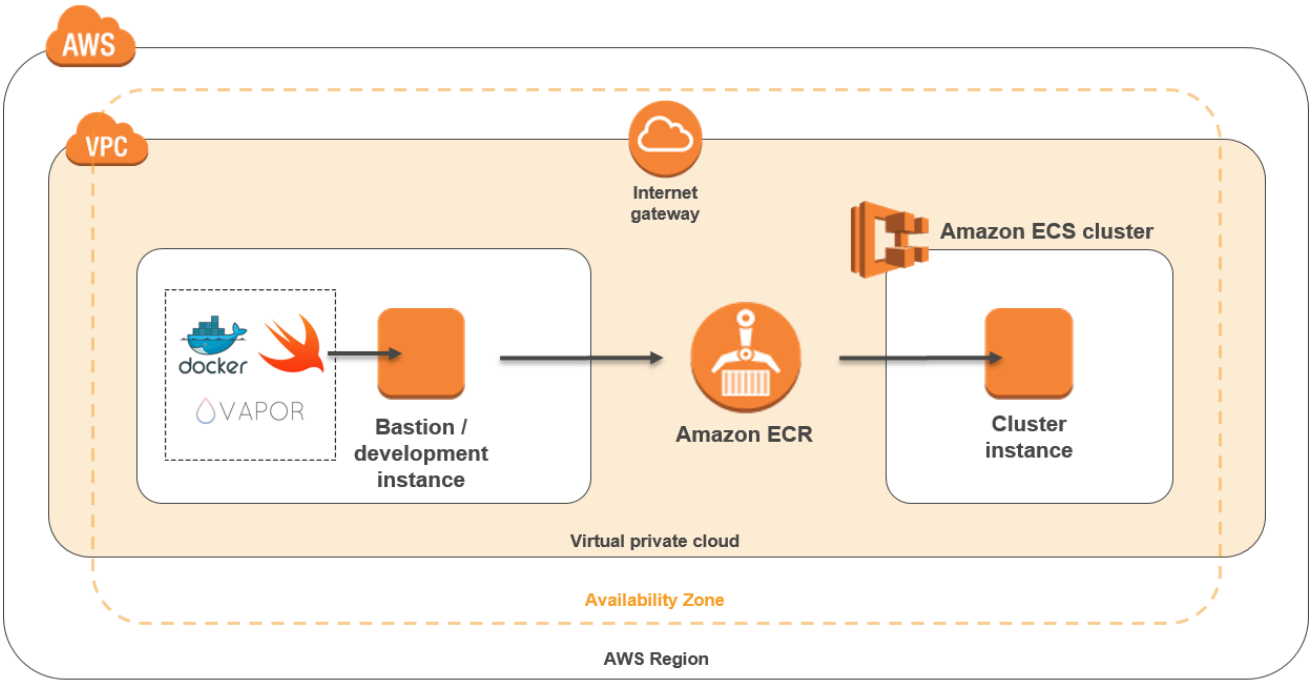

Figure 1: Quick Start architecture for Swift web applications on AWS

The Quick Start architecture consists of the following components:

- VPC – The Quick Start sets up a virtual private cloud (VPC) to provide a virtual network for the deployment.
- Bastion host – The Quick Start deploys a bastion host to provide secure access to the EC2 instances in the VPC. The bastion host includes preinstalled dependencies for building a Swift project. It also includes preinstalled versions of AWS Command Line Interface (AWS CLI), Docker, and Git. This software provides a prebuilt Swift development environment for your convenience.
- Amazon ECS – Amazon EC2 Container Service (Amazon ECS) is a highly scalable, high-performance container management service that supports Docker containers and allows you to easily run applications on a managed cluster of Amazon EC2 instances. Amazon ECS eliminates the need for you to install, operate, and scale your own cluster management infrastructure. You can also define task definitions and services that specify the Docker container images you'd like to run across these clusters.
- Amazon ECR – Amazon EC2 Container Registry (Amazon ECR) is a fully managed Docker container registry that makes it easy for developers to store, manage, and deploy Docker container images.

After you run the Quick Start to set up this architecture, you can follow the step-by-step instructions in this guide to configure the Swift development environment and to run a demo app. The demo app displays example Swift applications deployed with Vapor on Amazon ECS. You can extend the demo app to develop APIs or other services.

Prerequisites

Specialized Knowledge

Before you deploy this Quick Start, we recommend that you become familiar with the following AWS services. (If you are new to AWS, see Getting Started with AWS.)

- Amazon VPC
- Amazon EC2
- Amazon ECS
- Amazon ECR

The Quick Start also assumes familiarity with Swift and Vapor.

Deployment Steps

Step 1. Prepare an AWS Account

1. If you don't already have an AWS account, create one at http://aws.amazon.com by following the on-screen instructions.
2. Use the region selector in the navigation bar to choose the AWS Region where you want to deploy Swift web applications on AWS.
3. Create a key pair in your preferred region.

Step 2. Launch the Quick Start

1. Deploy the AWS CloudFormation template into your AWS account.

   Launch

   The template is launched in the US West (Oregon) region by default. You can change the region by using the region selector in the navigation bar.

   This stack takes approximately 10 minutes to create.

> **Note** You are responsible for the cost of the AWS services used while running this Quick Start reference deployment. There is no additional cost for using this Quick Start. See the pricing pages for each AWS service you will be using or the AWS Simple Monthly Calculator for full details.

You can also download the template to use it as a starting point for your own implementation.

2. On the **Select Template** page, keep the default setting for the template URL, and then choose **Next**.
3. On the **Specify Details** page, review the parameters for the template. Enter values for the parameters that require your input. For all other parameters, you can customize the default settings provided by the template.

Parameter	Default	Description
ECRRepository	*Requires input*	The name of the Amazon ECR repository that will contain the Docker container image.
InstanceType	t2.small	The EC2 instance type for the web server.
KeyName	*Requires input*	Public/private key pair, which allows you to connect securely to your instance after it launches. When you created an AWS account, this is the key pair you created in your preferred region.
KeyPairPath	*Requires input*	The path to the private key (.pem) file associated with the **KeyName** parameter that you've saved on your computer.
SSHLocation	*Requires input*	The IP address range that can be used to whitelist clients connecting to the bastion host using SSH. We recommend that you set this value to a trusted IP range (for example, to restrict access to your corporate network.) If you use 0.0.0.0/0, your EC2 instances will be open to public Internet access.

When you finish reviewing and customizing the parameters, choose **Next**.

4. On the **Options** page, you can specify tags (key-value pairs) for resources in your stack and set advanced options. When you're done, choose **Next**.
5. On the **Review** page, review and confirm the template settings. Under **Capabilities**, select the check box to acknowledge that the template will create IAM resources.
6. Choose **Create** to deploy the stack.
7. Monitor the status of the stack. When the status is **CREATE_COMPLETE**, the deployment is complete.

8. You can use the URL displayed in the **Outputs** tab for the stack to view the resources that were created.

The AWS CloudFormation stack also outputs a few commands that you'll need during the demo in step 3. You'll also need the cluster IP Address to connect to the application for the final step of the demo.

Step 3. Configure Your Swift Development Environment

1. Connect to the bastion instance by using Secure Shell (SSH):

```
{{SSHToBastion}}
```

2. Build, tag, and push a Docker image to Amazon ECS.
   a. Retrieve the Docker login command:

```
aws ecr get-login
```

   b. Run the output of the previous command to log in to Docker.
   c. Build a Docker image from your Swift web app:

```
docker build \
  -t swift-on-ecs \
  --build-arg SWIFT_VERSION=DEVELOPMENT-SNAPSHOT-2016-06-06-a \
  --build-arg REPO_CLONE_URL=<your-GitHub-repo> \
  ~/swift-on-ecs/demo/example
```

   You can find an example demo at https://github.com/thomasphorton/swift-on-ecs.git. Or you can extend one of the sample projects provided in the Vapor GitHub repository.

   d. When the build is complete, tag the image:

```
{{TagPreBuiltImage}}
```

   e. Push the image to the Amazon ECR repository:

```
{{PushPreBuiltImage}}
```

3. Create a task definition. This definition is required to run Docker containers in Amazon ECS. For more information about each step, see the Amazon ECS documentation.
   a. Open the Amazon ECS console at https://console.aws.amazon.com/ecs/.

b. In the navigation pane, choose **Task Definitions**.

c. On the **Task Definitions** page, choose **Create a New Task Definition**.

d. For the task definition name, enter: `swift-on-ecs-task`.

e. To add a container, choose **Add Container Definition** and fill out these fields:

- Image: `{{RepositoryURL}}:latest`
- Maximum memory: `300`
- Host port: `80`
- Container port: `8080`
- Protocol: `tcp`

f. Choose **Add Container** to add your container to the task definition.

g. Choose **Create**.

4. Create a service in Amazon ECS. The service enables you to specify the desired number of task definition instances you'd like to maintain in your cluster, to protect against task failures. For more information about services, see the Amazon ECS documentation.

   In the Amazon ECS console, in the **Task Definition name** page for the task definition you created in the previous step:

   a. Choose the task definition revision: `swift-on-ecs-task:1`.

   b. For service name, enter: `swift-on-ecs-service`.

   c. For the number of tasks, enter: `1`.

   d. Choose **Create Service**.

5. Wait for the service to stabilize, and then connect to the cluster instance in a browser with **http://<clusterip>:8080**. You should see the Vapor demo page.

> **Optional** You can also associate a service in Amazon ECS with an application load balancer for the Elastic Load Balancing service, to distribute traffic across the tasks in your service. The application load balancer supports a target group that contains a set of instances and ports.

Step 4. Clean Up

After you run the demo, you can either reset the demo so that it can be run again, or remove the demo.

To reset the demo:

1. Scale the service down to zero running tasks by following the instructions in the Amazon EC2 documentation.
2. In the AWS CloudFormation console, delete the AWS CloudFormation stack associated with this Quick Start.
3. Create a new stack by repeating step 2 earlier in this guide.

To remove the demo:

1. Scale the service down to zero running tasks by following the instructions in the Amazon EC2 documentation.
2. In the AWS CloudFormation console, delete the AWS CloudFormation stack associated with this Quick Start.

Troubleshooting FAQ

Q. I cannot connect to Amazon ECR.

A. If you cannot connect to Amazon ECR, check your AWS CLI settings. Use the `aws configure` command to input your access key and secret access key. This will set up your account credentials so you can connect to AWS and Amazon ECR.

Q. I cannot compile the Swift project on the bastion host.

A. If you cannot compile the project, check the Swift version by using the `swift -version` command, and make sure that it matches the version listed in `docker-compose`.

Q. I encountered a CREATE_FAILED error when I launched the Quick Start. What should I do?

A. If AWS CloudFormation fails to create the stack, we recommend that you relaunch the template with **Rollback on failure** set to **No**. (This setting is under **Advanced** in the AWS CloudFormation console, **Options** page.) With this setting, the stack's state will be retained and the instance will be left running, so you can troubleshoot the issue. (You'll want to look at the log files in `%ProgramFiles%\Amazon\EC2ConfigService` and `C:\cfn\log`.)

> **Important** When you set **Rollback on failure** to **No**, you'll continue to incur AWS charges for this stack. Please make sure to delete the stack when you've finished troubleshooting.

For additional information, see Troubleshooting AWS CloudFormation on the AWS website or contact us on the AWS Quick Start Discussion Forum.

Additional Resources

AWS services

- AWS CloudFormation
  http://aws.amazon.com/documentation/cloudformation/
- Amazon EC2
  http://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/
- Amazon ECS
  https://aws.amazon.com/documentation/ecs/
- Amazon ECR
  http://aws.amazon.com/documentation/ecr

Swift and related resources

- Swift
  https://swift.org/
- Vapor
  http://vapor.codes/
- Docker
  http://www.docker.com/

Quick Start reference deployments

- AWS Quick Start home page
  https://aws.amazon.com/quickstart/
- Community Quick Starts
  https://aws.amazon.com/quickstart/community/

Send Us Feedback

We welcome your questions and comments. Please post your feedback on the AWS Quick Start Discussion Forum.

You can visit our GitHub repository to download the templates and scripts for this Quick Start, and to share your customizations with others.

Notices

This document is provided for informational purposes only. It represents AWS's current product offerings and practices as of the date of issue of this document, which are subject to change without notice. Customers are responsible for making their own independent assessment of the information in this document and any use of AWS's products or services, each of which is provided "as is" without warranty of any kind, whether express or implied. This document does not create any warranties, representations, contractual commitments, conditions or assurances from AWS, its affiliates, suppliers or licensors. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

The software included with this paper is licensed under the Apache License, Version 2.0 (the "License"). You may not use this file except in compliance with the License. A copy of the License is located at http://aws.amazon.com/apache2.0/ or in the "license" file accompanying this file. This code is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.