Architecture Diagrams

Serverless Web Hosting on AWS App Runner

Serverless Web Hosting on AWS App Runner : Architecture Diagrams

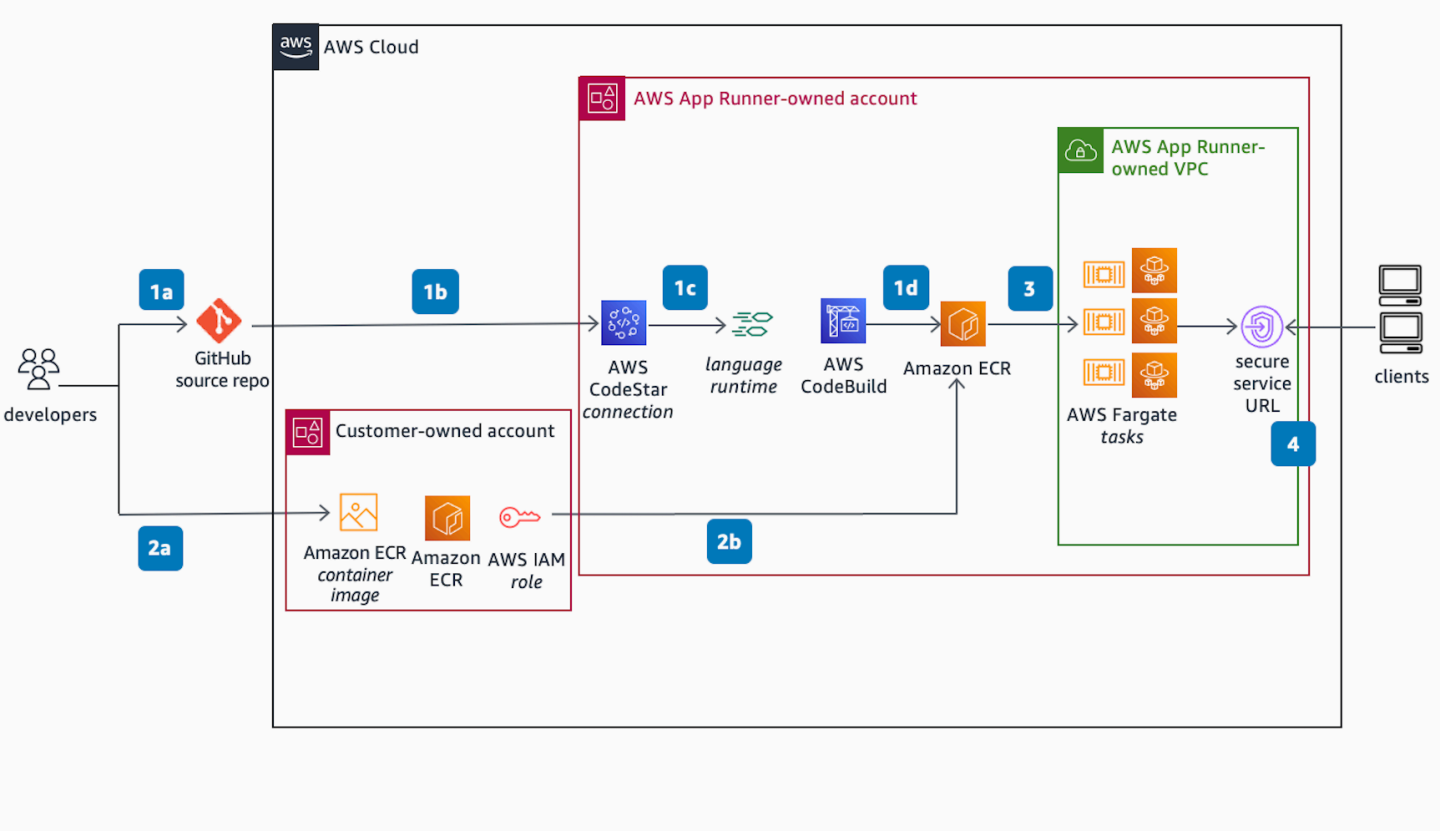
Table of Contents

Serverless Web Hosting on AWS App Runner

Publication date: **May 3, 2023 (Diagram history)**

This reference architecture details two scenarios for how to run serverless, containerized web applications without the need to provision or manage infrastructure.

Serverless Web Hosting on AWS App Runner



Developer experience: Use **AWS App Runner** to create and manage web services based on two types of service sources: source code and source image.

1. **Scenario 1 using a source code:**

   a. Commit code to GitHub source repo.

   b. Authorize access to source code in GitHub using **AWS CodeStar**.

c. Provide instructions for building and running the web service and specify a managed runtime environment supported by **AWS App Runner**. **AWS CodeBuild** then auto-packages the code and its dependencies and builds a new container image.

d. The container image is then pushed and stored into an **Amazon Elastic Container Registry (Amazon ECR)** repository, hosted within the **AWS App Runner**-owned account.

2. **Scenario 2 using a source image:**

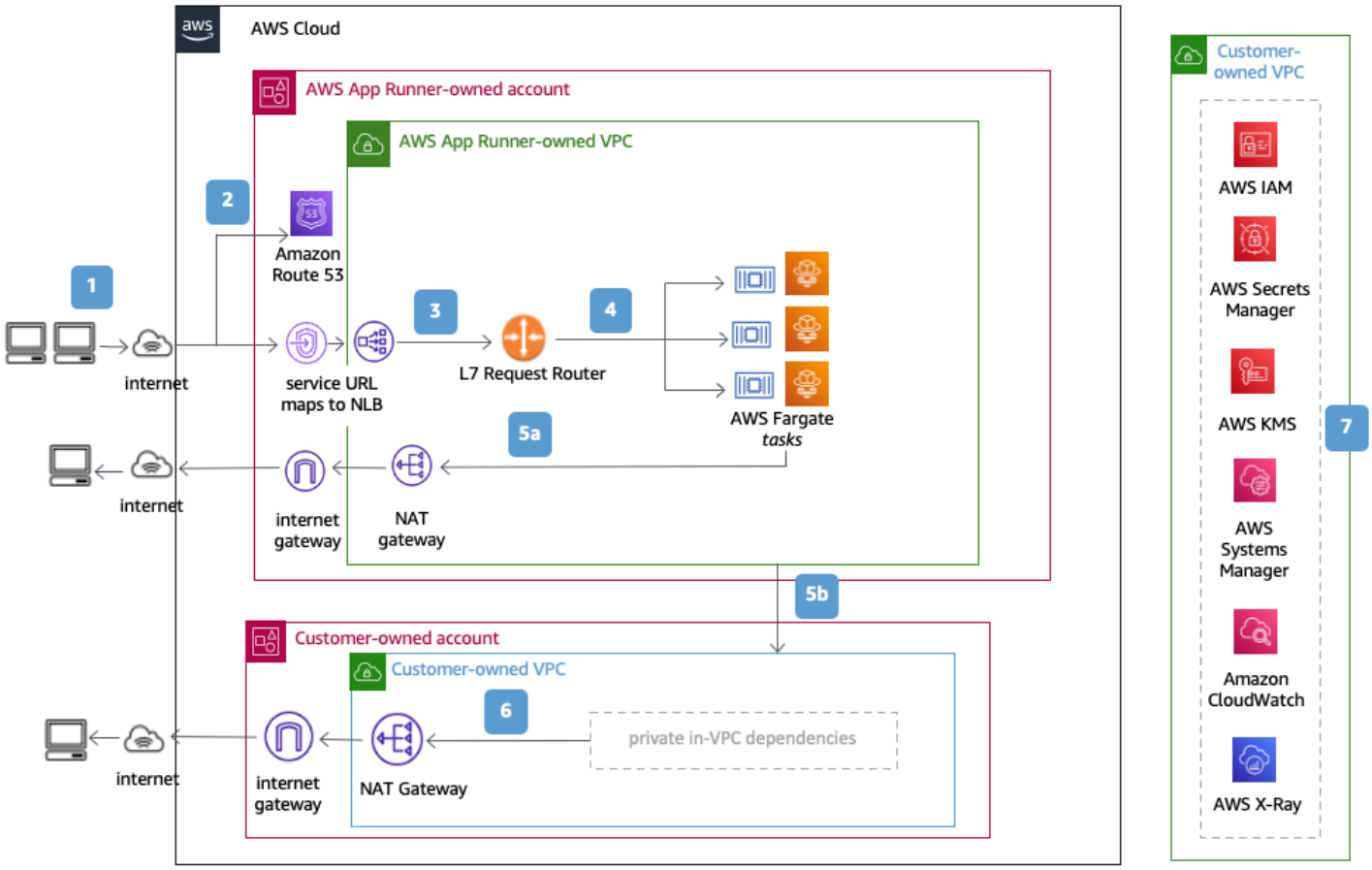
   a. Alternatively, start with a pre-built image stored in a customer-managed **Amazon ECR** repository, hosted within the customer-owned account.

   b. By authorizing access to the container image using an **AWS IAM Identity Center** Role, the image is then pushed and stored into **AWS App Runner**-managed **Amazon ECR** repository.

   App Runner Service creation:

3. The container image stored in **Amazon ECR** is then pulled for deployment as serverless **AWS Fargate** Tasks inside the **AWS App Runner**-owned **Amazon Virtual Private Cloud**, also know as **AWS App Runner** Service VPC. The **AWS Fargate** Tasks are deployed on an **AWS Fargate** cluster running across multiple Availability Zones for high availability in private subnets, and configured with **AWS Application Auto Scaling**.
4. A secure service URL is created/published after a successful deployment. The Secure Service URL maps to a public-facing **AWS Network Load Balancer** and is assigned a default domain name by **AWS App Runner**. Customers can also associate/map a custom domain name they own, but it must be publicly resolvable and can be registered with **Amazon Route 53** Public Hosted Zone or with any DNS provider.

Serverless Web Hosting on AWS App Runner



AWS App Runner is a secure, consistent solution for exposing web applications using the public endpoint or service URL.

Inbound traffic path:

1. Client initiates a request to the service URL.
2. The service URL is resolved to a public-facing **Network Load Balancer** using **Amazon Route 53**. The **AWS NLB** is automatically provisioned and owned by the **AWS App Runner** service.
3. The request enters the **AWS App Runner**-owned VPC through the **AWS Network Load Balancer** (AWS NLB) and is redirected to a L7 Request Router.
4. The L7 Request Router is routes incoming requests to a particular web service instance or **AWS Fargate** Task.

   Outbound traffic path, Scenario 1 (web service doesn't require access to downstream dependencies within a customer/private VPC):

5. a. Outbound (or return) traffic is routed to the internet through an **AWS**-managed NAT Gateway and an internet gateway provisioned within the **AWS App Runner**-owned VPC, or default networking mode.

   Outbound Traffic Path, Scenario 2 (web service requires access to downstream dependencies within a customer/private VPC):

   b. The outbound traffic is forwarded to the customer/private-owned VPC where the private resources/dependencies are manually provisioned by the end-user/developer. Traffic is forwarded to the destination endpoint per the VPC routing table.
6. Traffic to the internet is routed through a customer-managed NAT Gateway and an internet gateway provisioned within the customer/private-owned VPC, or VPC Egress networking mode.
7. **AWS Identity and Access Management** (AWS IAM), **AWS Key Management Service** (AWS KMS), **AWS Secrets Manager**, and **AWS Systems Manager** Parameter Store ensure role-based access and securely store confidential data. **Amazon CloudWatch** and **AWS X-Ray** maintain observability.

Download editable diagram

To customize this reference architecture diagram based on your business needs, download the ZIP file which contains an editable PowerPoint.

Create a free AWS account

Sign up now

Sign up for an AWS account. New accounts include 12 months of AWS Free Tier access, including the use of Amazon EC2, Amazon S3, and Amazon DynamoDB.

Further reading

For additional information, refer to

- AWS Architecture Icons
- AWS Architecture Center
- AWS Well-Architected

Diagram history

To be notified about updates to this reference architecture diagram, subscribe to the RSS feed.

Change	Description	Date
Initial publication	Reference architecture diagram first published.	May 3, 2021

Note

To subscribe to RSS updates, you must have an RSS plugin enabled for the browser you are using.