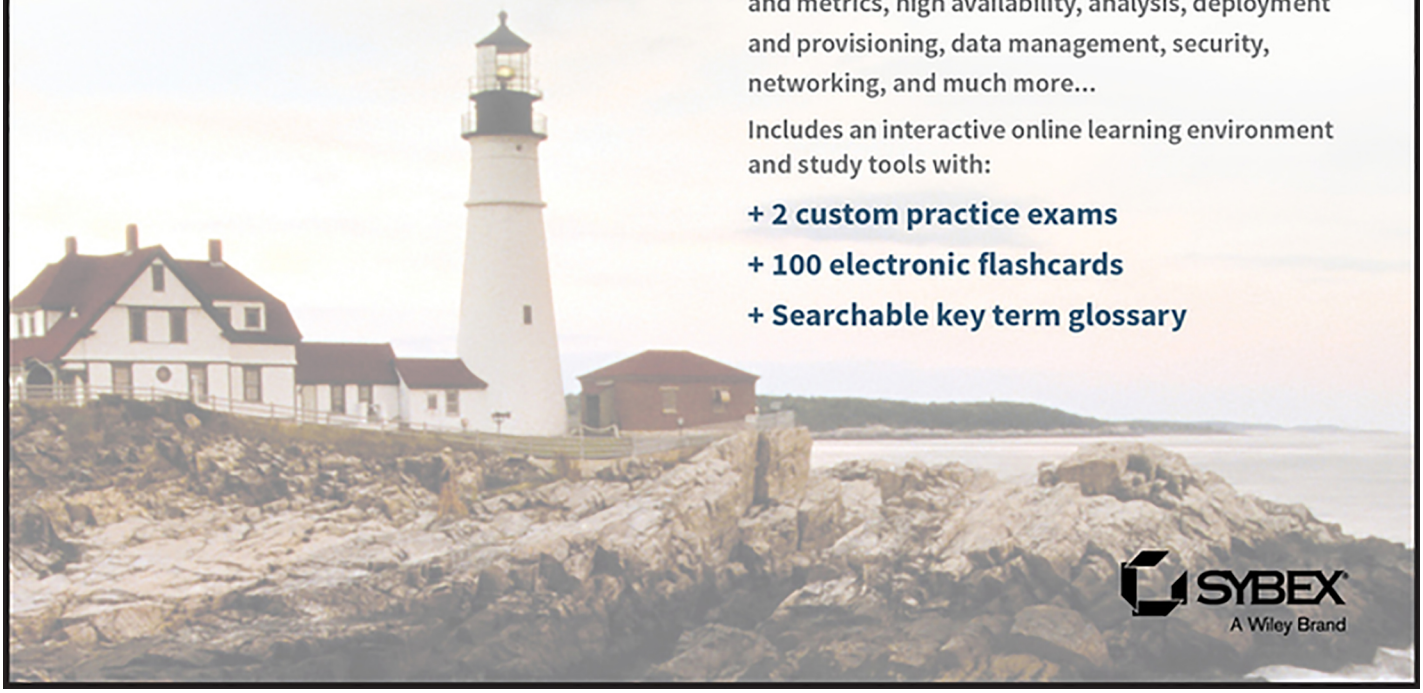
amazon web services

Stephen Cole, Gareth Digby, Chris Fitch,
Steve Friedberg, Shaun Qualheim, Jerry Rhoads,
Michael Roth, Blaine Sundrud

AWS Certified SysOps Administrator

OFFICIAL STUDY GUIDE

ASSOCIATE EXAM



Covers exam objectives, including monitoring and metrics, high availability, analysis, deployment and provisioning, data management, security, networking, and much more...

Includes an interactive online learning environment and study tools with:

+ 2 custom practice exams

+ 100 electronic flashcards

+ Searchable key term glossary

SYBEX
A Wiley Brand

groups to control access on a specific interface. With administrative access, you again can control access with placement of Amazon EC2 instances in an Amazon VPC, the use of security groups, and the use of public/private key pairs both to encrypt traffic and control access.

Controlling Administrative Access

AWS provides a number of tools to manage the security of your instances. These tools include the following:

- IAM
- AWS Trusted Advisor
- AWS Service Catalog
- Amazon Inspector

IAM allows you to create policies that control access to APIs and apply those policies to users, groups, and roles.

AWS Trusted Advisor, which comes as part of AWS Support, looks at things like open ports on security groups and level of access allowed to Amazon EC2 instances, and it makes recommendations to improve the security of your AWS infrastructure.

AWS Service Catalog allows IT administrators to create, manage, and distribute portfolios of approved products to end users who can then access the products they need in a personalized portal. AWS Service Catalog allows organizations to manage commonly deployed IT services centrally, and it helps organizations achieve consistent governance and meet compliance requirements while enabling users to deploy quickly only the approved IT services they need within the constraints the organization sets.

Amazon Inspector is a security vulnerability assessment service that helps improve the security and compliance of your AWS resources. Amazon Inspector automatically assesses resources for vulnerabilities or deviations from best practices and then produces a detailed list of security findings prioritized by level of severity.

AWS also has a number of developer documents, whitepapers, articles, and tutorials to help you in securing your environment. These are available on the Cloud Security Resources page of the AWS website.

Amazon EC2 Container Service (Amazon ECS)

Amazon ECS is a highly scalable container management service. Amazon ECS makes it easy to run, stop, and manage Docker containers on Amazon EC2 instances.

With Amazon ECS, you can launch and place containers across your cluster using API calls. You schedule the placement of containers based on your resource needs, isolation

policies, and availability requirements. You can use Amazon ECS both for cluster management and for scheduling deployments of containers onto hosts.

Amazon ECS eliminates the need for you to operate your own cluster management and configuration management systems. Amazon ECS can be used to manage and scale both batch and Extract, Transform, Load (ETL) workloads. It can also be used to build application architectures based on the microservices model.

Amazon ECS is a regionally-based service that can be used to run application containers in a highly available manner across all Availability Zones within an AWS Region.

Implementation

To implement Amazon ECS, you need to install an Amazon ECS agent on your Amazon EC2 instances. If you use Amazon ECS-optimized AMIs, that agent is already installed. Additionally, the container instance needs to have an IAM role that authenticates to your account and will need external network access to communicate with the Amazon ECS service endpoint.

Clusters are the logical grouping of container instances that you can place tasks on. Clusters are region specific and can contain multiple, different instance types (though an instance can only be a member of a single cluster).

Amazon ECS obtains a Docker image for repository. This repository can be on AWS or on other infrastructure.

Deploying a Container

To deploy a container, you need to do the following:

1. **Define a task**. This is where you assign the name, provide the image name (important for locating the correct image), and decide on the amount of memory and CPU needed.
2. **Define the service**. In this step, you decide how many instances of the task you want to run in the cluster and any Elastic Load Balancing load balancers that you want to register the instances with.
3. **Create the Amazon ECS cluster**. This is where the cluster is created and also where you specify the number of instances to run. The cluster can run across multiple Availability Zones.
4. **Create the stack**. A stack of instances is created based on the configuration information provided. You can monitor the creation of the stack in the AWS Management Console. Creation of the stack is usually completed in less than five minutes.

Management

Amazon ECS can be configured using the AWS Management Console, the AWS CLI, or the Amazon ECS CLI.

Monitoring Amazon ECS

The primary tool used for monitoring your Amazon ECS clusters is Amazon CloudWatch. Amazon CloudWatch collects Amazon ECS metric data in one-minute periods and sends them to Amazon CloudWatch. Amazon CloudWatch stores these metrics for a period of two weeks. You can monitor the CPU and memory reservation and utilization across your cluster as a whole and the CPU and memory utilization on the services in your cluster. You can use Amazon CloudWatch to trigger alarms, set notifications, and create dashboards to monitor the services.

Once it's set up, you can view Amazon CloudWatch metrics in both the Amazon ECS console and the Amazon CloudWatch console. The Amazon ECS console provides a maximum 24-hour view while the Amazon CloudWatch console provides a fine-grained and customizable view of running services.

The other tool available is AWS CloudTrail, which will log all Amazon ECS API calls.

AWS Trusted Advisor is another source for monitoring all of your AWS resources, including Amazon ECS, to improve performance, reliability, and security.

There is no additional cost for using Amazon ECS. The only charges are for the Amazon EC2 instances or AWS Lambda requests and compute time.

Security

With Amazon ECS, you need to do the following:

1. Control who can create task definitions.
2. Control who can deploy clusters.
3. Control who can access the Amazon EC2 instances.

IAM is the tool used for the first two necessities. For controlling access to Amazon EC2 instances, the tools described in the Amazon EC2 section still apply. You can use IAM roles, security groups, and (because these Amazon EC2 instances are located in an Amazon VPC) network Access Control Lists (ACLs) and route tables to control access to the Amazon EC2 instances.

AWS Elastic Beanstalk

AWS Elastic Beanstalk enables you to deploy and manage applications in the AWS Cloud without worrying about the infrastructure that runs those applications. AWS Elastic Beanstalk reduces management complexity without restricting choice or control. You simply upload your application, and AWS Elastic Beanstalk automatically handles the details of capacity provisioning, load balancing, scaling, and application health monitoring.

AWS Elastic Beanstalk provisions resources to support a web application that handles HTTP(S) requests or an application that handles background-processing tasks. An environment tier whose web application processes web requests is known as a *web server tier*. An environment tier whose application runs background jobs is known as a *worker tier*. The environment is the heart of the application. When you create an environment, AWS Elastic Beanstalk provisions the resources required to run your application.

If your application performs operations or workflows that take a long time to complete, you can offload those tasks to a dedicated worker environment. Decoupling your web application front end from a process that performs blocking operations is a common way to ensure that your application stays responsive under load. For the worker environment tier, AWS Elastic Beanstalk also creates and provisions an Amazon Simple Queue Service (Amazon SQS) queue if you don't already have one. When you launch a worker environment tier, AWS Elastic Beanstalk installs the necessary support files for your programming language of choice and a daemon on each Amazon EC2 instance in the Auto Scaling group. The daemon is responsible for pulling requests from an Amazon SQS queue and sending the data to the application running in the worker environment tier that will process those messages.

Environment Platforms

AWS Elastic Beanstalk supports Java, .NET, PHP, Node.js, Python, Packer, Ruby, Go, and Docker, and it is ideal for web applications. Due to AWS Elastic Beanstalk's open architecture, non-web applications can also be deployed using AWS Elastic Beanstalk. Additionally, AWS Elastic Beanstalk supports custom platforms that can be based on an AMI that you create from one of the supported operating systems and that can include further customizations. You can choose to have your AWS Elastic Beanstalk environments automatically updated to the latest version of the underlying platform running your application during a specified maintenance window. AWS Elastic Beanstalk regularly releases new versions of supported platforms with operating system, web and application server, language, and framework updates.

Managing Environments

AWS Elastic Beanstalk makes it easy to create new environments for your application. You can create and manage separate environments for development, testing, and production use, and you can deploy any version of your application to any environment. Environments can be long-running or temporary. When you terminate an environment, you can save its configuration to re-create it later.

As you develop your application, you will deploy it often, possibly to several different environments for different purposes. AWS Elastic Beanstalk lets you configure how deployments are performed. You can deploy to all of the instances in your environment simultaneously or split a deployment into batches with rolling deployments.

Managing Application Versions

AWS Elastic Beanstalk creates an application version whenever you upload source code. This usually occurs when you create a new environment or upload and deploy code using

the environment management console or AWS Elastic Beanstalk CLI. You can also upload a source bundle without deploying it from the application console.

A single-environment deployment of AWS Elastic Beanstalk performs an in-place update when you update your application versions, so your application may become unavailable to users for a short period of time. It is possible to avoid this downtime by performing a blue/green deployment with AWS Elastic Beanstalk, where you deploy the new version to a separate environment and then use AWS Elastic Beanstalk to swap the CNAMEs of the two environments to redirect traffic to the new version instantly.

Blue/green deployments require that your application's environments run independently of your production database—if your application uses one. If your AWS Elastic Beanstalk green environment has an Amazon RDS DB instance included in it, the data will not transfer over to the AWS Elastic Beanstalk blue environment and will be lost if you terminate the original environment.

Amazon EC2 Container Service

Amazon EC2 Container Service (Amazon ECS) is a highly scalable, high-performance container management service that makes it easy to run, stop, and manage Docker containers on a cluster of Amazon EC2 instances. *Docker* is a technology that allows you to build, run, test, and deploy distributed applications that are based on Linux containers. Amazon ECS uses Docker images in task definitions to launch containers on Amazon EC2 instances in your clusters.

Amazon ECS lets you launch and stop container-based applications with simple API calls, allowing you to get the state of your cluster from a centralized service and giving you access to many familiar Amazon EC2 features, such as security groups, Amazon EBS volumes, and IAM roles. Amazon ECS is a good option if you are using Docker for a consistent build and deployment experience or as the basis for sophisticated distributed systems. Amazon ECS is also a good option if you want to improve the utilization of your Amazon EC2 instances.

You can use Amazon ECS to schedule the placement of containers across your cluster based on your resource needs, isolation policies, and availability requirements. Amazon ECS eliminates the need for you to operate your own cluster management and configuration management systems or worry about scaling your management infrastructure. It also can be used to create a consistent deployment and build experience, manage and scale batch and Extract-Transform-Load (ETL) workloads, and build sophisticated application architectures on a microservices model.

While AWS Elastic Beanstalk can also be used to develop, test, and deploy Docker containers rapidly in conjunction with other components of your application infrastructure, using Amazon ECS directly provides more fine-grained control and access to a wider set of use cases.

Clusters

Amazon ECS is a regional service that simplifies running application containers in a highly available manner across multiple Availability Zones within a region. You can create

Amazon ECS clusters within a new or existing Amazon VPC. After a cluster is up and running, you can indicate task definitions and services that specify which Docker container images to run across your clusters. Container images are stored in and pulled from container registries, which may exist within or outside of your AWS infrastructure. A *cluster* is a logical grouping of Amazon EC2 instances on which you run tasks. When running a task, Amazon ECS downloads your container images from a registry that you specify and runs those images on the container instances within your cluster.

Instances

An *Amazon ECS container instance* is an Amazon EC2 instance that is running the Amazon ECS container agent and has been registered into a cluster. When you run tasks with Amazon ECS, your tasks are placed on your active container instances. A container instance must be running the Amazon ECS container agent to register into one of your clusters. If you are using the Amazon ECS-optimized AMI, the agent is already installed. To use a different operating system, install the appropriate agent.

The Amazon ECS container agent makes calls to Amazon ECS on your behalf. You must launch container instances with an instance profile that has an appropriate IAM role. The IAM role must have a policy attached that authenticates to your account and provides the required resource permissions. Additionally, the container instances need external network access to allow the agent to communicate with the Amazon ECS service endpoint. If your container instances do not have public IP addresses, then they must use Network Address Translation (NAT) or an HTTP proxy to provide this access. When the Amazon ECS container agent registers an instance into your cluster, the container instance reports its status as ACTIVE and its agent connection status as TRUE. This container instance can accept run task requests.

The Amazon ECS-optimized AMI is the recommended AMI for you to use to launch your Amazon ECS container instances. The Amazon ECS-optimized AMI is preconfigured and has been tested on Amazon ECS by AWS engineers. It provides the latest, tested, minimal version of the Amazon Linux AMI, Amazon ECS container agent, recommended version of Docker, and container services package to run and monitor the Amazon ECS agent. Typically, the Amazon ECS config file on the instance is modified with a user data script to specify parameters such as the cluster with which to register the instance.

Containers

To deploy applications on Amazon ECS, your application components must be architected to run in containers. A Docker container is a standardized unit of software development that contains everything that your software application needs to run including operating system, configuration, code, runtime, system tools, and system libraries. Containers are lighter in weight and have less memory and computational overhead than virtual machines, so they make it easy to support applications that consist of hundreds or thousands of small, isolated microservices. A properly containerized application is easy to scale and maintain and makes efficient use of available system resources.

Building your application as a collection of tight, focused containers allows you to build them in parallel with strict, well-defined interfaces. With better interfaces between microservices, you have the freedom to improve and even totally revise implementations without fear of breaking running code. Because your application's dependencies are spelled out explicitly and declaratively, less time will be lost diagnosing, identifying, and fixing issues that arise from missing or obsolete packages. Using containers allows you to build components that run in isolated environments, thus limiting the ability of one container to disrupt the operation of another accidentally while still being able to share libraries and other common resources cooperatively. This opportunistic sharing reduces memory pressure and leads to increased runtime efficiency.

Images

Containers are created from a read-only template called an *image*. Images are typically built from a *Dockerfile*, a manifest that describes the base image to use for your Docker image and what you want installed and running on it. When you build the Docker image from your Dockerfile, the images are stored in a registry from which they can be downloaded and run on your container instances.

Repository

Docker uses images that are stored in repositories to launch containers. The default repository for the Docker daemon is Docker Hub. Although you don't need a Docker Hub account to use Amazon ECS or Docker, having a Docker Hub account gives you the freedom to store your modified Docker images so that you can use them in your Amazon ECS task definitions.

Another registry option is Amazon EC2 Container Registry (Amazon ECR). Amazon ECR is a managed AWS Docker registry service. Customers can use the familiar Docker CLI to push, pull, and manage images. Amazon ECR supports private Docker repositories with resource-based permissions using IAM so that specific users or Amazon EC2 instances can access repositories and images.

Tasks

A *task definition* is like a blueprint for your application. Every time you launch a task in Amazon ECS, you specify a task definition so that the service knows which Docker image to use for containers, how many containers to use in the task, and the resource allocation for each container.

Amazon ECS provides three task features:

- A service scheduler for long-running tasks and applications
- The ability to run tasks manually for batch jobs or single-run tasks, with Amazon ECS placing tasks on your cluster for you
- The ability to run tasks on the container instance that you specify so that you can integrate with custom or third-party schedulers or place a task manually on a specific container instance

A task is the instantiation of a task definition on a container instance within your cluster. Before you can run Docker containers on Amazon ECS, you must create a task definition. You can define multiple containers and data volumes in a task definition.

Services

Amazon ECS allows you to run and maintain a specified number (the "desired count") of instances of a task definition simultaneously in an Amazon ECS cluster. This is called a *service*. If any of your tasks should fail or stop for any reason, the Amazon ECS service scheduler launches another instance of your task definition to replace it and maintain the desired count of tasks in the service. In addition to maintaining the desired count of tasks in your service, you can optionally run your service behind a load balancer. The load balancer distributes traffic across the tasks that are associated with the service.

The service scheduler is ideally suited for long-running, stateless services and applications. You can update your services that are maintained by the service scheduler, such as deploying a new task definition or changing the running number of desired tasks. By default, the service scheduler spreads tasks across Availability Zones, but you can use task placement strategies and constraints to customize task placement decisions.

AWS OpsWorks Stacks

AWS OpsWorks is an application management service that makes it easy for developers and operations personnel to deploy and operate applications of all shapes and sizes. AWS OpsWorks works best if you want to deploy your code, have some abstraction from the underlying infrastructure, and have an application more complex than a Three-Tier architecture. AWS OpsWorks is also recommended if you want to manage your infrastructure with a configuration management system such as Chef.

AWS OpsWorks Stacks provides a simple and flexible way to create and manage stacks and applications. It has a rich set of customizable components that you can mix and match to create a stack that satisfies your specific purposes. Additionally, if you have existing computing resources, you can incorporate them into a stack along with instances that you created with AWS OpsWorks Stacks. Such resources can be existing Amazon EC2 instances or even on-premises instances that are running on your own hardware. You can then use AWS OpsWorks Stacks to manage all related instances as a group, regardless of how they were created.

Stacks

The *stack* is the top-level AWS OpsWorks Stacks entity. It represents a set of instances that you want to manage collectively, typically because they have a common purpose such as serving PHP applications. In addition to serving as a container, a stack handles tasks that apply to the group of instances as a whole, such as managing applications and cookbooks. For example, a stack whose purpose is to serve web applications might contain a set of application server instances, a load balancer to direct traffic to the instances, and a database instance that serves as a back end of the application instances. A common practice is to have multiple stacks that represent different environments (such as development, staging, and production stacks).