Other Services

What is CloudFormation

- CloudFormation is a declarative way of outlining your AWS Infrastructure, for any resources (most of them are supported).
- For example, within a CloudFormation template, you say:
  - I want a security group
  - I want two EC2 instances using this security group
  - I want an S3 bucket
  - I want a load balancer (ELB) in front of these machines
- Then CloudFormation creates those for you, in the right order, with the exact configuration that you specify.

Benefits of AWS CloudFormation (1/2)

Infrastructure as Code

- No resources are manually created, which is excellent for control.
- Changes to the infrastructure are reviewed through code.

Cost

- Each resource within the stack is tagged with an identifier, so you can easily see how much a stack costs.
- You can estimate the costs of your resources using the CloudFormation template.
- Savings strategy: In development environments, you can automate the deletion of templates at 5 PM and recreate them at 8 AM, safely.

Benefits of AWS CloudFormation (2/2)

Productivity

- Ability to destroy and re-create an infrastructure on the cloud on the fly.
- Automated generation of diagrams for your templates.
- Declarative programming: no need to figure out ordering and orchestration.

Don’t Re-invent the Wheel

- Leverage existing templates available on the web.
- Leverage the official documentation.

Resource Support

- Supports (almost) all AWS resources.
- Everything covered in this course is supported.
- You can use “custom resources” for resources that are not supported.

CloudFormation + Infrastructure Composer

Example: WordPress CloudFormation Stack

- Allows you to visualize all the resources.
- Shows the relationships between the components.

CloudFormation Infrastructure Composer

- A tool that helps you design and visualize your CloudFormation templates more easily.

CloudFormation – Service Role

- An IAM role that allows CloudFormation to create, update, or delete stack resources on your behalf.

Benefits

- Enables users to manage stack resources (create/update/delete) without having direct permissions on those resources.
- Useful for enforcing the **least privilege principle**.

Use Case

- You want to allow a user to deploy infrastructure using CloudFormation without granting them full permissions to all AWS services.

Requirements

- The user must have the `iam:PassRole` permission.
- The CloudFormation service role must have permissions such as:
  - `cloudformation:*`
  - `s3:*`

Amazon Simple Email Service (Amazon SES)

- Fully managed service to send emails securely, globally, and at scale.
- Supports both inbound and outbound email capabilities.

Features

- Reputation dashboard, performance insights, and anti-spam feedback.
- Provides statistics such as:
  - Email deliveries
  - Bounces
  - Feedback loop results
  - Email opens
- Supports email authentication protocols:
  - DKIM (DomainKeys Identified Mail)
  - SPF (Sender Policy Framework)
- Flexible IP deployment options:
  - Shared IPs
  - Dedicated IPs
  - Customer-owned IPs
- Emails can be sent using:
  - AWS Console
  - APIs
  - SMTP

Use Cases

- Transactional emails
- Marketing campaigns
- Bulk email communications

Amazon Pinpoint

- Scalable 2-way (outbound/inbound) marketing communications service.

Supported Channels

- Email
- SMS
- Push notifications
- Voice messages
- In-app messaging

Features

- Allows segmentation and personalization of messages with targeted content.
- Supports receiving customer replies.
- Scales to billions of messages per day.

Use Cases

- Running marketing campaigns.
- Sending bulk or transactional SMS messages.

Comparison with Amazon SNS and Amazon SES

- With **SNS/SES**, you manage:
  - Each message's audience
  - Message content
  - Delivery schedule
- With **Pinpoint**, you can:
  - Create message templates
  - Define delivery schedules
  - Build highly-targeted segments
  - Run full-scale campaigns

Integration

- Can stream events (e.g., `TEXT_SUCCESS`, `TEXT_DELIVERED`) to services like:
  - Amazon SNS
  - Kinesis Data Firehose
  - CloudWatch Logs

Systems Manager – SSM Session Manager

- Allows you to start a secure shell session on your EC2 and on-premises servers.

Key Features

- No need for SSH access, bastion hosts, or SSH keys.
- No requirement for port 22 (improved security).
- Supports Linux, macOS, and Windows instances.
- Can send session log data to:
  - Amazon S3
  - CloudWatch Logs

How It Works

- The EC2 instance runs the SSM Agent.
- Users must have proper IAM permissions.
- Session Manager allows executing commands securely without opening SSH access.

Systems Manager – Run Command

- Enables you to execute a document (script) or run individual commands on EC2 instances.

Features

- Run commands across multiple instances using resource groups.
- No need for SSH access.
- Command output can be:
  - Viewed directly in the AWS Console
  - Sent to an S3 bucket
  - Sent to CloudWatch Logs
- Notifications can be sent to Amazon SNS regarding command status (e.g., In progress, Success, Failed).
- Integrated with:
  - IAM (for access control)
  - CloudTrail (for auditing)
- Can be triggered programmatically via Amazon EventBridge.

Systems Manager – Patch Manager

- Automates the process of patching managed instances.

Supported Updates

- Operating System (OS) updates
- Application updates
- Security updates

Supported Systems

- EC2 instances
- On-premises servers
- Linux, macOS, and Windows

Features

- Supports patching on-demand or based on a schedule using **Maintenance Windows**.
- Can scan instances to generate **patch compliance reports** (e.g., list of missing patches).

Integration

- Can be triggered via:
  - AWS Console
  - AWS SDK
  - Run Command (`AWS-RunBatchBaseline`)

Systems Manager – Maintenance Windows

- Defines a schedule for performing actions on your instances.

Example Use Cases

- Operating System patching
- Updating drivers
- Installing software

Components of a Maintenance Window

- **Schedule**: When the window starts (e.g., every 24 hours).
- **Duration**: How long the window lasts.
- **Registered Instances**: The instances on which tasks will run.
- **Registered Tasks**: The operations to be executed (e.g., Run Command).
- Works with EC2 instances that have the SSM Agent installed.

Systems Manager – Automation

- Simplifies common maintenance and deployment tasks on EC2 instances and other AWS resources.

Example Tasks

- Restart instances
- Create an AMI
- Take an EBS snapshot

Automation Runbook

- Also known as **SSM Documents**.
- Define the actions to perform on EC2 instances or AWS resources.
- Can be pre-defined by AWS or custom-defined by the user.

Trigger Methods

- Manually via:
  - AWS Console
  - AWS CLI
  - AWS SDK
- Automatically via:
  - Amazon EventBridge
  - Maintenance Windows
  - AWS Config for rules remediations

Supported Resources

- EC2
- EBS
- AMI
- RDS
- Other AWS resources

Cost Explorer

- Visualize, understand, and manage your AWS costs and usage over time.

Features

- Create custom reports to analyze cost and usage data.
- Analyze data at different levels:
  - High-level overview across all accounts
  - Monthly, hourly, or per-resource granularity
- Identify and choose optimal **Savings Plans** to reduce costs.
- Forecast future usage and spending up to 12 months based on historical data.

Cost Explorer – Monthly Cost by AWS Service

- Allows you to break down and visualize your AWS monthly costs by individual services.
- Helps identify which AWS services contribute the most to your total spending.
- Useful for budgeting, cost optimization, and identifying unexpected charges.
- Enables tracking of trends in service usage and costs over time.

Cost Explorer – Hourly & Resource Level

- Enables detailed analysis of AWS costs and usage at the hourly and individual resource level.
- Helps pinpoint specific resources or time periods that drive costs.
- Useful for:
  - Identifying underutilized or expensive resources
  - Performing detailed cost attribution
  - Supporting chargeback or showback models in organizations

Cost Explorer – Savings Plan

- Savings Plans are an alternative to Reserved Instances for reducing AWS costs.

Key Points

- Provide flexible pricing models in exchange for a commitment to a consistent amount of usage (measured in $/hour) over a 1 or 3-year term.
- Automatically apply to matching usage across compute services.
- Cost Explorer helps you:
  - Analyze your current usage
  - Identify potential savings
  - Choose the optimal Savings Plan based on historical usage patterns

Cost Explorer – Forecast Usage

- Allows forecasting of AWS usage and costs up to 12 months into the future.
- Forecasts are based on historical usage data and trends.
- Useful for:
  - Budget planning
  - Financial forecasting
  - Predicting future infrastructure needs
- Helps identify when usage might exceed budget thresholds in advance.

AWS Cost Anomaly Detection

- Continuously monitors your AWS cost and usage using machine learning to detect unusual spending patterns.

Features

- Learns your unique historical spend patterns.
- Automatically detects:
  - One-time cost spikes
  - Continuous cost increases
- No need to manually define thresholds.

Monitoring Scope

- AWS services
- Member accounts
- Cost allocation tags
- Cost categories

Notifications & Reports

- Sends anomaly detection reports including root-cause analysis.
- Get notified via:
  - Individual alerts
  - Daily or weekly summaries using Amazon SNS

Benefits

- Helps reduce cost surprises.
- Allows for root-cause analysis to understand the impact on your AWS bill.

AWS Outposts

- AWS Outposts enable hybrid cloud architecture by extending AWS infrastructure and services to on-premises environments.

Use Case

- Designed for businesses that operate both on-premises infrastructure and cloud infrastructure.

Traditional Challenges

- Two separate management systems:
  - One for AWS cloud (AWS Console, CLI, APIs)
  - One for on-premises systems

What is AWS Outposts?

- Physical **server racks** installed in your on-premises data center.
- Provide the same:
  - AWS infrastructure
  - AWS services
  - AWS APIs and tools
- Enables building and running applications on-premises using familiar AWS tools and services.

Management and Responsibility

- AWS handles setup and ongoing management of the Outposts racks.
- You are responsible for the **physical security** of the hardware.

Benefit

- Seamless extension of AWS services to your on-premises infrastructure.

AWS Outposts – Benefits and Supported Services

Benefits

- **Low-latency access** to on-premises systems.
- Enables **local data processing** for workloads requiring on-site compute.
- Supports **data residency** requirements (keep data within a specific location).
- Facilitates **easier migration** from on-premises infrastructure to the AWS cloud.
- Provided as a **fully managed service** by AWS.

Supported AWS Services on Outposts

- Amazon EC2
- Amazon EBS
- Amazon S3
- Amazon EKS
- Amazon ECS
- Amazon RDS
- Amazon EMR

AWS Batch

- Fully managed batch processing service at any scale.

Key Features

- Efficiently runs hundreds of thousands of computing batch jobs on AWS.
- A **batch job** is defined by having a clear **start** and **end**, unlike continuous jobs.
- Dynamically launches EC2 instances or Spot Instances as needed.
- Automatically provisions the appropriate amount of compute and memory.

How It Works

- You submit or schedule batch jobs.
- AWS Batch handles provisioning, scheduling, and execution.
- Batch jobs are defined as **Docker images** and run on **ECS**.

Benefits

- Reduces operational overhead.
- Enables cost optimization by using Spot Instances.
- Lets you focus on job logic rather than infrastructure management.

AWS Batch – Simplified Example

Flow Overview

1. **Trigger**: An event (e.g., object upload to S3) initiates the batch job.
2. **AWS Batch**:
    - Schedules and runs the job.
    - Automatically selects compute environment (EC2 or Spot Instance).
3. **ECS**: Executes the batch job using a Docker container.
4. **Amazon S3**: Processed output is stored back into S3.

Summary

- The entire pipeline is managed automatically by AWS Batch.
- Ideal for workflows that require automated, event-driven batch processing with output persistence in S3.

-## Batch vs Lambda

AWS Lambda

- **Time limit** for execution.
- **Limited runtimes** (predefined by AWS).
- **Limited temporary disk space**.
- **Serverless**: No infrastructure to manage.

AWS Batch

- **No time limit** on job execution.
- **Any runtime** supported, as long as it's packaged as a Docker image.
- Uses **EBS or instance store** for disk space.
- Relies on **EC2 instances**, which can be automatically managed by AWS.

Amazon AppFlow

- Fully managed integration service to securely transfer data between SaaS applications and AWS.

Sources

- Salesforce
- SAP
- Zendesk
- Slack
- ServiceNow

Destinations

- AWS services such as:
  - Amazon S3
  - Amazon Redshift
- Non-AWS services such as:
  - Snowflake
  - Salesforce

Triggering Options

- On a schedule
- In response to events
- On demand

Features

- Built-in data transformation capabilities:
  - Filtering
  - Validation
- Secure data transfer:
  - Encrypted over the public internet
  - Or privately over AWS PrivateLink

Benefits

- No need to write custom integration code.
- Quickly leverage SaaS and AWS APIs for data movement.

AWS Amplify – Web and Mobile Applications

- A set of tools and services to develop and deploy scalable full-stack web and mobile applications.

Features

- Authentication
- Storage
- APIs (REST and GraphQL)
- CI/CD (Continuous Integration/Delivery)
- PubSub (real-time data)
- Analytics
- AI/ML Predictions
- Monitoring

Backend Configuration

- Use the **Amplify CLI** to configure backend resources.
- Backend services include:
  - Amazon S3
  - Amazon Cognito
  - AWS AppSync
  - API Gateway
  - DynamoDB
  - AWS Lambda
  - Amazon SageMaker
  - Amazon Lex

Frontend Integration

- Use **Amplify Frontend Libraries** to connect frontend apps to the backend services.

Deployment

- Connect source code from:
  - GitHub
  - AWS CodeCommit
  - Bitbucket
  - GitLab
  - Or upload directly
- Build and deploy using the **Amplify Console**, served via **Amazon CloudFront**.

Instance Scheduler on AWS

- An AWS solution (not a native service) deployed via **CloudFormation** to automate the start/stop of AWS resources.

Purpose

- Reduce costs by automatically stopping resources outside of business hours (e.g., save up to 70%).

Supported Resources

- EC2 instances
- EC2 Auto Scaling Groups
- RDS instances

How It Works

- **Schedules** are stored and managed in a **DynamoDB** table.
- Uses **tags** on resources and **AWS Lambda** functions to control start/stop actions.
- Supports **cross-account** and **cross-region** scheduling.

Use Case Example

- Automatically stop all EC2 instances after 6 PM and start them at 8 AM on weekdays.

More Info

- AWS Instance Scheduler Solution