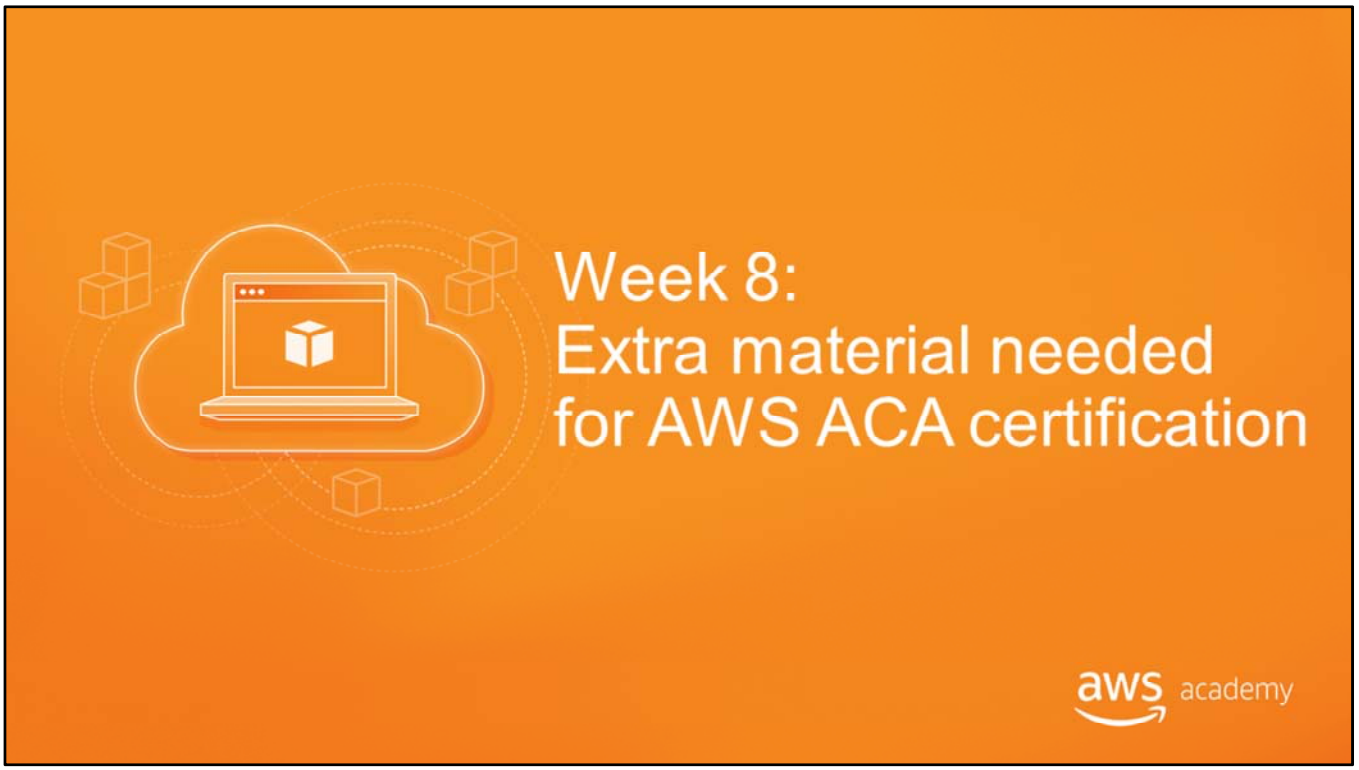
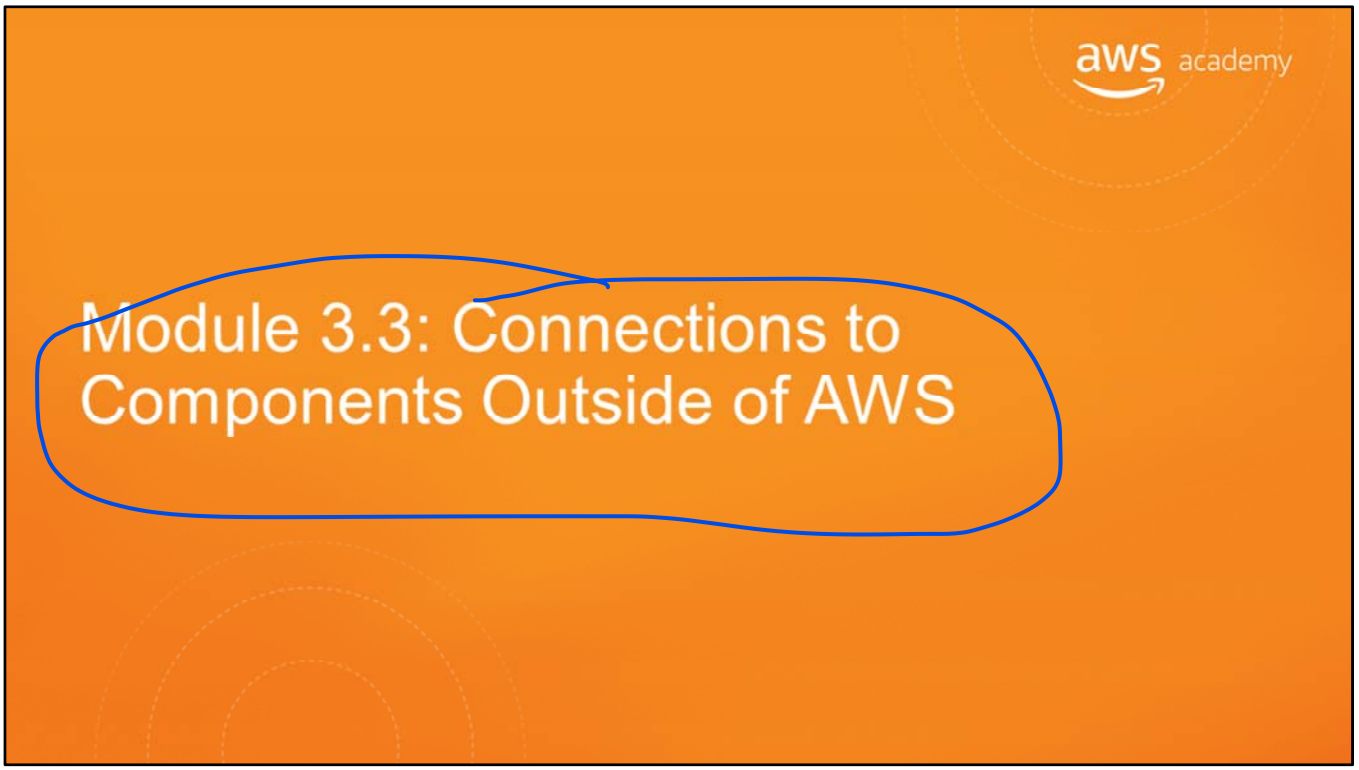



Thanks for participating! You can now exit this module.

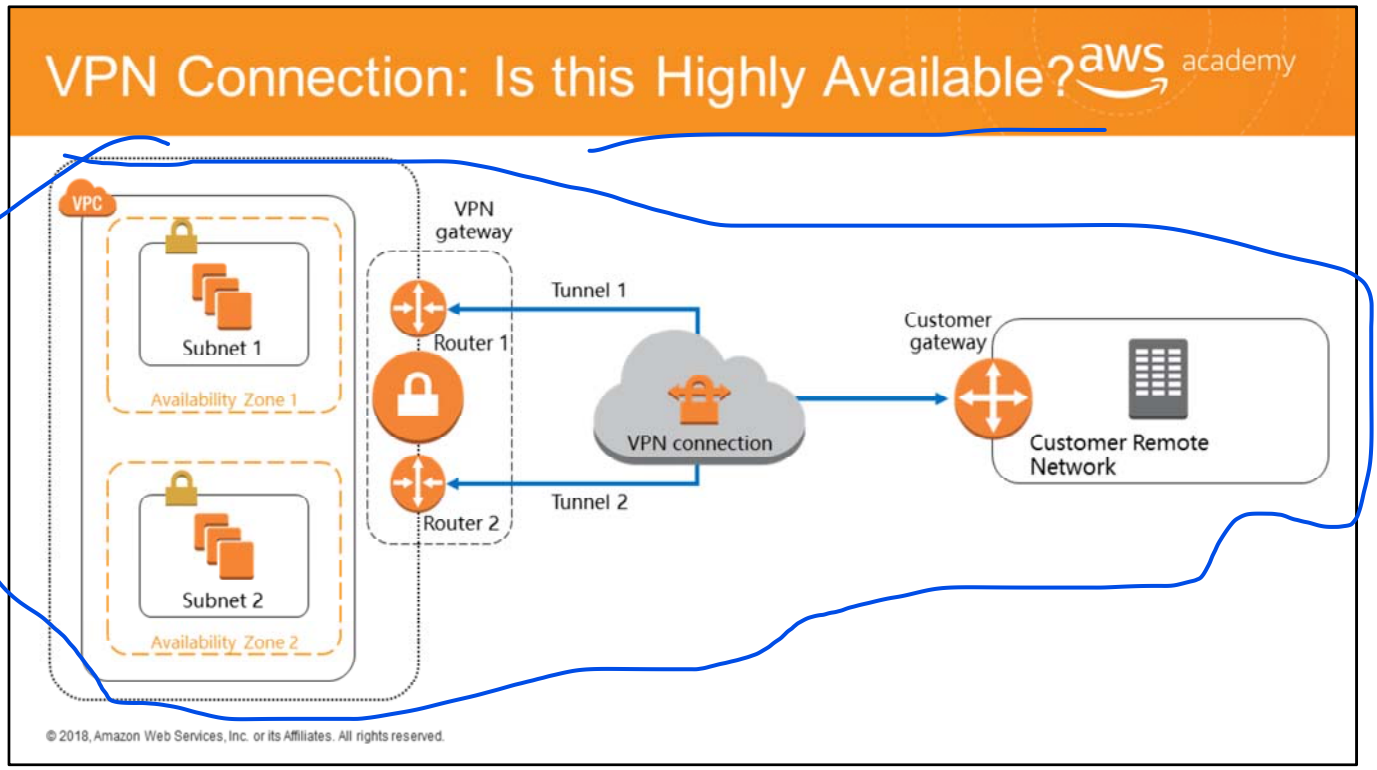
# Extras for ACA certification (not covered in COS20019)

- ACA Mod 3.3: Connections Outside of Amazon VPC
- ACA Mod 4.3: EC2 Auto-recovery
- ACA Mod 4.5: Lambda scaling Case Study
- ACA Mod 5.4: Other AWS Infrastructure as code services

SWINBURNE UNIVERSITY OF TECHNOLOGY



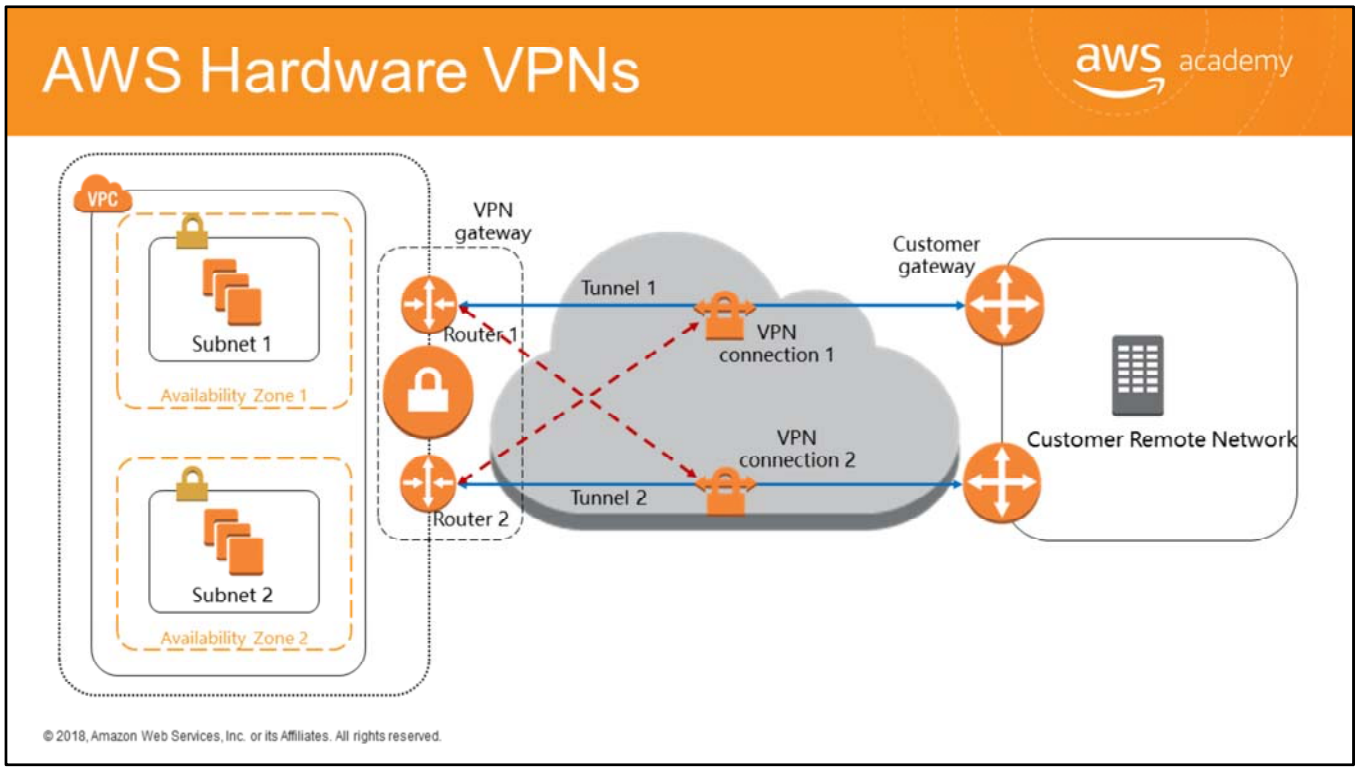

Next, we'll review connections to components outside of AWS.




Virtual private network—or VPN—connections can help you create a highly available connection. In this example, we are looking at the AWS Cloud and a customer network.

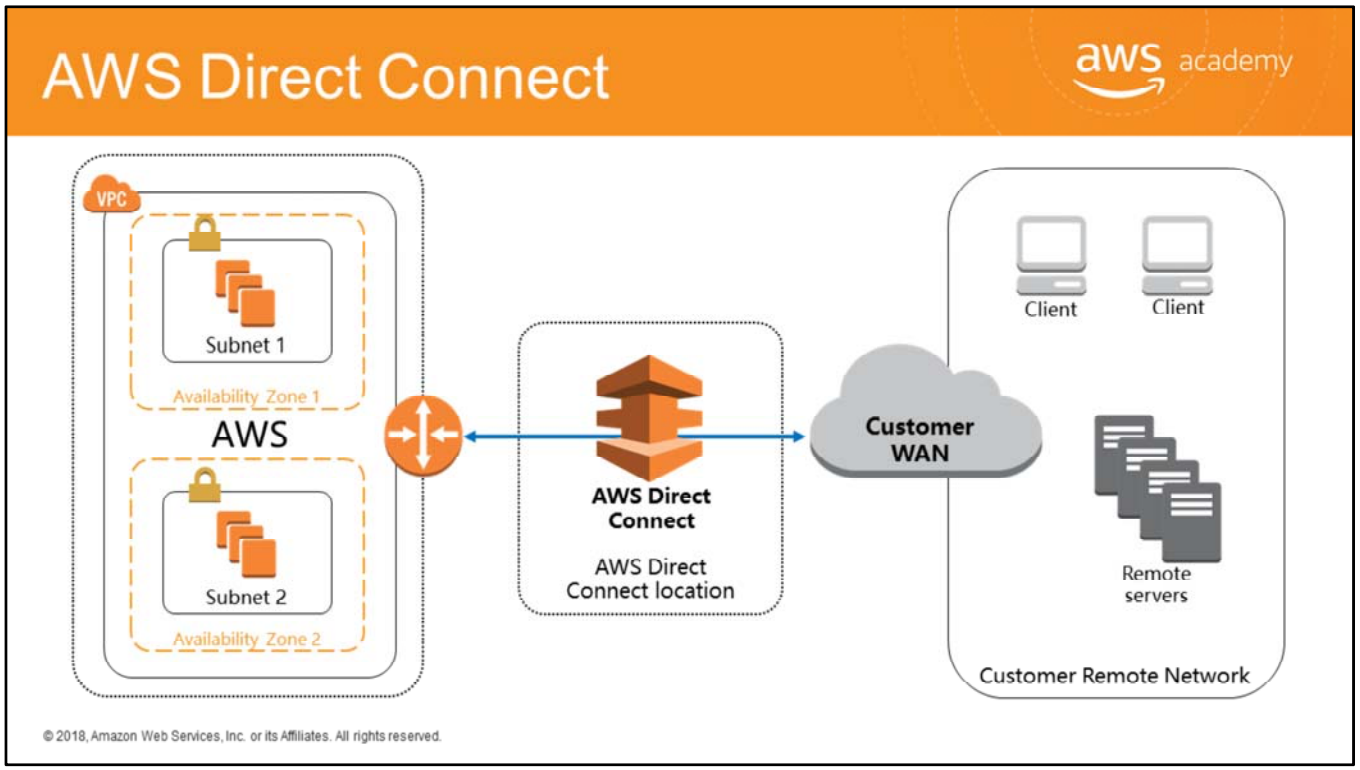
VPN technology is based on the idea of tunneling. VPN tunneling involves establishing and maintaining a logical network connection. This process creates what is known as a "tunnel" between you and the internet, encrypting your internet connection and preventing ISPs, hackers, and even the government from seeing your browsing activity.

In this diagram, each VPN connection has two tunnels, and each tunnel uses a unique virtual private gateway public IP address. This provides automatic, but limited failover protection for your VPN connection. If one tunnel goes down, the other tunnel will still be available. However, if your VPN connection goes down or needs to be taken down for maintenance, your entire connection is severed.



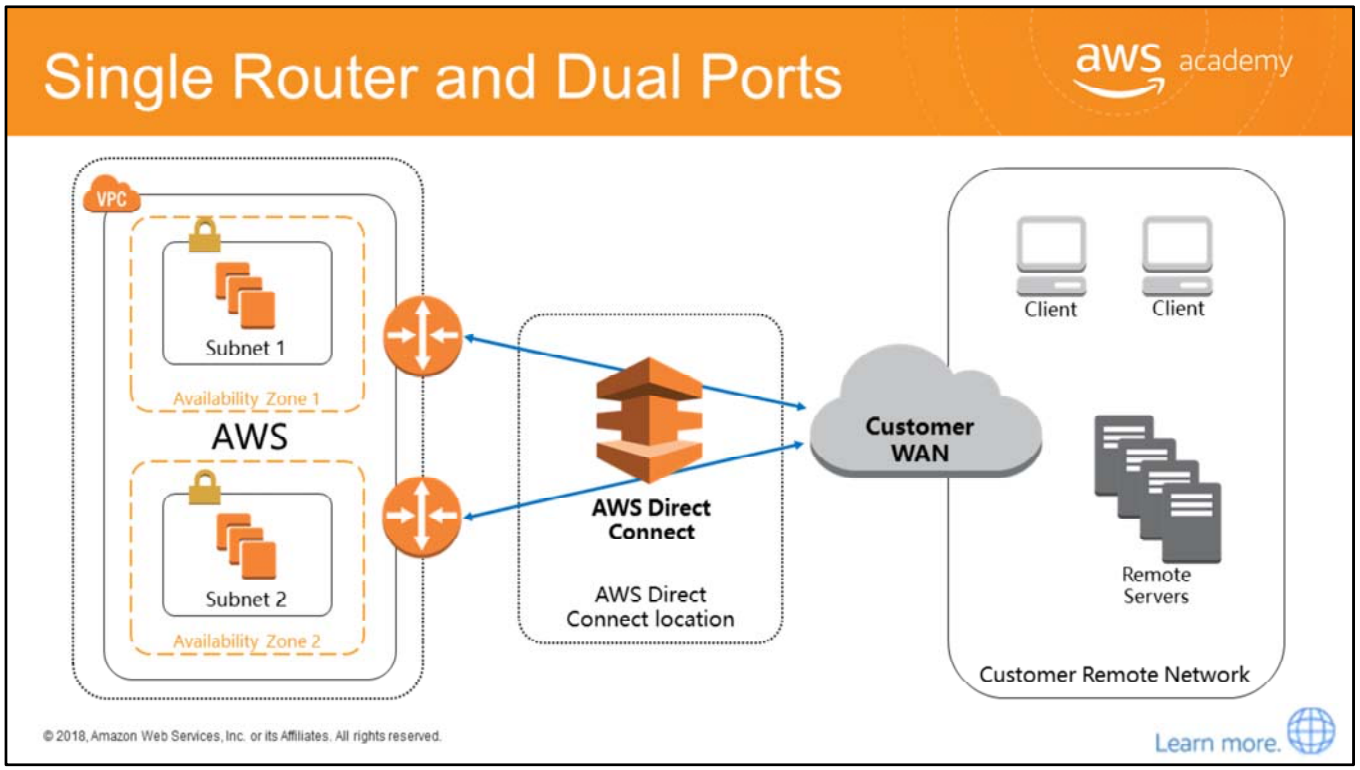

If you lose connections, you lose your connectivity. To protect against a loss of connectivity, or if your customer gateway becomes unavailable, you can use a second customer gateway to set up a second VPN connection to your VPC and virtual private gateway. By using redundant VPN connections and customer gateways, you can perform maintenance on one of your customer gateways while traffic continues to flow over the second customer gateway's VPN connection. To establish redundant VPN connections and customer gateways on your network, you need to set up a second VPN connection. The customer gateway IP address for the second VPN connection must be publicly accessible.

Dynamically routed VPN connections use the Border Gateway Protocol—or BGP—to exchange routing information between your customer gateways and the virtual private gateways. Statically routed VPN connections require you to enter static routes for the network on your side of the customer gateway. BGP-advertised and statically entered route information allow gateways on both sides to determine which tunnels are available, and to reroute traffic if a failure occurs. We recommend that you configure your network to use the routing information provided by BGP, if it’s available, to select an available path. The exact configuration depends on the architecture of your network.



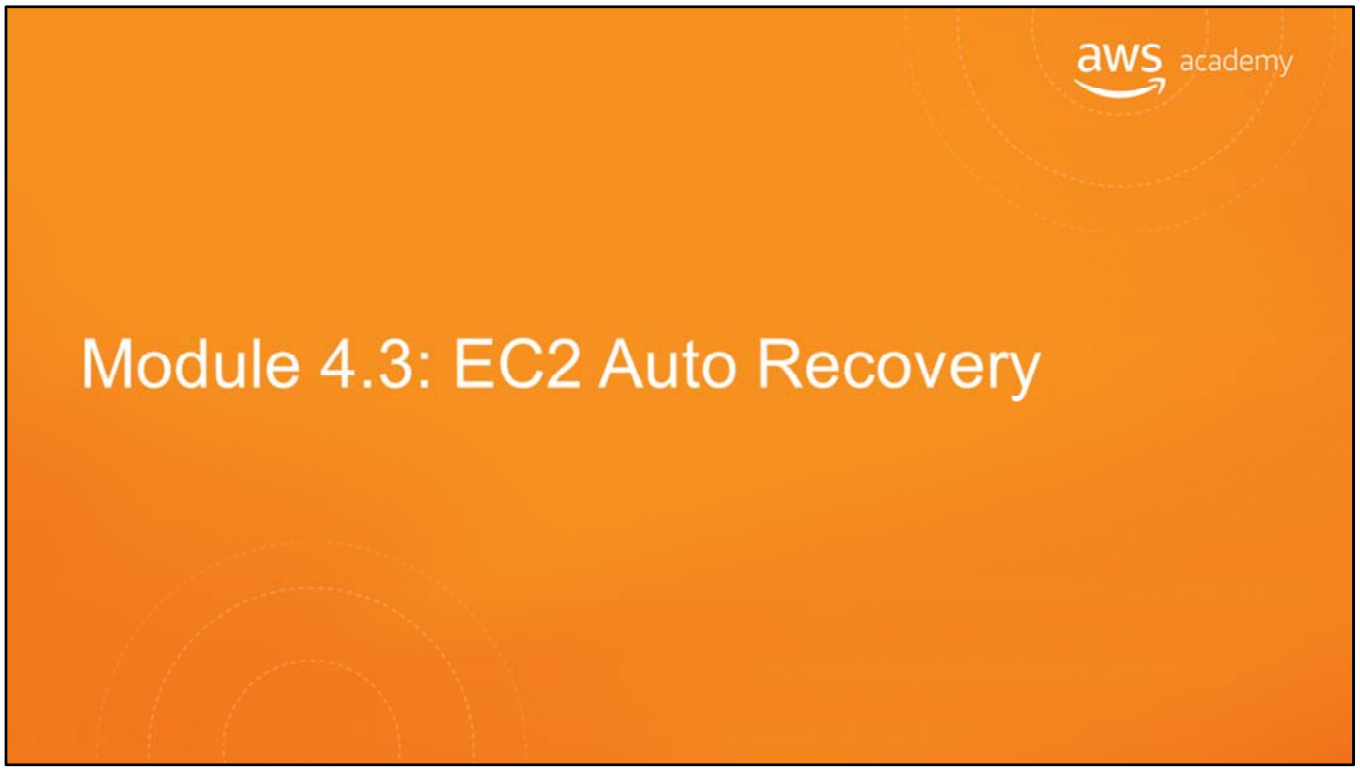

Another option is to use AWS Direct Connect, or DX. Having DX requires planning and configuration, but you will always be able to connect your remote servers to your AWS Cloud.

So, is this solution highly available? No, it is not. AWS Direct Connect is considered highly available, as long as you have two ports available. If you only have one port and the connection is lost, DX could become a single point of failure.

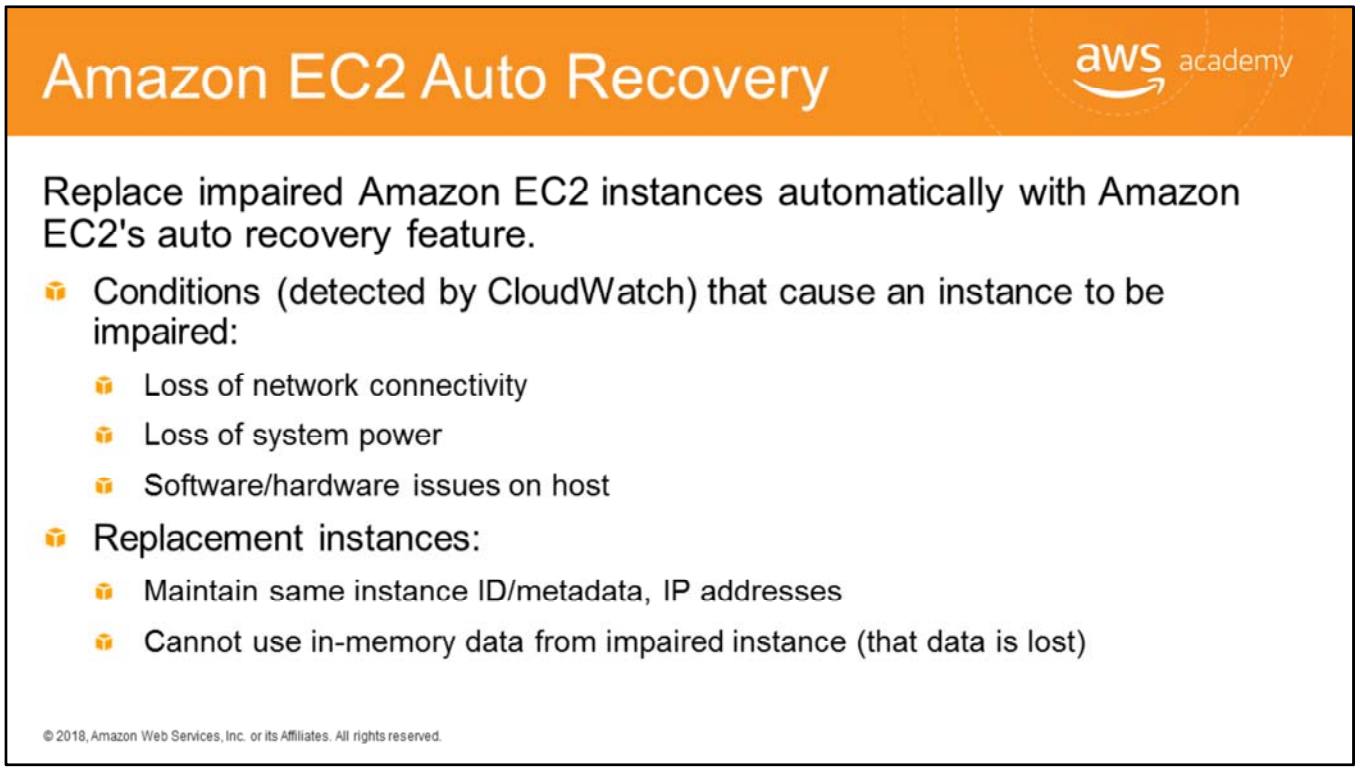



The slide shows a default configuration where you establish two physical links from your router to different AWS Routers, which provides redundancy.

Redundancy is based on the BGP policies. Customers can set BGP policies—or local preference and MED—that causes one link to be preferred for both inbound and outbound traffic. If the local preference and MED values are the same between the two links, traffic will be roughly load=balanced across both.
For more information, including steps for how to set up redundant DX links, see the documentation for getting started with AWS Direct Connect.
http://docs.aws.amazon.com/directconnect/latest/UserGuide/getstarted.html




Next, we'll review connections to components outside of AWS.



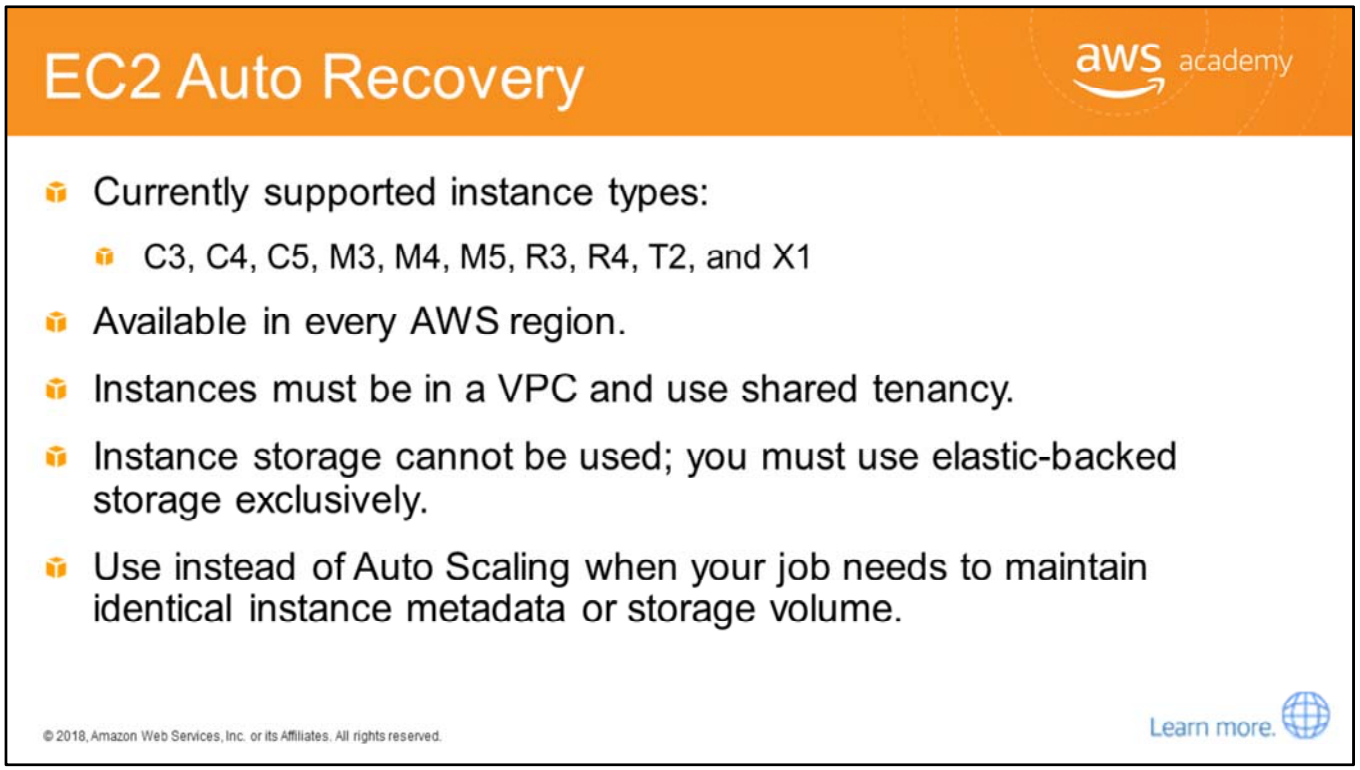

Amazon EC2 Auto Recovery  is a way to automatically repair from a failure with a legacy application that can't be automatically scaled, so you can only have one instance of that type. As long as the system doesn't need to be recovered more than five times per day, the auto recovery feature works great.

Conditions that are detected by CloudWatch, and that can cause an instance to be impaired include:

- Loss of network connectivity
- Loss of system power
- And software or hardware issues on the host

Replacement instances:

- Maintain the same instance ID or metadata, and IP addresses.
- Cannot use in-memory data from an impaired instance. That data is lost.



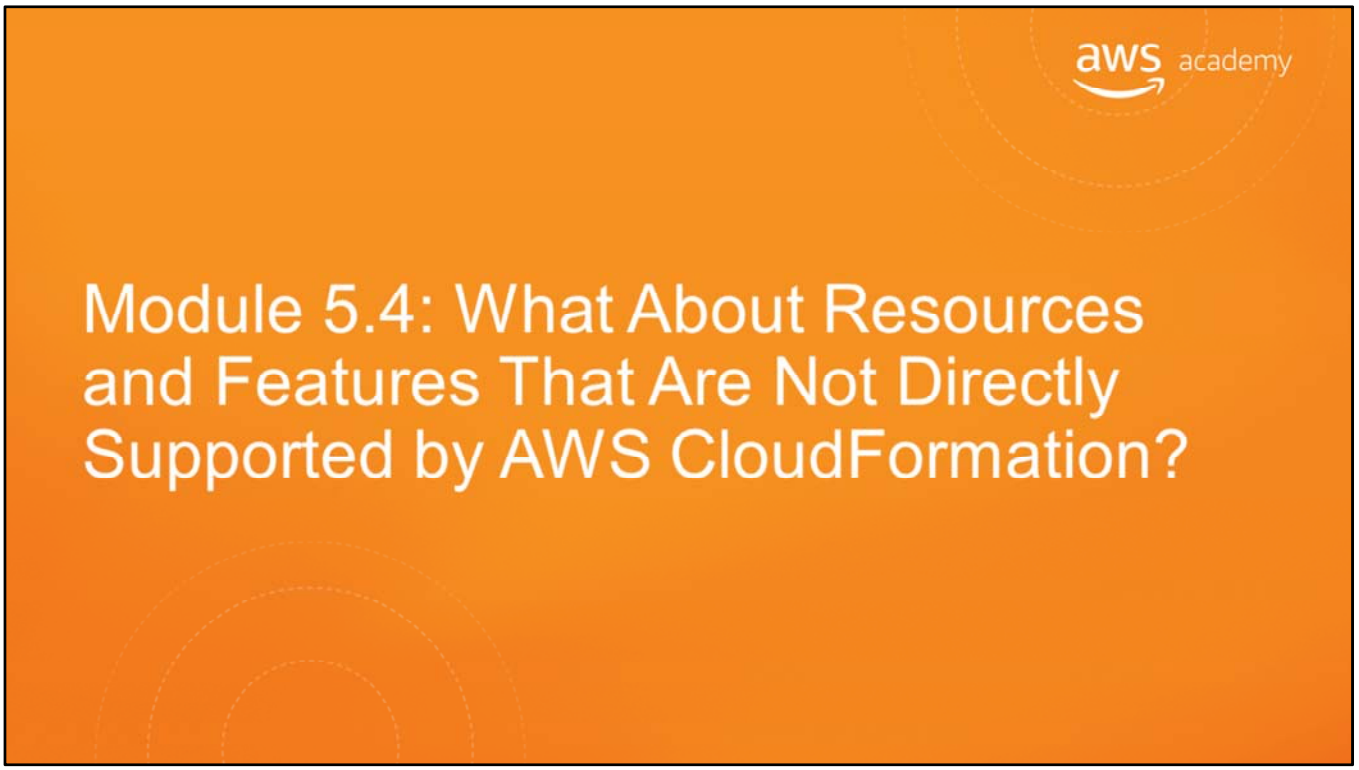

- Currently supported instance types include C3, C4, C5, M3, M4, M5, R3, R4, T2, and X1.
- Are available in every AWS Region.
- Must be in a VPC and use shared tenancy.

- Cannot use instance storage. You must exclusively use elastic-backed storage.

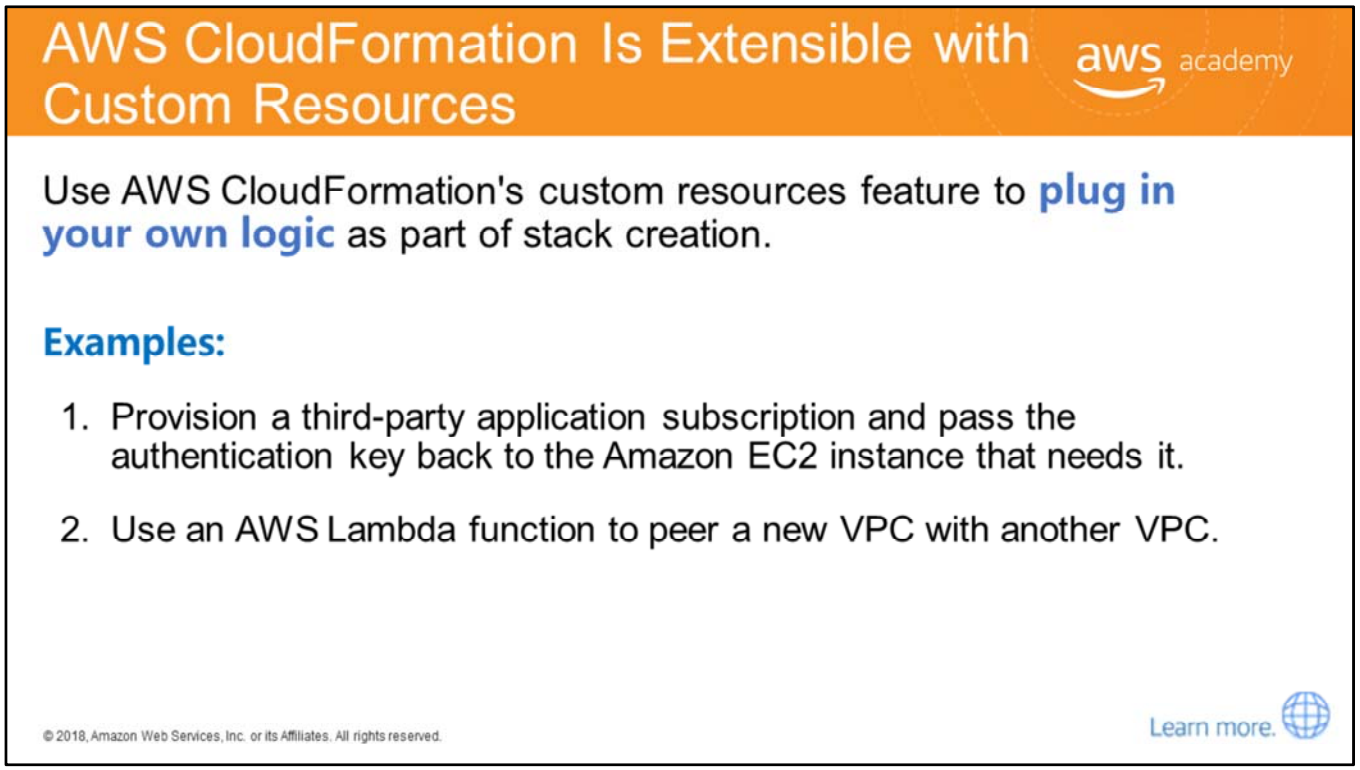
This feature should be used instead of Auto Scaling when your job needs to maintain identical instance metadata or storage volume.

For more information, see the Amazon EC2 documentation on recovering your instance.

http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-recover.html




Introducing Part 4: What About Resources and Features That Are Not Directly Supported by AWS CloudFormation?

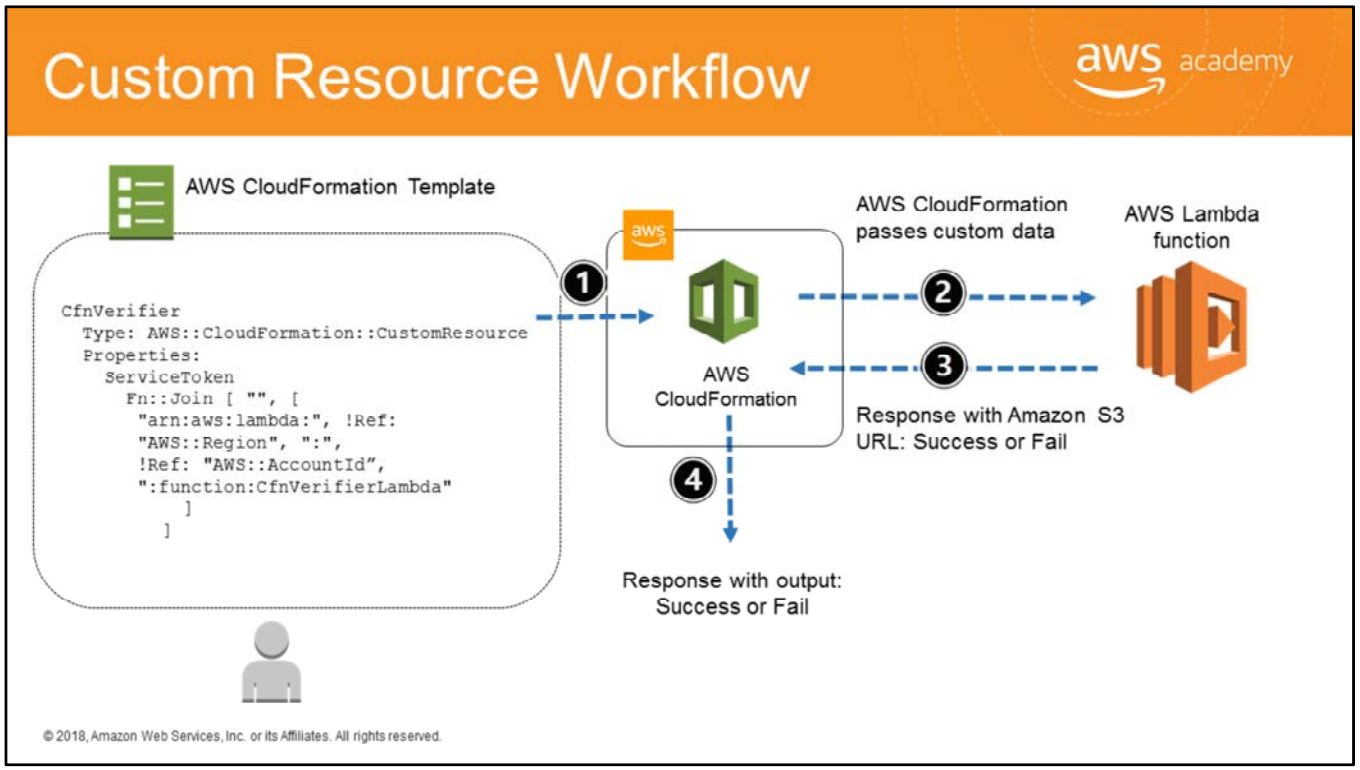



AWS CloudFormation is extensible with custom resources, so you can use part of your own logic to create stacks. With custom resources, you can write custom provisioning logic in templates. AWS CloudFormation runs your custom logic when you create, update, or delete stacks. For example, you might want to include resources that are not available as AWS CloudFormation resource types. You can include those resources by using custom resources, which means that you can still manage all your related resources in a single stack.

Use the AWS::CloudFormation::CustomResource or Custom::String resource type to define custom resources in your templates. Custom resources require one property: the service token, which specifies where AWS CloudFormation sends requests to, such as an Amazon SNS topic.

Examples include provisioning a third-party application subscription and passing the authentication key back to the Amazon EC2 instance that needs it. You could also use an AWS Lambda function to peer a new VPC with another VPC.

For more information, go to the AWS CloudFormation documentation on custom resources.
http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/template-custom-resources.html




In this example, a user creates an AWS CloudFormation template by using a stack that has a custom resource operation. This custom resource operation was defined by using AWS::CloudFormation::CustomResource or Custom::CustomResource . The template includes a service token—or ServiceToken—from the third-party resource provider, which is used for authentication. The template also includes any provider-defined parameters required for the custom resource.

AWS CloudFormation communicates with the custom resource provider by using an Amazon Simple Notification Service—or Amazon SNS—message that includes a Create, Update, or Delete request. The message also includes any input data that is stored in the stack template and an Amazon S3 URL for the response.

The custom resource provider processes the message and returns a Success or Fail response to AWS CloudFormation. The custom resource provider can also return the names and values of resource attributes if the request succeeded—that is, output data—or send a string that provides details when the request fails.

AWS CloudFormation sets the stack status according to the response that is received, and it provides the values of any custom resource output data.

You can use an AWS Lambda function to act as a custom resource. To implement this , you can replace the ServiceToken for your custom resource with the Amazon Resource Name, or ARN, of your Lambda custom resource. In other words, you do not need to create an Amazon SNS topic for a custom resource when you use AWS Lambda because AWS CloudFormation is Lambda-aware.

As in the previous scenarios, your code is responsible for doing any required processing. It uses the pre-signed URL—which is sent by AWS CloudFormation—to signal to the service that the creation of the custom resource either succeeded or failed.

# Other Infrastructure as Code Services on AWS

AWS Elastic Beanstalk

AWS OpsWorks

Amazon EC2 Run Command and Third-Party Applications on Amazon EC2



While AWS CloudFormation is the most common way to deploy infrastructure automatically, AWS offers other options that might be more suitable for specific use cases.

You could treat code as a service by using AWS Elastic BeanStalk, AWS OpsWorks, Amazon EC2 run command, and third-party applications on Amazon EC2 instances.

# AWS Elastic Beanstalk

- AWS Elastic Beanstalk is an **automated deployment and scaling service** for web applications.
- AWS Elastic Beanstalk:
  - Accepts Java, .NET, PHP, Node.js, Python, Ruby, Go, or Docker code.
  - Deploys on Apache, Nginx, Passenger, and IIS servers.
- A deployment with AWS Elastic Beanstalk can automatically handle:
  - Load balancing
  - Health monitoring
  - Automatic scaling
  - Application platform management
  - Code deployment



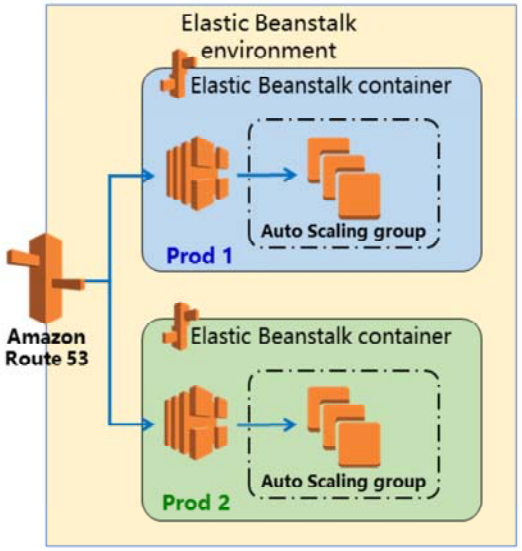
AWS Elastic Beanstalk is an automated deployment scaling service for web applications.

If you have an application that's written in Java, .NET, PHP, Node.js, Python, Ruby, Go, or Docker, you can roll out an AWS Elastic Beanstalk application. The application would deploy on Apache, Nginx, Passenger, and Internet Information Services—or IIS—servers.

A deployment with AWS Elastic Beanstalk can automatically handle load balancing, health monitoring, automatic scaling, application platform management, and code deployment.

# Blue/Green Deployment on AWS Elastic Beanstalk

- Production is fully scaled
- Pre-production is minimally scaled
- Both Elastic Load Balancing load balancers are kept warm
- If anything goes wrong, can roll back quickly






In the case of a 24-hour flash sale, you might not worry about the deployment of a web application. However, in general, you want to minimize the downtime for your web application. Therefore, blue/green deployment is highly desirable.

One of the challenges of automating deployment is the cutover, which is when you take software from the final stage of testing to live production. So how can you quickly deploy the application without any downtime?

One simple technique is to use blue/green deployments, where the live production environment is “blue,” and the matching environment is “green.” Blue/green deployment on AWS Elastic Beanstalk provides a way for you to test new hardware or applications without going fully into production.

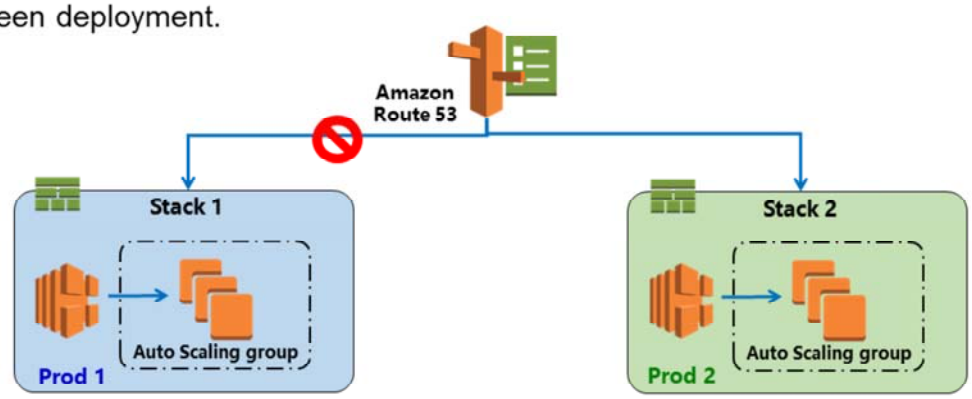
“Green” represents your new deployment. You deploy updates to the green deployment and attach it to your load balancer. After the green deployment is complete and functional, you can begin to shut down or upgrade the blue deployment. This approach also gives you the opportunity for a rapid rollback by switching back to the blue deployment if the green environment is not working properly.

AWS Elastic Beanstalk provides automated infrastructure management and code deployment for your application. It includes:

- Load balancing
- Health monitoring
- Auto scaling
- Application platform management
- Code deployment

# Blue/Green Deployment on AWS CloudFormation

- It takes little more effort than the Elastic Beanstalk approach.
- **Customer story:** An internal security team has not approved the usage of Elastic Beanstalk; therefore, the customer had to use CloudFormation to implement blue/green deployment.






You can also use AWS CloudFormation for a blue/green deployment rollout. In this example, AWS CloudFormation templates were used instead of Elastic Beanstalk. It takes little more effort than the Elastic Beanstalk approach.

This example is from a customer whose internal security team had not approved the usage of Elastic Beanstalk. As a result, the customer had to use AWS CloudFormation to implement the blue/green deployment. Traffic was trickled from Stack 1 to Stack 2 until it was apparent that Stack 2 was functional. After Stack 2 was functional, the connection to Stack 1, which is the former production environment, was taken away. Stack 2 became the new production environment, and the old production environment was torn down.

# AWS OpsWorks

aws academy

AWS OpsWorks is a **configuration management service** that helps you configure and operate applications of all shapes and sizes using Puppet or Chef.

- Define the application's whole architecture and specifications, including:
  - Package installation
  - Software configuration
  - Resources (compute, storage, etc.)
- Start from templates for common technologies (app servers, databases, etc.) or build your own.
- Use AWS OpsWorks as a lifecycle tool to:
  - Simplify management of an application.
  - Reduce the number of deployment cycles.



AWS OpsWorks is a configuration management service that helps you configure and operate applications in a cloud enterprise by using Puppet or Chef. AWS OpsWorks Stacks and AWS OpsWorks for Chef Automate let you use Chef cookbooks and solutions for configuration management. AWS OpsWorks for Puppet Enterprise lets you configure a Puppet Enterprise master server in AWS. Puppet offers a set of tools for enforcing the desired state of your infrastructure, and automating on-demand tasks.

You can define the entire architecture of the application, along with specifications that include package installation, software configuration, and the type of compute and storage resources that are needed. You can start from your own template, or you can use common templates for application servers and database servers.

AWS OpsWorks also works as a lifecycle tool to help you simplify the management of an application and reduce the number of deployment cycles.

# AWS Systems Manager

- Enables **automated configuration** and **ongoing management of systems at scale** through a set of capabilities
  - Works across all of your Windows and Linux workloads.
  - Runs in Amazon EC2 or on premises.
- Automatically collects software inventory
- Applies OS patches
- Creates system images
- Run command



AWS Systems Manager is a management service that helps you automatically collect software inventory, apply OS patches, create system images, and configure Windows and Linux operating systems. These capabilities help you define and track system configurations, prevent drift, and maintain software compliance of your Amazon EC2 and on-premises configurations. By providing a management approach that is designed for the scale and agility of the cloud but extends into your on-premises data center, AWS Systems Manager allows you to bridge your existing infrastructure with AWS.

AWS Systems Manager is straightforward to use. Access AWS Systems Manager from the Amazon EC2 console, select the instances that you want to manage, and define the management tasks you want to perform. AWS Systems Manager is now available at no cost to manage both your Amazon EC2 and on-premises resources.

# Amazon EC2 Run Command

aws academy

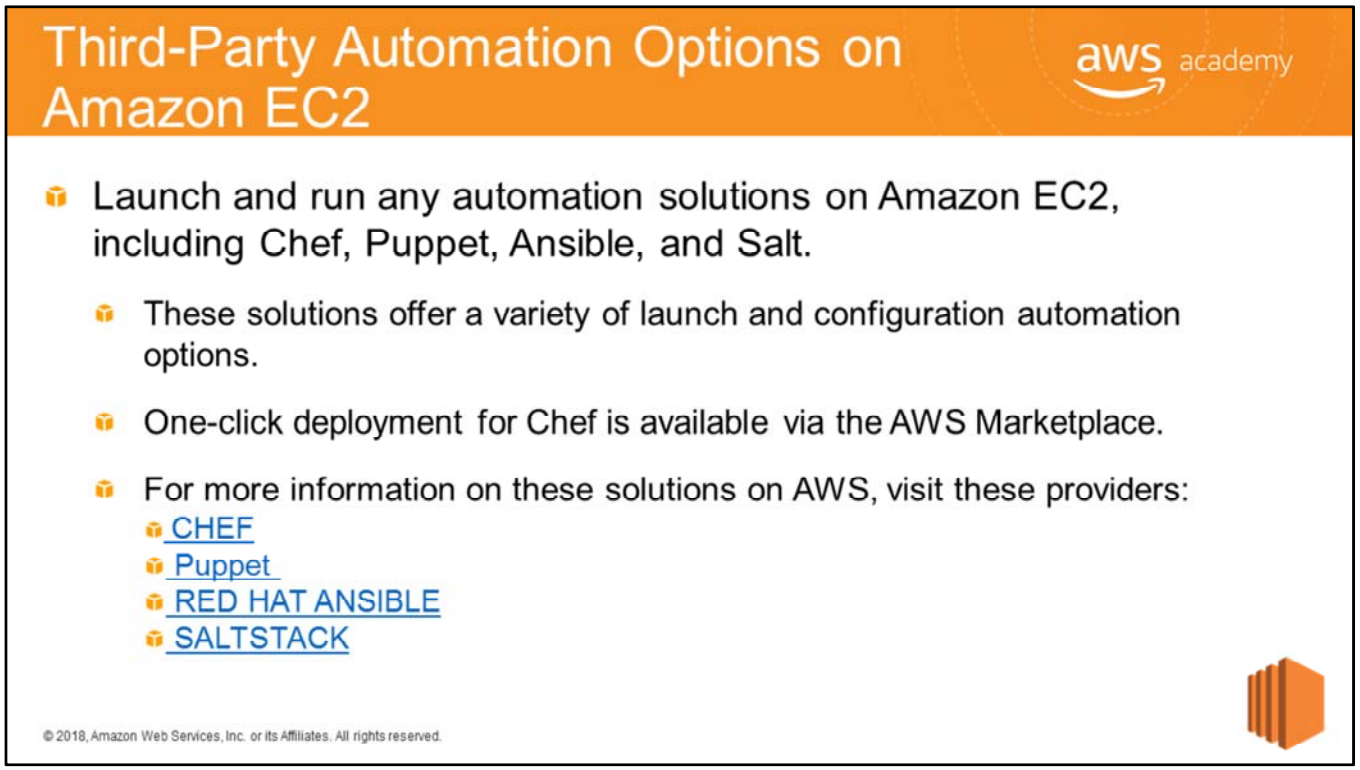
Amazon EC2 Run Command enables you to **securely manage the configuration** of your Amazon EC2 instances. Run Command:

- Provides a simple way to automate common administrative tasks, such as:
  - Executing Shell scripts and commands on Linux.
  - Running PowerShell commands on Windows.
  - Installing software or patches.
  - Managing and sharing documents.
- Allows you to execute commands across multiple instances.
- Offers visibility into the results.



Amazon EC2 Run Command helps you administer your instances—no matter how many you have—in a manner that is both straightforward and secure. This feature was designed to support a wide range of enterprise scenarios that involve automating common administrative tasks, like executing Shell scripts and commands on Linux, running PowerShell commands on Windows, installing software or patches, and more. It is accessible from the AWS Management Console, the AWS Command Line Interface or the AWSCLI, the AWS Tools for Windows PowerShell, and the AWS software development kits, or SDKs. If you currently administer individual Windows instances by running PS1 scripts or individual PowerShell commands, you can now run them on one or more instances. You can use new predefined commands to simplify your administration of Windows instances. The Linux version of the on-instance agent is also available in open source form on GitHub.

Command execution is secure, reliable, convenient, and scalable. You can create your own commands and exercise fine-grained control over execution privileges by using AWS Identity and Access Management, or IAM. For example, you can specify that administrative commands can be run on a specific set of instances by a tightly controlled group of trusted users. All of the commands are centrally logged to AWS CloudTrail for easy auditing.




You can use many third-party automation options. You can launch and run many automation solutions on Amazon EC2, including Chef, Puppet, Ansible, and Salt. These solutions offer a variety of launch and configuration automation options. For example, Chef offers a one-click deployment.

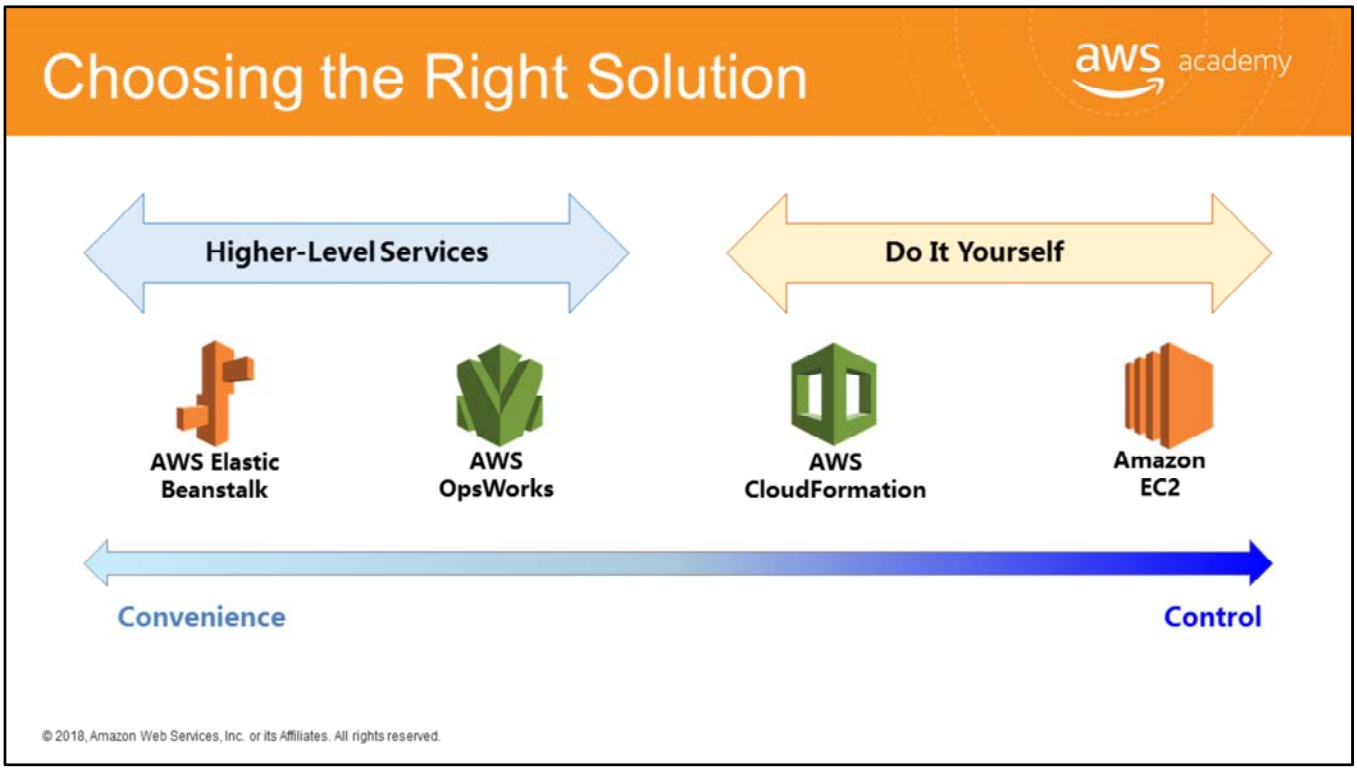
If you have a tool that you're already using and you would like to continue using it in the cloud, you can continue to do so. You can visit the AWS Marketplace to learn about solutions that offer one-click deployments for tools like Chef.

For more information on using these solutions on AWS, visit the solution providers:
https://www.chef.io/solutions/aws/

https://puppet.com/product/managed-technology/aws

https://www.ansible.com/aws

https://docs.saltstack.com/en/latest/topics/cloud/aws.html

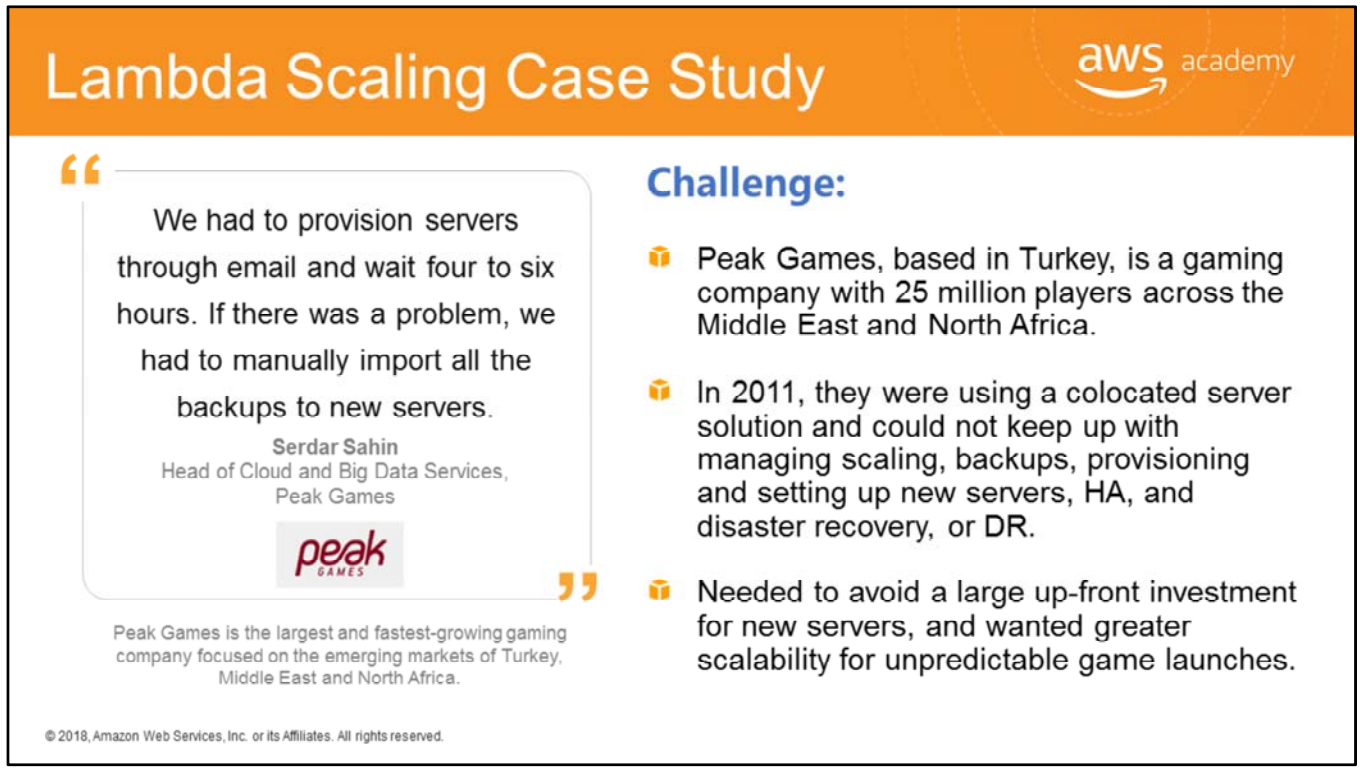



A frequently asked question is about the multiple services that provide application management capabilities, and where the line between them is. Choosing the right solution is a balance between convenience and having total control.

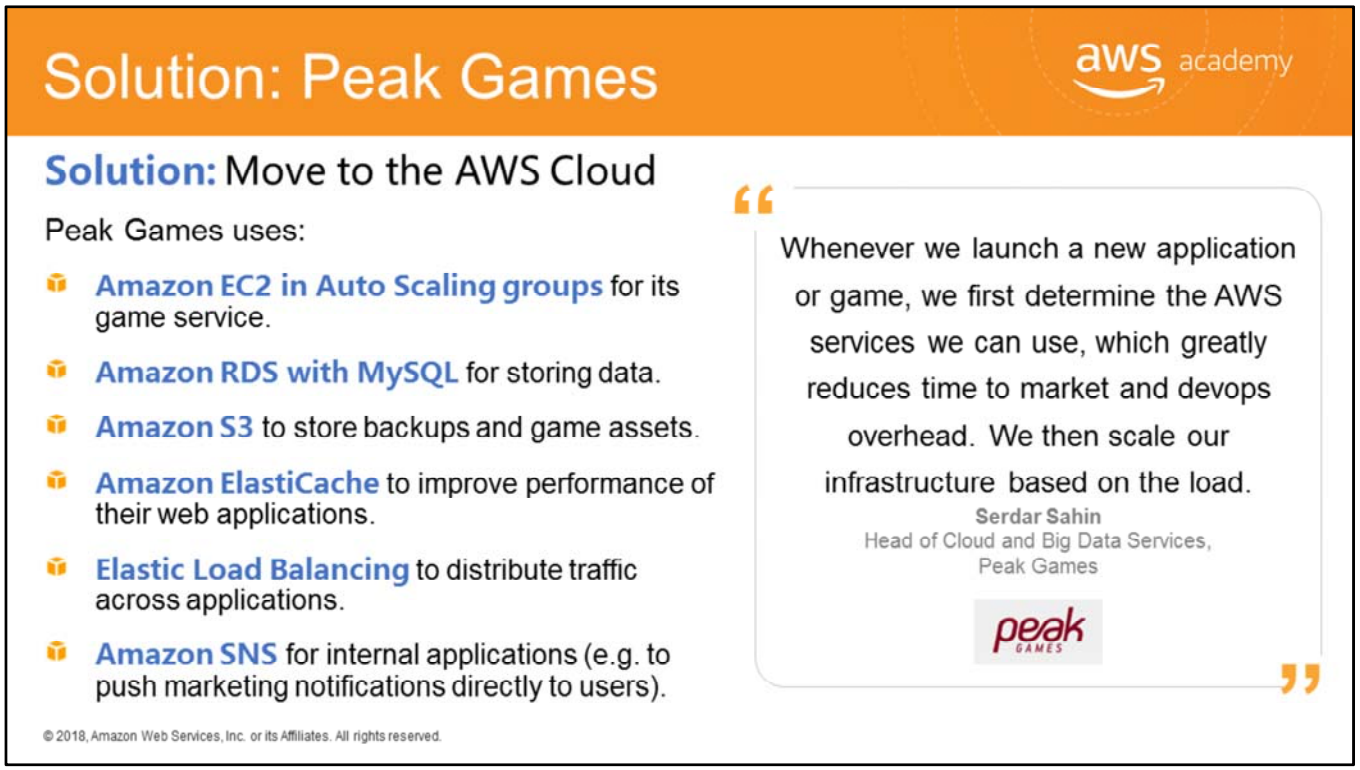
AWS Elastic Beanstalk is an application service for building web applications with a popular container— you can use Java, PHP, Node.js, Python, Ruby, and Docker. If you want to upload your code and don't need to customize your environment, Elastic Beanstalk might be a good choice for you.

AWS OpsWorks lets you launch an application and define its architecture and the specification of each component, including package installation, software configuration and resources, such as storage. You can use templates for common technologies—such as application servers, databases, etc.—or you can build your own template instead.

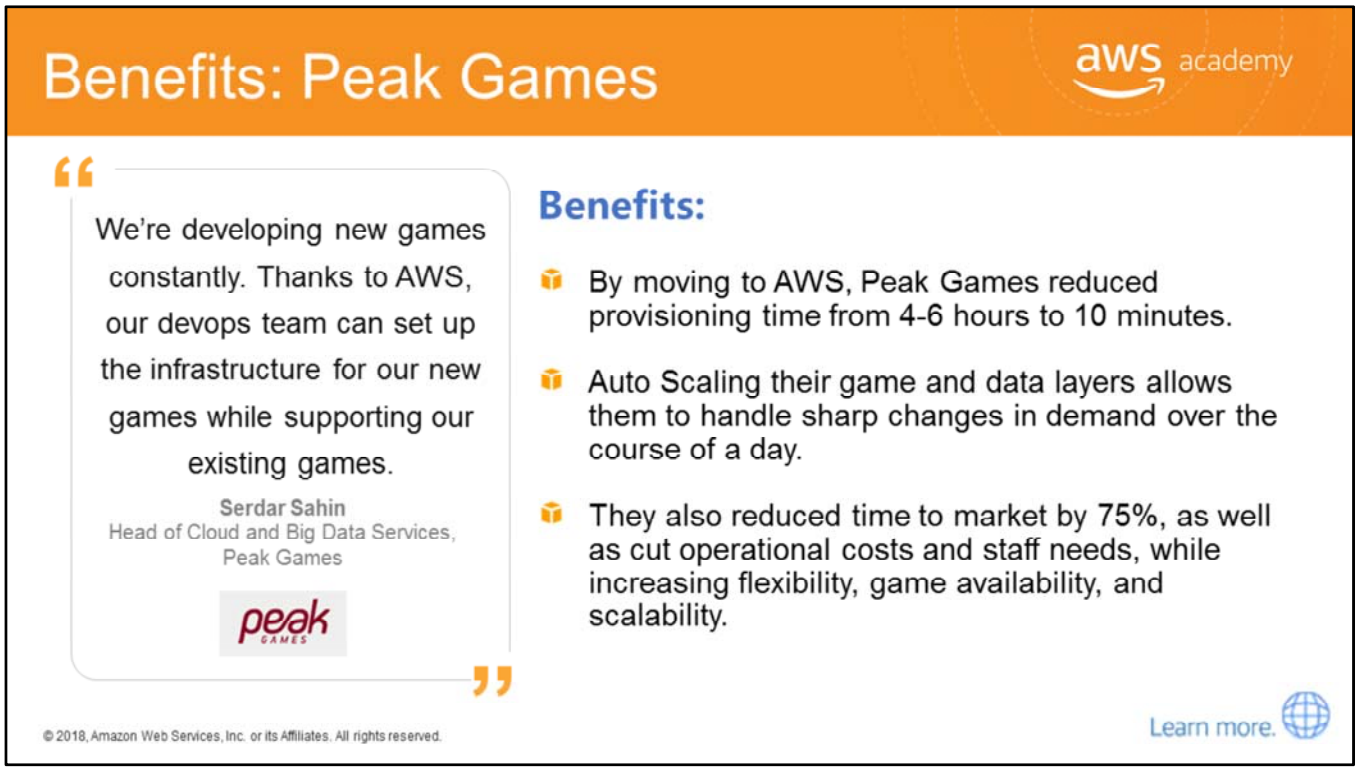
# Lambda Scaling Case Study




Take a moment to review this case study by Peak Games, who uses AWS to scale its games platform.




This slide shows the services that Peak Games chose in their solution to move to the AWS Cloud.
It's important to note that all of the selected services scale either inherently or by implementation within the Peak Games' environment.




Take some time to review the benefits that Peak Games experienced as a result of moving to AWS.
For more information about the case study, go to the AWS Case Study page for Peak Games.
https://aws.amazon.com/solutions/case-studies/peak-games/

_EC2 auto scaling can launch instances in multiples AZ, which increases overall reliability for an implementation
_Schedule scaling is most appropriate when you have a well-established pattern of demand (SET AMOUNT EACH DAY)
_Target tracking scaling provides average CPU utilization as a standard target metrics. You can specify the target value and let EC2 auto scaling handle the rest
_Separating read and write nodes is a form of horizontal scaling (RDS should read only)
_An Aurora cluster is made up of a primary Aurora DB instance and one or more Aurora replicas, can scale horizontally by adding replicas
_Amazon DynamoDB does not use instances. Amazon RDS uses instance. Amazon DB uses Application Auto Scaling service to adjust the provisioned throughput capacity on your behalf. When workload decrease, Application AS decreases throughput so that you do not pay for unused provisioned capacity
_ A Network load balancer can handle TCP, UDP and TLS traffic. As a feature of ELB, NLB is highly available
_ Latency-based routing is based on traffic between users and AWS data centers, not application
_ The ASG can automatically launch instances to handle increase load , and terminate instances when unhealthy or load decrease. Application Load Balancer is highly available and distributes the load across the instances
_ Automation does not necessarily lead to fewer cost. Although automation can assist with building highly available implementations, it is not requirement. Finally, automation is not a requirement for creating resources.
_ AWS cloud Formation provides a simplified way to model, create and manage a collection of AWS resources
_AWS cloud Formation Designer to author AWS cloud Formation templates in the AWS Management Console.