AWS Partner: AWS Well-Architected Best Practices (Technical)

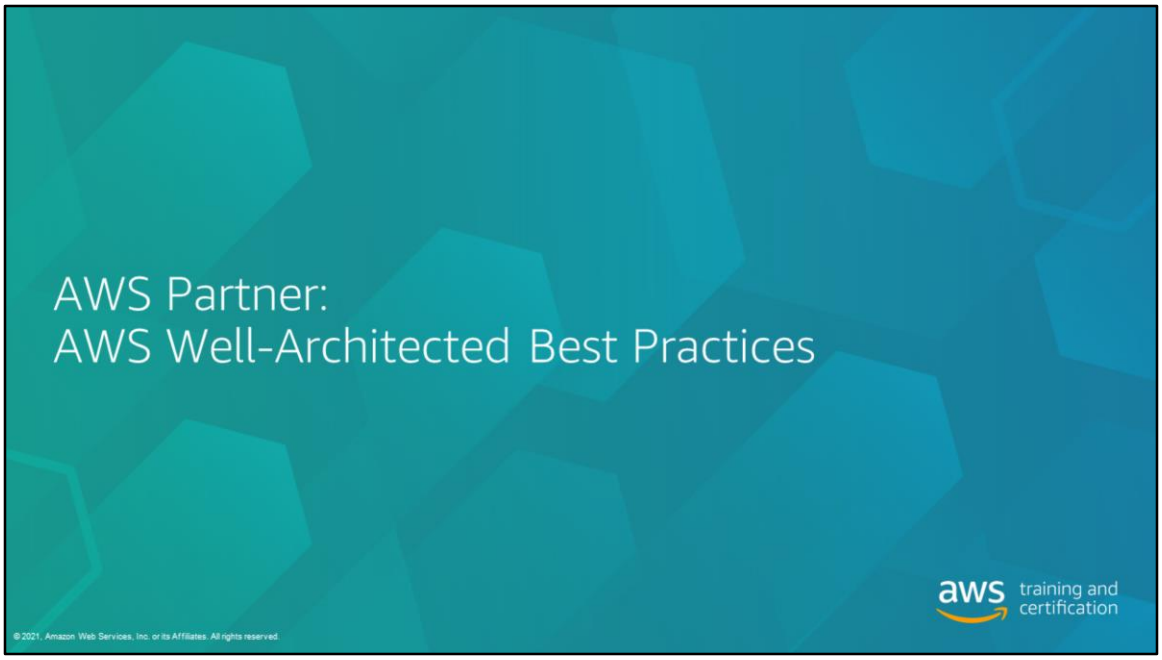
Student Guide

Version 1.0.0

200-PTWABP-10-EN-SG

Corrections, feedback, or other questions? Contact us at https://support.aws.amazon.com/#/contacts/aws-training.

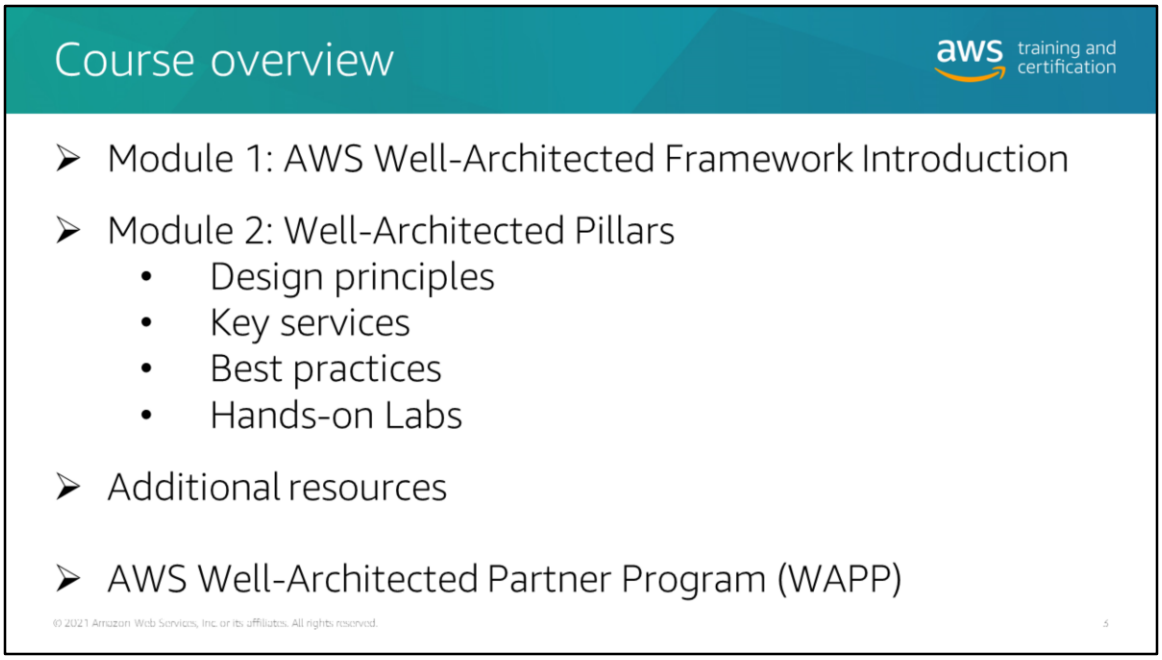
Contents



Course objectives

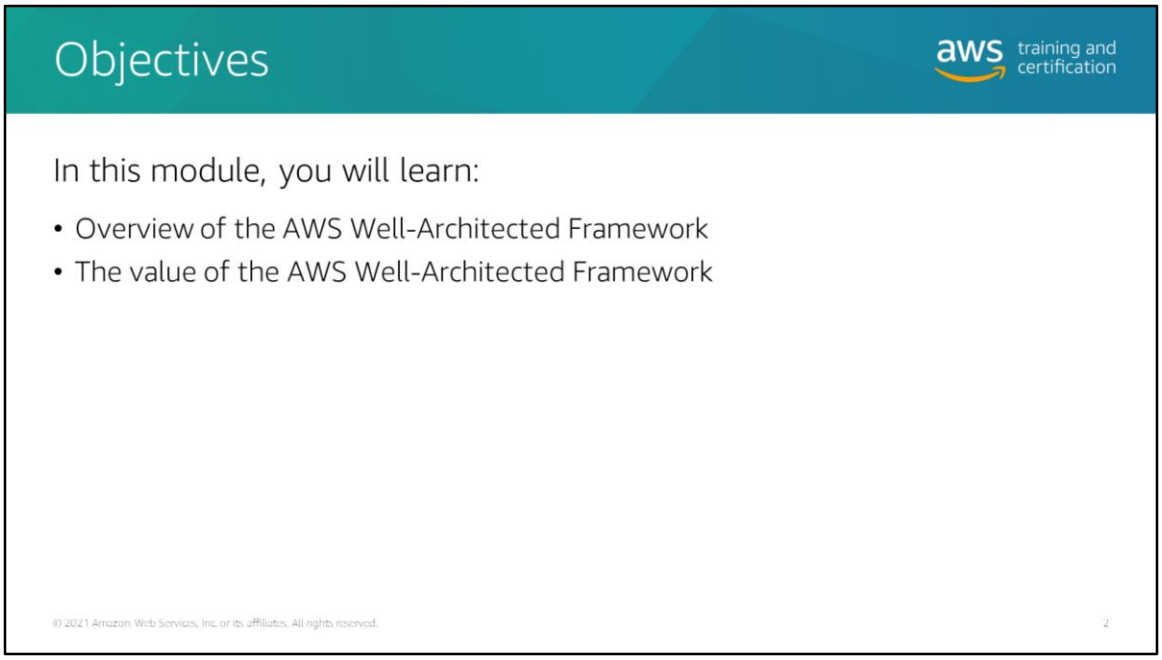
In this course, you will learn to:

- Identify the Well-Architected Framework features, design principles, design pillars, and common uses.
- Apply the design principles, key services, and best practices for each pillar of the Well-Architected Framework.
- Use the AWS Well-Architected Tool to conduct Well-Architected reviews.



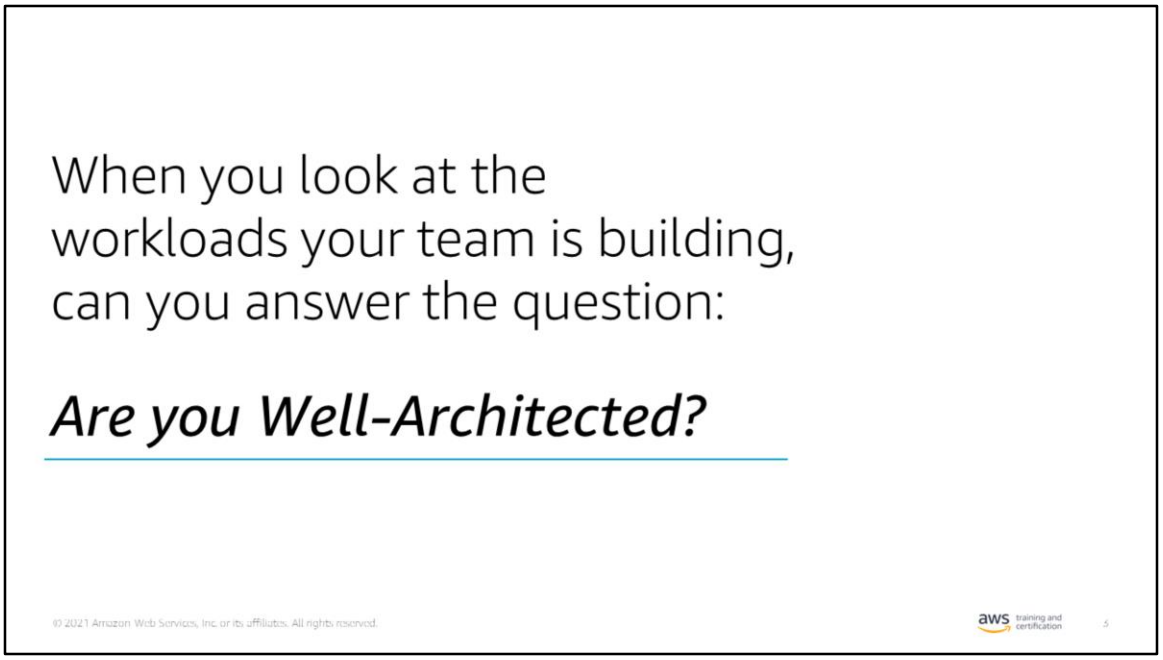
For instructor only:
Ask the students about their previous training experience with Amazon Web Services (AWS). Ideally, they should have attended the Foundations Technical or Cloud Practitioner course.

Module 1: Well-Architected Introduction

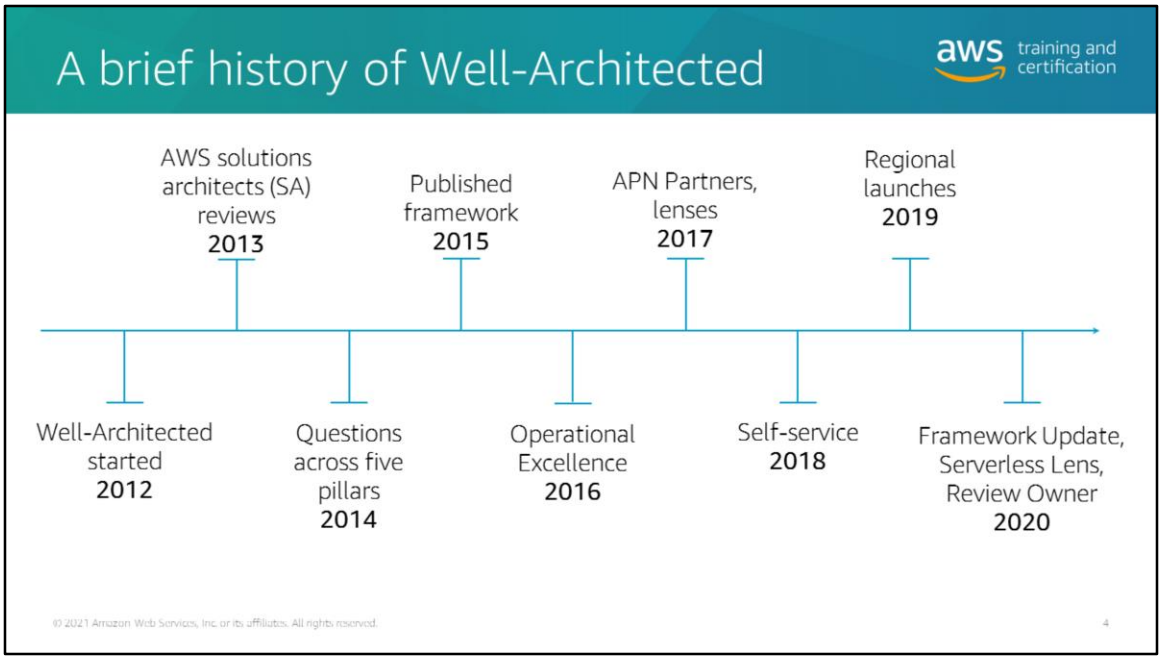


In this module, you will learn:

- Overview of the AWS Well-Architected Framework
- The value of the AWS Well-Architected Framework



Look at the workloads and systems within your organization. How **confident** are you that **those systems** are **built following best practices** for the **cloud**?
At AWS, we ask, "Are you Well-Architected?“ "Are you production ready?"



History

AWS started asking, “Are you Well-Architected?” [2012]

AWS solutions architects (SAs) review workloads. [2013]

Questions across four pillars (security, reliability, performance, cost) [2014]

Published **AWS Well-Architected Framework** [2015]

Added Operational Excellence [2016]

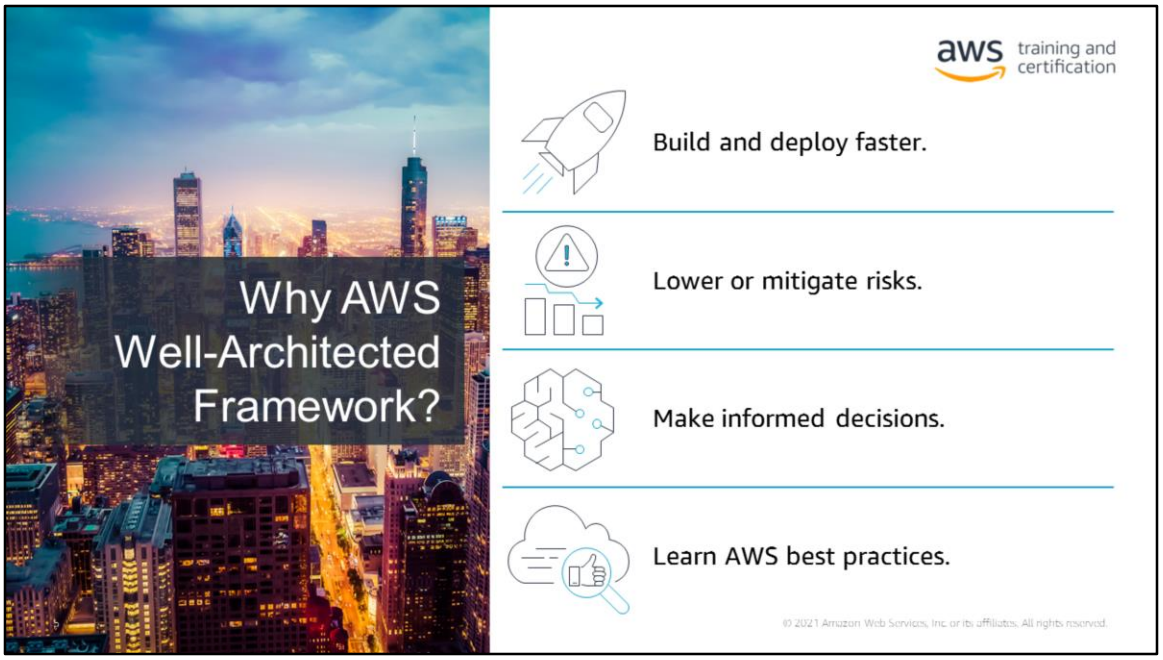
Select **APN Partners** trained to review workloads [2017]

Introduce the concept of AWS Well-Architected Lenses, which extend the guidance offered by AWS Well-Architected to specific industry and technology domains, such as machine learning, analytics, serverless, high performance computing (HPC), IoT (Internet of Things), and financial services. To fully evaluate your workloads, use applicable lenses together with the AWS Well-Architected Framework and the five pillars.

Launch self-service AWS Well-Architected Tool [2018]

Launched the AWS Well-Architected Tool and the Well-Architected Partner Program in multiple Regions [2019]

https://aws.amazon.com/architecture/well-architected/?wa-lens-whitepapers.sort-by=item.additionalFields.sortDate&wa-lens-whitepapers.sort-order=desc&awsm.page-wa-lens-whitepapers=1

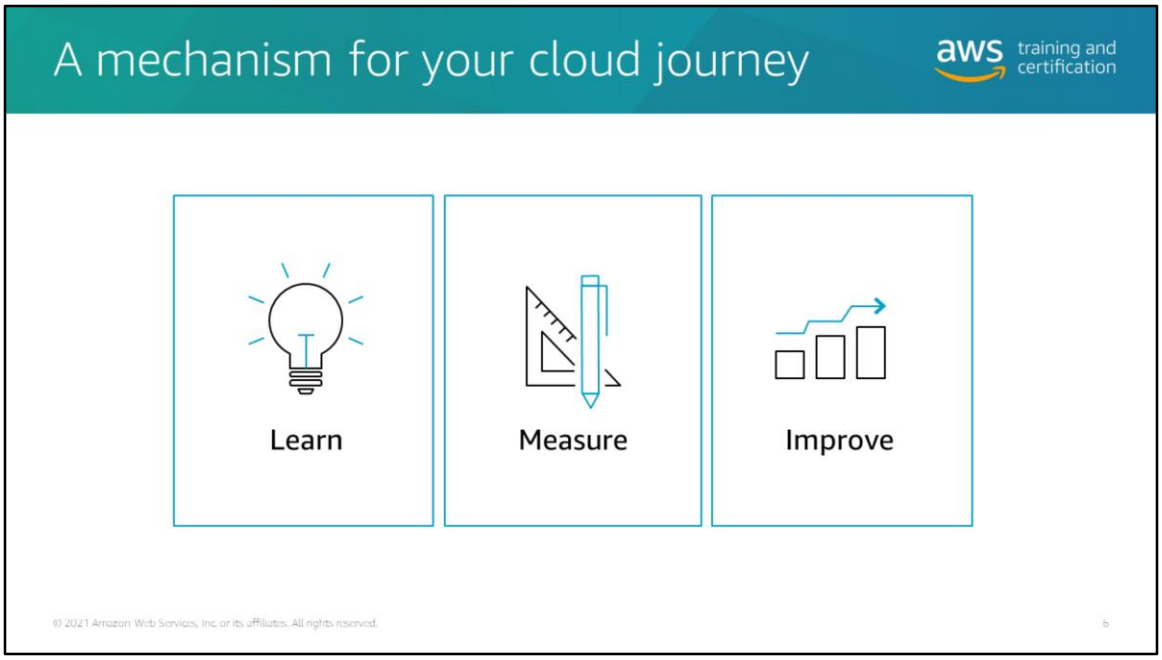


Why would I want to apply the AWS Well-Architected Framework?

Because you want to:

- **Build and deploy faster:** By reducing emergency response and capacity management and by using automation, you can experiment and release value more often.
- **Lower or mitigate risks:** Understand where you have risks in your architecture, and address them before they impact your business and distract your team.
- **Make informed decisions:** Ensure that you have made active architectural decisions that highlight how they might impact your business outcomes.
- **Learn AWS best practices:** Make sure your teams are aware of best practices that we have learned through reviewing thousands of customers' architectures on AWS.

We have seen customers use the AWS Well-Architected Framework to successfully achieve all of these.

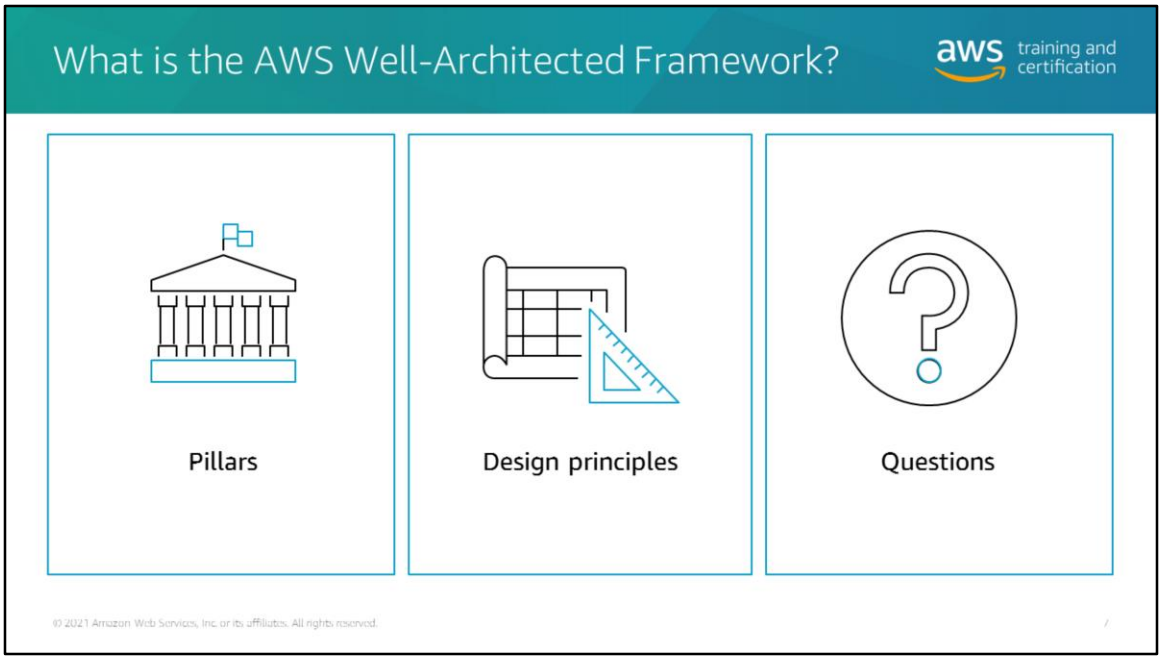


Well-Architected is a risk quantification tool. We must be able to learn what the best practices are, measure what currently exists against those best practices, and then improve over time.

Well-Architected is a **mechanism** that helps you be **successful** in your cloud journey.

- **Learn** the **strategies** and **best practices** for **architecting in the cloud.**
  - What should our mindset be when building in the cloud?
- **Measure** your architecture against best practices.
  - When we have learned best practices and we have a system designed and deployed, we want to measure our architecture against these best practices and look to see if there are opportunities for further improvement.
  - Ongoing improvement is the goal. We strive to remove 100% of issues, but cloud infrastructure is dynamic and always changing.
- **Improve** your architecture by **addressing any issues.**
  - Now that you have measured the workload against the best practices that you have learned, you can implement changes to address the issues you have uncovered.

- The workload is evolving and taking advantage of new services to ensure that it's providing the optimum capability to generate better business outcomes for the customer and their customers.



So what is the Well-Architected Framework?

The Well-Architected Framework provides **a set of questions that are based on design principles** across **five pillars.**

All of this comes from about 12 years of tribal knowledge across Amazon and AWS that we have refined into a framework (a set of whitepapers).

Hundreds of thousands of customers from across our support, SA, ProServ, TAM org, and even internally, build and deploy the 175+ services we have on the console today.

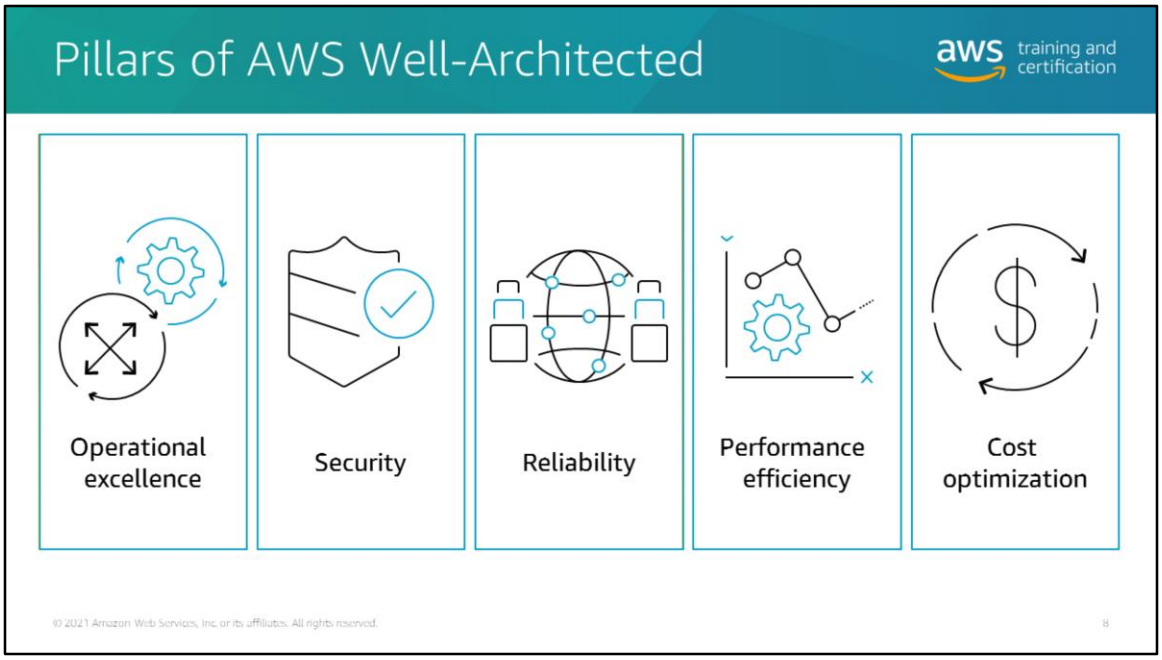
All these people gave input on common pain points that customers dealt with and the common architectural problems and oversights that were happening. And we started to create questions to address each of these cases. The best practices applied in these questions were summarized into white papers called *design principles*. Those design principles have been organized into five pillars.

Design principles are concepts and ways of thinking that you must have in mind when designing a workload. For example, under security, one of the things you need to

have in mind is encryption. What needs to be encrypted; when does it need to be encrypted?

For example, a question you might have is, “How are you encrypting data at rest?” This could lead to: “Do we need to encrypt data at rest?” “How do you encrypt data at rest?” “What’s the impact to the business if it’s not encrypted?”

This opens a conversation that gives people insight and ensures that if this wasn’t previously addressed, it is addressed at this point.

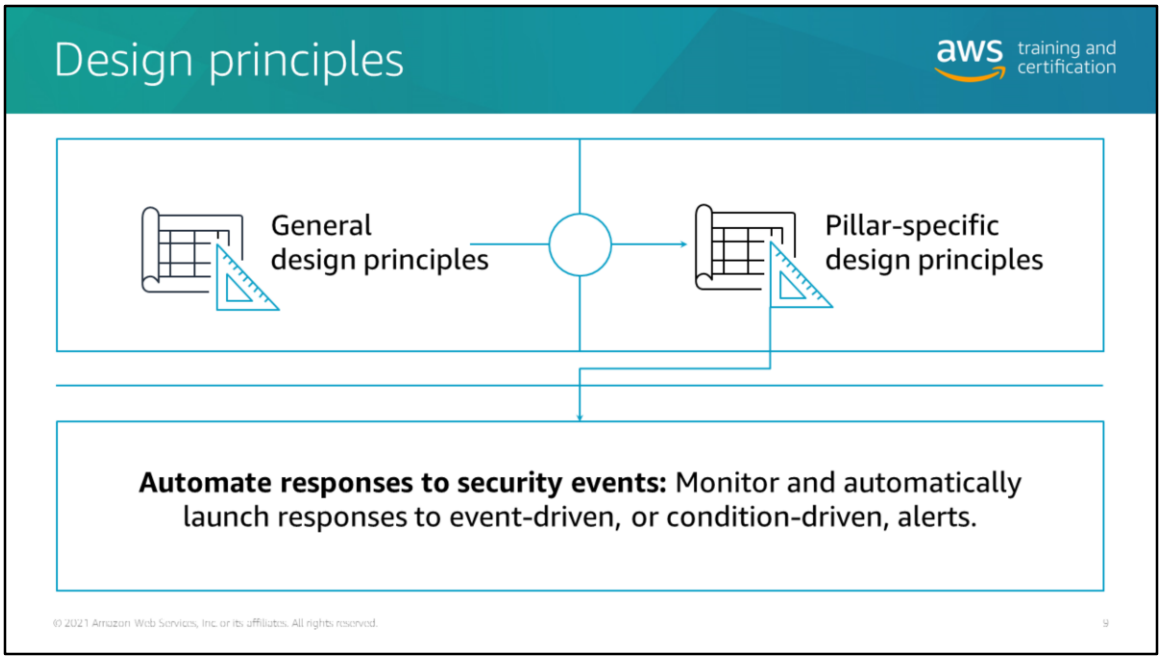


Creating technology solutions is a lot like **constructing a physical building**.

If the **foundation is not solid**, it can **cause structural problems** that **undermine** the **integrity** and **function** of the **building**.

If you **neglect** the **five pillars** when **architecting technology solutions**, it can **become** a **challenge** to **build** a system that **delivers functional requirements** and **meets your expectations**.

When you **incorporate** these pillars, it will help you **produce stable and efficient systems**, allowing you to **focus on functional requirements.**

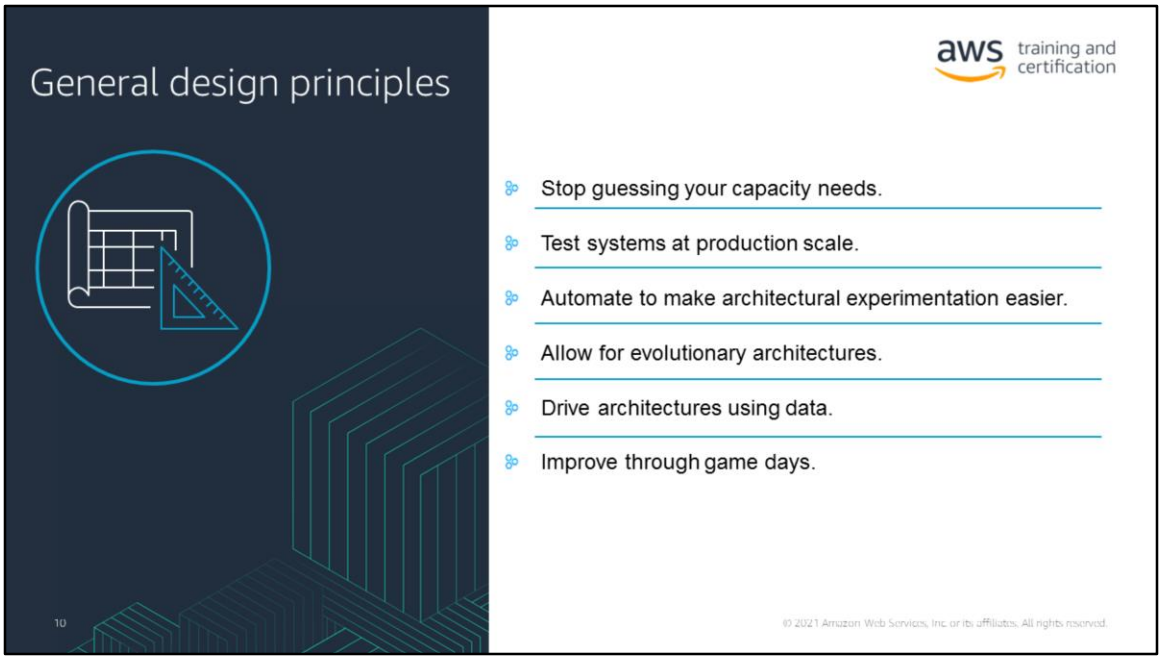


Design principles help you **adopt** the **appropriate mental model** when **building for the cloud.** This ensures you **take advantage** of the **capabilities** of AWS and **free yourself** from the **constraints** of **traditional approaches.**

There are general design principles and pillar-specific design principles. This slide shows a design principle from the security pillar as an example.

In this module, we will do a **deep dive** on the **general design principles. Later today, we will cover the pillar-specific principles.**

Security design principles reference:
https://wa.aws.amazon.com/wellarchitected/2020-07-02T19-33-23/wat.pillar.security.en.html



The Well-Architected Framework identifies a set of general design principles to facilitate good design in the cloud. They are as follows:

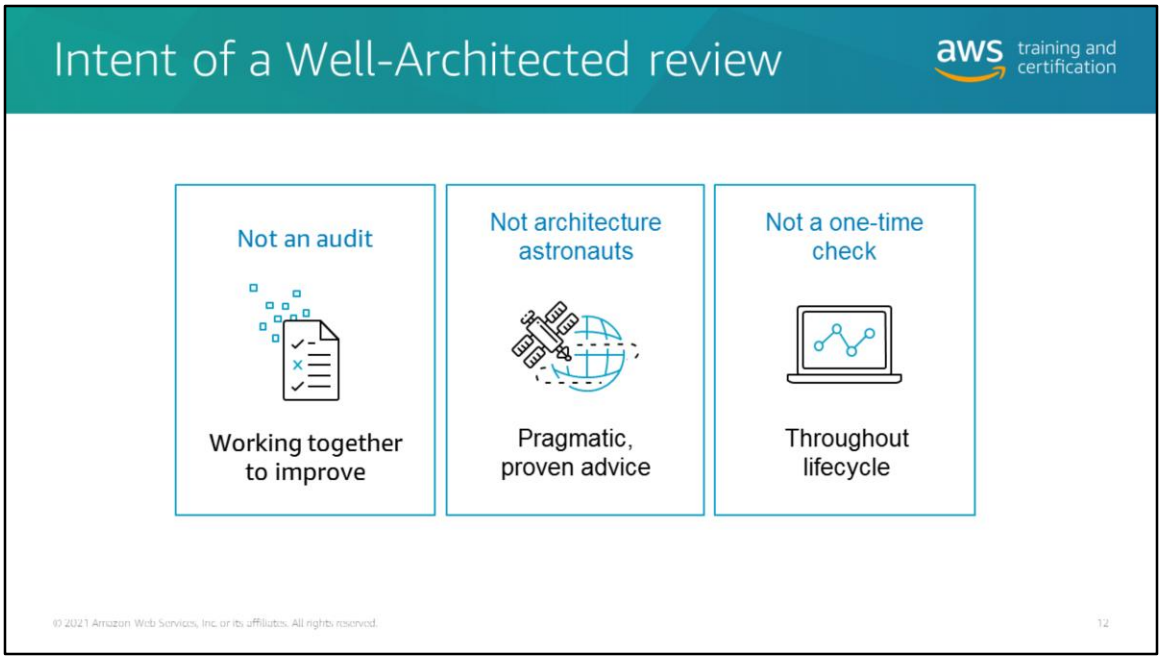
- You had to **guess how much infrastructure you needed**, often based on very **high-level business requirements** and demand and often before a **line of code is written**.
- You could **not afford to test at scale.** A complete duplicate of production costs is hard to justify, especially with low utilization. So when you went into production, you normally found a **whole new class of issues at high scale.**
- Any **proof of concepts or architectural experimentation was done by hand** and was generally only done at the **start of the project**.
- You generally had **static architectures** and it was **difficult** to even **think** about **making a change**.
- You generally **couldn't generate data sets** that would allow you to make informed decisions. So you probably **used models and assumptions to size your architecture.**
- In a traditional environment, you would **only exercise** your **runbook** when something **bad happened** in production.

In the cloud, **constraints have been removed**, so you can **use these principles to take advantage of that**.

Performing Well-Architected reviews is a best practice to evaluate if a specific workload is following the best practices of the AWS Well-Architected Framework.

What have we **learned** from doing reviews?

- The most common problem we see is **not bad decisions;** it is people **neglecting a decision**. Don't decide to not backup data, they forget to talk about it.
- **Most work**loads **have high-risk items** that must be addressed.
    - Finding them is **not a bad thing;** they were **always there.**
    - **If you address them**, that's **one less thing** that can **damage** or **slow your business**.



The **intent of a review** is to **improve outcomes.** It is **not an audit;** it's about a **team working out how to improve.**

An audit usually has some sort of pass/fail mechanism. A Well-Architected review does not. This should not be finger-pointing. It should be a chance to get many stakeholders from the business to work together to make sure we are addressing the needs of the entire business and not just one siloed department.

It's **pragmatic;** we look at **advice** that we see actually **helps people.**

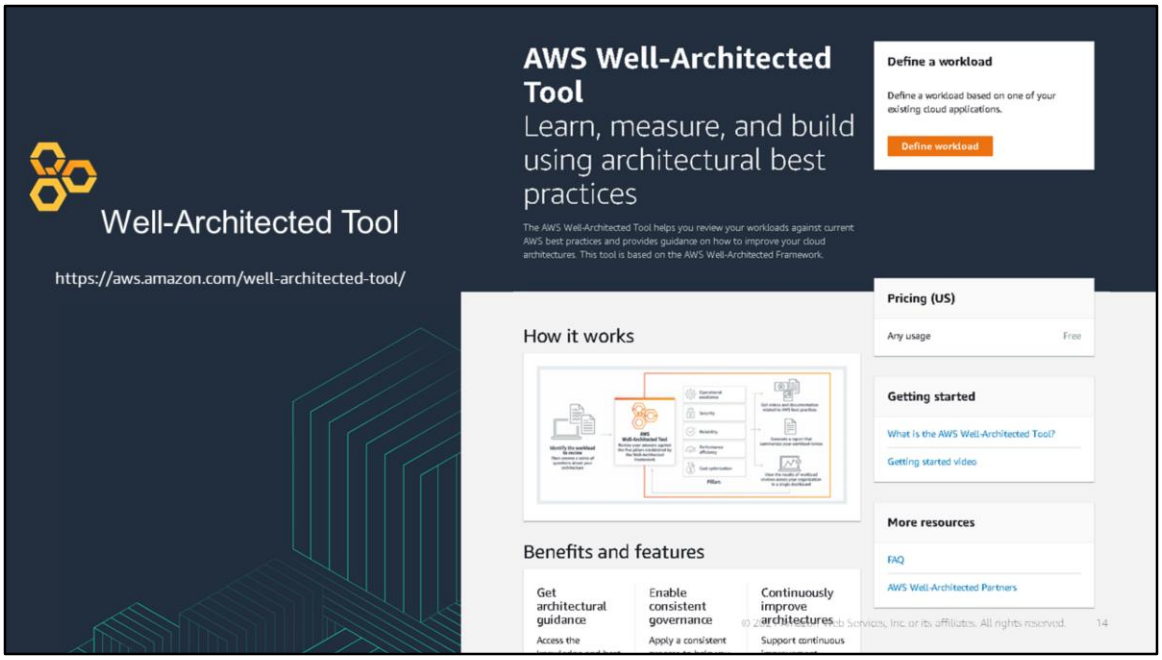
After you have reviewed a workload, you should **continually update it. Use milestones** to see how it's **improving over time**.

We have seen teams use the review changes to explain to their business the value they are adding.

During COVID, we saw how consumer behavior changed rapidly due to localized restrictions. Our customers who were well-architected were better prepared to serve their customers during this unprecedented time.

We must stay **agile.**

The AWS Well-Architected Tool helps you review the state of your workloads and compares them to the latest AWS architectural best practices. The tool is based on the AWS Well-Architected Framework, developed to help cloud architects build secure, high-performing, resilient, and efficient application infrastructure.



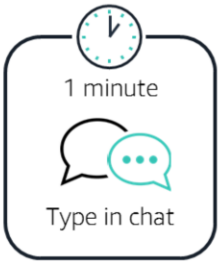
To complete AWS Well-Architected reviews, we use the tool in the AWS management console. All details are stored securely in your account. Workloads can be shared for collaboration on the review or remediation steps using workload sharing.

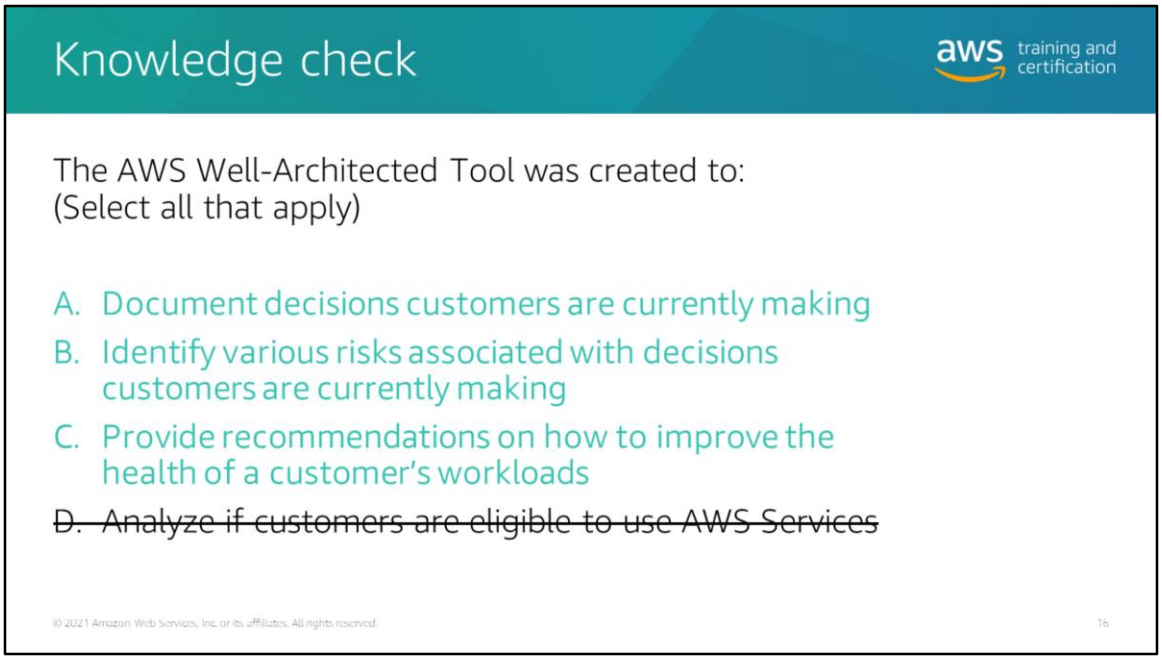
Knowledge check

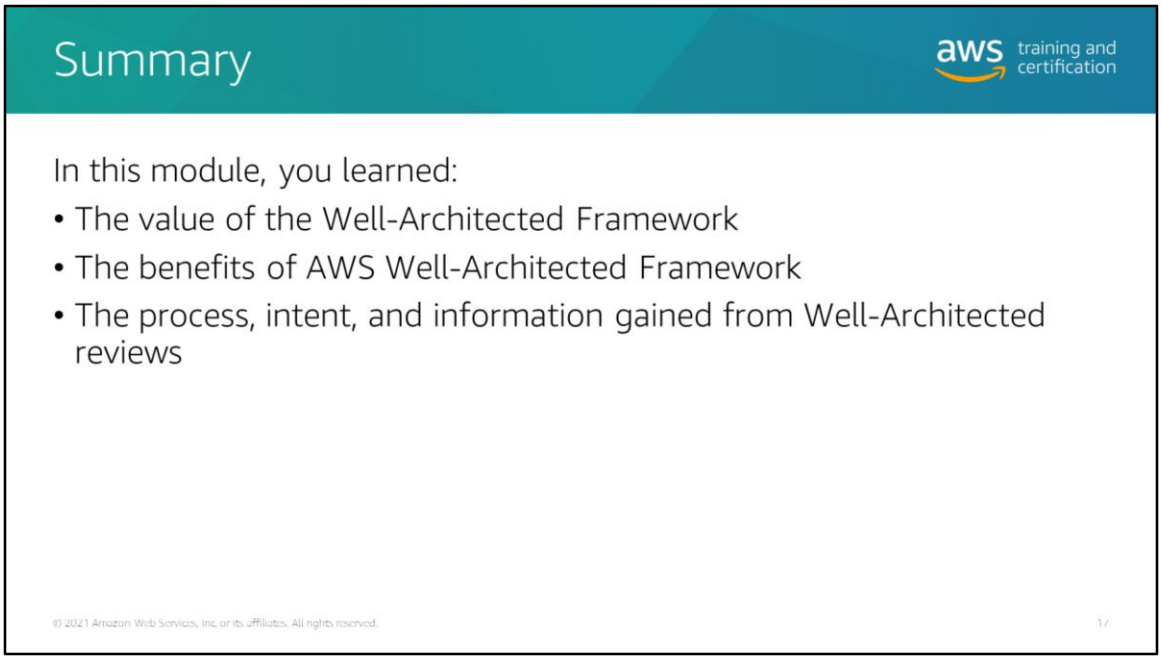
aws training and certification

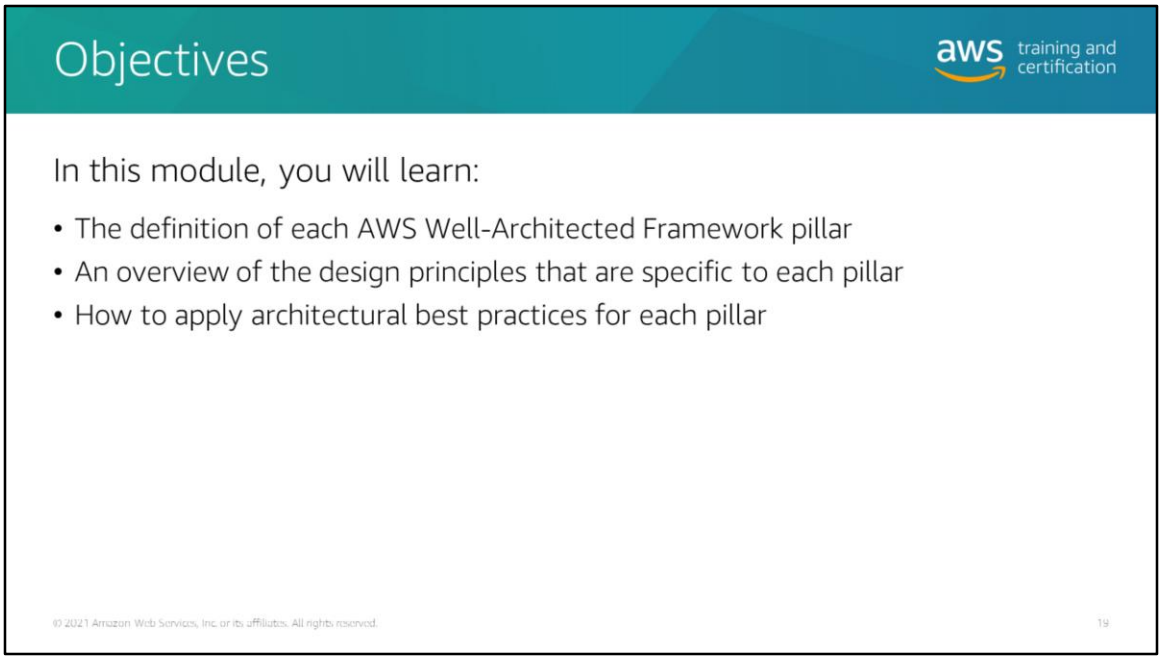
The AWS Well-Architected Tool was created to:
(Select all that apply)

A. Document decisions customers are currently making
B. Identify various risks associated with decisions that customers are currently making
C. Provide recommendations on how to improve the health of a customer's workloads
D. Analyze if customers are eligible to use AWS services





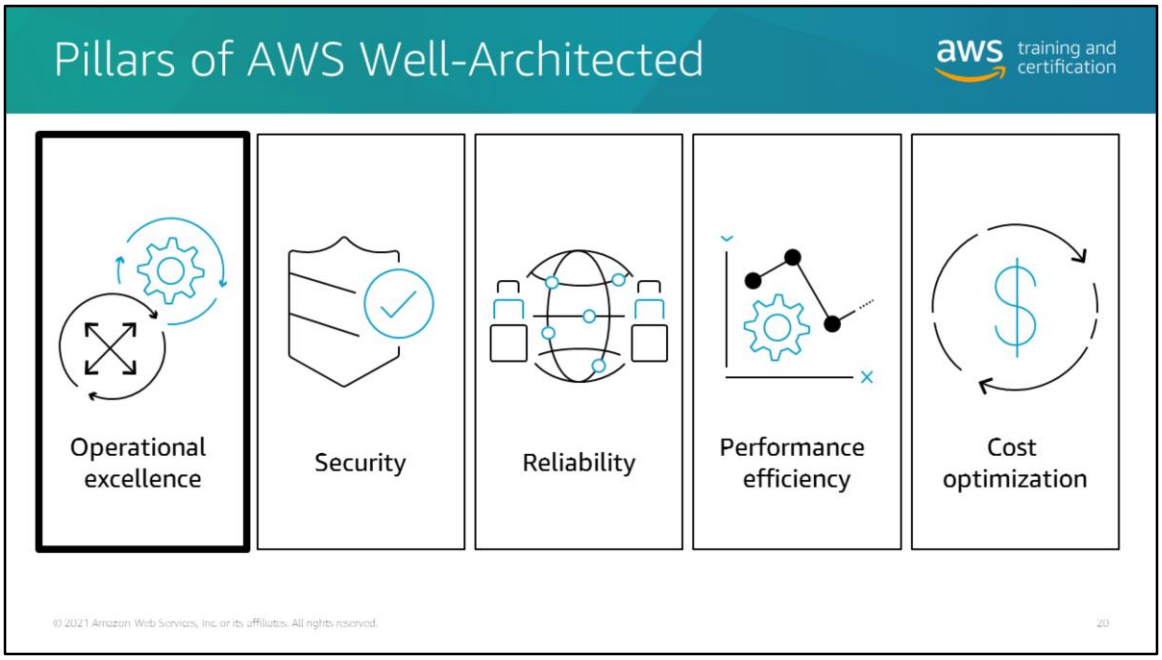


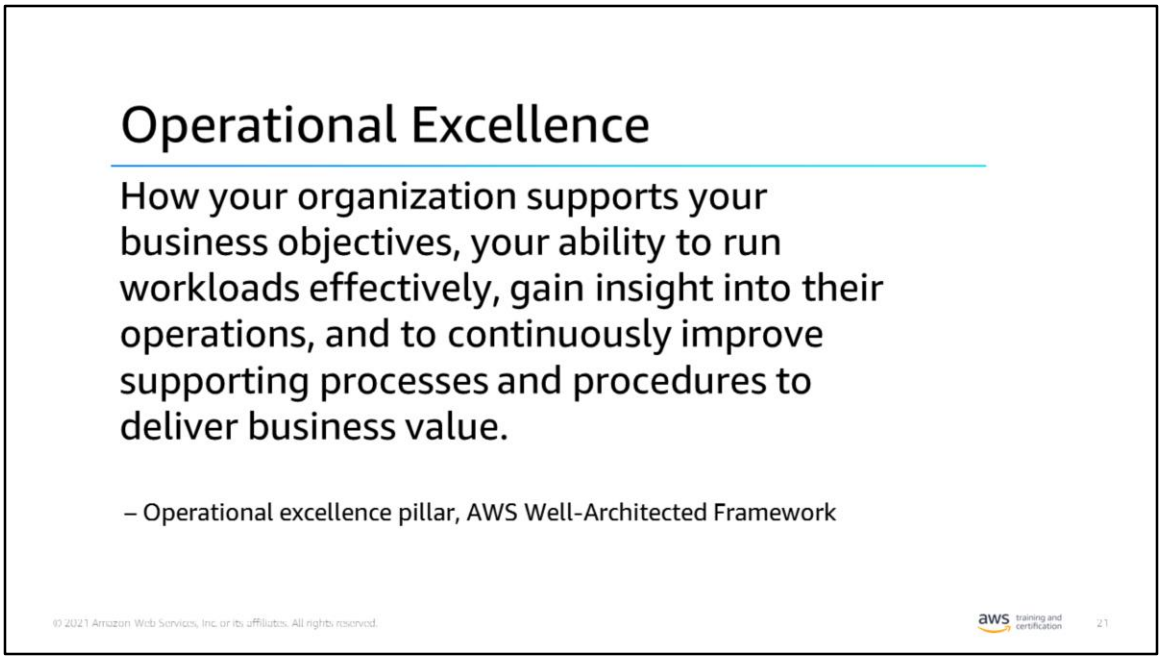


Important note for trainers only:
We need to make sure that messaging around well architected is for the cloud and not AWS-specific.

Well Architected is not a means to sell AWS services or adopt AWS. Well Architected principles can be used with other clouds too.

This is important messaging because our examples after each principle contain AWS services. However, if there were other technical solutions, that would be okay too.





Operational excellence is primarily concerned with the people and processes that support the workload. This includes how your organization supports your business objectives. It also includes your ability to run workloads effectively, gain insight into their operations, and continuously improve processes and procedures to deliver business value.
https://docs.aws.amazon.com/wellarchitected/latest/operational-excellence-pillar/operational-excellence.html

Operational excellence means that your organization is optimized to prepare, operate, and evolve your workload. Grouping brilliant people into teams is no guarantee of success if they don't share information or understand how to work together.

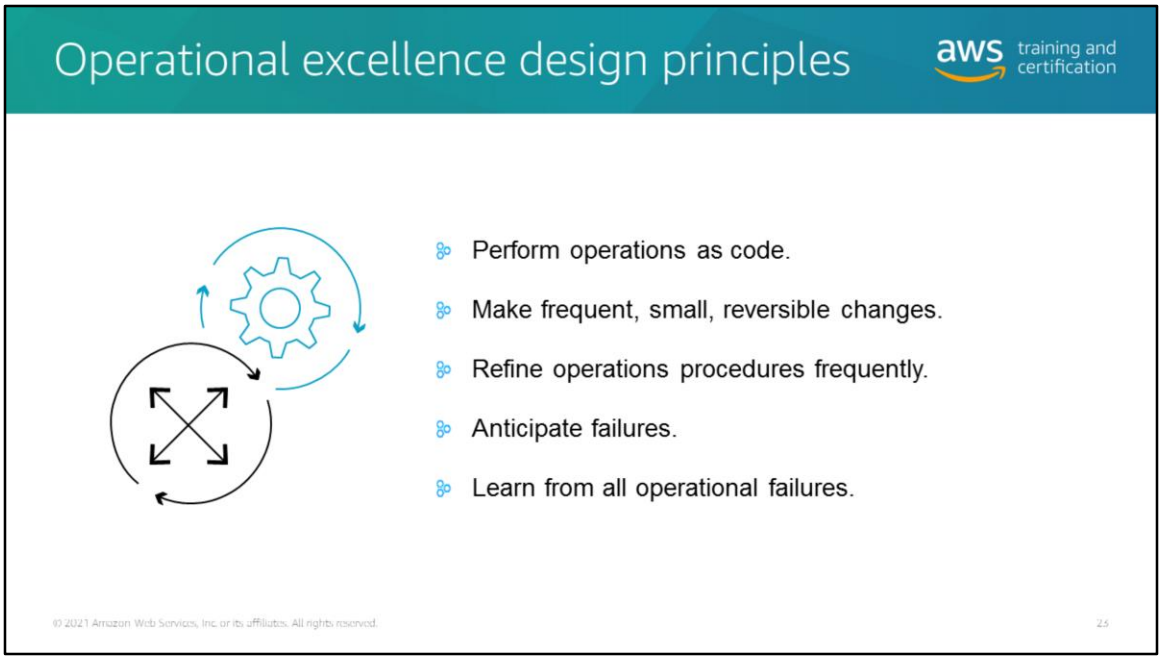
It's key to understand that Well-Architected won't help you fix all those challenges you have. But it will help you understand those risks and challenges. Sometimes it's appropriate to accept them; you can't fix them all, but you can have visibility of them.

Some of the fundamentals of operational excellence are ensuring that your workloads, process, and procedures deliver business value to your organization. Your

workloads must deliver effective business value. And all the supporting functions around that workload must help that workload deliver effective business value.

It's about running your workloads effectively. Having a workload that is secure, cost optimized, reliable, and high performing is fantastic. But if your teams cannot run and operate it effectively, it becomes an overhead to the business.

Let’s dive into the design principles of the operational excellence pillar with some examples for each.

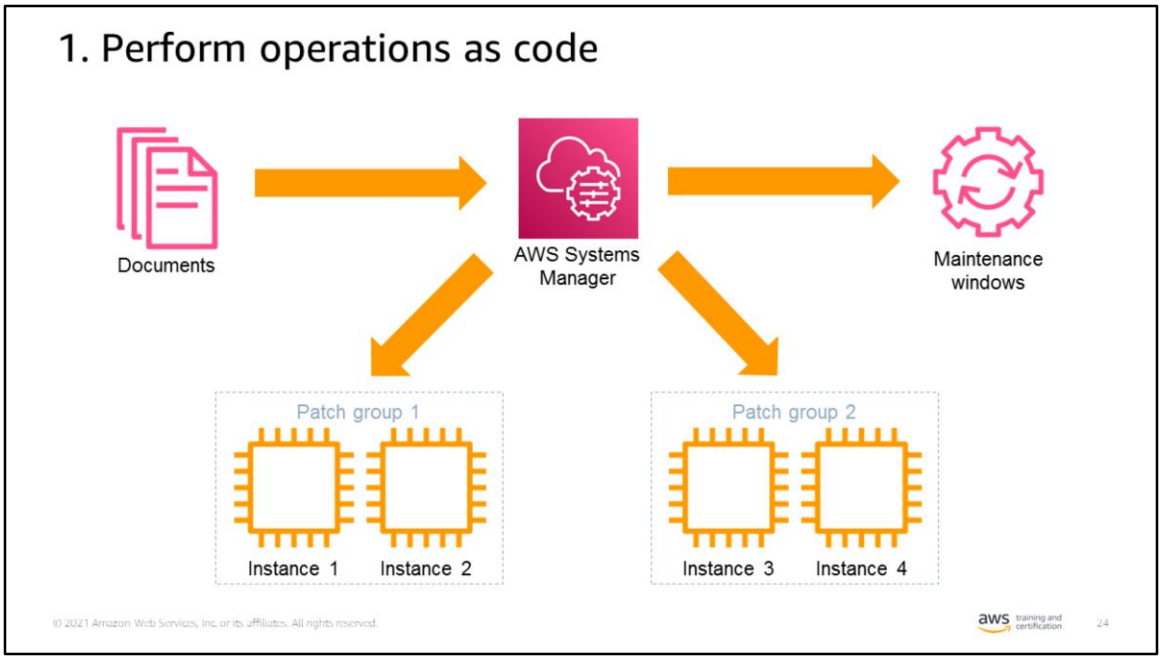


- Most **changes** were **made** by **human beings following runbooks** that were **often out of date.**
- It was **easy** to become very **focused** on the **technology metrics rather than business outcomes.**
- Because **making change** was **difficult and risky**, we tended **not to want to do it often** and therefore tended to **batch changes** into **large releases.**
- We **rarely simulated failures** or **events** because we were **too busy responding to emergencies** from real failures.
- We were **so busy reacting to situations that** it was **hard to take the time to extract learnings.**
- It was **hard to keep information current** because we were **making changes** to everything in response to emergencies**.**

In the cloud, constraints of a traditional environment are removed. You can use the design principles of the operational excellence pillar to make all changes by code with business metrics that you can measure your success against. By automating change and using code, you can move to making incremental changes and reduce risk. You can build organizational muscle memory by running game days that simulate failures to test your recovery processes. And you can learn from these and other operational

events to improve your responses. Finally, because infrastructure is now code, you can detect when documentation is out of date and even generate documentation.

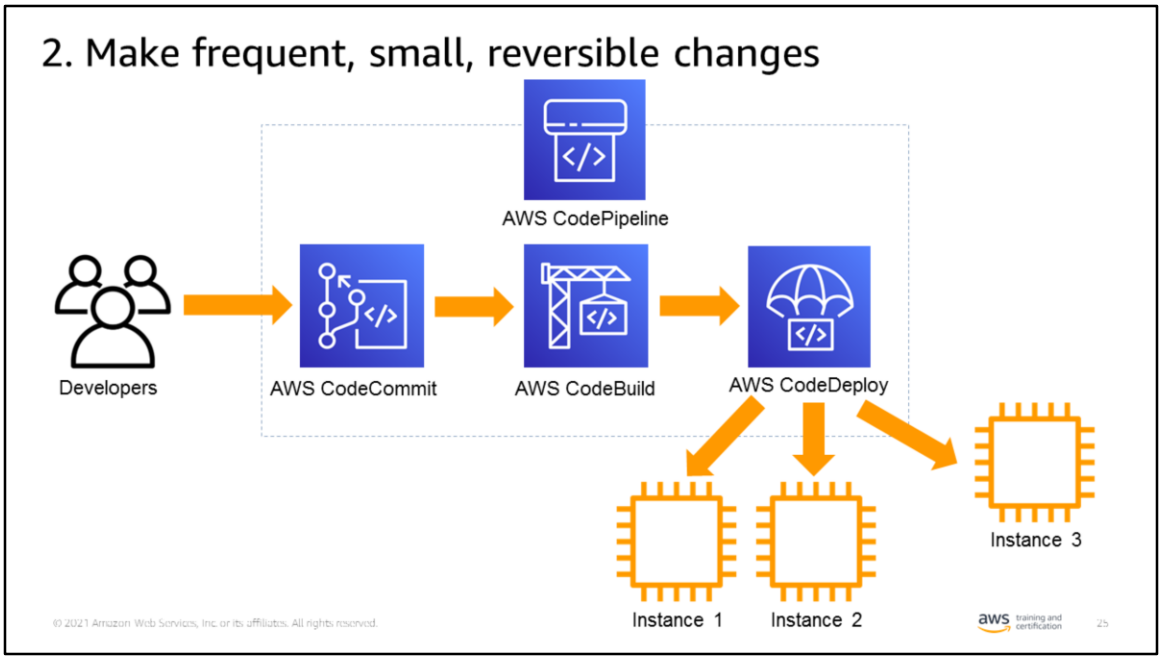
- **Perform operations as code** – This not only means infrastructure as code, but **security, operations, and process as code**. Don't rely on manual intervention to do things. In the cloud, you can define your entire workload as code and perform updates to it with code. By performing operations as code, you limit human error and ensure consistent responses to events.
- **Make frequent, small, reversible changes** – Small changes made often reduce overall risk and increase certainty. Reversible doesn't mean rollback, but it means that the recovery process is known and doable.
- **Refine operations procedures frequently** – As your workload evolves, so should your operations procedures. Operating a workload running on Amazon Elastic Compute Cloud (Amazon EC2) requires different processes than running on Amazon Elastic Container Service (Amazon ECS) or purely serverless.
- **Anticipate failures** – Accept that everything will eventually fail and make sure that you have playbooks ready to guide your team in unfamiliar circumstances.
- **Learn from all operational failures** – When something unexpected happens, you should identify the root causes as a team and put processes or automation in place to handle or prevent it.



Consider something like patching your infrastructure, in this case Amazon EC2 instances. This can be time consuming and complex to perform manually. But if we automate this, a lot of the difficulties can disappear.

Using a patch management solution like AWS Systems Manager, we can configure patching groups. Because we don't want to patch everything at the same time in case something goes wrong, we do small incremental changes. We can also configure maintenance windows. We don't want instances being patched when they are at their most busy because it could degrade performance. All of this can be managed using a template or document, typically written in something like JSON or YAML.

This document can be maintained along with your source code. Changes would be checked in and out of a repository, and you have full visibility as to what changes have been made.

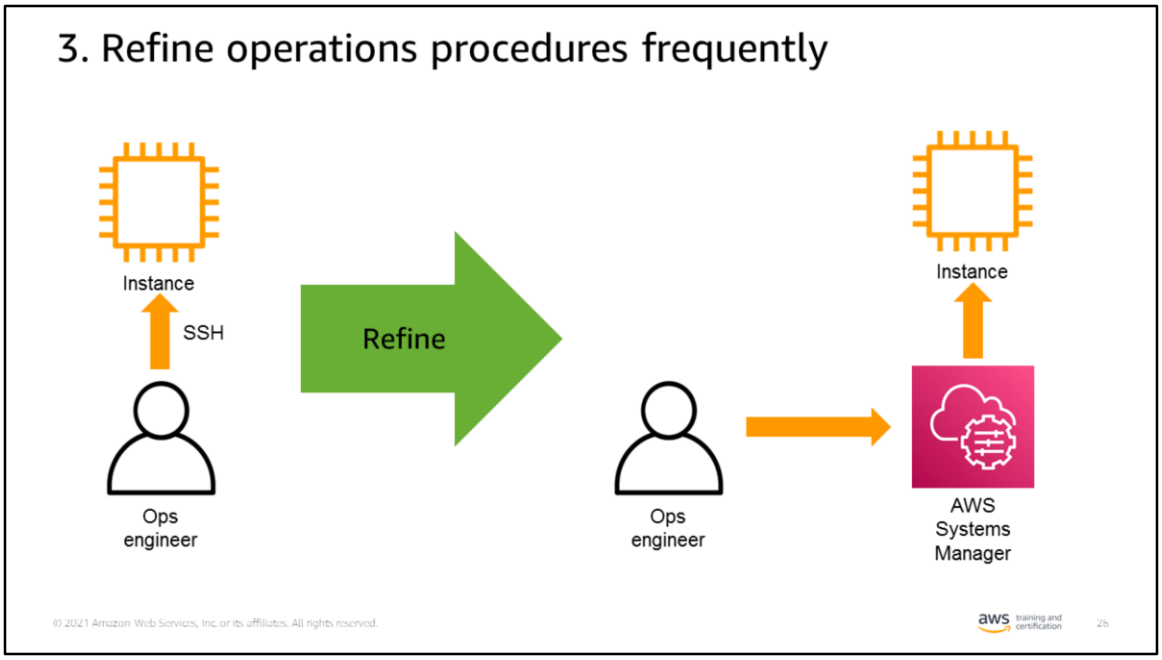


Making large changes that impact multiple parts of your workload at the same time has a larger impact to your business. Some of this can be covered by process, ensuring teams for different parts of your workload don't all make changes at the same time. You don't want folks changing security group configurations at the same time as you are patching instances. Because if something goes wrong, troubleshooting becomes more complex.

Using a pipeline is a great way to achieve this. AWS Code suite of services fits together well to support this best practice and allows you to get up and running very quickly. Refer to this blog: https://aws.amazon.com/blogs/devops/complete-ci-cd-with-aws-codecommit-aws-codebuild-aws-codedeploy-and-aws-codepipeline/

Use a source control service to host your repos. In this case AWS CodeCommit, but it could be anything else like GitHub or GitLab.
Use a continuous integration (CI) service that compiles and tests your code, in this case AWS CodeBuild.
Use a deployment service to deploy your code into your compute services, like Amazon EC2, AWS Lambda, or even on premises.

AWS CodePipeline is the continuous delivery (CD) service wrapper around all of this. It lets you automate all of this and is the glue that holds it all together.



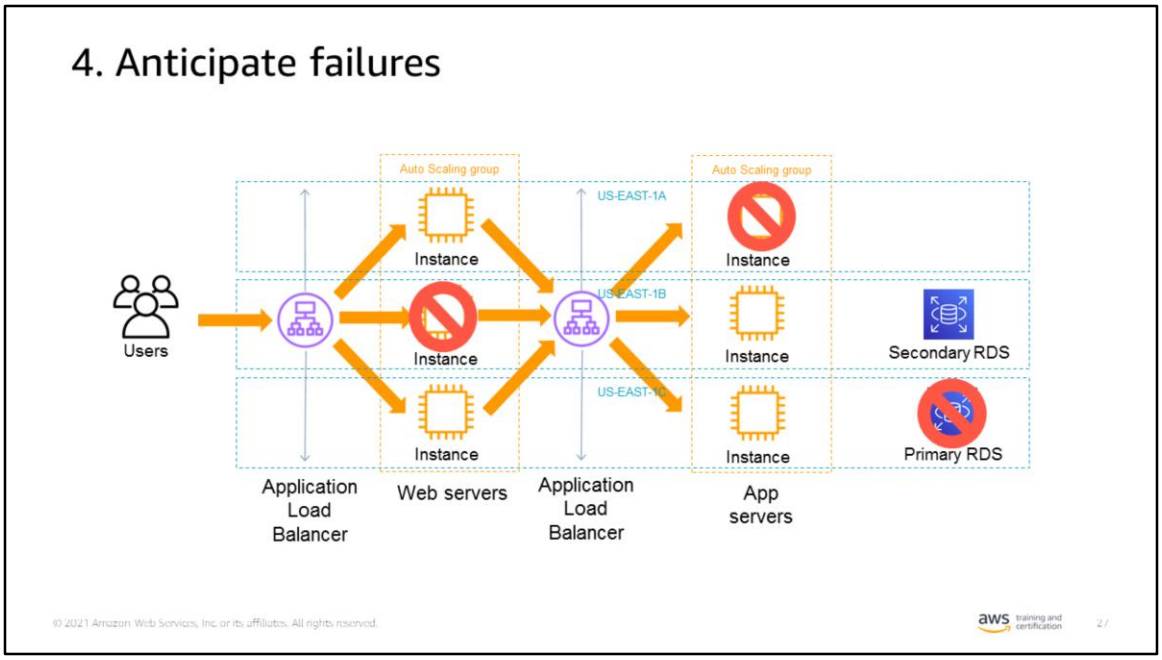
Let's take a migration. You had a load of Linux servers that you migrated to EC2 instances. Your Ops team manages them using Secure Shell (SSH) to connect and run commands as needed.

This design principle also must speak to operational cadences that the organization has to constantly evolve operations. For example, are you using thins like the Well-Architected Framework Review on a cadence to look at opportunities to evolve your workload.

For this example, the evolution of this type of operation came from a migration.
[1] It was on premises, then it was migrated to the cloud.
[2] Then the cloud and ops engineers would simply connect by using SSH as they did on premises.
[3] Finally, they moved to using cloud technology services such as Systems Manger.
Forward looking, [4] as new services come out, they will continue to evolve the workload by ops mechanisms like well architected reviews during regular intervals or business rhythms.

Staying static and living with the decisions made previously can be an inhibitor.

Making sure you evolve your process and procedures on a frequent basis allows you to take advantage of new technology more quickly, and save time, money, and effort.



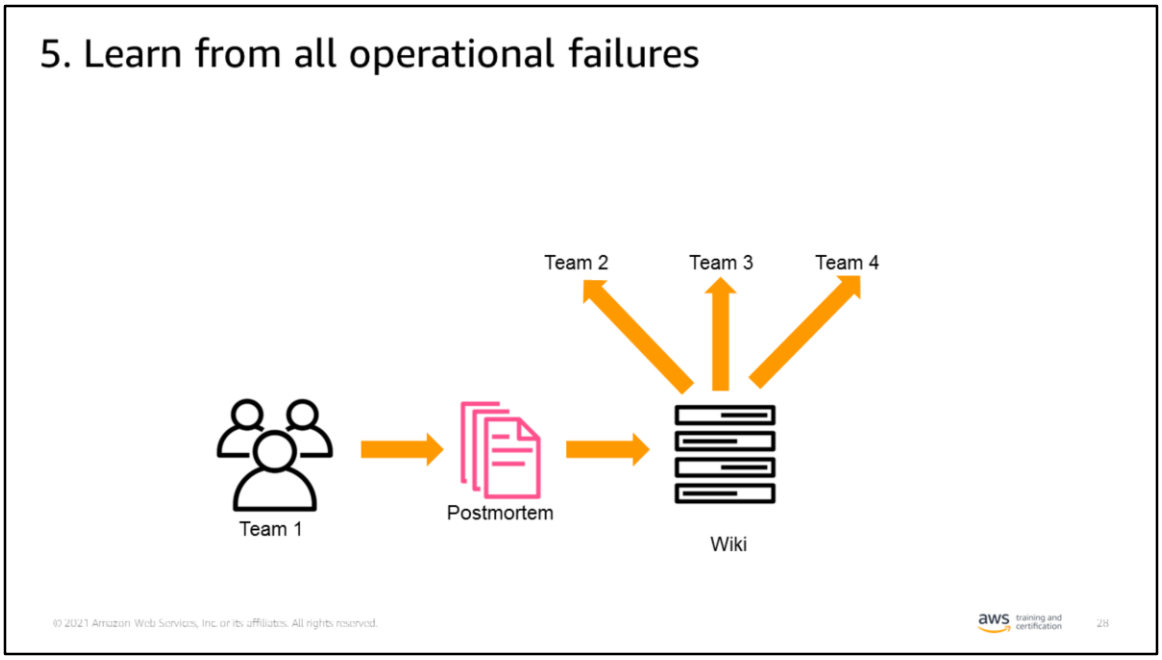
Accept that at some point, “everything fails, all the time” (a quote from Werner Vogels, CTO of Amazon) with your architectures. And plan for as much of this failure as possible.

If we take a three-tier web application, decoupling the three tiers will help us in case we have a failure within those tiers.

Spreading instances across different subnets and Availability Zones anticipates the failure of an instance. But it also anticipates the misconfiguration of a network access control list (network ACL), or networking issues that impact access to that subnet or Availability Zone.

Having primary and secondary Amazon Relational Database Service (Amazon RDS) instances in different Availability Zones helps maintain availability and anticipate failure.

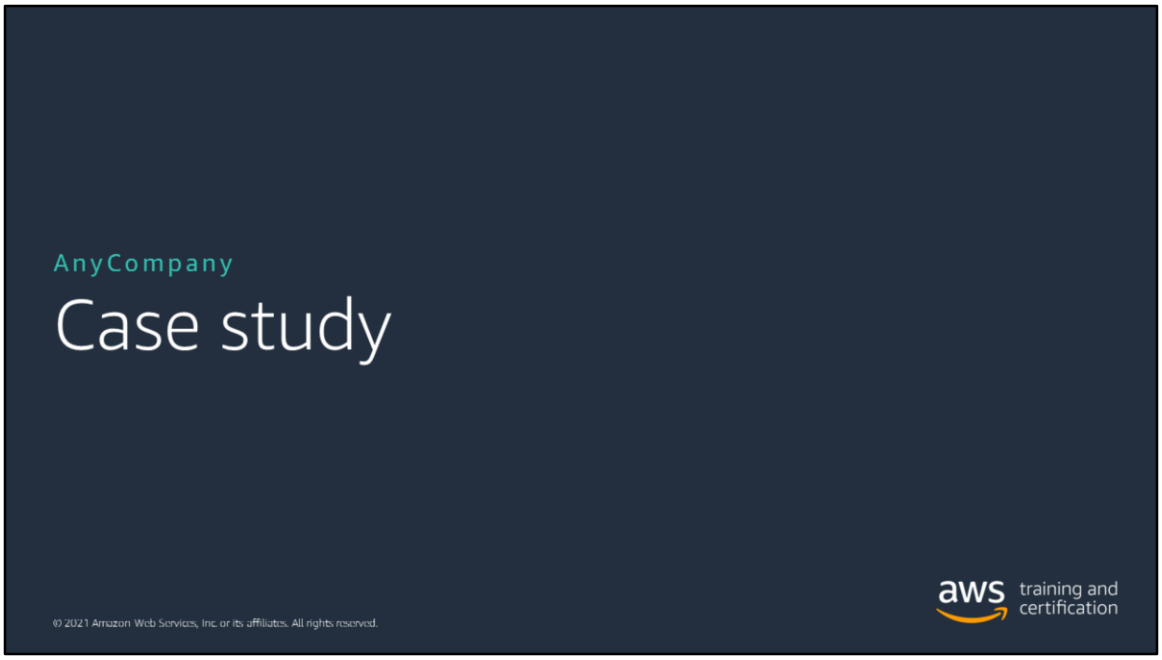
What happens if you have instances in web tier, app tier, and DB tier fail at the same time? Does that impact your availability?

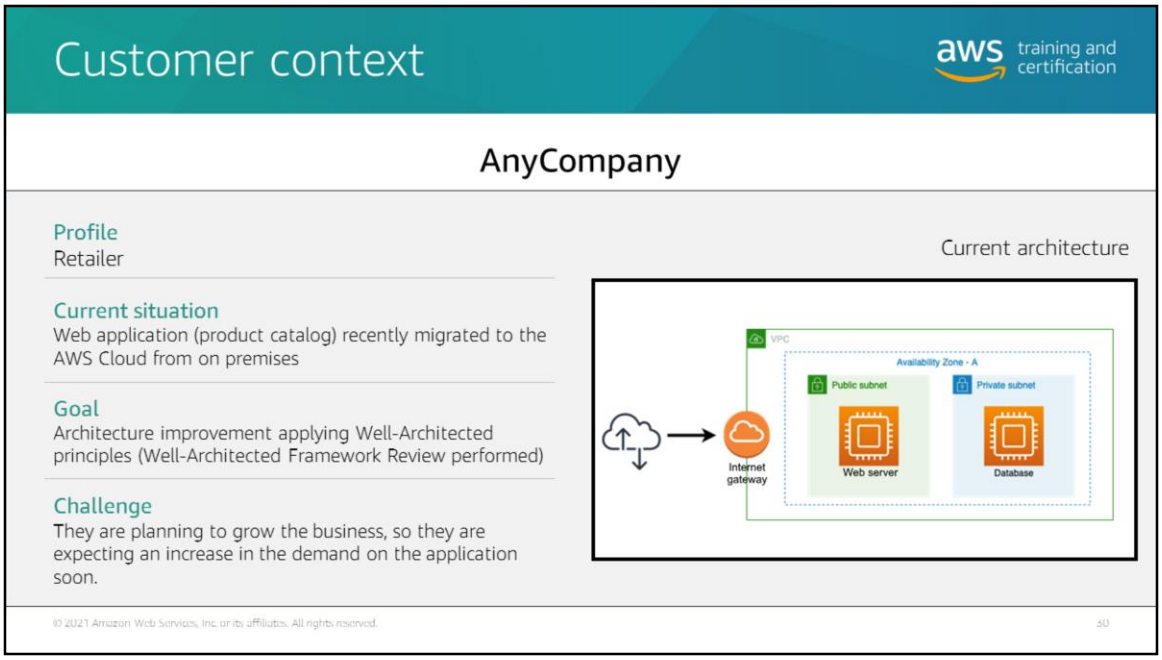


When failures happen, whether simulated or real world, have a postmortem to discover what the issue was. Use it as a learning experience. Use game days to simulate the events that happen and cause failures, rerun simulations and use all the data from them, along with what you know from running a Well-Architected Framework Review on your workload to understand what caused those failures.

Keeping that data from the postmortem and sharing it across your organization so other teams can learn from those failures is key to scaling beyond just one team and one workload. **We've seen customers be successful using a shared collaboration platform, like a wiki, to keep this information ever green on, and to have a repeatable process to share the findings** (and as an ops organization also having root cause analysis documented)

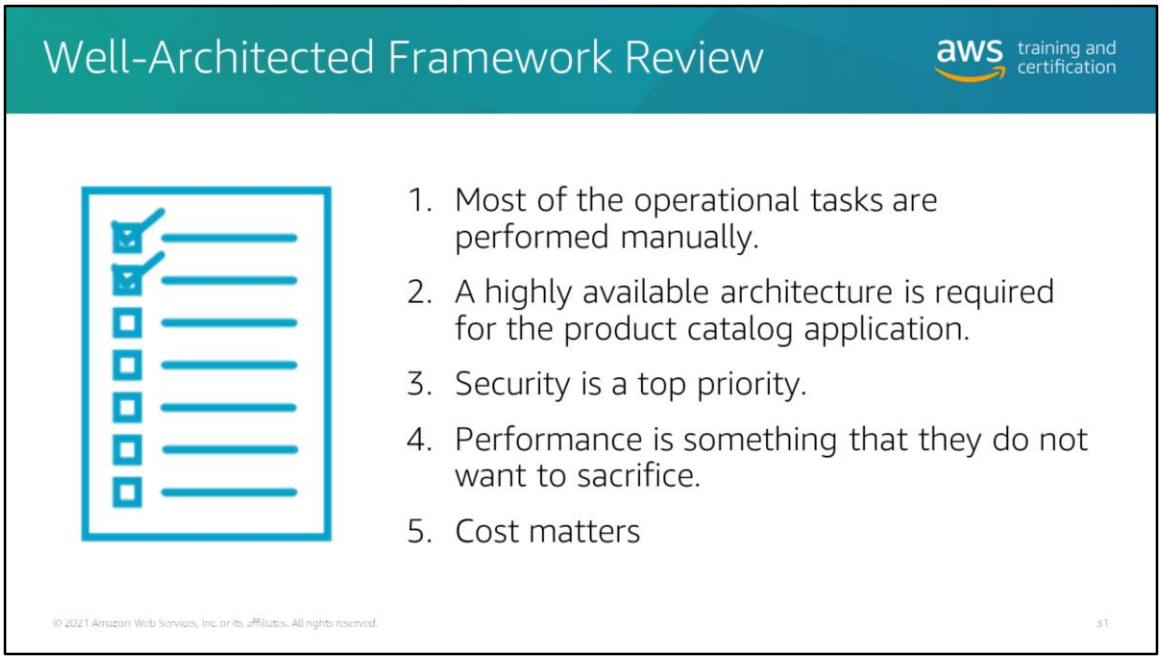
example you can automatically gather logs after a failure for analysis, and they can go into your postmortem and be updated on the wiki





AnyCompany is a retail business. One of the main applications for the company is a product catalog, a web application that they recently migrated to the AWS Cloud from their on-premises environment. Even though the application is functional, they want to have an architecture with best practices applied, because the business is growing. They are looking for an architecture that meets the new performance requirements, mitigates risks, and saves money. For them, it is crucial to use automation.

This is the initial architecture. Your mission is to improve upon it by applying some of the Well-Architected principles, according to the company's needs.

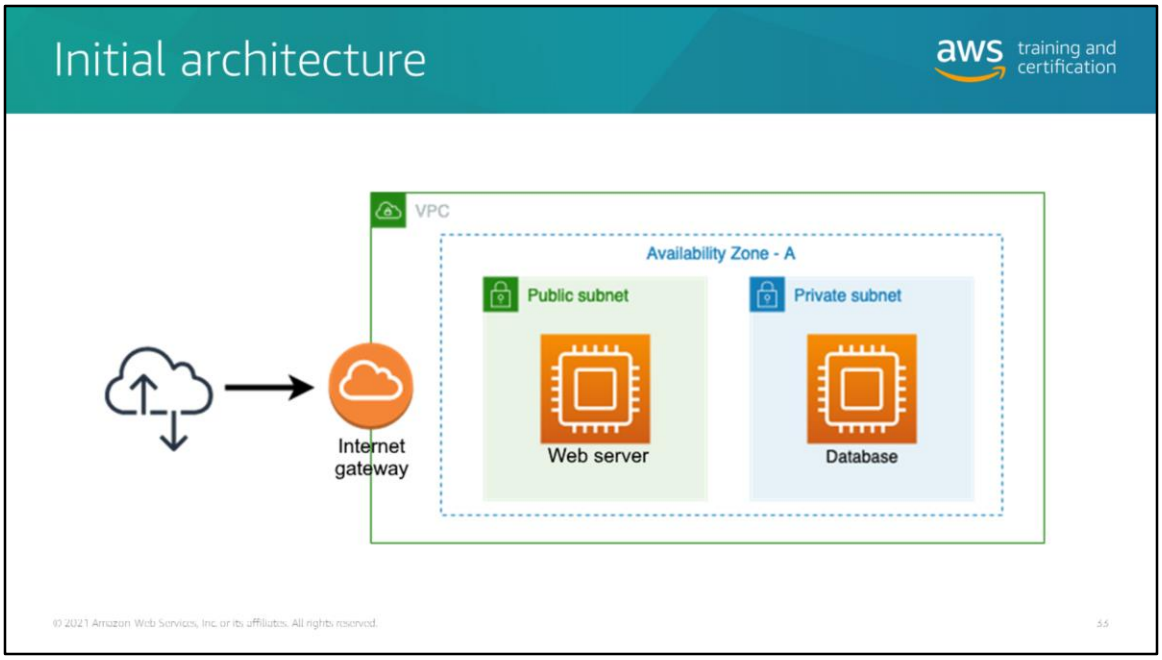


You proposed doing a Well-Architected Framework Review of the application's environment to better understand the current status and the needs. After that review, you identified some insights, the most relevant of which are listed below:

1. Most of the operational tasks are performed manually. AnyCompany wants to automate the process to provide visibility into some important performance metrics, like memory or disk utilization. Additionally, a centralized log monitoring for DB and App is needed.
2. A highly available architecture is required for the product catalog application.
3. Security is a top priority. The more insights available related to this topic, the better.
4. They are not sure about the decision that they made when they chose a t2.micro instance to run the application. Performance is something that they do not want to sacrifice. AnyCompany people want to do some stress tests for the application, especially because they are expecting an increase in the demand on the application in the near future.
5. Cost matters. Some applications are not using approved instance types in accordance with AnyCompany's architecture standard. This has been driving unnecessary cost due to over-provisioned resources in non-production environment.

6. The information above is your starting point to help enhancing the architecture and achieving organization objectives. You may identify more opportunities for improvement in this architecture but, for the purposes of the workshop, just focus on these findings.

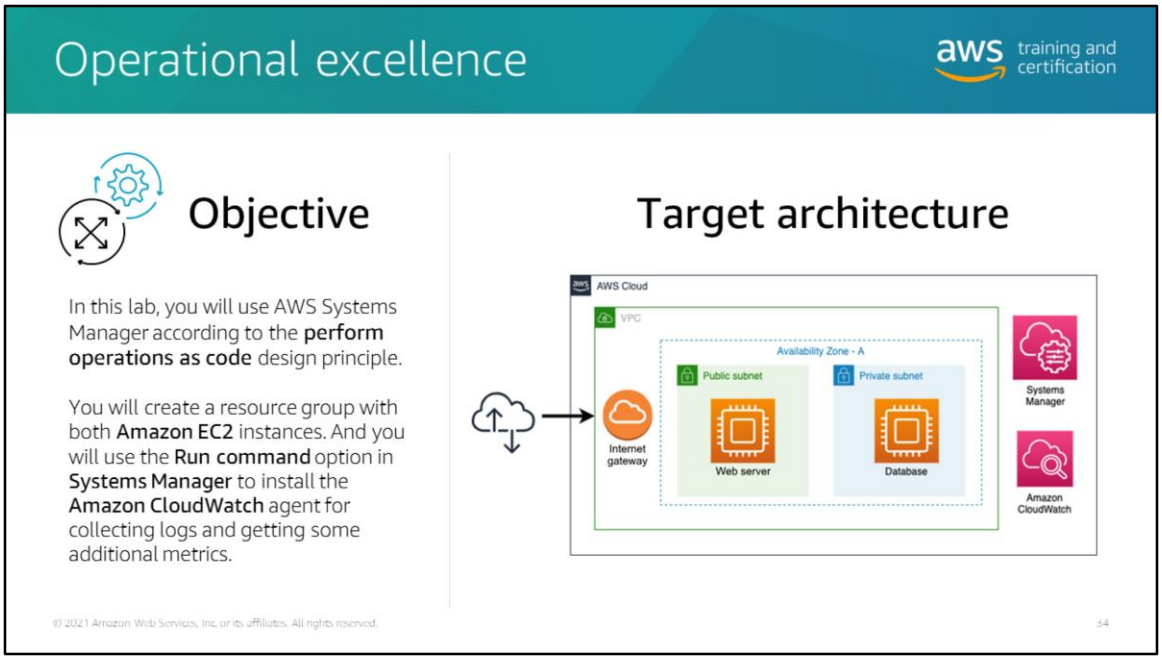
Lab guide: https://main.d2azidedm760yt.amplifyapp.com/work0/



Let's apply what we have learned so far, starting with this basic architecture.

Remember that one of the insights found in the Well-Architected Framework Review was about the need to automate tasks. AnyCompany people are **performing many operational tasks manually.**

One of the issues they mentioned was a lack of visibility into important metrics like memory and disk utilization for the Amazon EC2 instances. They would like an **automated** process to get that information. Additionally, they need centralized log monitoring for the DB and App instances.

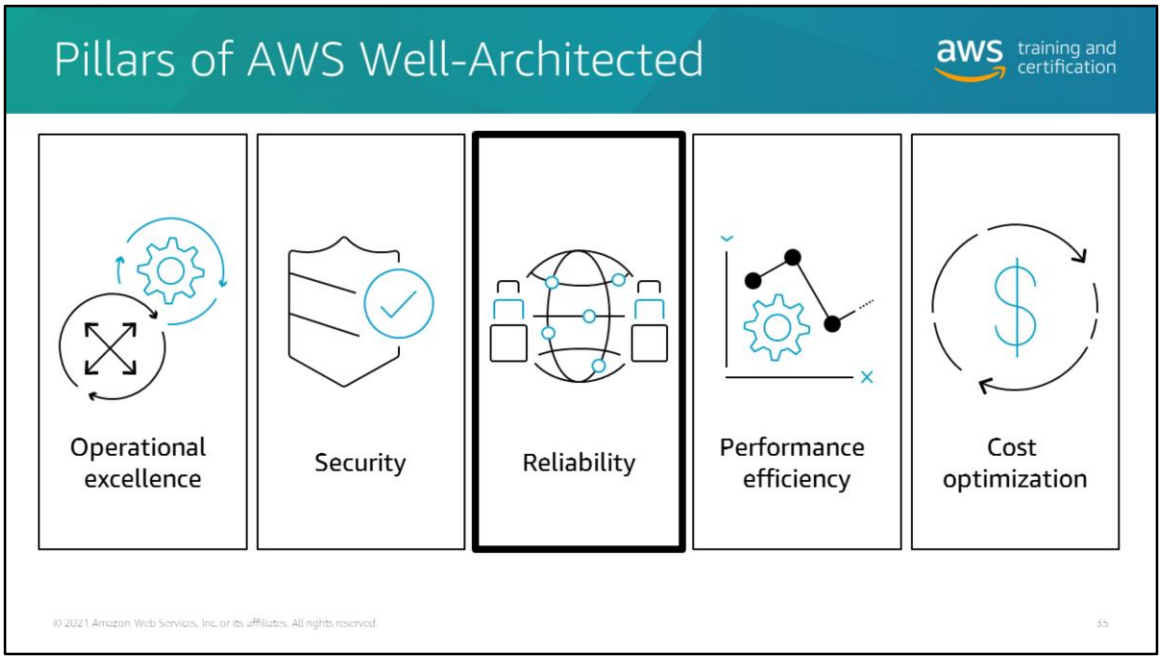


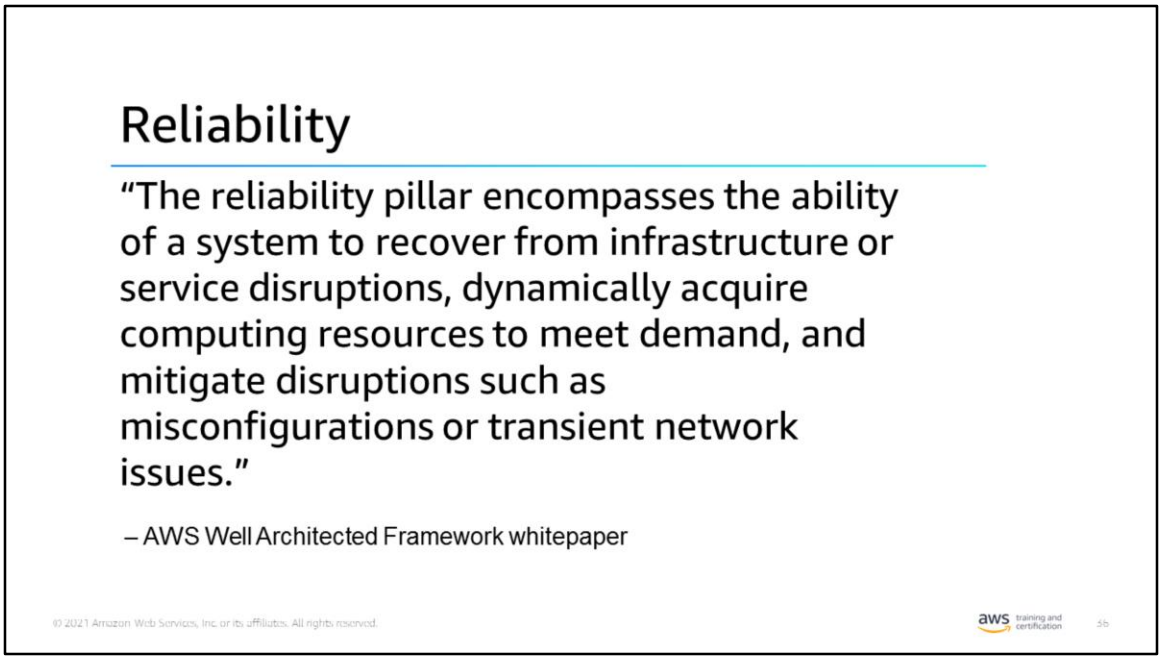
Estimated duration: 1 hour

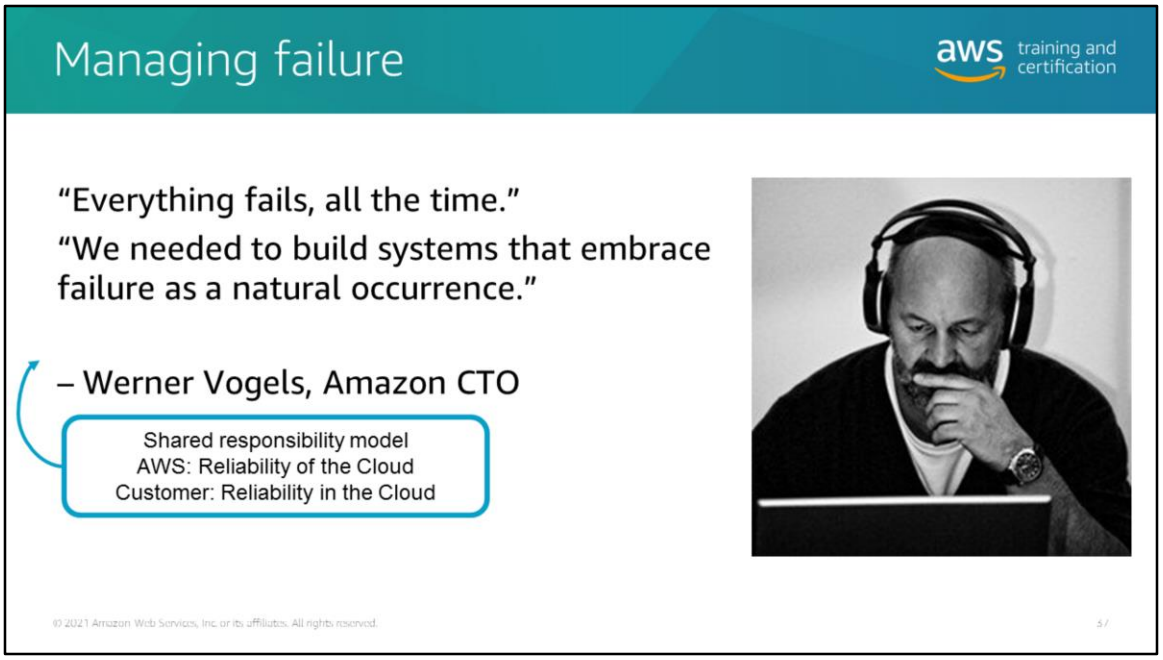
Wrap-up after lab completion:

In a traditional environment, you would have to set up the systems and software to perform administration activities. You would require a server to run your scripts. You would need to manage authentication credentials across all your systems.

Operations as code reduces the resources, time, risk, and complexity of performing operations tasks. And it ensures consistent running, so **your organization can focus on delivering more value to customers instead of reacting to emergencies**. You can take operations as code and automate operations activities by using scheduling and event triggers. Through integration at the infrastructure level, you avoid “swivel chair” processes that require multiple interfaces and systems to complete a single operations activity.





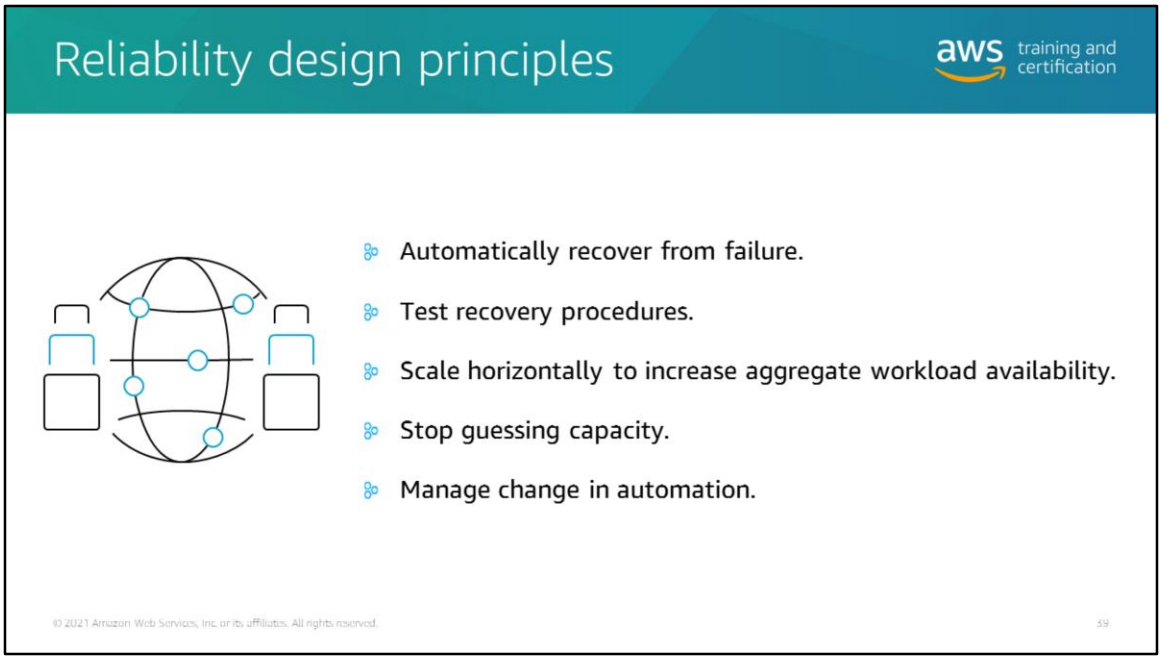


I am sure most of you must have heard this quote from Werner Vogels, CTO of Amazon. This was true before the cloud existed and before the dawn of computers, and certainly in the age of distributed computers. In your on-premises environment as well, you have network failure, HDD failure, and power supply or cooling failure. And this remains true. You have to deal with that in an on-premises environment and work around it, and it can be quite a burden. The nice thing about AWS Cloud is that we are running a physical data center as well (although, we manage it). Our computers have failures too, but we generally abstract that from you because we have redundancy. We communicate our service health dashboard to you in a transparent manner. Everyone will have failures and you have to design your systems to withstand and recover from those failures.

Note: In your own data centers, it was often difficult to simulate failure, so the recovery planning was often only a thought experiment.

Reliability is a shared responsibility between AWS and customers.

- AWS: AWS components abstract away low-level failures like hard-drives and power supplies. AWS services are built using best practices like redundancy.
- Customer: Architect and build reliable solutions using AWS tools and services.



Let's think about how we might consider reliability in a traditional environment. We often **test if things work normally**—we check if it meets expectations. But we **rarely test what happens after things fail.** So the first time we test our recovery process is in the middle of a **live reliability failure**…which is not a great learning experience! This is why you used to see **lots of systemic failures**—X failed and then Y failed (Y being the thing we never got to test).

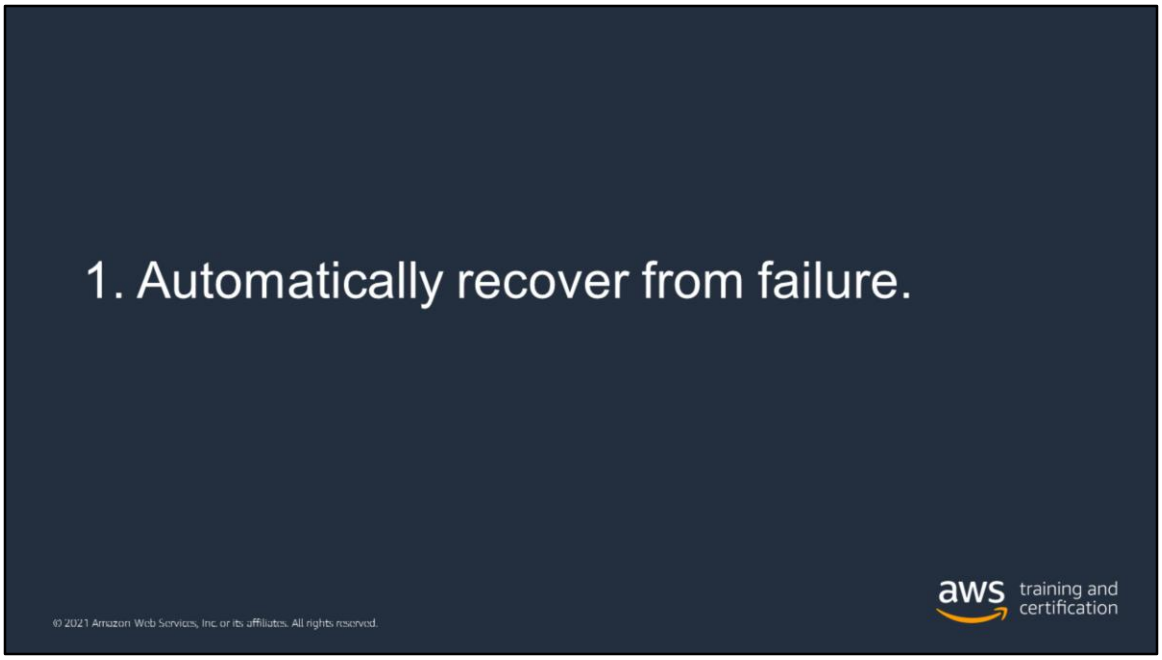
In the cloud, constraints have been removed. This allows us to adopt these design principles to build and operate cloud-native architectures.

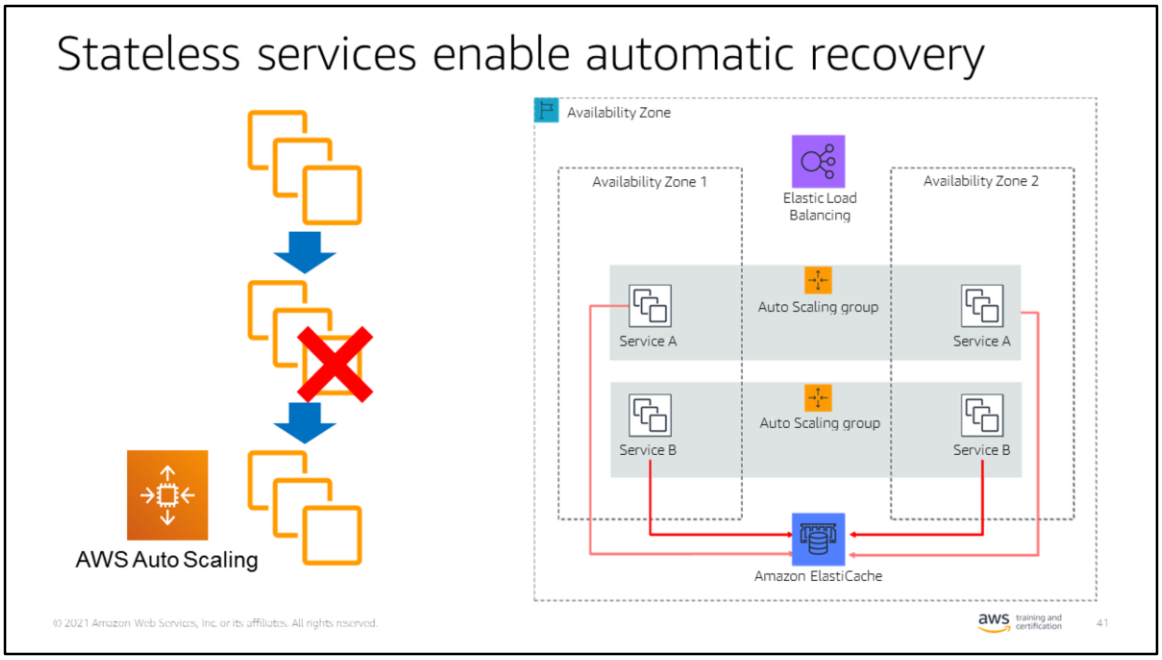
to **test beyond destruction** to make sure recovery procedures are **automatic and successful**,

we can have **multiple resources answering requests** – such that a failure in any single component always has **siblings who can step in** and absorb the load

we can use the **horizontal scaling** to meet demand

and when we make **changes to our environment,** we can do that **through code,** and apply the same best practices we would apply to application code.





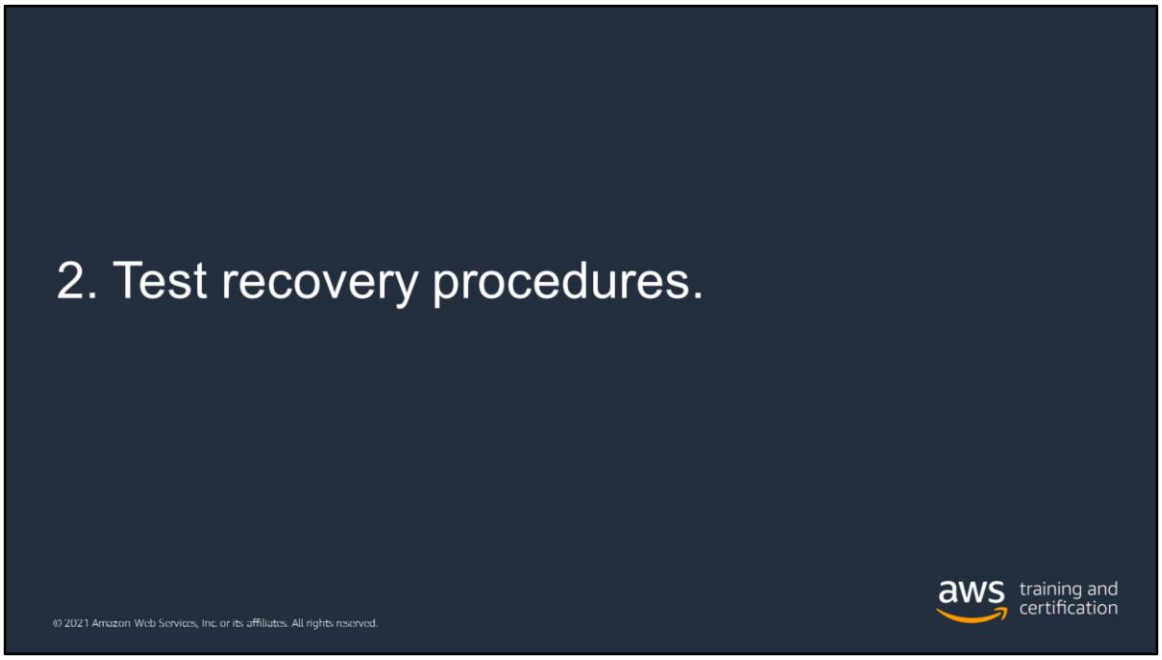
To recover from many failures, killing and replacing the EC2 instance is often a good approach.

You need to configure Auto Scaling to do this, with the right health checks. And design your application to be stateless.

You can offload state to a store like Amazon ElastiCache.

Note: Auto Scaling will drain requests before terminating an instance.

=====
In systems that apply a recovery-oriented approach, many different categories of failures are mapped to the same recovery strategy. An instance can fail due to hardware failure, operating system bug, memory leak, or other causes. Rather than building custom remediation for each, treat any cause as an instance failure, terminate the instance, and replace the instance.

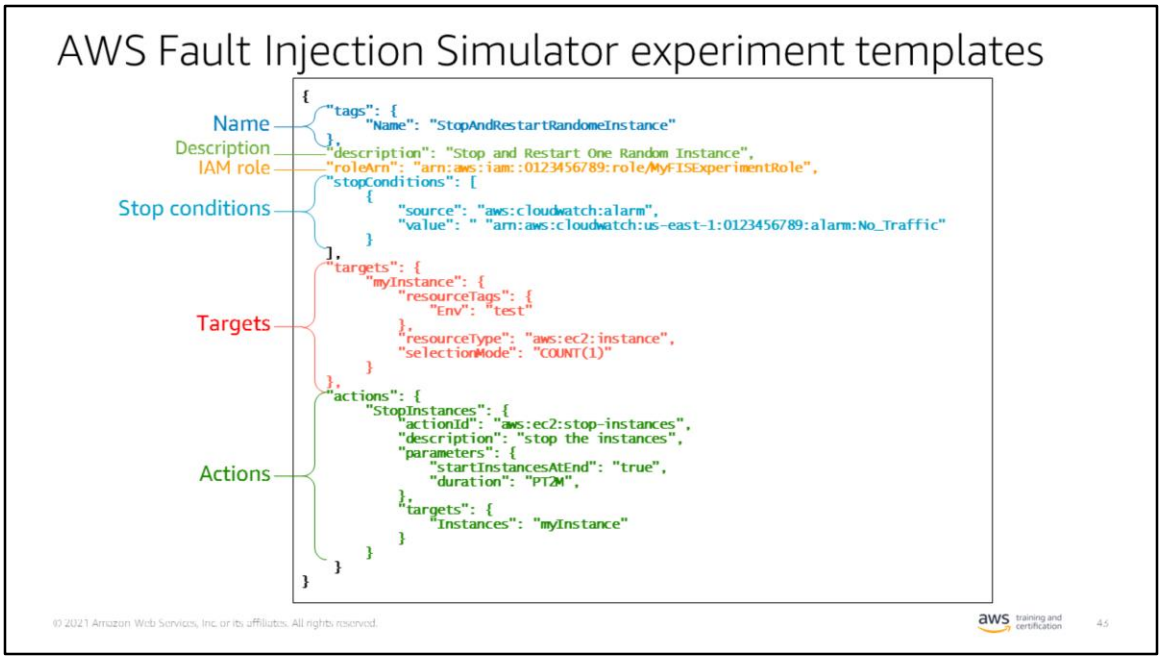


We talked about testing and recovery, but this whole talk today is about resiliency. Let's discuss that.

We already spoke about capability to withstand turbulent conditions, like component failure and network issues. What is the key term here? I think it is *experimenting*.

It is important to realize that outages rarely happen because of one single failure. It is often a combination of small failures happening at the same time or in a sequence…and that's just hard to reproduce.

You might hear terms like fault injection and failure testing. Those are important parts of **chaos engineering**, but not the complete picture. Chaos engineering is a scientific method of setting up hypotheses. If this failure occurs, my system will react in a certain way. If it happens, great; otherwise fix that and iterate and move to the next one. This is the important part, experimenting and continuously iterating.



AWS Fault Injection Simulator is a fully managed service for running fault injection experiments on AWS that makes it easier to improve an application's performance, observability, and resiliency.

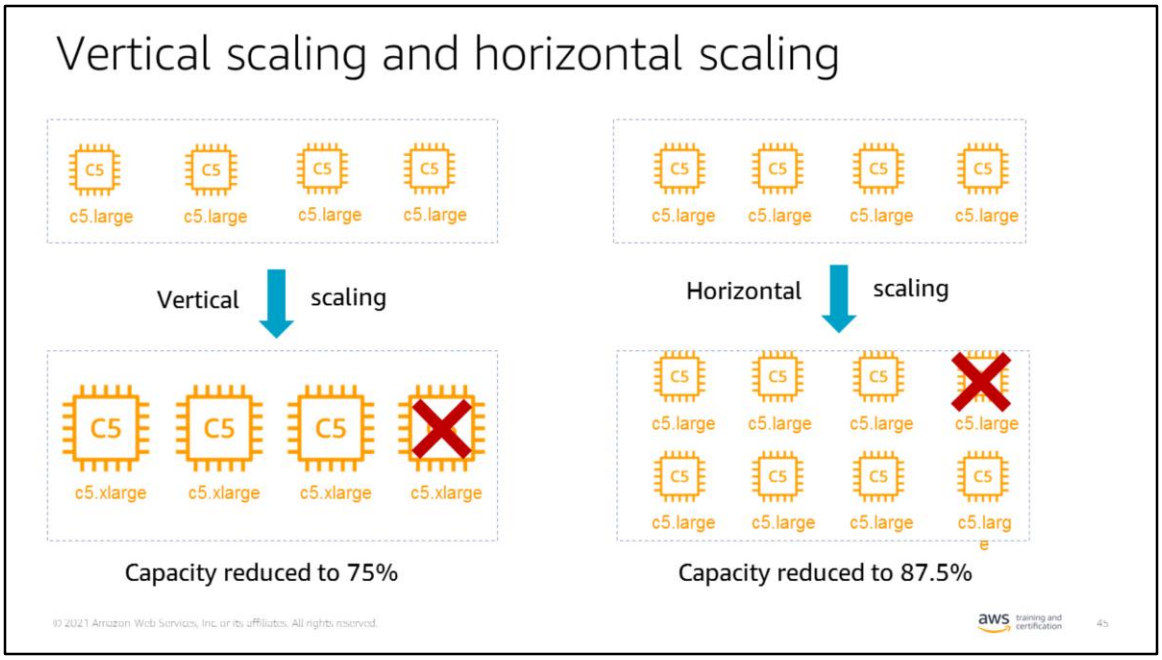
When you create your chaos experiment from the console, AWS Fault Injection Simulator automatically creates a template for you—a template you can reuse later.

When you start your experiment, AWS Fault Injection Simulator injects failures, real failures, on your AWS resources.

It is best practice to have CloudWatch alarms monitoring your AWS resources and workload. And define some stop conditions that will automatically stop the experiment if they change status. You will be able to integrate your own monitoring solutions with Amazon EventBridge to initiate the stop conditions.

So a template looks something like that. Again, you can use JSON or YAML as you prefer. You have the name (using tags), the description, the AWS Identity and Access Management (IAM) role, the stop condition, the targets, and finally the actions to perform on the targets.

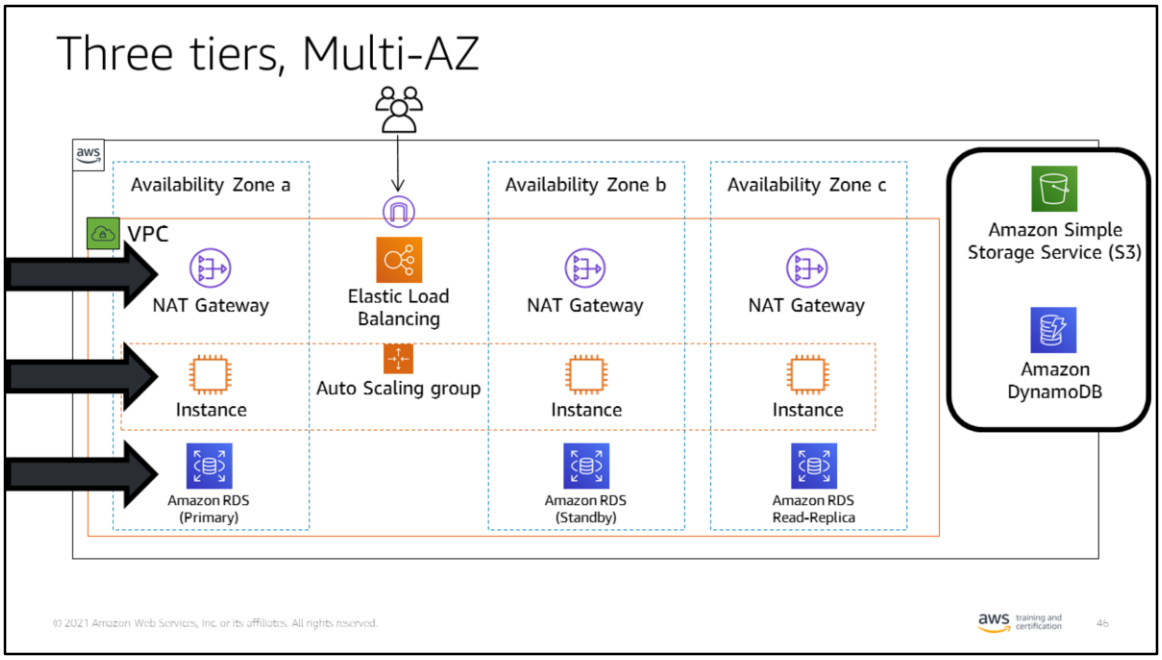
Notice the use of tags to refine the selection. It is the best practice to always use tags. And in this case, only the instances tagged with "Env": "test" will be targeted.



Replacing instances with larger instance types is vertical scaling. It will reach a limit eventually.

Adding more instances to your total pool is horizontal scaling. There is no limit to this.

Horizontal scaling will ensure failures have a lesser impact than vertical scaling does.



Animation

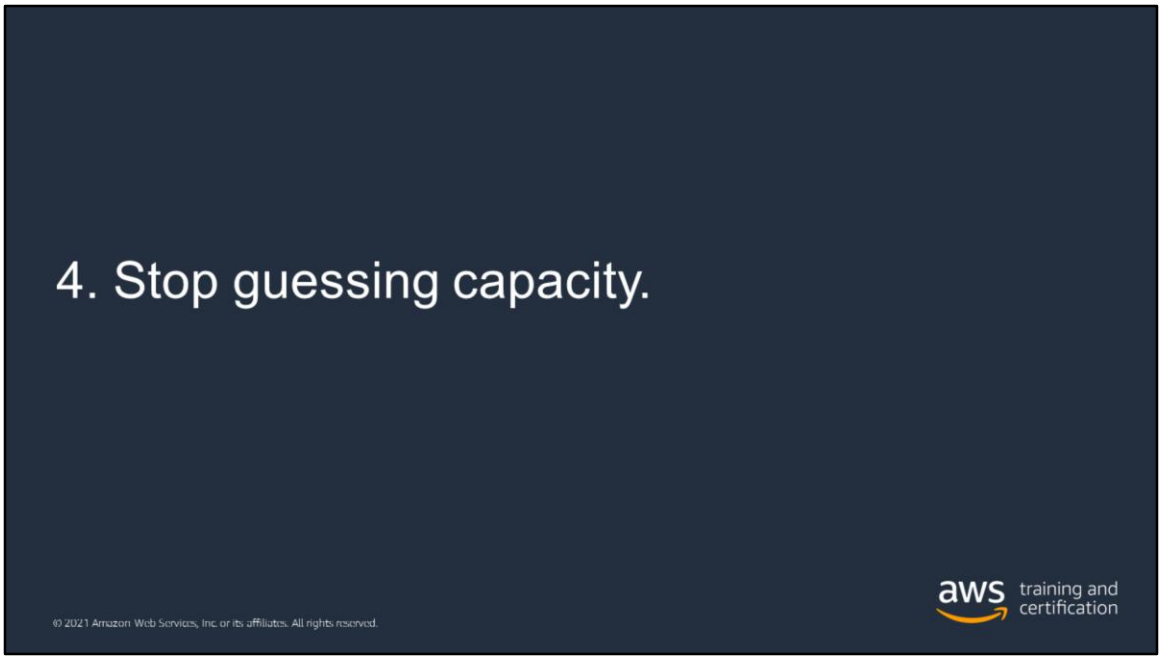
1. **Click:** Elastic Load Balancing is redundant in all Availability Zones.
2. **Click:** If an EC2 instance in an Availability Zone fails, traffic goes to other Availability Zones and a new EC2 instance is replaced in working Availability Zones.
3. **Click:** If the primary fails, the standby is promoted to primary.
4. **Click:** These services automatically replicate data and compute across multiple Availability Zones—regional, not zonal.

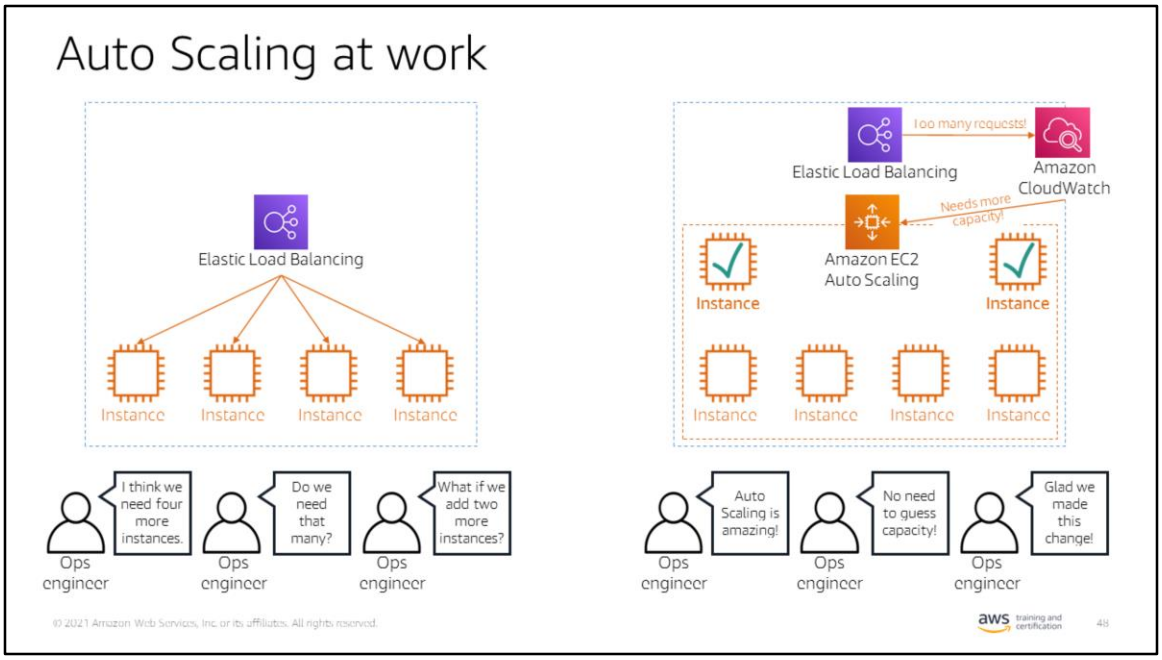
Amazon RDS you simply tell it to be Multi-AZ

Amazon Simple Storage Service (Amazon S3) is our durable, highly available, and infinitely scalable object store. Amazon DynamoDB is our fast and flexible nonrelational database service for any scale.

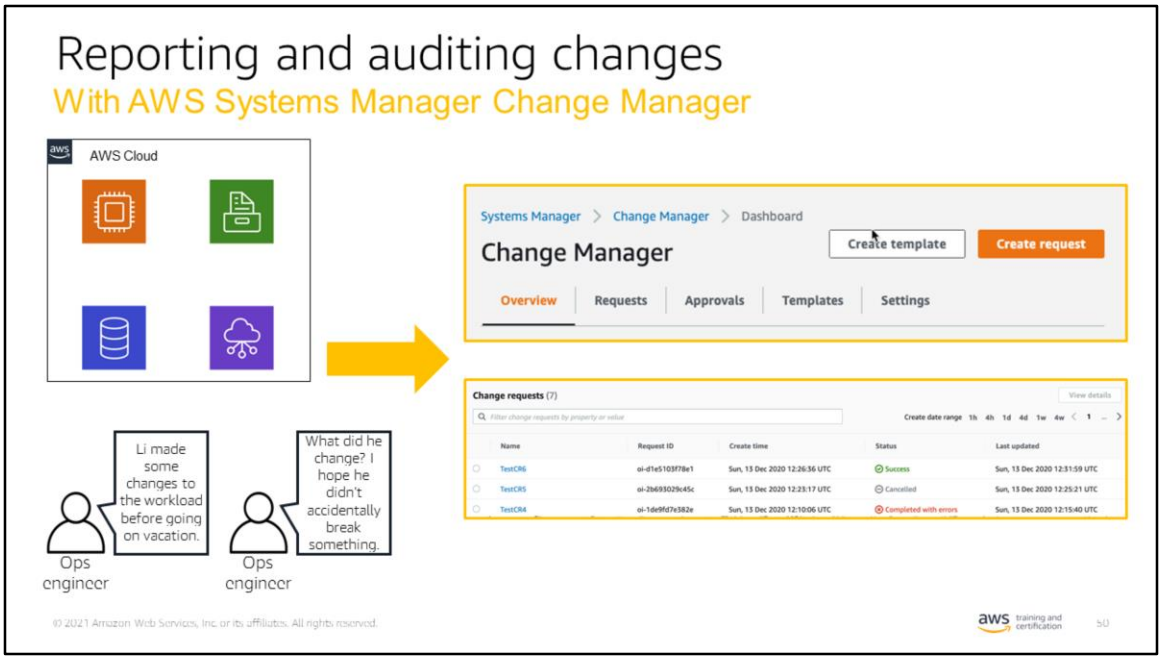
Amazon S3 is designed to provide 99.999999999% durability of objects over a given year. It redundantly stores your objects on multiple devices across a minimum of three Availability Zones in an Amazon S3 Region.
https://aws.amazon.com/s3/faqs/

DynamoDB: All your data is stored on solid state drives (SSDs) and is automatically replicated across multiple Availability Zones in an AWS Region, providing built-in high availability and data durability.
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html



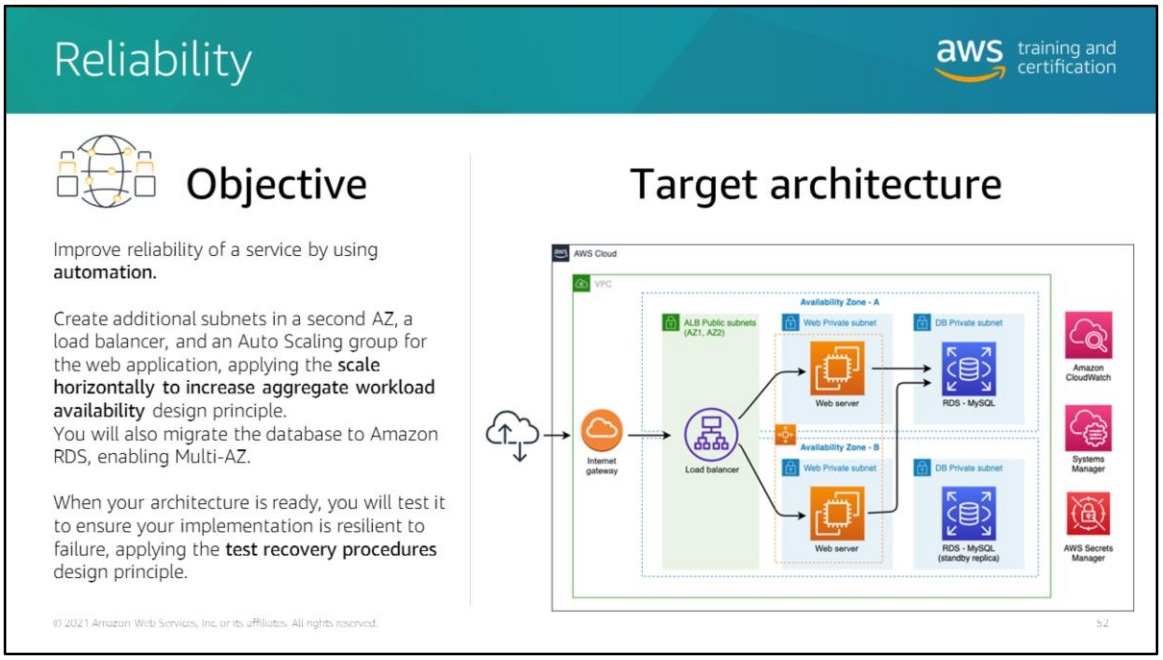


Provision capacity automatically based on demand. Identify the right metric or KPI that indicates demand. Don't do it just on CPU or memory. Auto Scaling can be applied to Amazon EC2, container tasks, DynamoDB throughput, and so forth.



Automated change control will reduce risk of manual error, increase speed, and ensure consistency. And it can be readily tracked.

Refer to this blog post:
Introducing AWS Systems Manager Change Manager
https://aws.amazon.com/blogs/aws/introducing-systems-manager-change-manager/



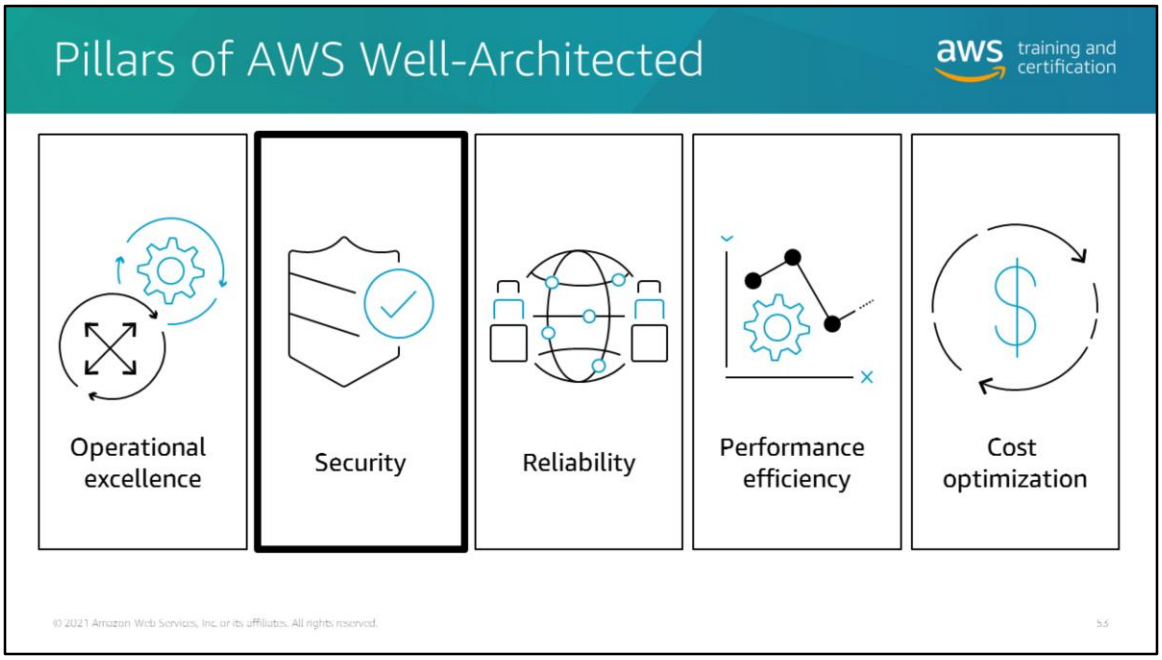
During the Well-Architected Framework Review , AnyCompany discovered how important it is to build a **resilient solution** as required to protect against failure.

The customer understood the risks and business impact in case of failure.
Estimate duration: 1 hour and 10 minutes

Wrap-up after lab completion:

Deploying multiple servers and Elastic Load Balancing lets a service recover after the loss of an instance with no availability disruptions. Traffic from the user is automatically routed to healthy instances. Amazon Auto Scaling ensures unhealthy hosts are removed and replaced with healthy ones to maintain high availability.

Availability Zones are isolated sets of resources within a Region. Each has redundant power, networking, and connectivity, housed in a separate facility. Each Availability Zone is isolated, but the Availability Zones in a Region are connected through low-latency links. AWS provides you with the flexibility to place instances and store data across multiple Availability Zones within each AWS Region for high resiliency.



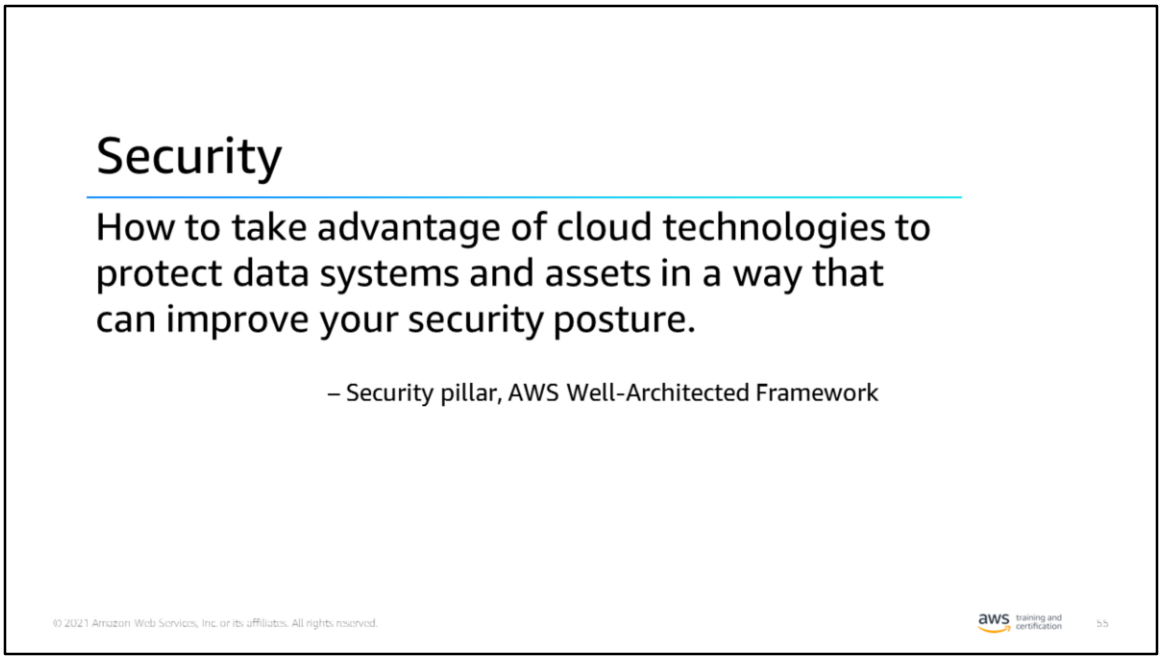
Security is always job zero

"Protecting your customers should always be your number one priority, and it certainly has been for AWS…from both an operational perspective as well as tools and mechanisms; it will forever be our number one investment area."

– Werner Vogels, Amazon CTO

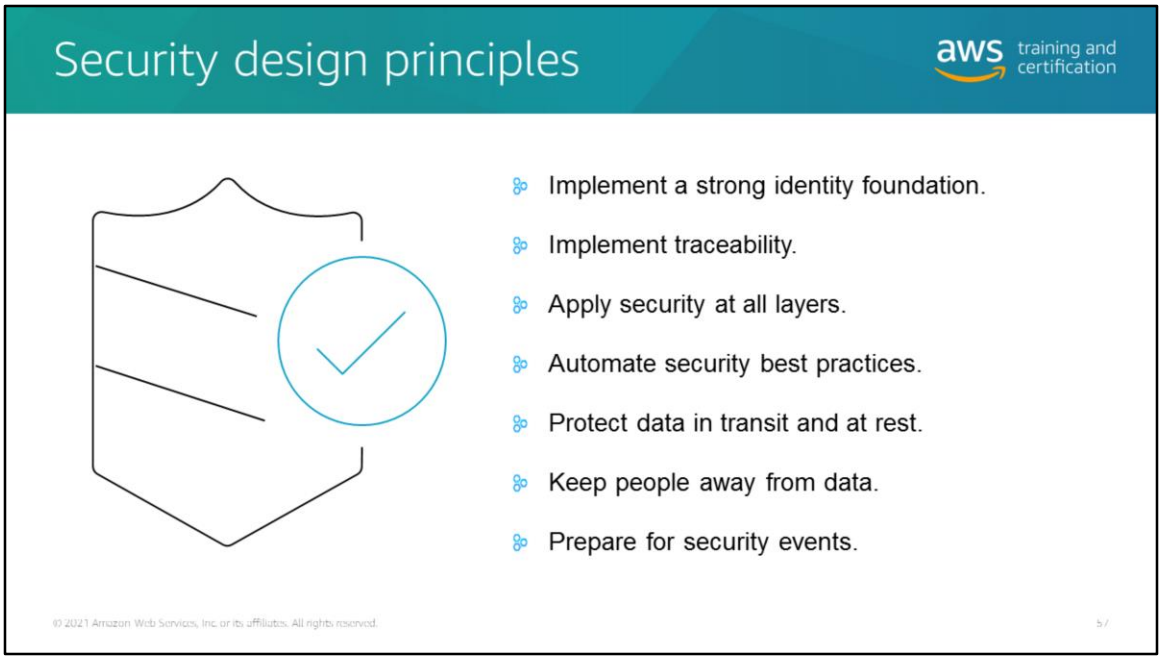
54

If you want to build workloads that are trusted by your customers, they will have to demonstrate security across a broad range of scenarios. AWS provides services that are specific in nature and can work to address the wide range of security challenges of our customers, regardless of their size and industry.



We want to ensure that our customers take advantage of the services we provide to protect data and assets. This can help them minimize risk exposure of their business and concentrate on providing great service to their customers.

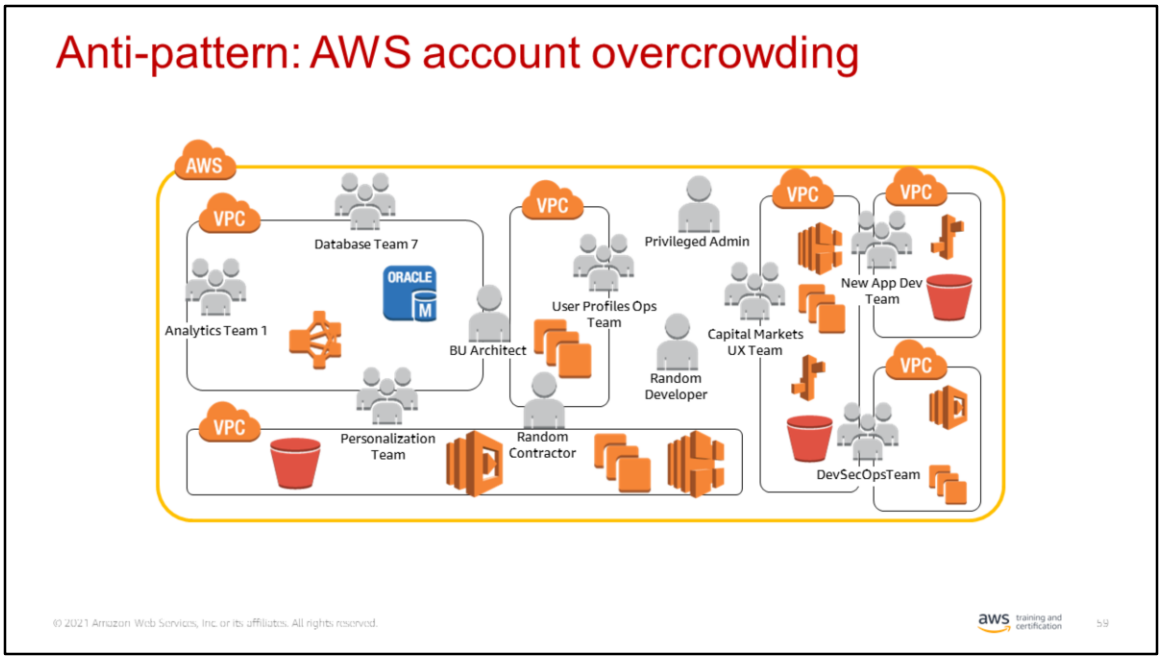
There are seven design principles within the security pillar. When determining if a customer's security posture is adequate, you should try to look for evidence of each principle within their architecture. We will go through each design principle and provide examples of where it can be applied.



There are seven design principles for security in the cloud.

Implement a strong identity foundation.

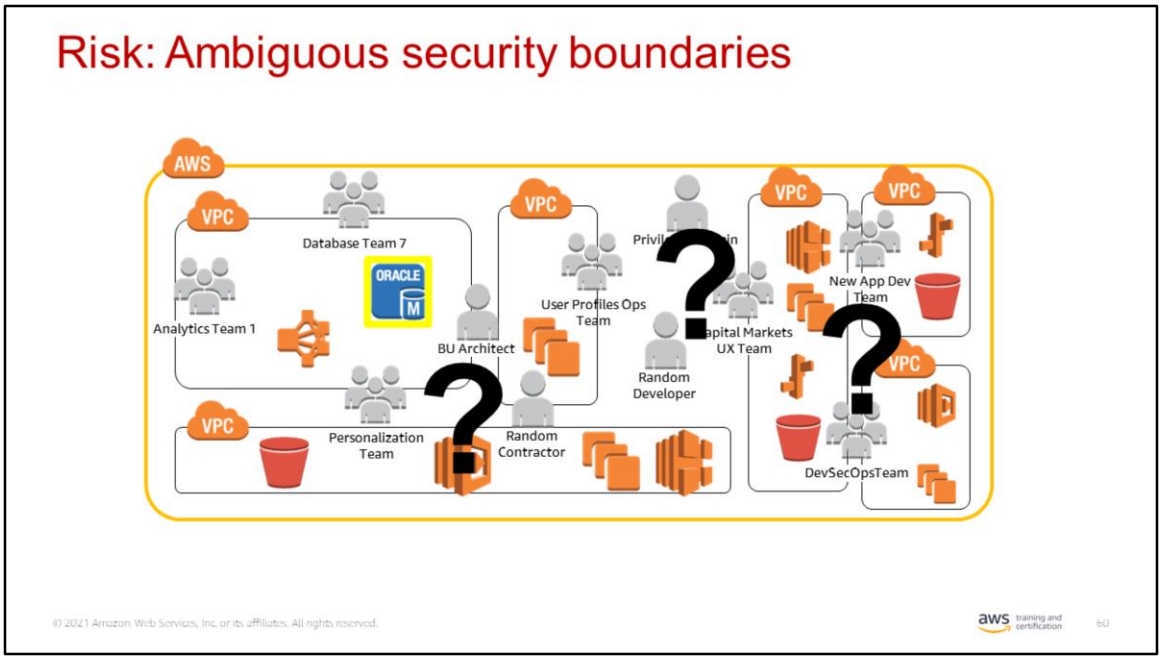
Implement the principle of least privilege and enforce separation of duties with appropriate authorization for each interaction with your AWS resources. Centralize identity management and aim to eliminate reliance on long-term static credentials.



Let's look at a quick anti-pattern (an example of a common mistake).

Look at this example. Initially, this environment made use of virtual private clouds (VPCs) within a single account structure to run several different workloads. Each had different teams owning and developing the architecture.

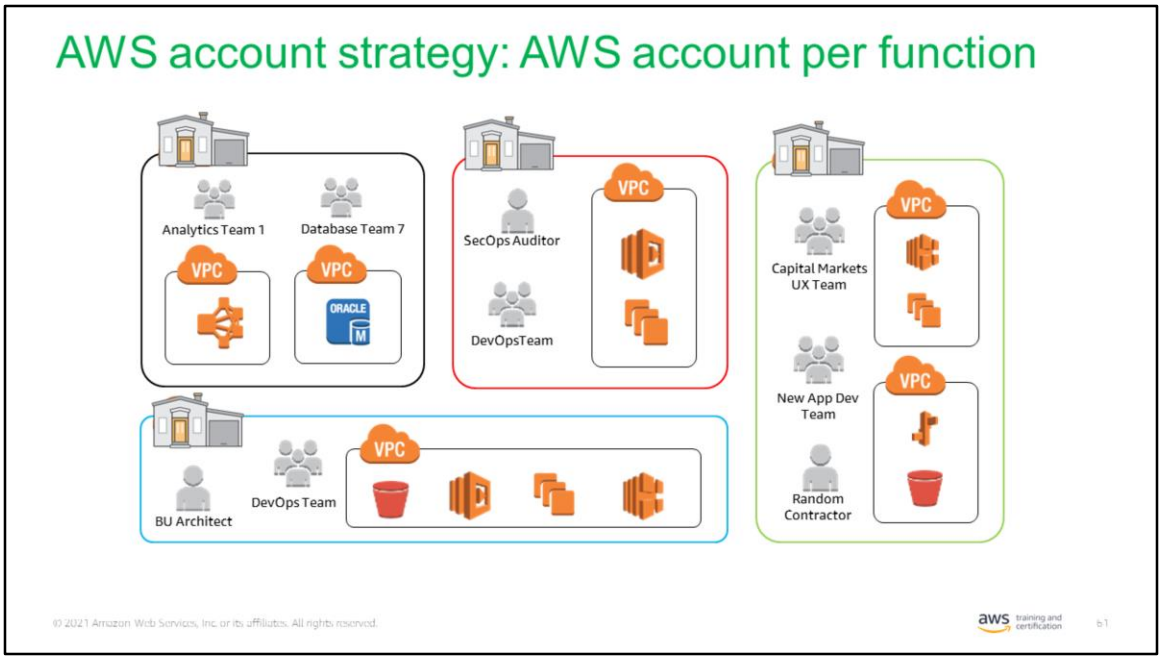
As the workload matures, more and more services are added, together with more employees to manage the increasing complexity of the environment and its associated business deliverables.



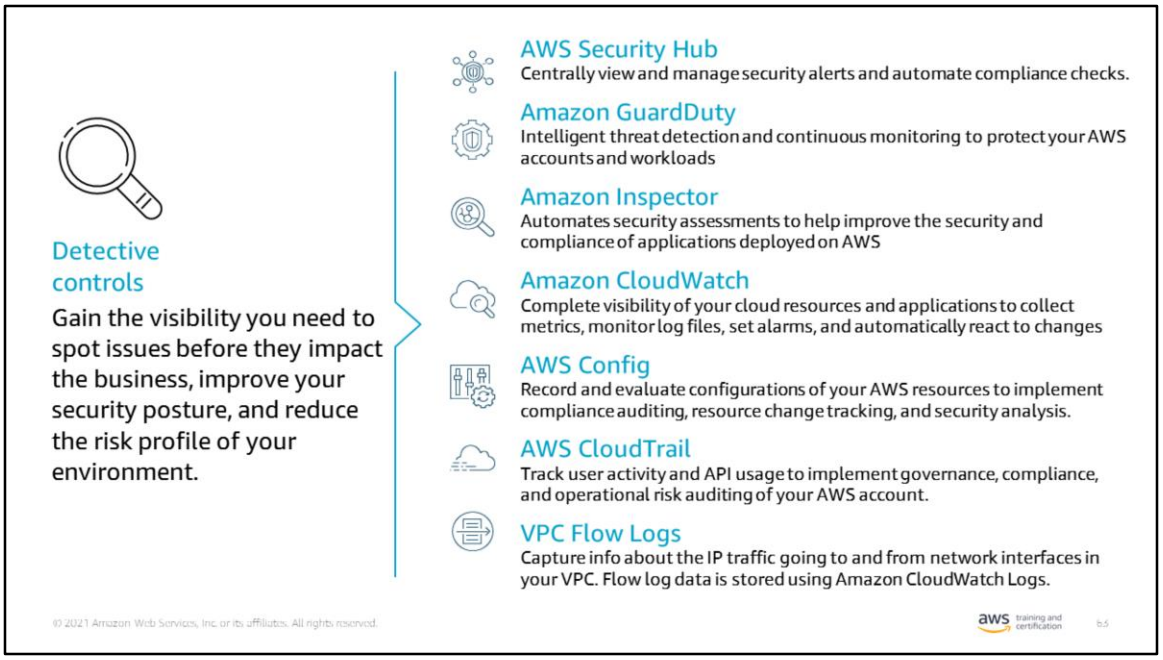
Now the company is at a reasonable size in terms of growth and customer demand. But it is still operating in the same manner with the same structures that they originally put in place from an account perspective.

Because of the size of the organization, they are now subject to audit, which selects a database at random, and seeks who has access and who is the rightful owner and maintainer of that data set.

It’s a difficult question to answer because there are no real lines of demarcation within the environment. In fact, it is increasingly difficult to see what is actually going on.



This is much better: minimum access for each team within an account, with logging being centrally pushed to an account solely designated for that purpose. Each workload can be permissioned with its own individual access policies and logging, which makes traceability much easier.



The emphasis on detective controls is to:
1. Identify an issue, and
2. To take action based on the event.

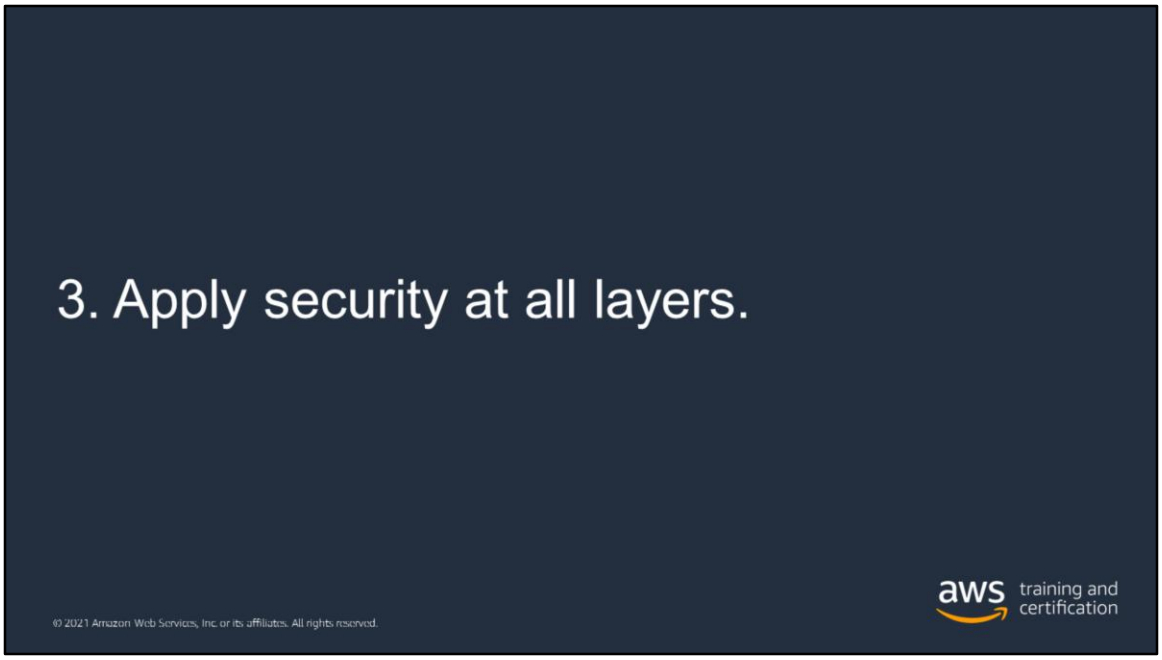
To highlight this, you can specifically talk about how Amazon GuardDuty performs continuous monitoring of workloads, and can when combined with the integration of Amazon CloudWatch Events to set off a Lambda function. Such as potentially updating ACLs based on seeing traffic from a block of IPs and sending a notification to the Devs/Admin team the event occurred.

To further highlight detective controls, you can say that part of having detective controls does mean having visibility into activity, but that's only half of it. Such as turning on CloudTrail to see account level API events or Flow Logs for networking traffic, but that's only half of it. You must have a second part, which is to alarm off set thresholds and actions taken off those alarms.

There is no point to set off an alarm if there is no specified manual or automated action. In the case that you might have a manual process laid out, where an alarm is set off and pages a dev to do something manually while the automation is being

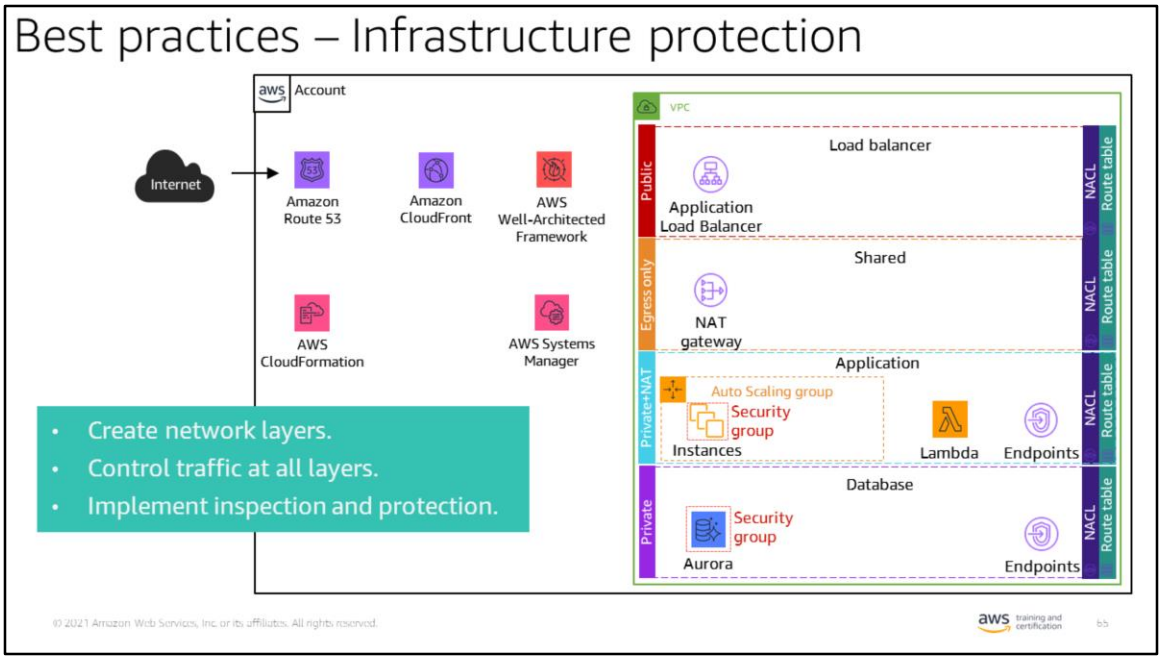
developed and will be implemented.

Detective controls allow for visibility of a particular event after it has occurred. Ensure that you are building an environment where traceability is possible throughout the architecture. Ensure that CloudTrail is turned on in all Regions. And ensure that you have configured VPC Flow Logs together with logging for all components present, from load balancers to databases. In addition to this, look to see if you can automate specific checks using AWS Config to pull specific events from logs. And escalate based on automation rather than manual effort after an event has occurred.



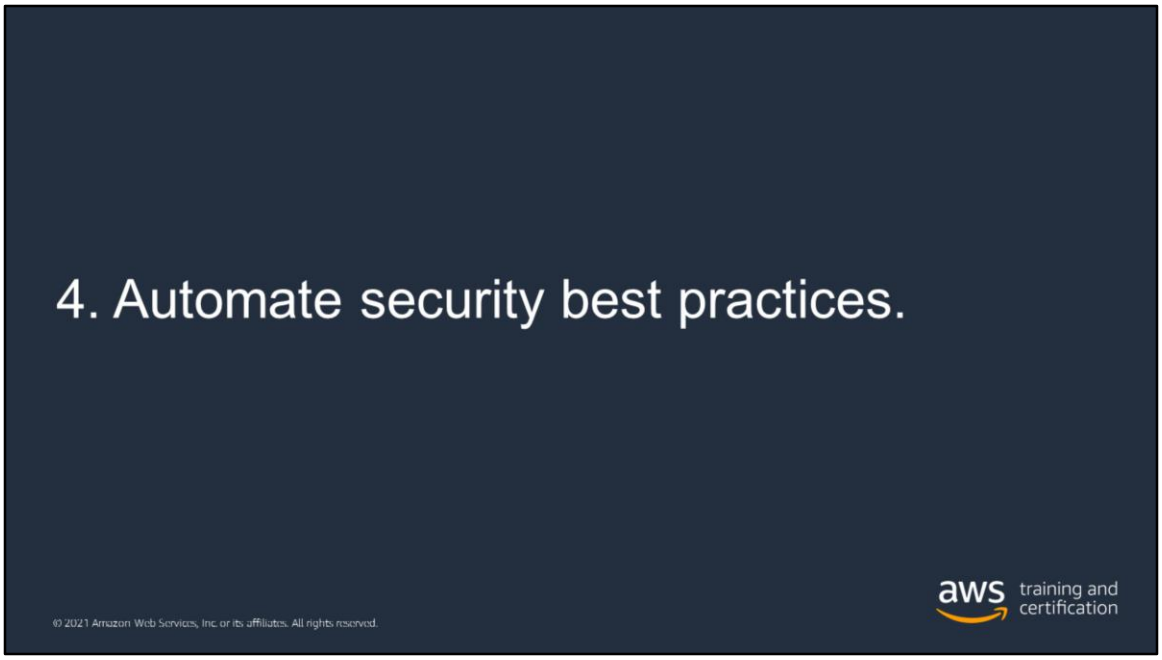
Apply security at all layers:

Apply a defence in depth approach with multiple security controls. Apply it to all layers (for example, edge of network, VPC, load balancing, every instance and compute service, operating system, application, and code).



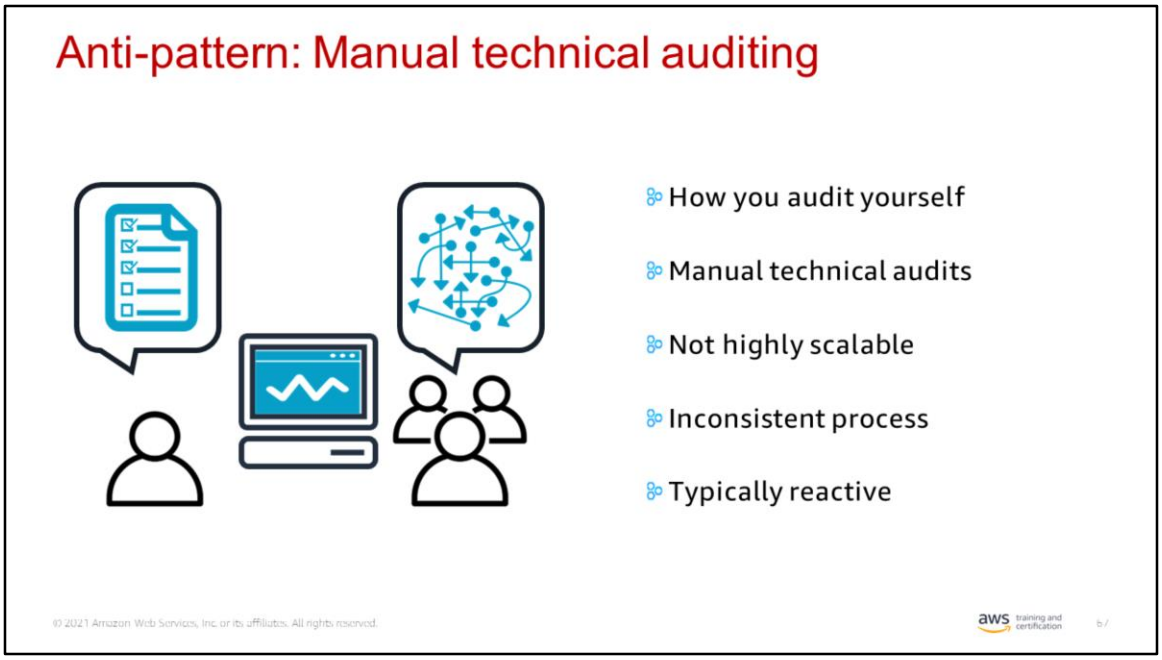
Make sure that you make full use of security at multiple layers of your infrastructure.

Where possible, use Amazon CloudFront to assist with distributed denial of service (DDoS) mitigation. This can also include AWS Well-Architected Framework integration. Within your VPC, reduce your attack surface with an Application Load Balancer through to your application layer. Add permissions with network access control lists (ACL) and security groups appropriately.



Automate security best practices.

Automated software-based security mechanisms improve your ability to securely scale more rapidly and cost effectively. Create secure architectures, including the implementation of controls that are defined and managed as code in version-controlled templates.



- Manually go team to team and run interviews.
- Time consuming and doesn't really scale
- Internal audits don't run as frequently as they should.
- Move toward automated audits within AWS.

Best practice: Continuous automated auditing

DevSecOps: Security as code:

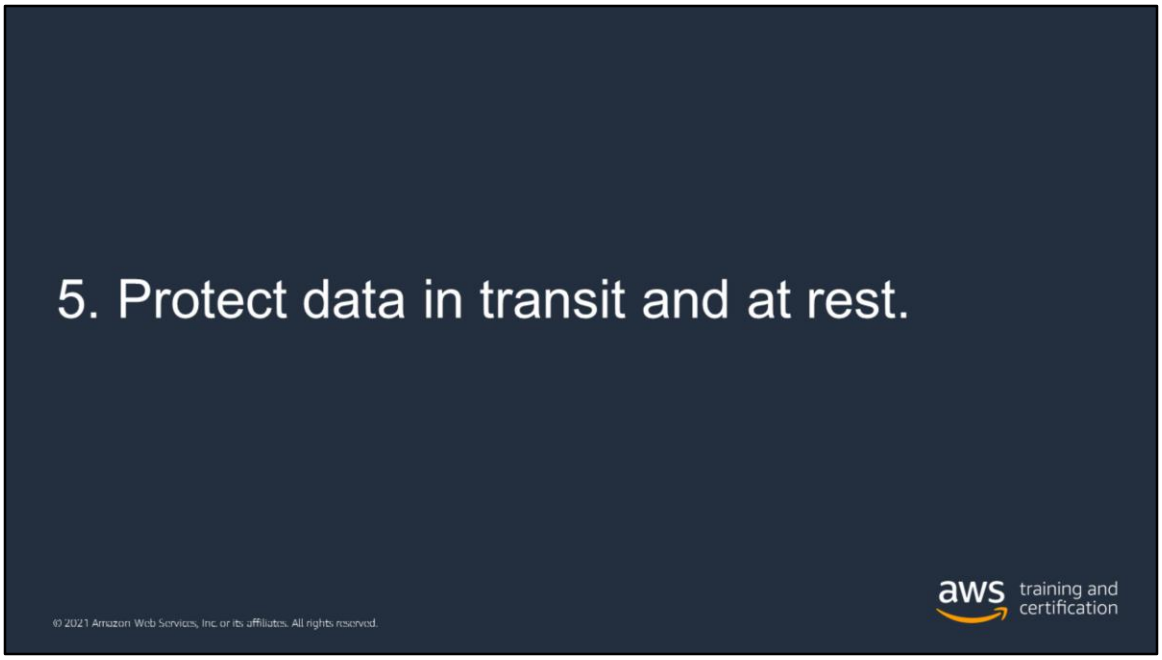
- Proactive controls enforced by code
- Continuous evidence-based auditing

Continuous detective controls:

- CloudWatch Logs and Alarms
- Amazon Inspector for Amazon EC2
- Amazon Macie for Amazon S3
- AWS Trusted Advisor
- AWS Config rules
- GuardDuty
- Cloud Conformity
- …and many more!

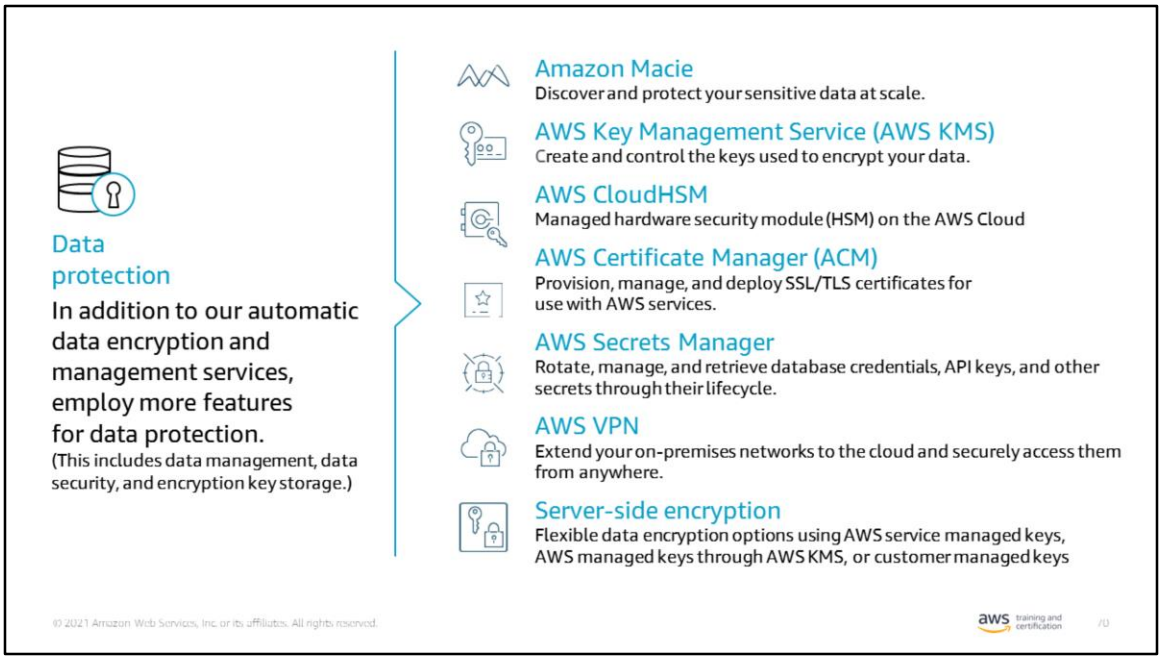
aws training and certification 68

Look at how easy audits become when you are running a continuous compliance mechanism such as config. Config provides a record of not only how long you have remained within a compliant state, but exactly when you deviated and what happened. Through immediate access to evidence, audit situations become easier to manage as issues are addressed through a combination of near real-time notification and appropriate remediation.



Protect data in transit and at rest.

Classify your data into sensitivity levels and use mechanisms such as encryption where appropriate.



Data classification provides a way to categorize organizational data based on levels of sensitivity. And encryption protects data by rendering it unintelligible to unauthorized access. These tools and techniques are important because they support objectives such as preventing financial loss or complying with regulatory obligations.

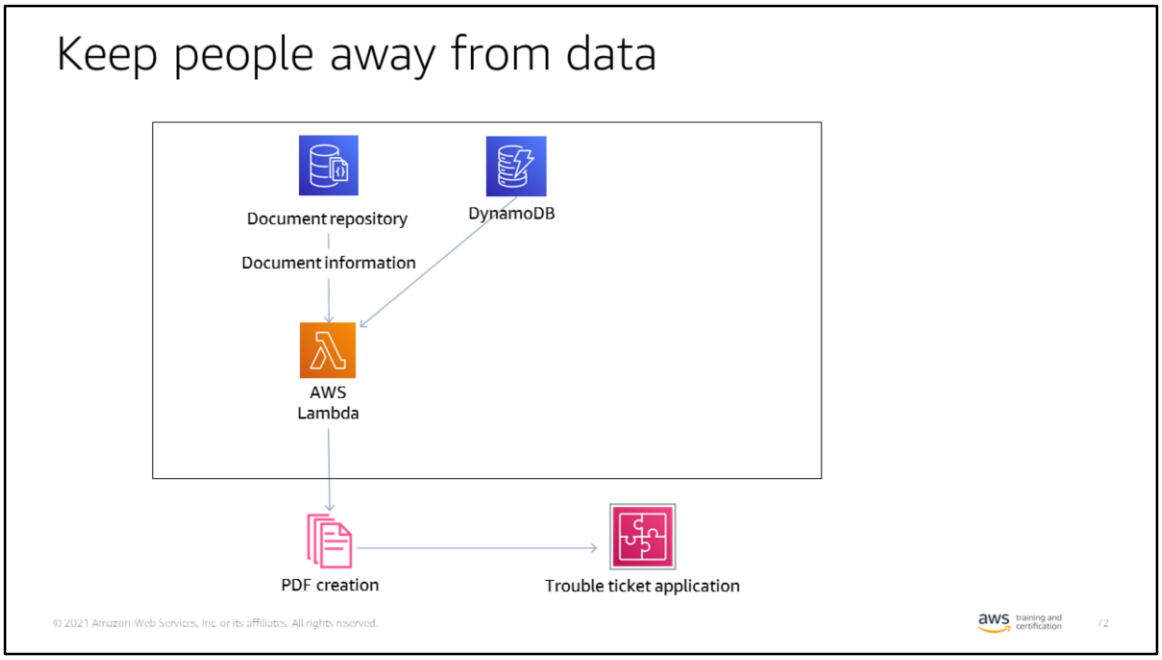
In AWS, the following practices facilitate protection of data:
- As an AWS customer, you maintain full control over your data.
- AWS makes it easier for you to encrypt your data and manage keys, including regular key rotation, which can be automated by AWS or maintained by you.

Amazon Macie is a fully managed data security and privacy service that helps you discover and protect your sensitive data. Macie applies machine learning and pattern matching techniques to the buckets you select to identify. And it alerts you to sensitive data such as personally identifiable information (PII).

For more information, see How do you classify your data? - AWS Well-Architected Framework

Keep people away from data.

Use mechanisms and tools to reduce or eliminate the need for direct access or manual processing of data. This reduces the risk of mishandling or modification and human error when handling sensitive data.

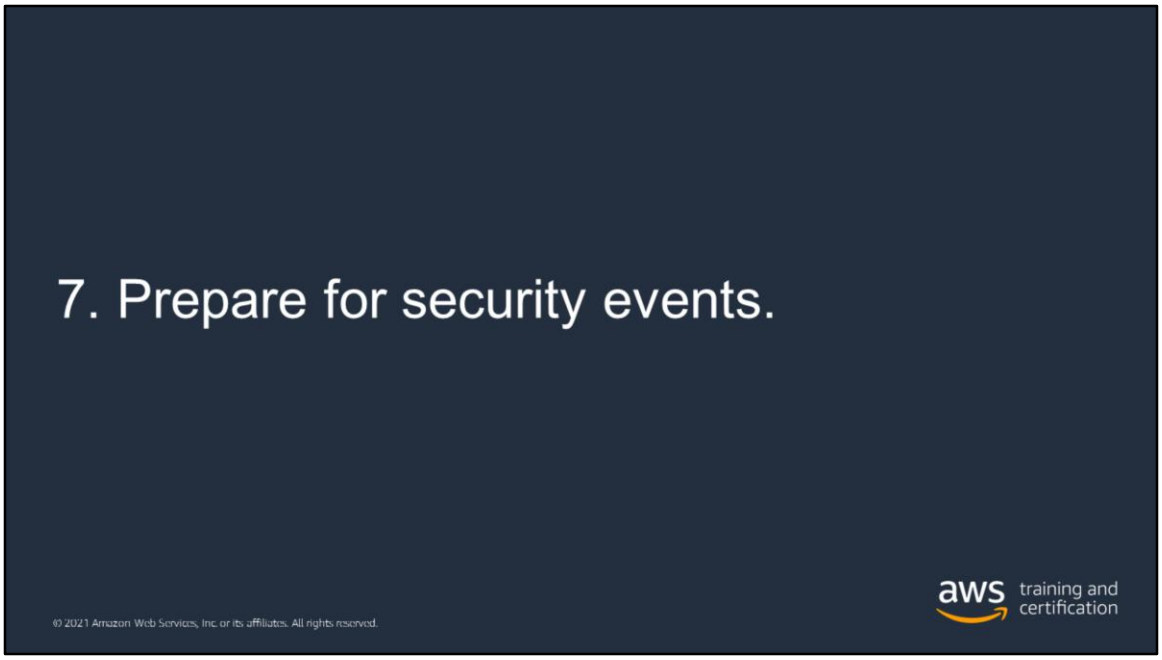


Keeping people away from data reduces the chance of corruption of your original data source through inadvertent changes. This can be achieved through several methods.

Only Lambda has permissions to access sets of data (within the document and DynamoDB repos) to extract data and create PDFs.

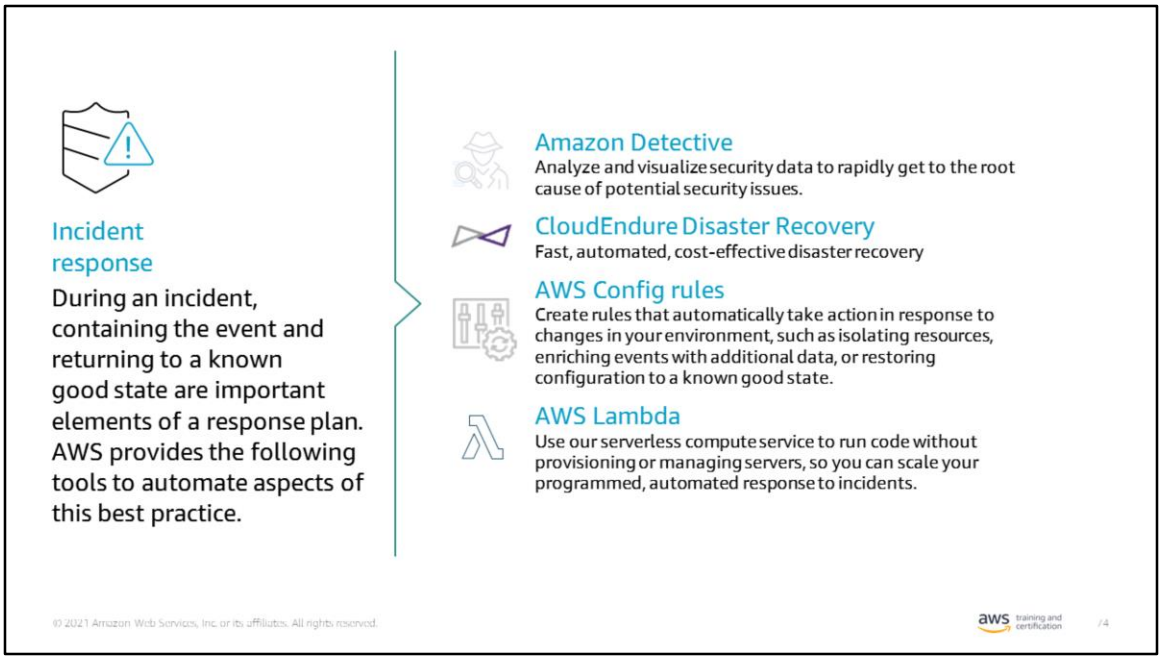
The message of *keep people away from data* in this case refers to people not needing the ability to access the data source directly. But instead, Lambda can be launched (with specific permissions) to create human-readable PDFs with information needed.

This method is used instead of humans manually accessing logs within these database repos.



Prepare for security events.

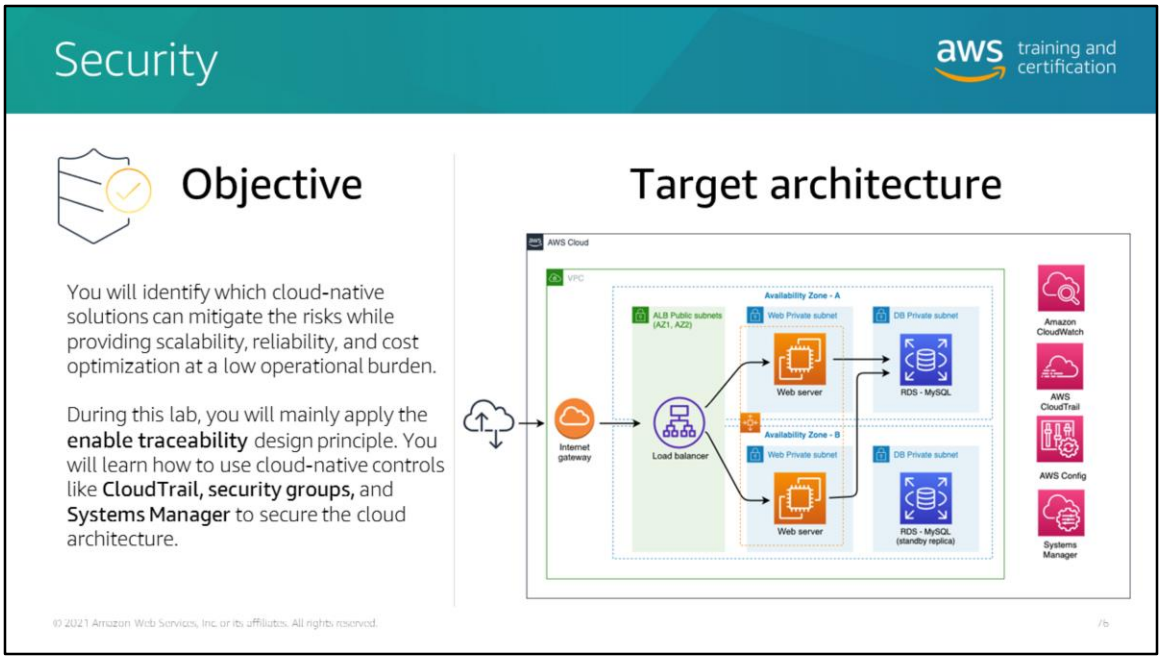
Prepare for an incident by having incident management and investigation policy and processes that align to your organizational requirements. Run incident response simulations and use tools with automation to increase your speed for detection, investigation, and recovery.



Preparing for security events is aligned to incident response. More accurately, you should be well prepared to deal with incidents before they occur. Through the use of game days, your can play out common security scenarios (data spill through misconfiguration or simply ensuring that when configurations are changed that you are aware of the change and can react accordingly).

Question: If someone changed your security group settings on a Friday night, how long would it take you to notice? And what would be needed to correct that change and remediate it accordingly?
https://aws.amazon.com/blogs/security/how-to-auto-remediate-internet-accessible-ports-with-aws-config-and-aws-system-manager/

Also try to Introduce Amazon Detective. Detective can analyze trillions of events from multiple data sources such as VPC Flow Logs, CloudTrail, and GuardDuty. And it automatically creates a unified, interactive view of your resources and users and the interactions between them over time, to quickly determine the root cause.



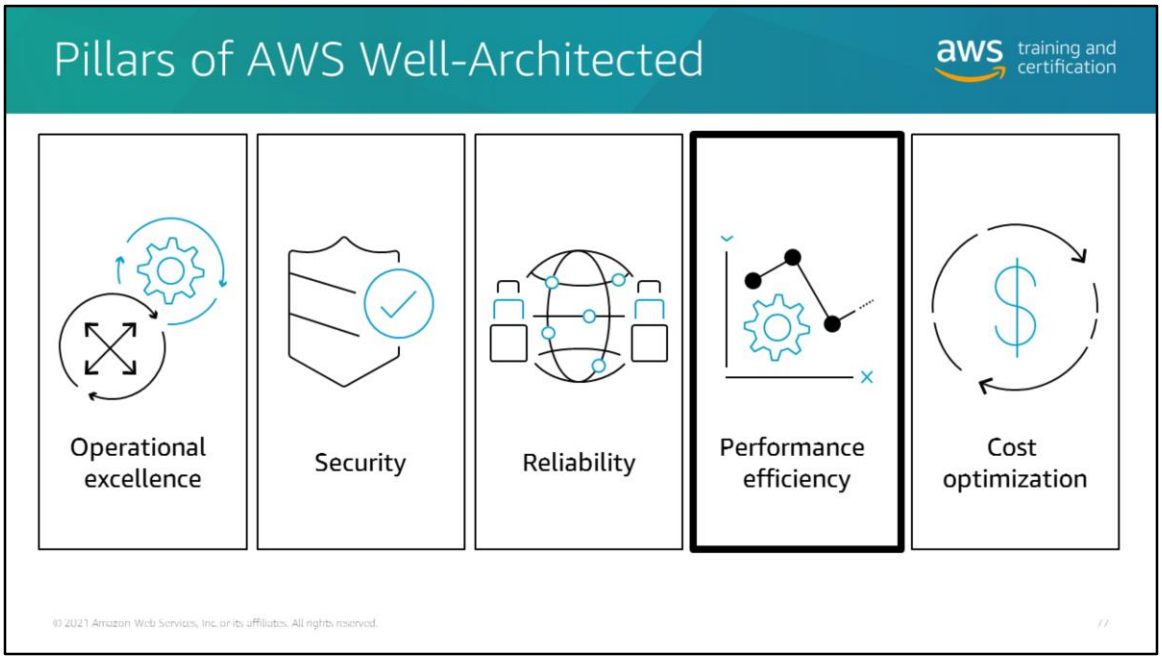
Security is our top priority, and it is also top priority for our AnyCompany customer.

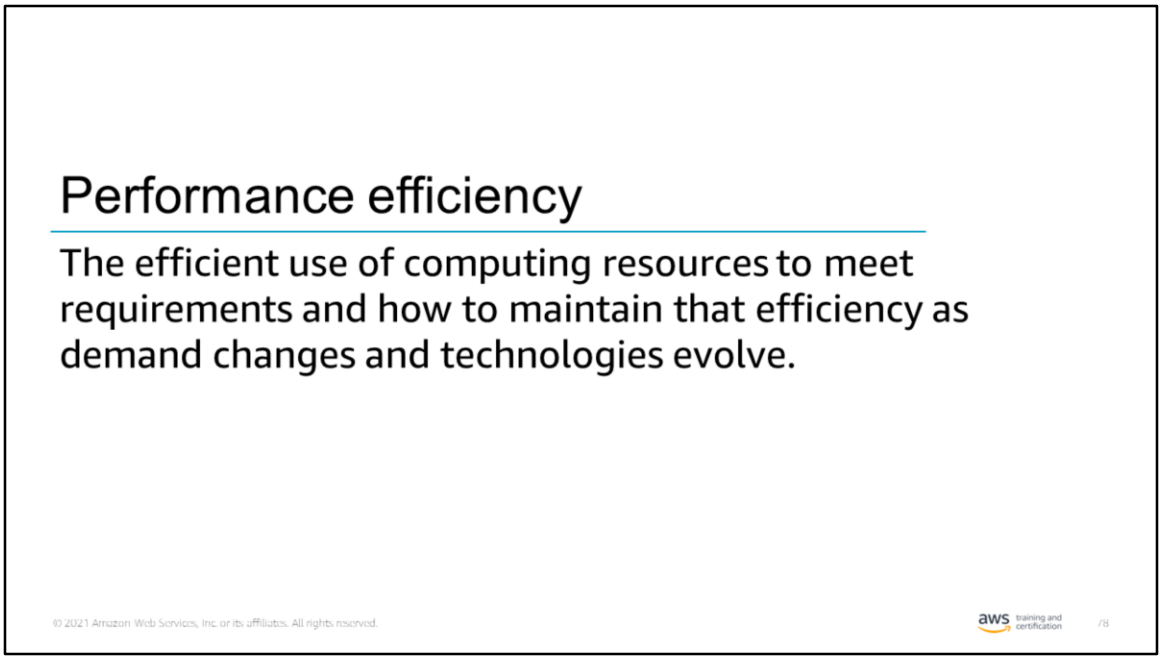
According to the Well-Architected Framework Review , implementing some cloud-native controls to prevent and detect security issues.

Estimated duration: 1 hour

Wrap-up after lab completion:

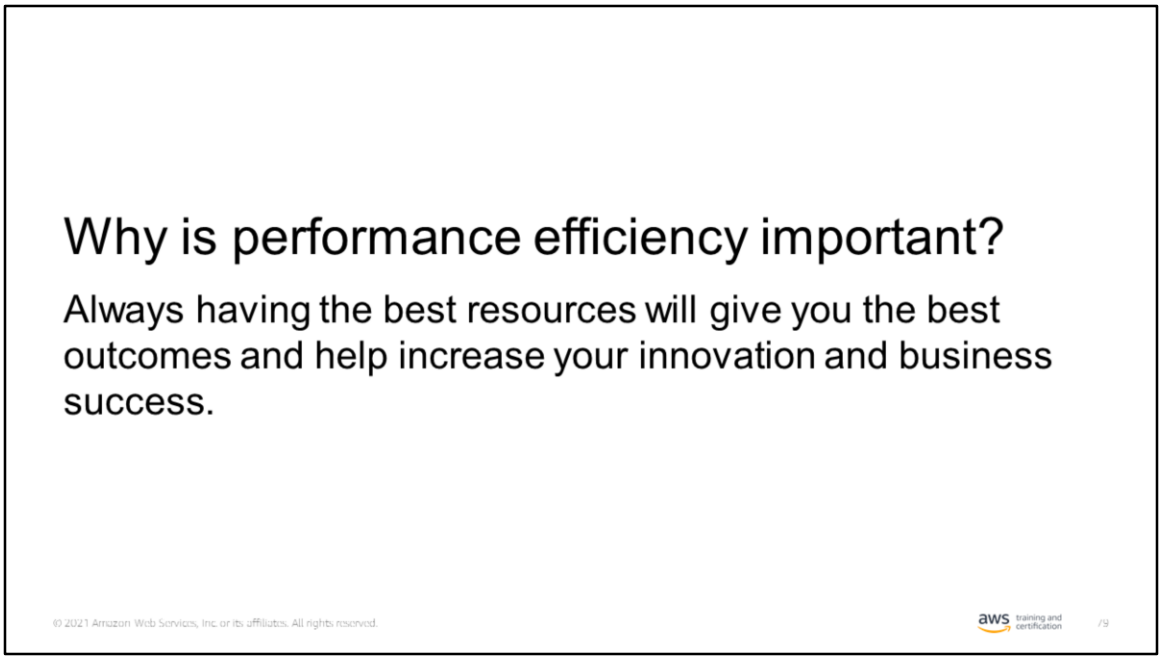
You have successfully set up this AWS environment for strong logging with CloudTrail and AWS Config, granular communication with security groups and network ACLs, and intelligent threat detection with GuardDuty. And you have configured your machines to have safe administrative access without requiring access from the public internet.





Performance efficiency is about making sure the resources that we are using meet both business and technical requirements. And they help us maintain that as demand changes and as our technologies evolve.

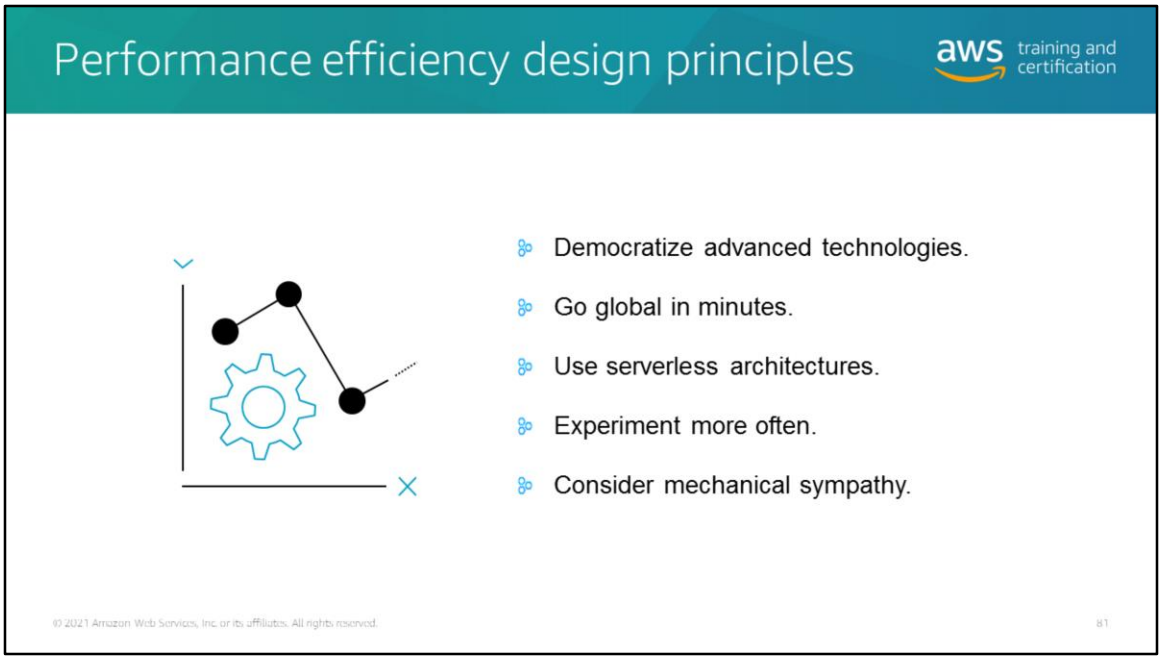
When we are designing, building, or operating workloads, we need to ensure that the decisions we make give us the flexibility to maintain that efficiency. And we are able to adapt to changes in demand, and take advantage of new technologies, features, and services when they become available.



Your systems will run better, with fewer operational issues and with less underlying technology to burden your company.

Free you up to focus on your business and what makes you great.

Remove the heavy lifting, allow you teams to focus on what makes them great and unique.



As we did before, let's think about the kinds of **constraints we had in a traditional environment** when thinking about performance efficiency:

- We tended to use the **same tech** for everything. **When the only tool you had is a hammer, every problem looks like a nail**. Generally, this is why you saw so many **RDBMS.**
- We stayed **local** because **global** was **too hard and too expensive.** Even the **thought** of **negotiating a contract** with a **supplier in a different country**, with a different **legal framework and language,** was enough to **stop most conversations** here.
- We used **many servers** that **did one thing.** And we had to have **people to manage all those** servers.
- It's **hard to get the resources for experimentation**. It takes a lot of **time** to set up, and it's not very common.
- We tended to **force technologies** to do what we need and then **hope** we could get the **performance we needed.**

In the cloud constraints have been removed, that allows us to adopt these design principles to build and operate cloud native architectures -

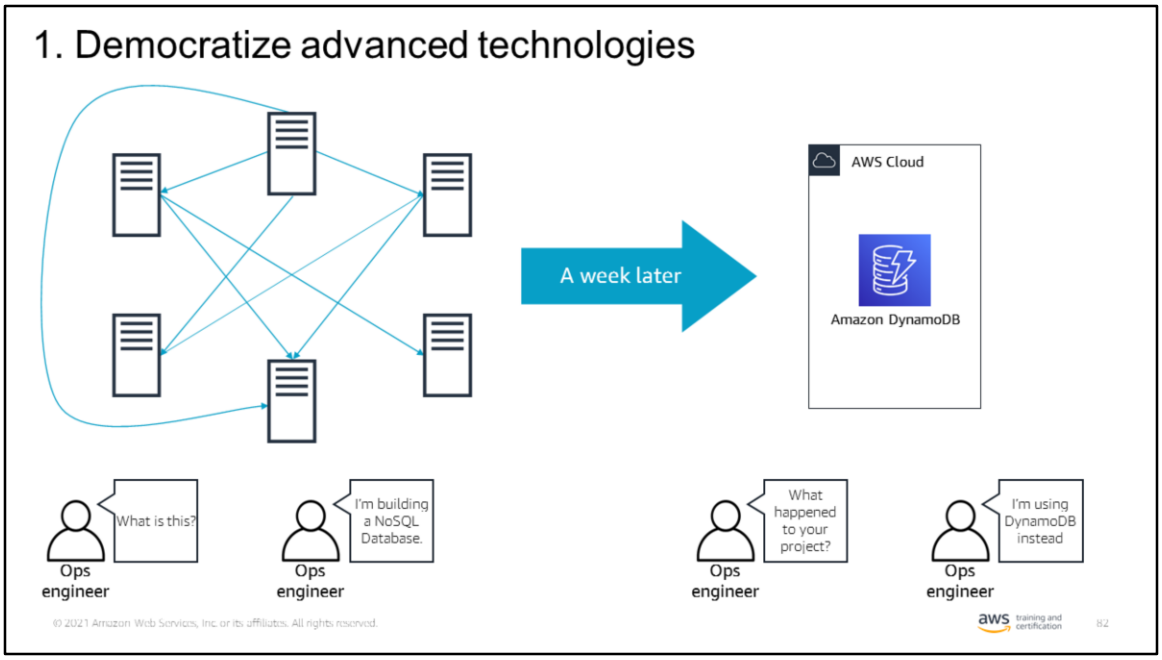
skills such as machine learning and media transcoding are not evenly distributed across technologists, so having AWS set up and configure those services for you, it makes adoption easier.

deploying to global locations is a click of a button and not a legal process,

we can create solutions that are fully managed so we can focus on the code that add values

and experimentation is something we can do continuously

and we have a bigger toolbox of techniques, and select the one that works best for what we are trying to do. For example, if you have relational information then you would use a relational database while if you needed internet scale lookups you would use a No SQL solution such as DynamoDB.

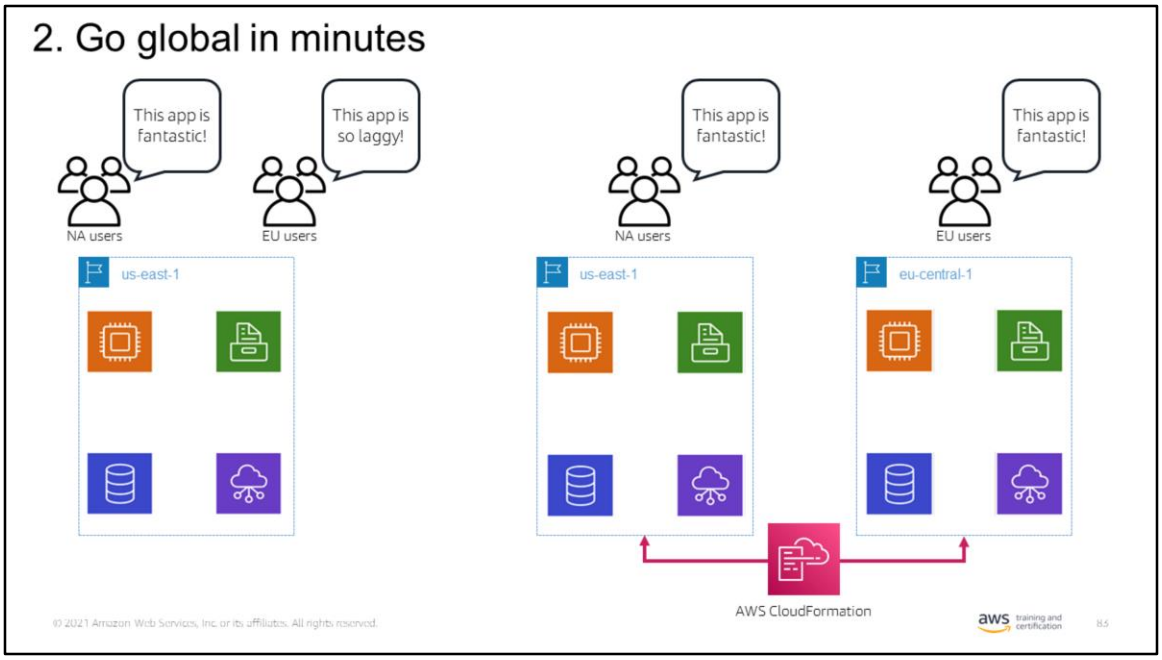


Make advanced technology implementation easier for your team. Delegate those complex tasks to your cloud vendor. Don't get your IT team to host a new technology; consider consuming it as a service.

We tended to use the **same tech** for everything. **When the only tool you had is a hammer, every problem looks like a nail**. Generally, this is why you saw so many **RDBMS.**

Think how long it would take to build, from scratch, a petabyte-scale, multi-Region, multi-master NoSQL database that offered the performance you needed for your project.

Think how much time your engineers could devote to actually building your project and adding value to it, rather than doing all of that heavy lifting.



Deploying into multiple Regions helps get your workload closer to your global audience, and can help reduce costs.

We stayed **local** because **global** was **too hard and too expensive.** Even the **thought** of **negotiating a contract** with a **supplier in a different country**, with a different **legal framework and language,** was enough to **stop most conversations** here.

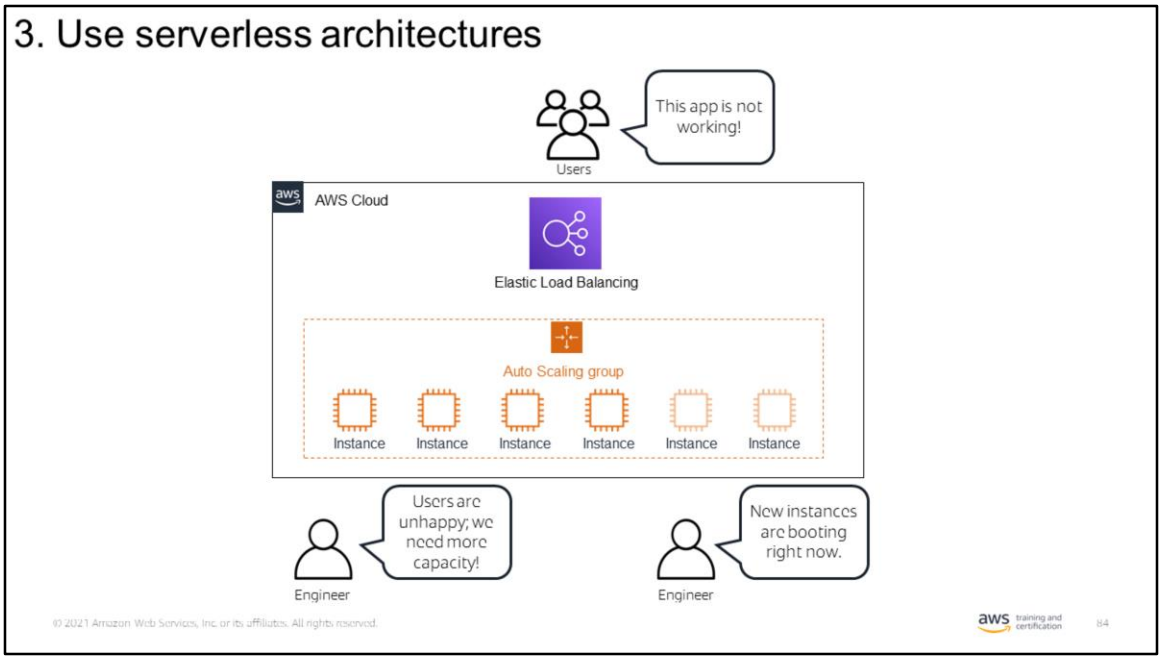
You have your app deployed into a single AWS Region (North America). Users who are close to that Region would get amazing performance. However, users who are geographically further away might get worse performance.

By using templating technology to deploy our infrastructure and application, such as CloudFormation, Terraform, Chef, and Puppet, we can quickly spin up resources in different global geographies with minimal overheads. This can offer better performance to our application users. But it also gives us the benefit of being able to select different features or functionality that may exist in different Regions.

Another example could be using CloudFront to scale to multiple areas without having to be in another Region. It might be one option for customers that have not

yet deployed in a new Region but have customers they need to serve.

Mention how Route53 can help direct NA and EU users to different Regions.



Don’t run and maintain physical servers; avoid using servers for traditional compute activities like patch management or website hosting.

We used **many servers** that **did one thing.** And we had to have **people to manage all those** servers.

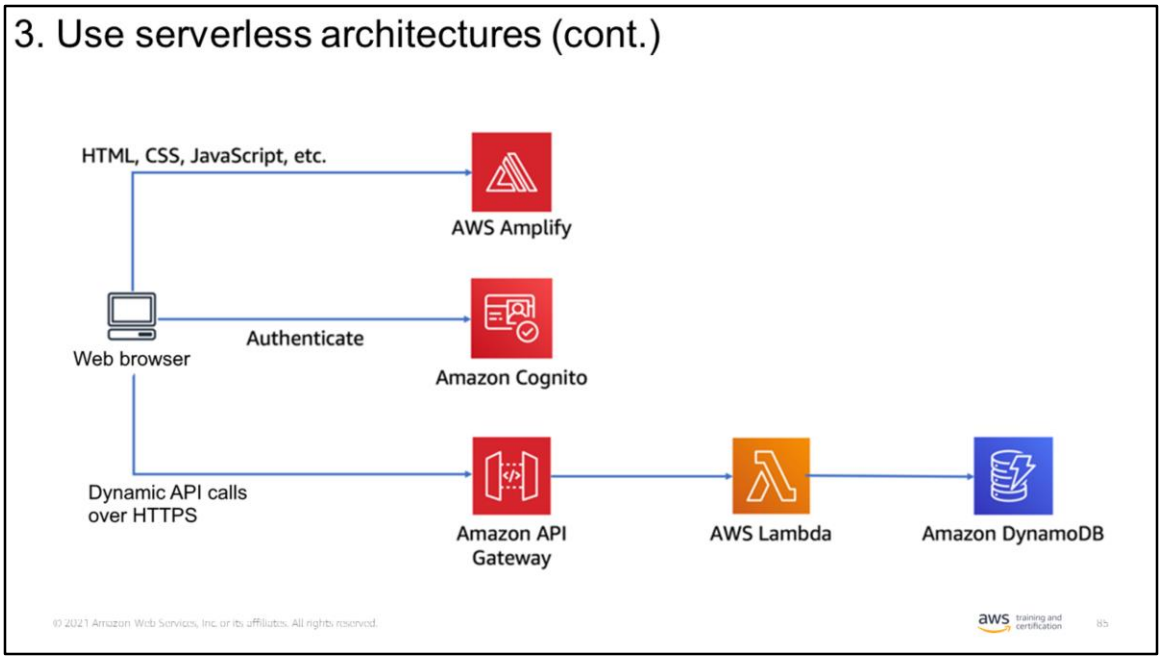
Serverless technology isn’t just Lambda functions. Any AWS service where you don’t manage servers is serverless.

Consider an application built on Amazon EC2 with a load balancer. Sudden increases of demand are still bound by scaling EC2 instances. And although this only takes a few seconds to start, your instances might take several minutes until they are serving requests.

Not only does this remove infrastructure that you don’t need to manage, it also makes scaling significantly easier. Because you often only need to configure settings like concurrency or throughput rather than CPU/memory/disk.

It also helps track usage against cost.

As AWS manages the scaling of these services, it can also bring cost benefits.

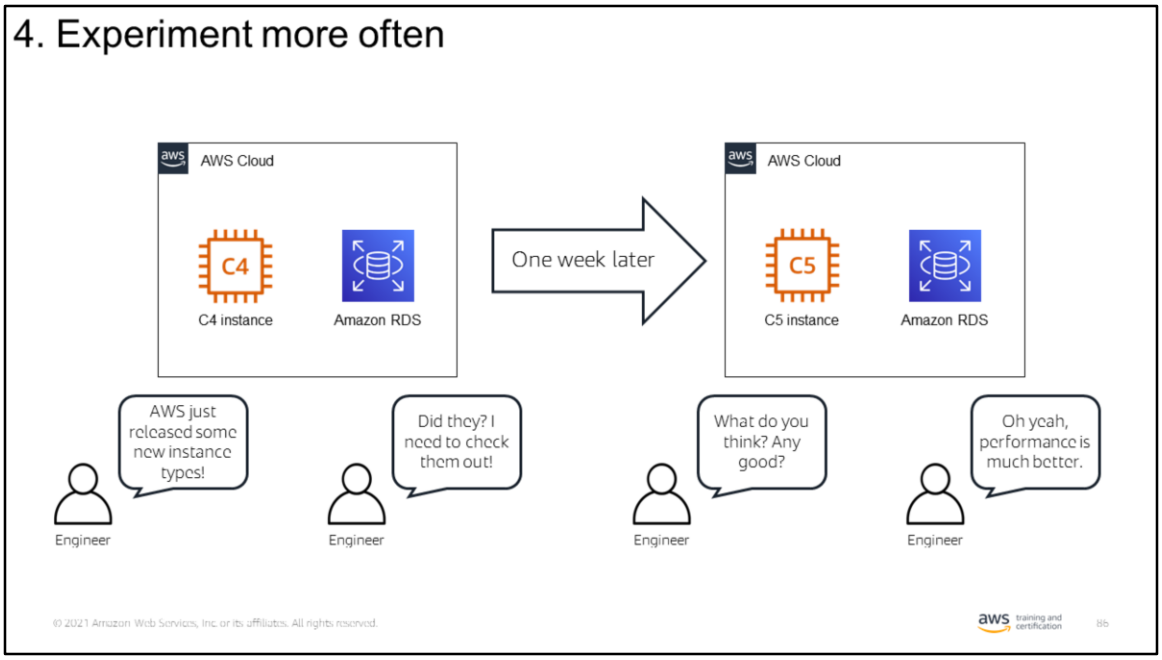


The application architecture uses AWS Lambda, Amazon API Gateway, DynamoDB, Amazon Cognito, and AWS Amplify. The Amplify console provides continuous deployment and hosting of the static web resources—including HTML, CSS, JavaScript, and image files—that are loaded in the user's browser. JavaScript run in the browser sends and receives data from a public backend API built using Lambda and API Gateway. Amazon Cognito provides user management and authentication functions to secure the backend API. DynamoDB provides a persistence layer where data can be stored by the API's Lambda function.

- **Static web hosting:** Amplify hosts static web resources—including HTML, CSS, JavaScript, and image files—that are loaded in the user's browser.
- **User management**: Cognito provides user management and authentication functions to secure the backend API.
- **Serverless backend:** DynamoDB provides a persistence layer where data can be stored by the API's Lambda function.
- **RESTful API:** JavaScript run in the browser sends and receives data from a public backend API built using Lambda and API Gateway.

https://aws.amazon.com/getting-started/hands-on/build-serverless-web-app-

lambda-apigateway-s3-dynamodb-cognito/



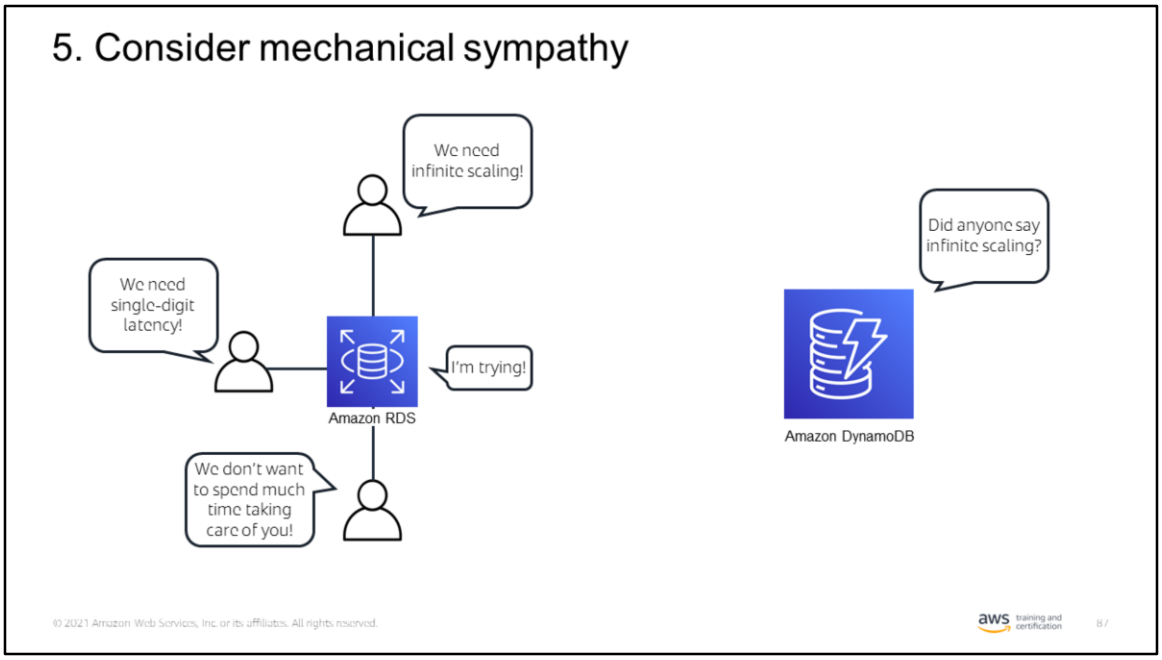
Virtually unlimited resources encourage comparisons of different configurations.

It's **hard to get the resources to perform experimentation**, it takes a lot of **time** to set up, and it's not very common.

Templating and using automation lets you quickly try different configurations on your workload and application. You can spin up resources, or a full copy of your production environment, in hours or minutes. You can test what you need to, get the results, analyze them, and see if it's a change you want to carry over.

You can experiment more quickly because barriers have been removed. You no longer need to procure physical hardware or equipment, and you can just spin up what you want to experiment with.

This can be as basic as a different-sized instance or type of storage, or trying totally different services (like running your code in Lambda rather than on an EC2 instance).



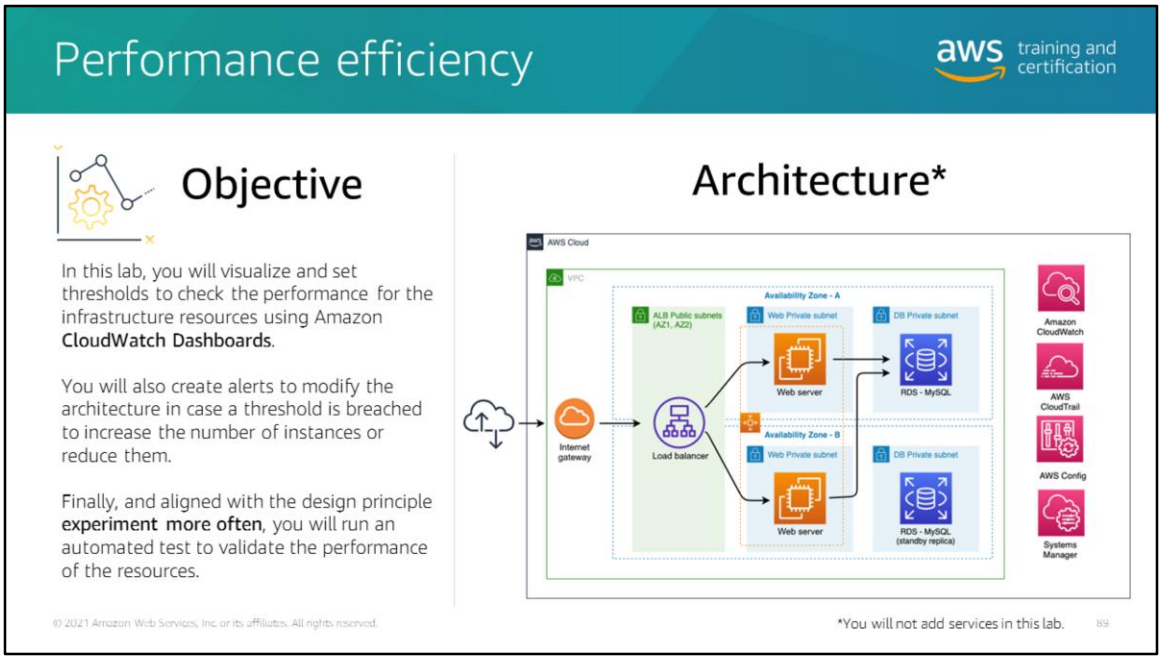
Mechanical sympathy is when you use a tool or system with an understanding of how it operates best. Align your technology approach to your overall business goals, not the other way around.

We tended to **force technologies** to do what we need and then **hope** we could get the **performance we needed.**

We often see lots of RDBMS out there. When your only tool was a hammer, everything looks like a nail.

You have access to a rich and varied array of services, with features and functionality to meet many different use cases across your workloads and applications. Using the right tool for the job will result in significantly improved performance, and often at a lower cost point than using a hammer.

Another example:
A better point would be to move to a purpose-built database. So for instance, it would be easier to explain going toward QLDB or Neptune.



The objective of this lab is to identify how AnyCompany can create and operate their infrastructure with the best performance while maintaining the alignment to the business and technical needs.

Estimated duration: 40 minutes

Wrap-up after lab completion:

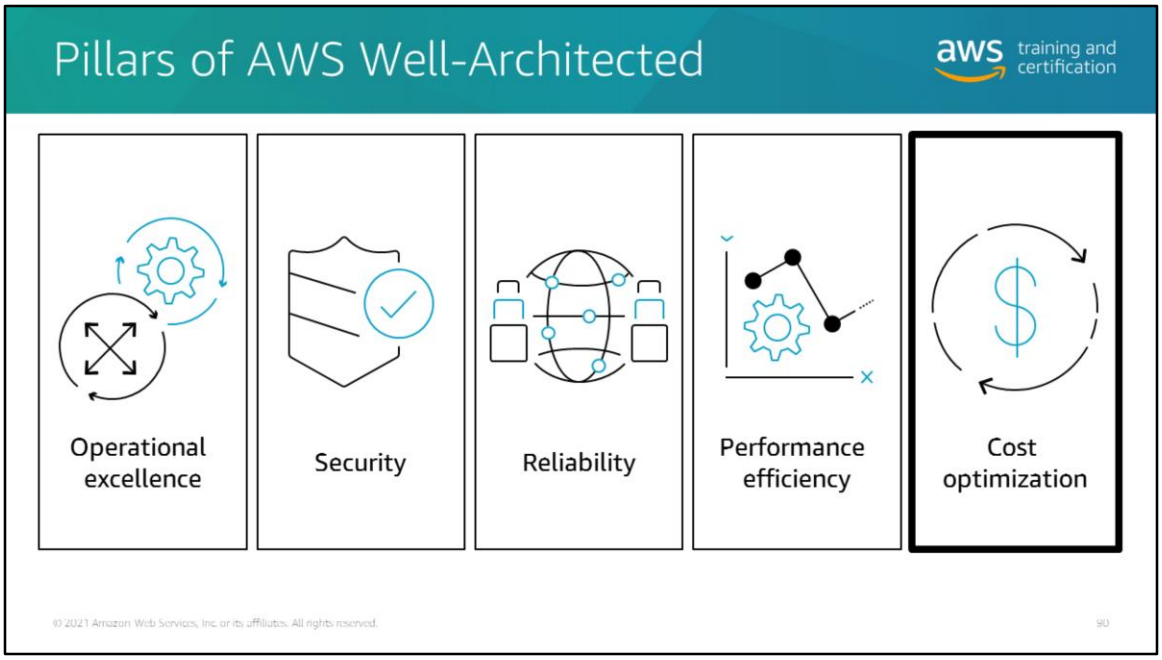
This lab has an open question:

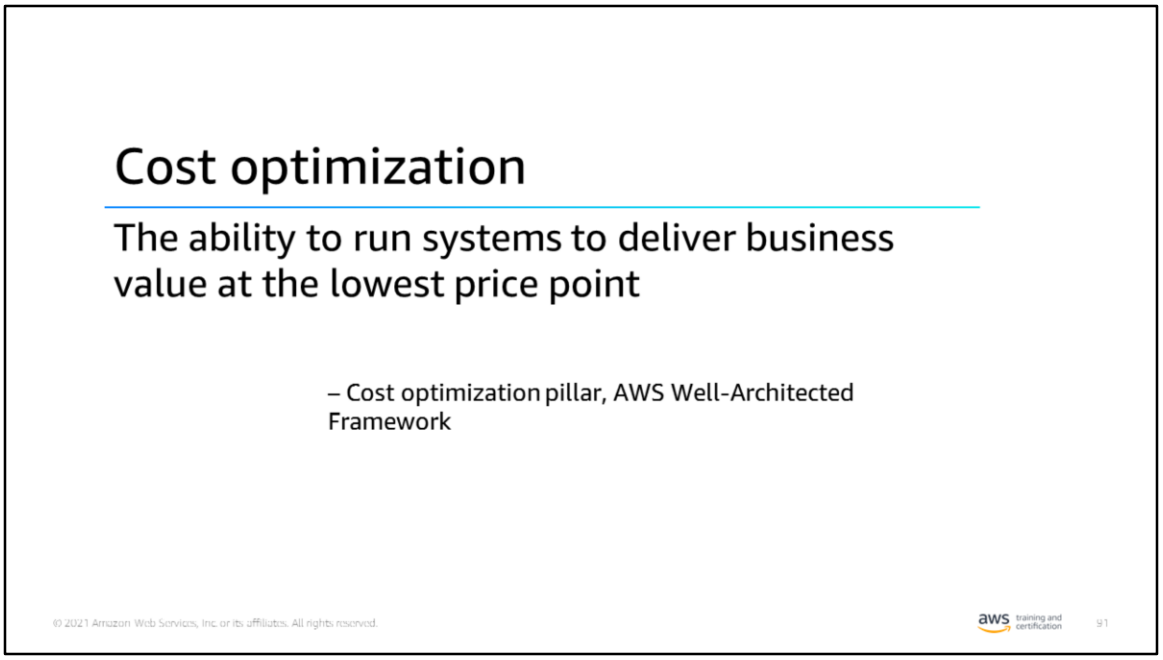
One of the more important aspects about monitoring performance and running experiments is that **you can use data to make fact-based decisions about your architecture**. In the case of AnyCompany and according to the insights that you got in this lab, **what would you recommend to the customer?** If the customer want to meet the demand in the near future, should the customer change the EC2 instance type? Should the customer increase the maximum limit configuration for the Amazon EC2 Auto Scaling group? Are there other activities to do before making decisions?

Final comments

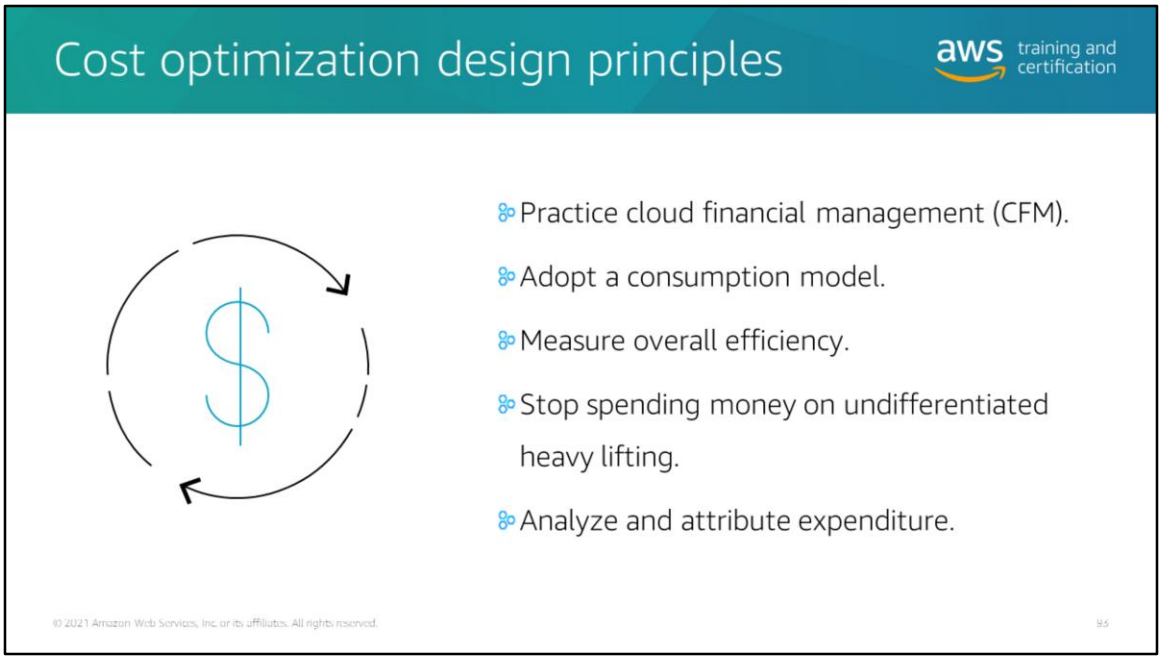
We have worked on the efficient use of computing resources to meet requirements, and how to maintain efficiency as demand changes and technologies evolve. From now on, you should keep reviewing your choices on a regular basis, ensures that you are taking advantage of the continually evolving of AWS Cloud.

When architecting workloads, there are finite options that you can choose from. However, over time, new technologies and approaches become available that could improve the performance of your workload. Fortunately, in the cloud it is much easier to experiment with new features and services.





Note the emphasis on delivering business value. Cost optimization is not just looking at your AWS bill. It is about measuring and understanding the value your resources and workloads are providing the business. A workload is not cost optimized if it has a very low bill but does not generate any value to the business. Technology exists to serve the needs of the business; cost optimization exists to do this as efficiently as possible.



Principles: Have these in the back of your mind at all times. Approach every problem with these in mind.

Consumption model: Don't run things; create and use things when you need them and destroy them.

Overall efficiency: Measure the workload output and the costs of that output. Do you need a static budget when you're always making money?

Data center: Focus on your core business, undifferentiated heavy lifting

Analyze and attribute: not possible before cloud, think of a network switch – did you allocate each port? Its power? Its heat? The cloud gives you the opportunity to drive efficiency and be lean like never before.

Managed/App services: similar to data center, keep pushing up the stack. Are you in the game of managing, patching operating systems? Databases?

Again, let's think about how we would approach cost optimization in a traditional

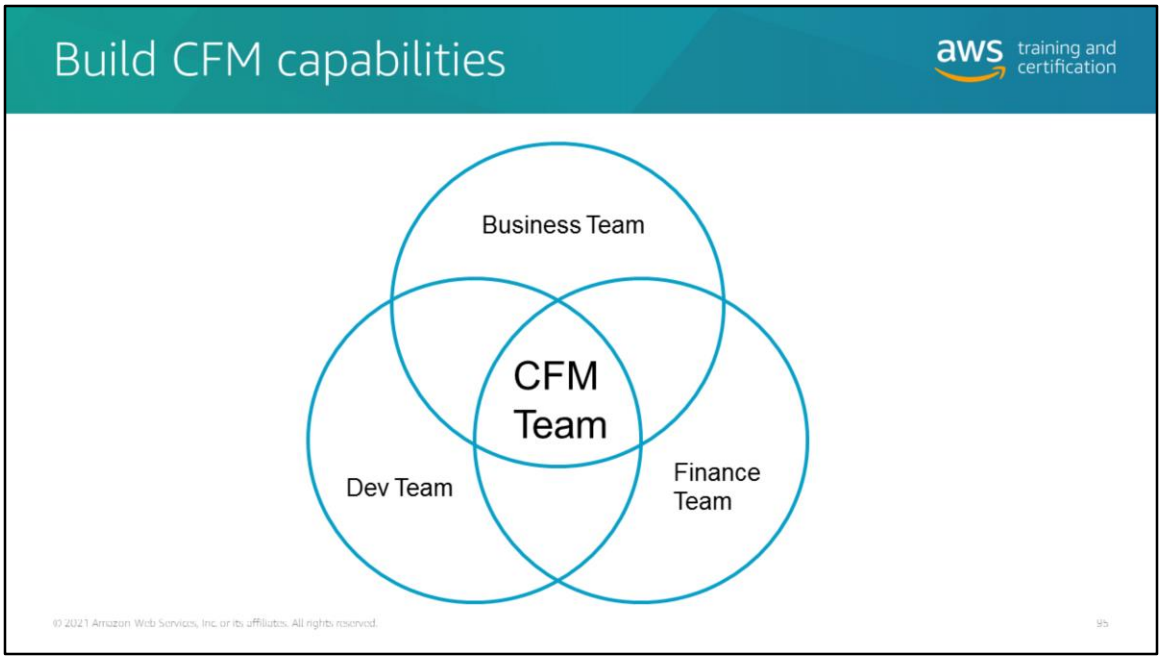
environment:

- You had to **invest capital expenditure (capex) up front** for new infrastructure before you needed it.
- Most companies are **not large enough** to benefit from **economies of scale**.
- You spent time and money on the **undifferentiated heavy lifting of building**, maintaining, stacking, and racking data centers.
- Often there are only **centralized costs that couldn't be attributed back** to others. So no one is incentivized to review costs and you had orphan systems.
- You **purchased and ran servers** to provide services, often with **low utilization** because they were hard to share.

In the cloud, constraints have been removed. That allows us to adopt these design principles to build and operate cloud-native architectures:

- **You pay** for computing resource **as you consume them**.
- AWS can use its **economies of scale** to drive down infrastructure costs and pass savings on to customers.
- We do the **heavy lifting** of managing the **physical bits**, so you can focus on the **value adding bytes.**
- You can **attribute costs back** to **business units** and **product owners**, so they can drive these down.
- Use **managed services** that have a lower cost and eliminate the time and cost of managing servers.

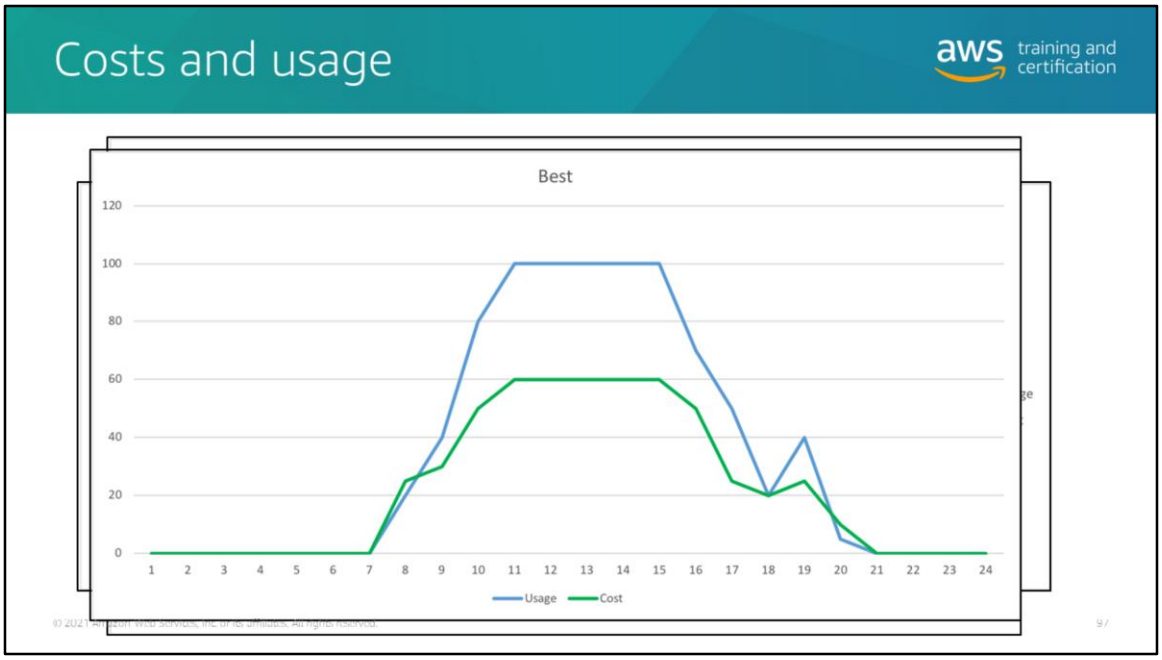
To achieve financial success and accelerate business value realization in the cloud, you need to invest in CFM and cost optimization. Your organization must dedicate time and resources to build capability in this new domain of technology and usage management. Similar to your security or operational excellence capabilities, you need to build capability through knowledge building, programs, resources, and processes to become a cost-efficient organization.



Create a team that is responsible for establishing and maintaining cost awareness across your organization. The team requires people from finance, technology, and business roles across the organization. Agree on a set of financial objectives and align goals accordingly.

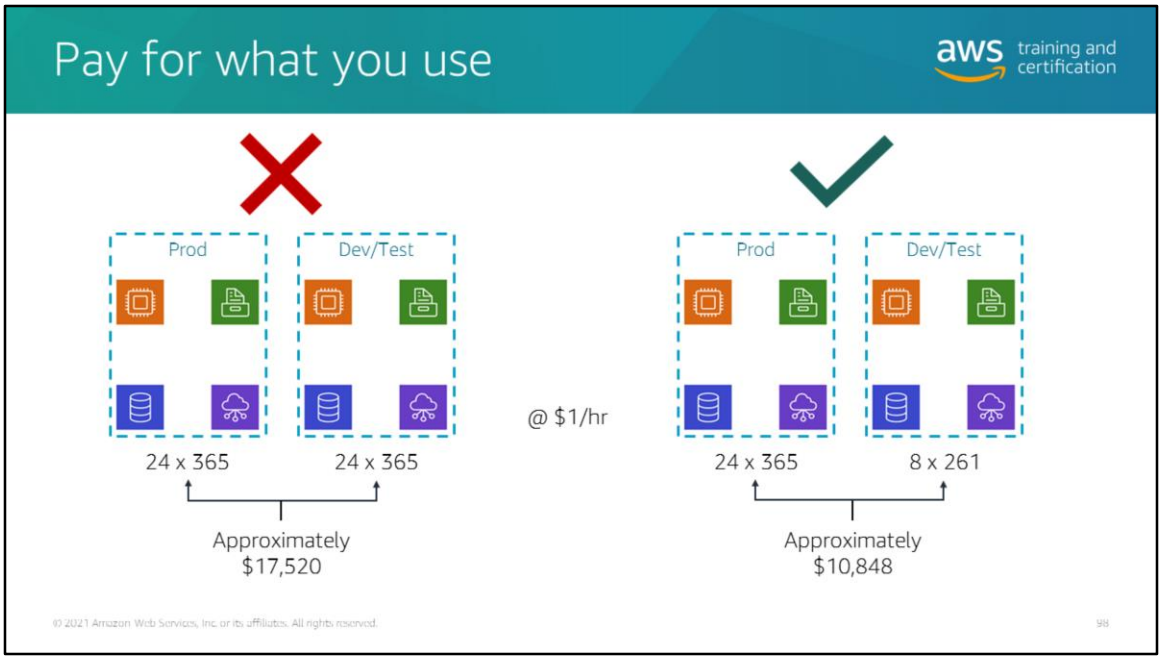
This team will help implement cost awareness into new or existing processes that impact usage, and will use existing processes for cost awareness.
Implement cost awareness into employee training.

Pay only for the computing resources that you require and increase or decrease usage depending on business requirements, not by using elaborate forecasting.



Cost should always be associated with usage and never be flat. Flat costs could result in periods of underutilization, which are missed opportunities for significant savings.

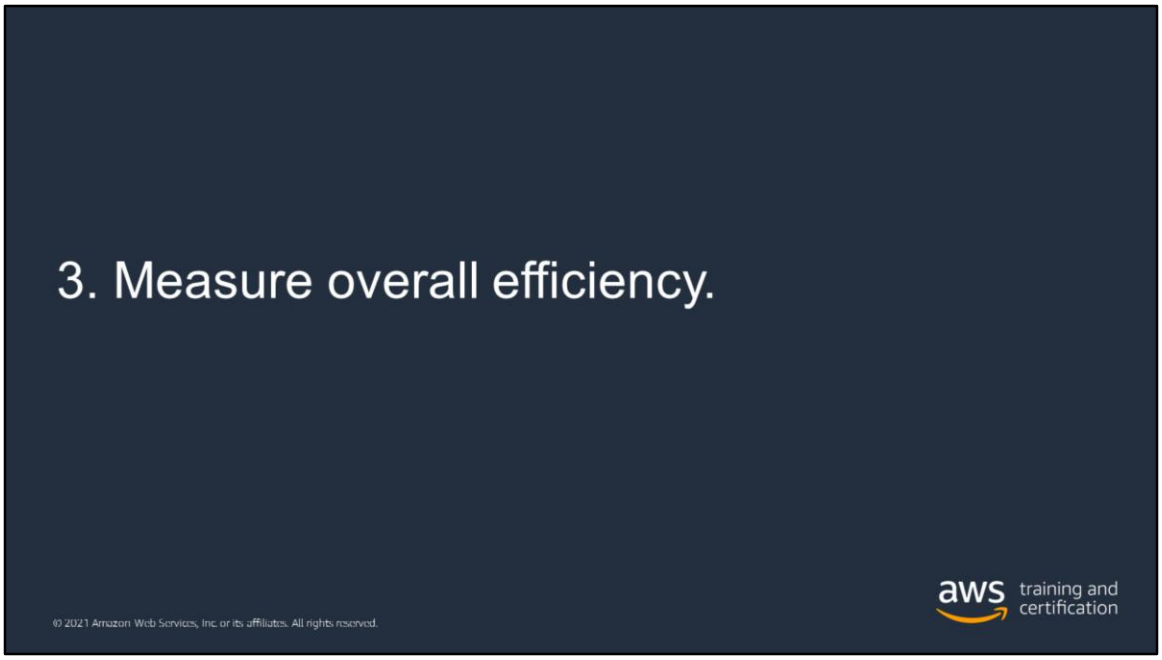
The cloud's pay-as-you go model, coupled with the speed of provisioning resources, makes it significantly easier to optimize workloads where costs are directly influenced by usage.



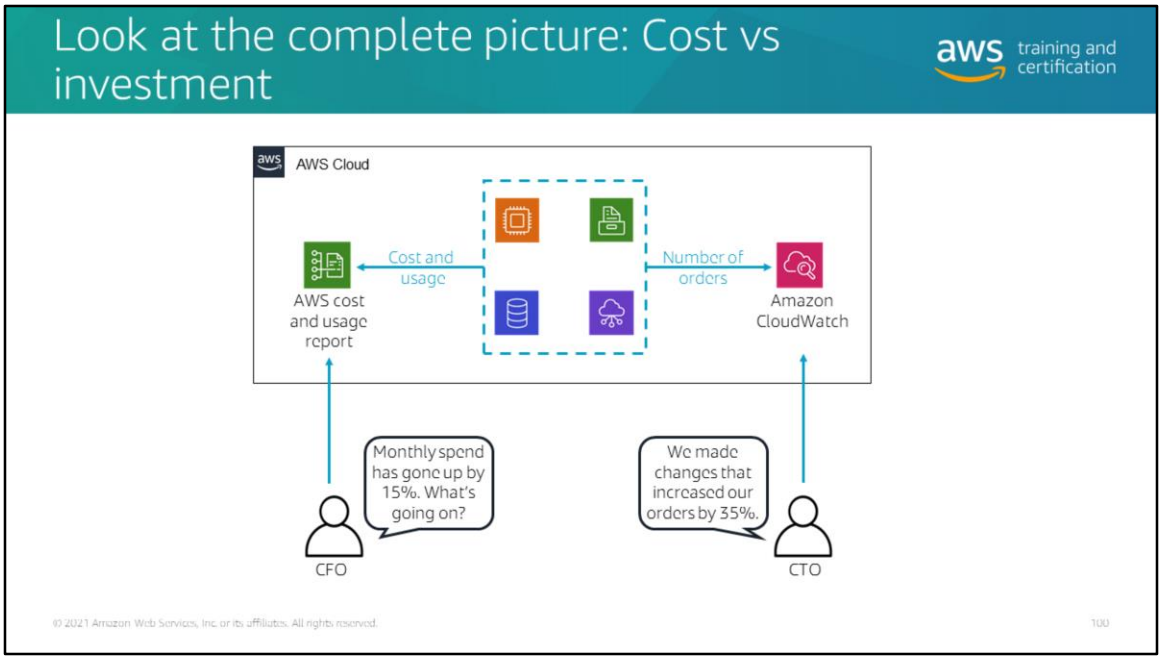
Shut down/suspend/terminate resources that are not in use. For example, production environments are usually required to run 24 hours a day, all year along. However, dev/test environments are usually only used during working/business hours and will sit idle outside of these times or over the weekends.

Use automation to assist with this.

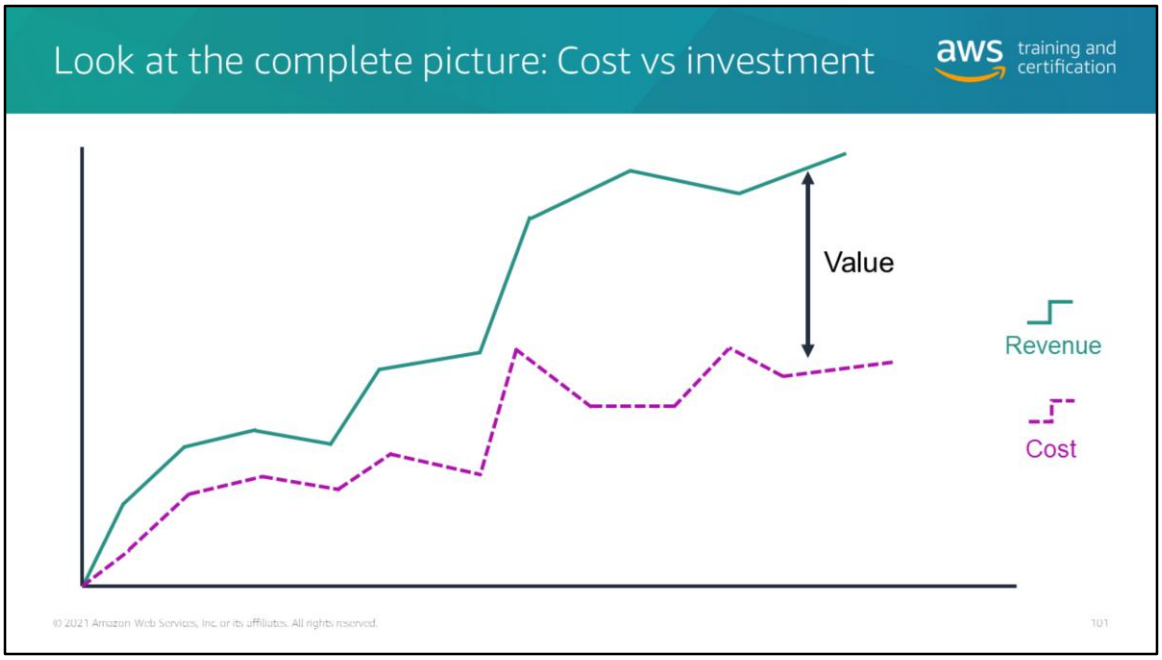
Massive savings: In this scenario, shutting down/suspending dev/test environments outside of usual work hours results in a lot of savings. Apply this to all resources that are not always in use (for example, instances, load balancers, AMIs/snapshots, and elastic IP addresses).



Measure the business output of the workload and the costs associated with delivering it. Use this measure to know the gains you make from increasing output and reducing costs. Identify business key performance indicators (KPIs) that are influenced by variations in cost. And implement mechanisms to track these KPIs to get a complete picture of the value the workload is providing to the business.

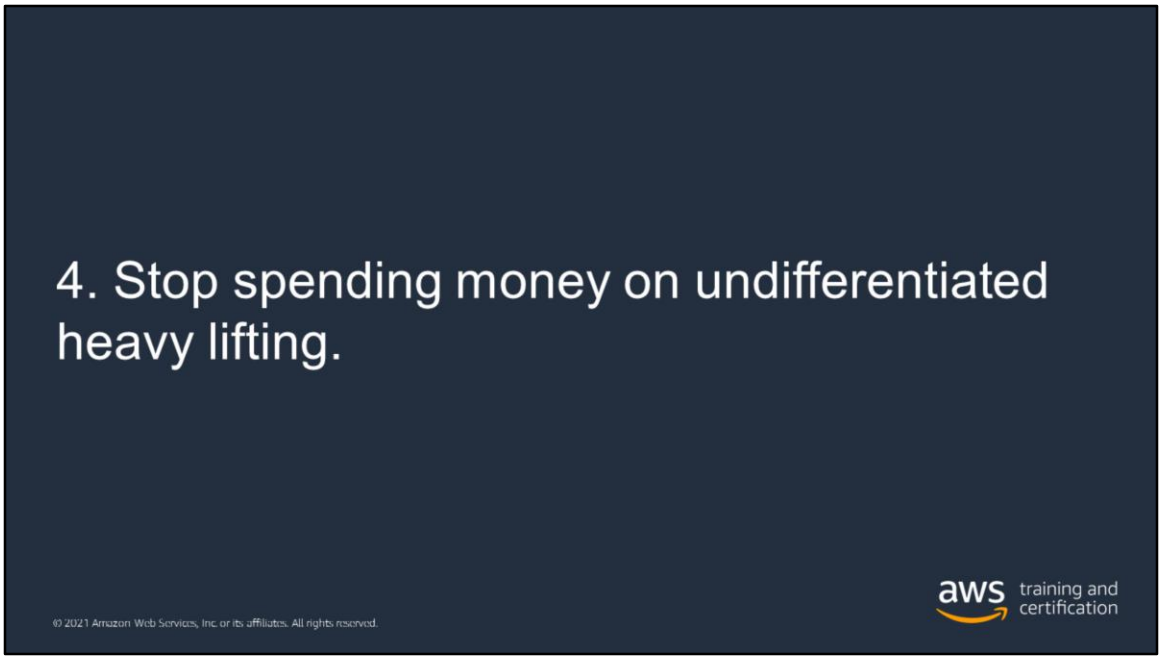


It's important to understand the reason for an increase in cost as well as the result it had on the business. Don't just think of increase or decrease in cost; think how the business is affected by this. Understand what is a cost and what is an investment. A CFO might be concerned with the increase in their monthly expenses. However, if that results in a positive impact to the business, it is no longer cost to the business, but an investment. This has to be measured with business KPIs such as increase in orders/revenue, customer usage/demand, improved customer satisfaction, etc.

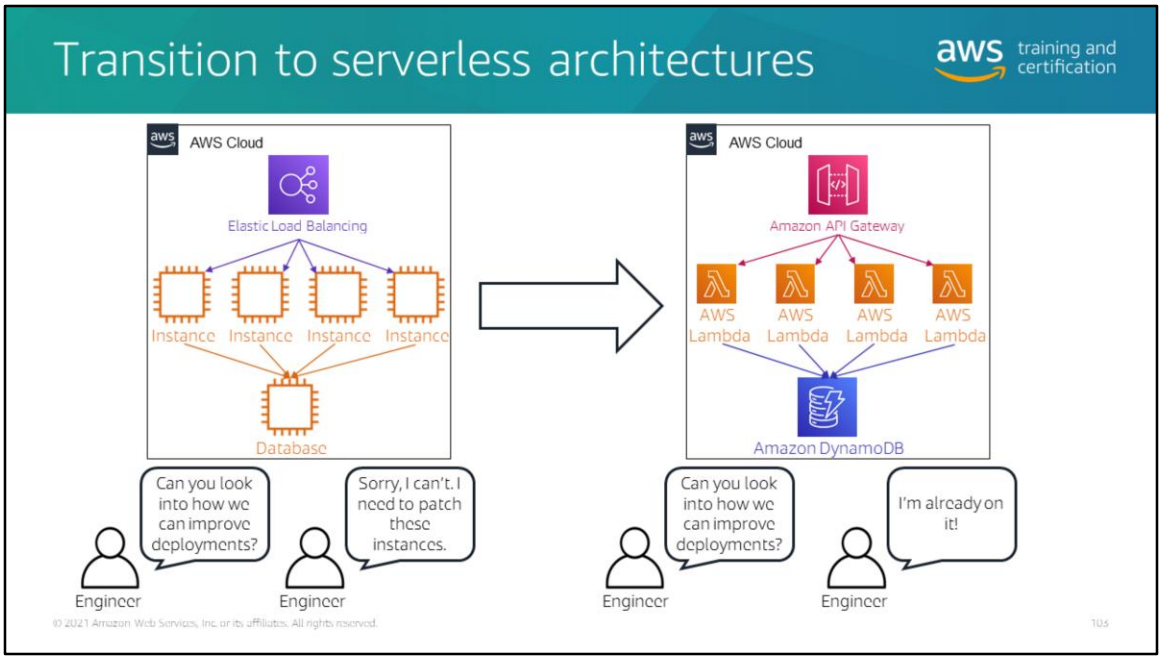


If you have a cost visualization dashboard, this is easy to understand.

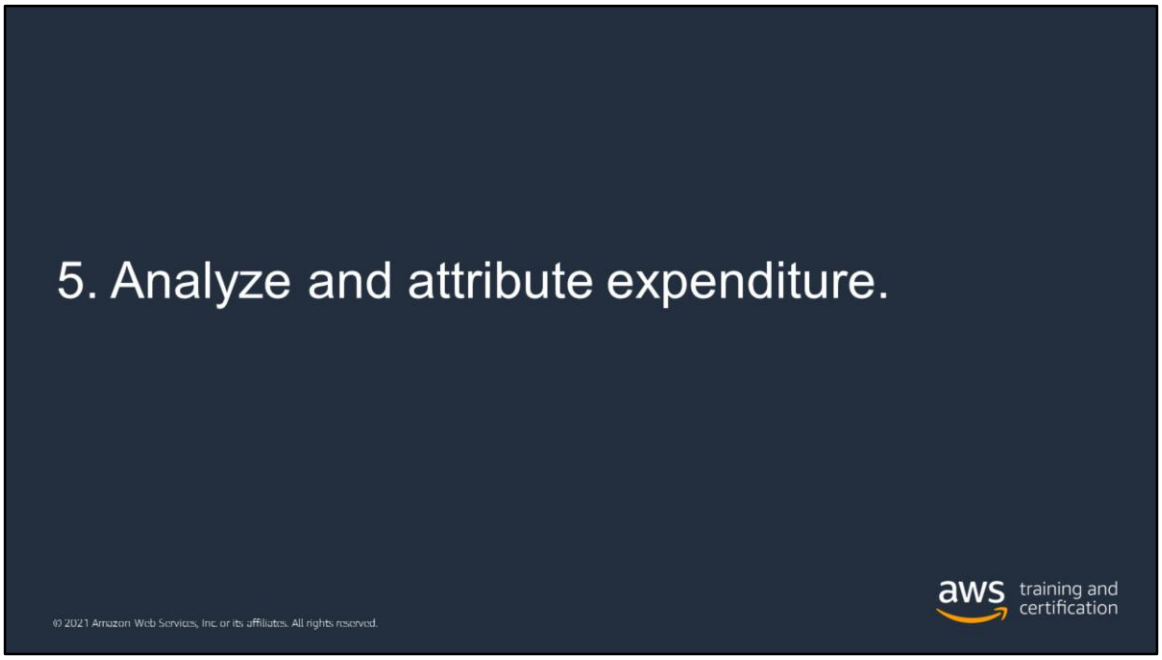
Notice increase in cost after a change to the workload. This is a cause for concern for most CTOs and finance teams. However, if we overlay this with changes in revenue, it is apparent that there is a significant increase in revenue as well. This is the return on the investment of making the change to the workload.



AWS does the heavy lifting of data center operations like racking, stacking, and powering servers. It also removes the operational burden of managing operating systems and applications with managed services. This allows you to focus on your customers and business projects rather than on IT infrastructure.

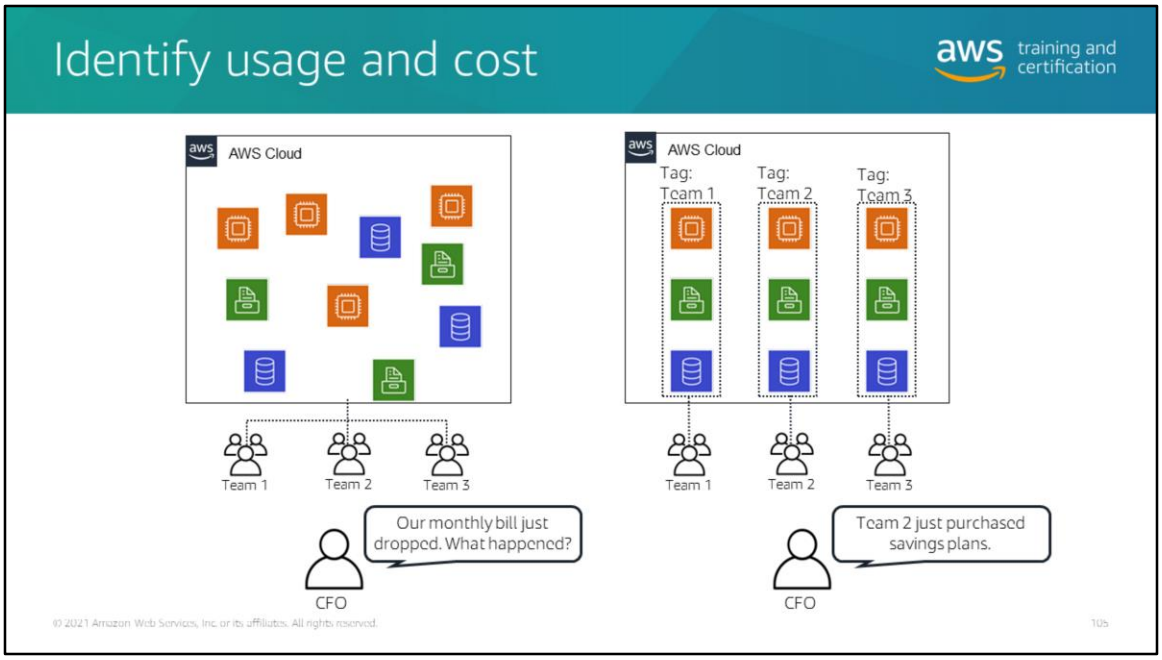


Using managed services will greatly reduce the operational burden on your teams. This allows them to focus on adding value to the business instead of being bogged down by routine mundane tasks. For example, it is certainly possible to build your own database cluster on EC2 instances or virtual machines. The effort involved in this as well as managing the cluster over a period of time is effort that could have been invested elsewhere—for example, reevaluating data formats and database engine choice, addressing slow query issues, or optimizing schemas. In this situation, using a service like Amazon DynamoDB will let you focus on improving the workload and adding value to the business instead of spending time on ensuring logs are being rotated, instances are patched, etc.



The cloud makes it easier to accurately identify the usage and cost of systems, which then allows transparent attribution of IT costs to individual workload owners. This helps measure return on investment (ROI) and gives workload owners an opportunity to optimize their resources and reduce costs.

Always consider how a cost or expense originated and where it originated from. Think about your credit card bill. If there was a massive purchase on your bill, you immediately look at the statement and look at individual line items. You want to understand where the expense came from so you can address it. You should have the same mentality when it comes to your AWS bill. Understand where the cost came from, who was responsible for it, and if it is just a cost or an investment.



When multiple teams or workloads are involved, ensure that you can precisely identify who or what team is responsible for every resource. Work with those specific stakeholders on optimizing their spend.

The CFO can now take learnings from Team 2 and share the knowledge with the other teams to optimize cost across the organization.

Using tags/different accounts/OUs, etc. are effective ways of being able to track and attribute expenditure to different teams/parts of the business. This will also help in measuring specific ROI for each workload.

Knowledge check 1

aws training and certification

It is recommended to design workloads to allow components to be updated regularly and making changes in big and irreversible increments.

- ❑ True
- ❑ False

 106

Knowledge check 1 (cont.)

aws training and certification

It is recommended to design workloads to allow components to be updated regularly and making changes in big and irreversible increments.

❑ True

✓ False

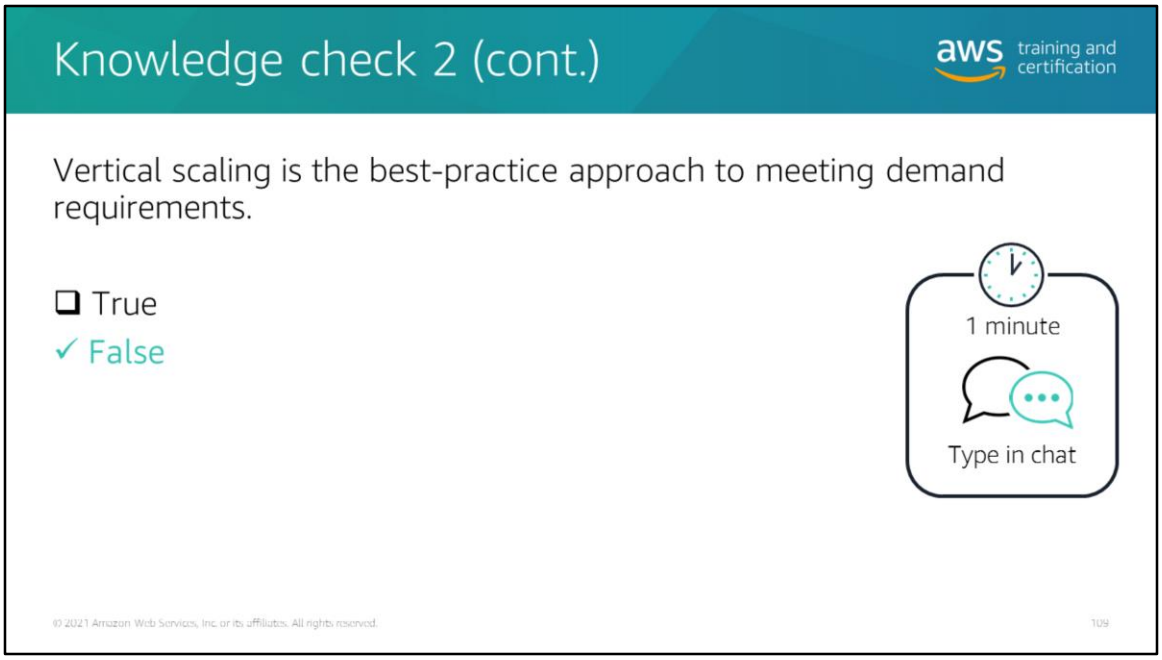
 10 /

Knowledge check 2

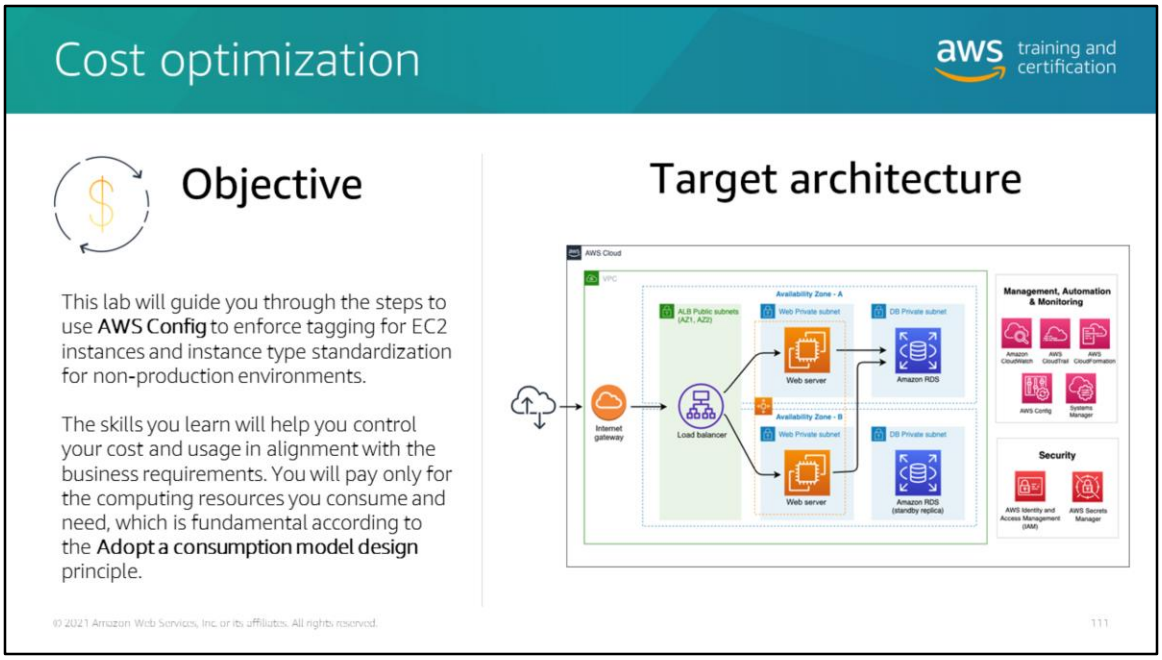
aws training and certification

Vertical scaling is the best-practice approach to meeting demand requirements.

- ❑ True
- ❑ False



Horizontal (not vertical)



AnyCompany wants to have clear visibility about **costs and usage** for AWS services. They want to use a flexible tool for creating their own dashboards according to their needs.

They understand that they need a new way of analyzing expenditures to make informed decisions about cost tradeoffs.

Based on that information, the objective of this lab is to identify how AnyCompany can create and operate their infrastructure with the best performance while maintaining alignment to the business and technical needs.

Estimated duration: 40 minutes

Wrap-up after lab completion:

When you have this kind of analysis, you will want to add more measuring and monitoring controls so that you can get better insights for IT costs on workloads. This is aligned with the **analyze and attribute expenditure** design principle.

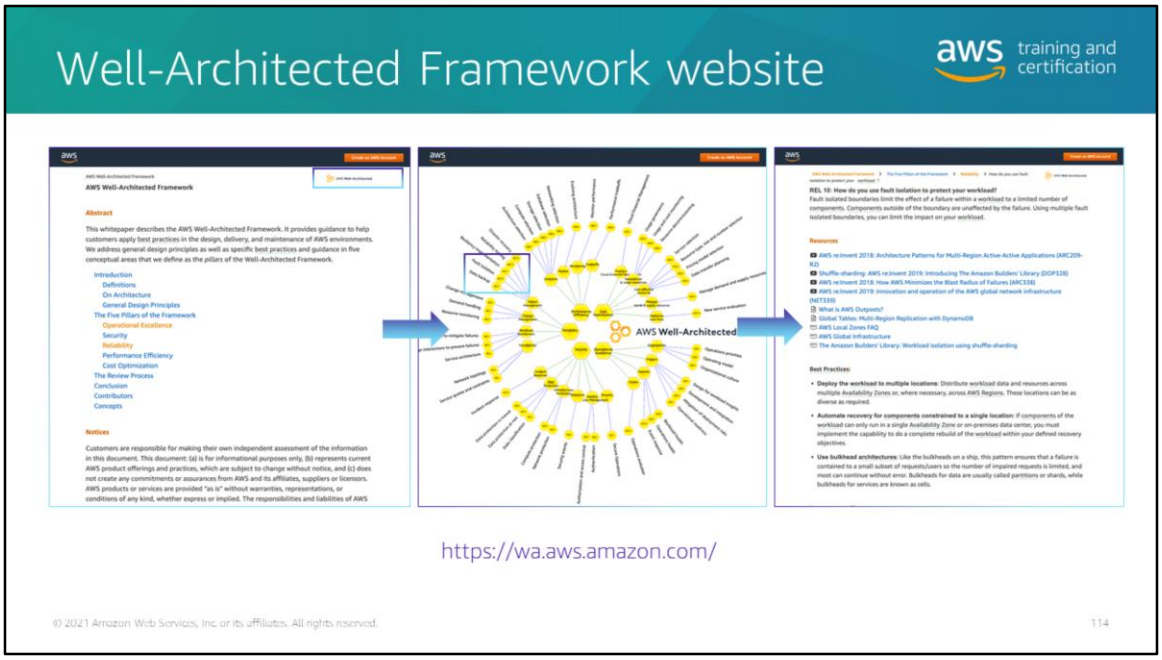
Using the appropriate services, instances, and resources for your workload is key to cost savings. A well-architected workload uses the most cost-effective resources, which can have a significant and positive economic impact. You also have the opportunity to use managed services to reduce costs. AWS offers a variety of flexible and cost-effective pricing options to acquire EC2 instances and other services in a way that best fits your needs.

Summary

aws training and certification

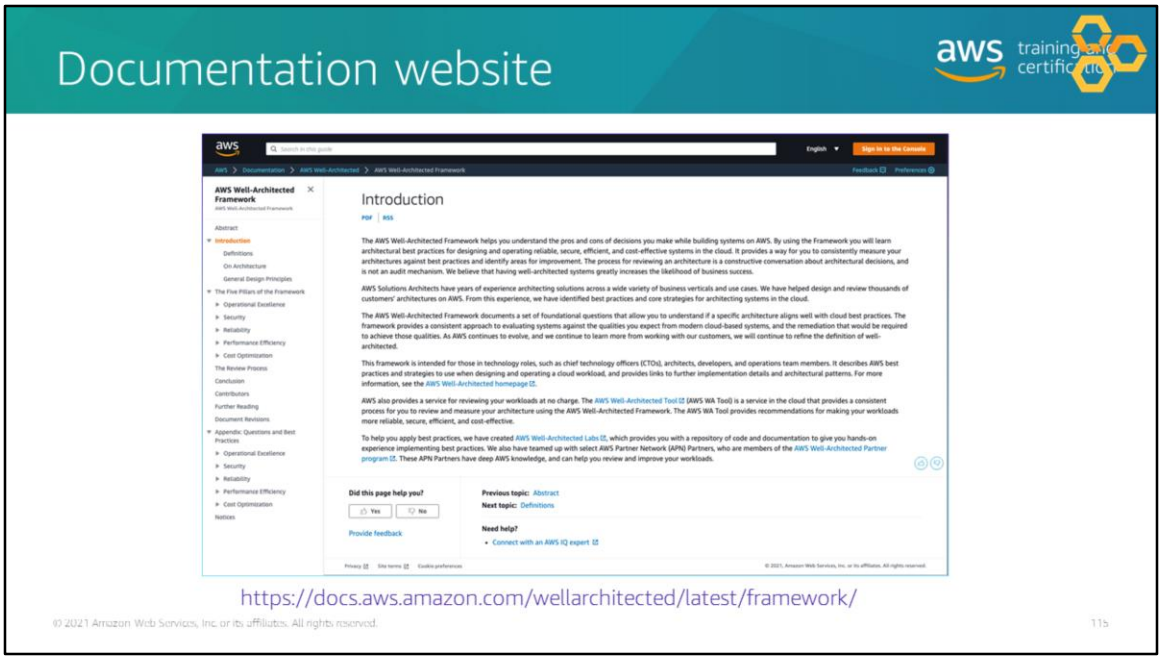
In this module, you learned:

- The definition of each AWS Well-Architected Framework pillar
- An overview of the design principles that are specific to each pillar
- How to apply architectural best practices for each pillar



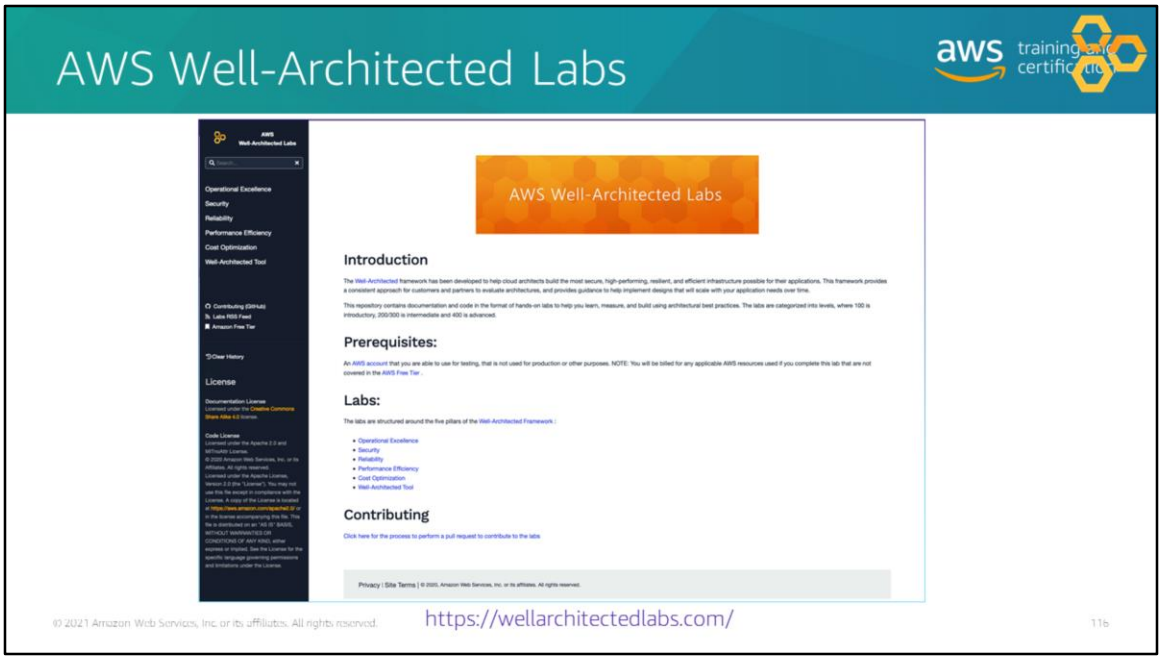
The **website contains the whole framework**, so you can **link to a particular section.**

You can **access the map** by clicking the logo**.** And **from the** map, you can **jump** to **any section.**

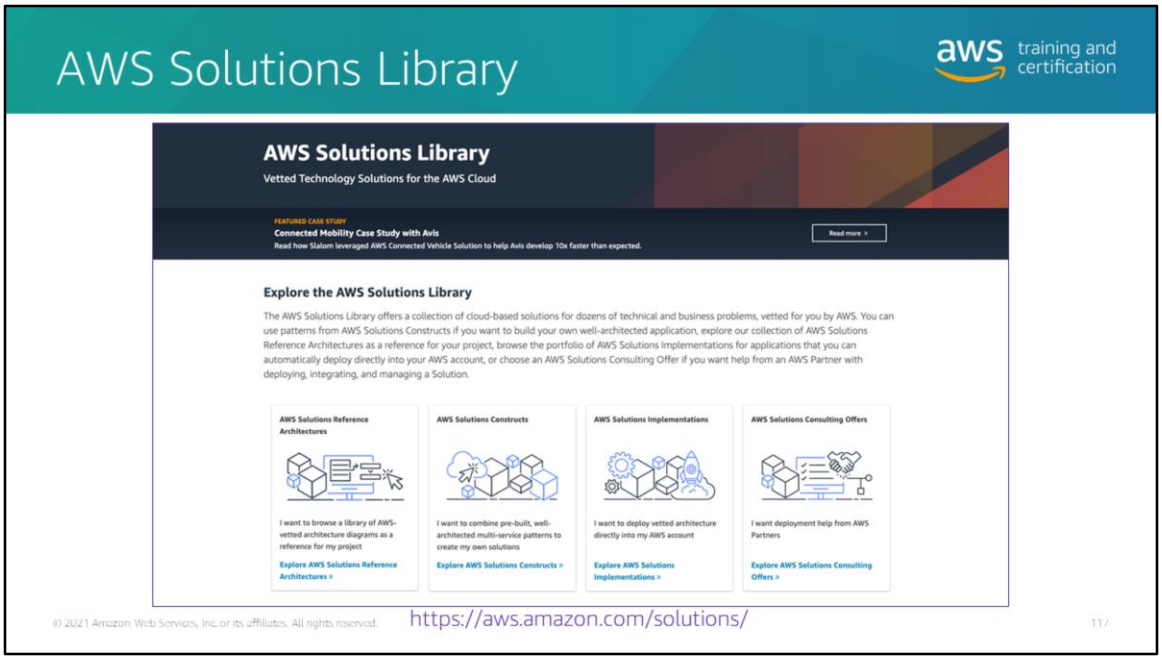


The **website contains the whole framework**, so you can **link to a particular section.**

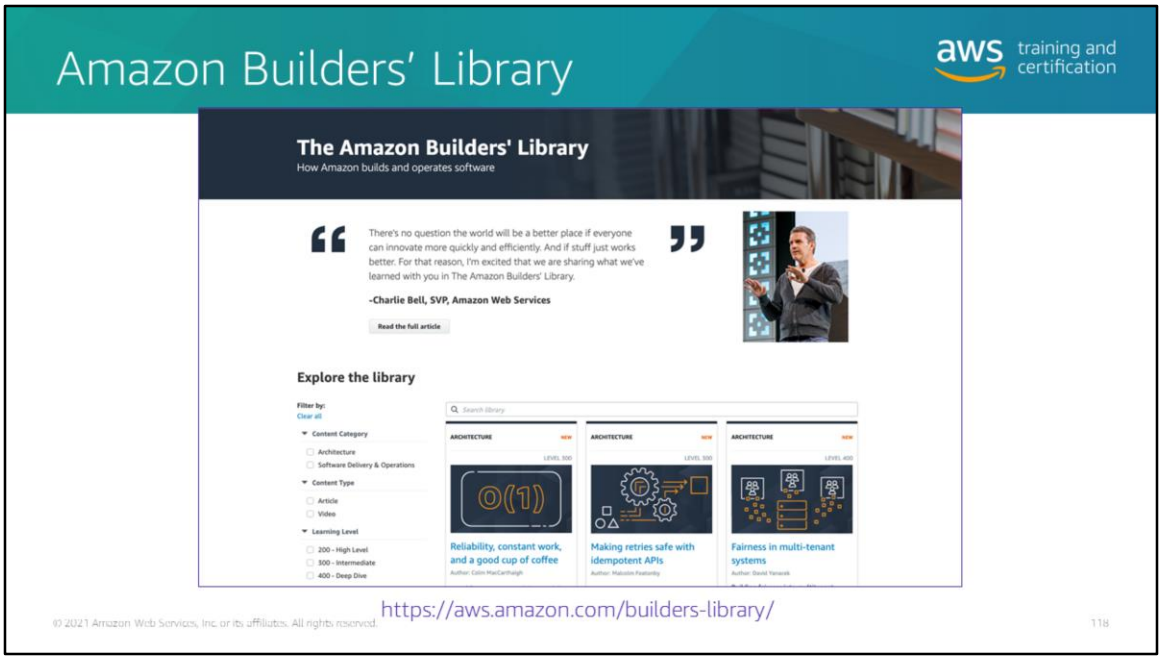
The site is hierarchical and searchable. And it is the location to get the newest versions of the Well-Architected Framework whitepaper content.



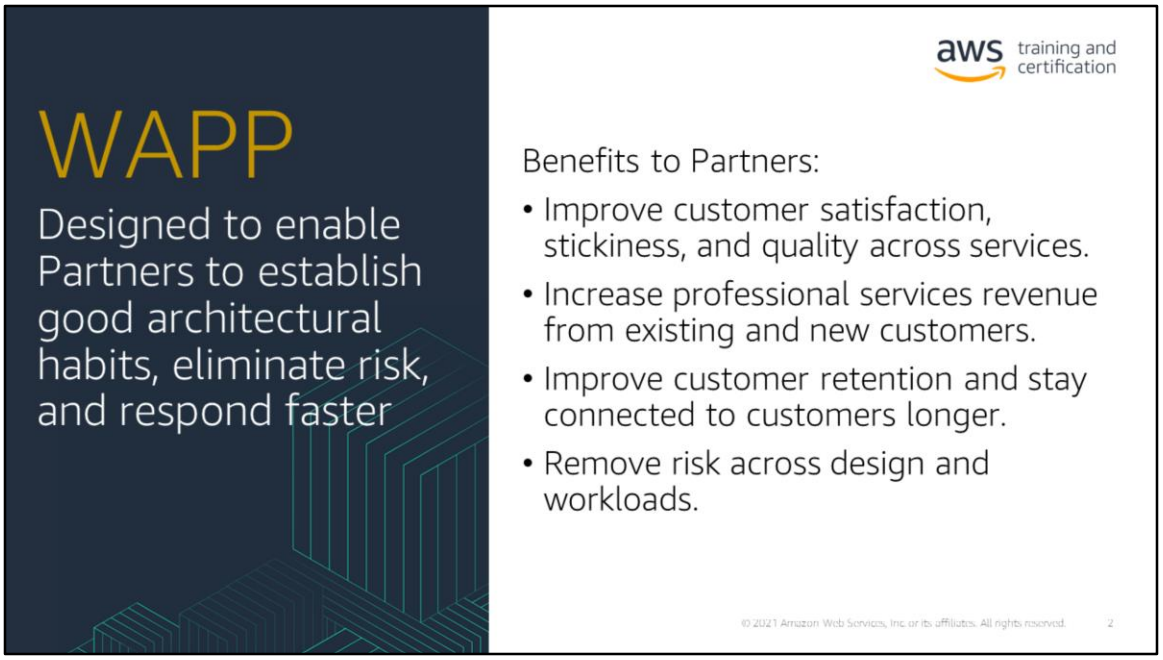
<Update to Hugo Site screenshot>



AWS Solutions Library is useful for finding prebuilt solutions or service patterns to help you get started. These have been vetted by AWS.



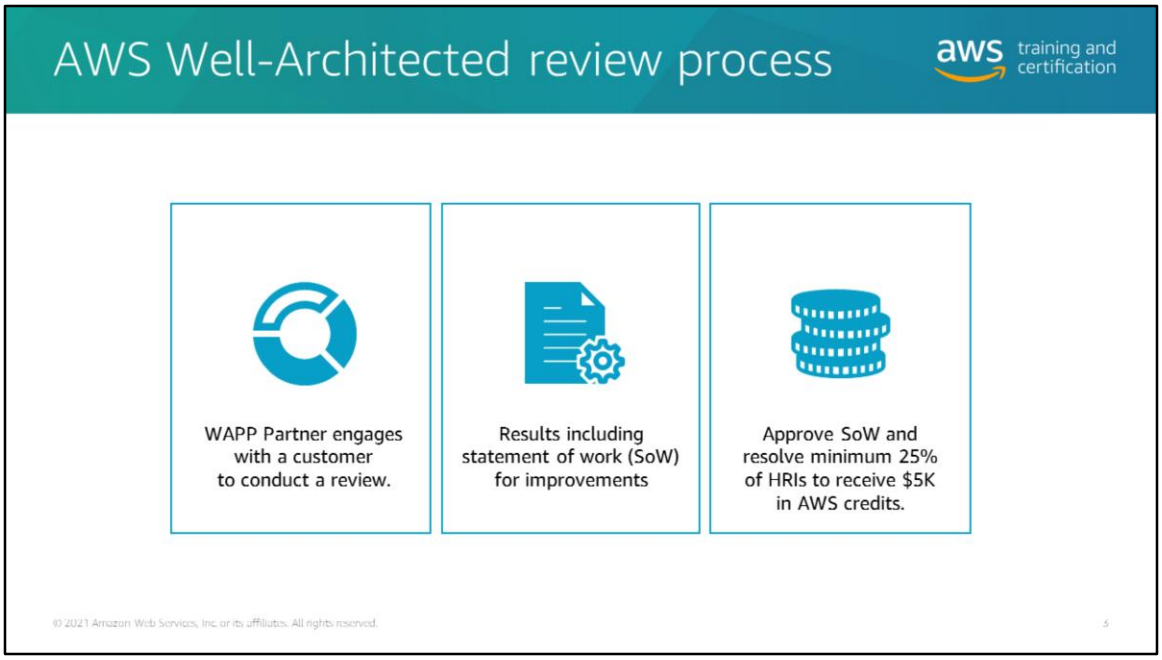
Amazon Builders' Library is useful for finding best practices and architectural implementations used successfully by Amazon and AWS.



We have created the Well-Architected Partner Program to educate and train APN Partners on how to engage leveraging Well-Architected experience that we've gathered across tens of thousands of Well-Architected engagements. These Partners have trained experts who have been validated to help customers through their Well-Architected journey, including education, advisory, discovery, engineering, and professional services. These Partners have experts who are trained on Well-Architected principles to help customers ensure that architectural best practices and processes are implemented. They help customers measure workloads against Well-Architected practices, and assist with workload remediations and improvements when resources are scarce or needed.

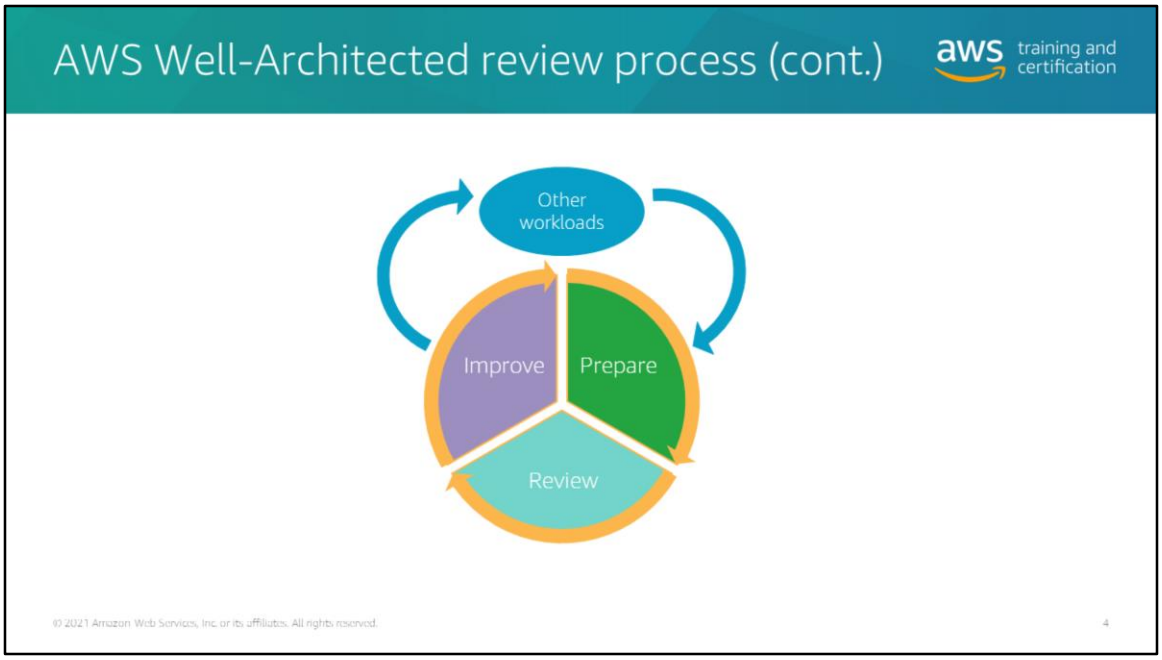
1. Partners who've institutionalized Well-Architected find themselves staying connected to customers longer as they build regular reviews into their business models and reviews provide a pipeline of ways to stay connected, making improvements of the workloads after critical risks are remediated.
2. **Up to 75% of the time deploying new services on the workloads they are reviewing (across all Partners delivering reviews)**
3. **99% of workloads reviewed uncover risks across security, cost, reliability, and operational excellence (across 394 reviews).**

4. **90% of reviews end in a paid engagement to remediate the workload (across 394 reviews).**

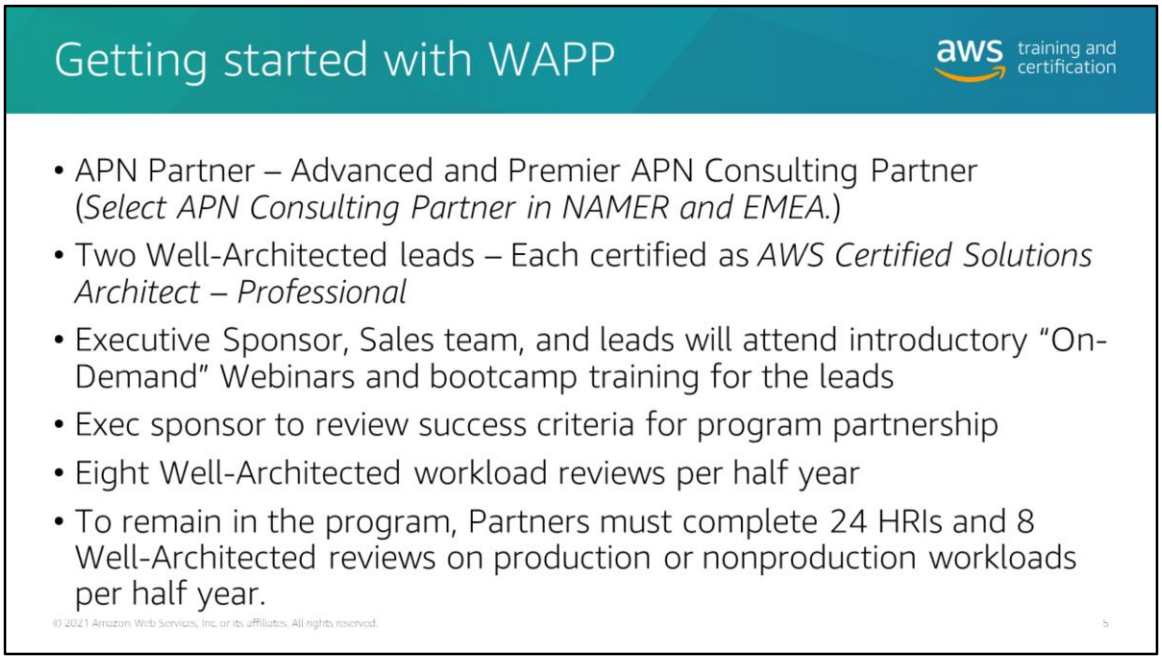


If you need a **Partner to help**:

- **Most** Partners have an **offering** where they will review your **workload and** provide you with **results** and **a statement of work (SoW) for addressing the top issues.**
- If you **approve that SoW,** AWS will provide you with **AWS Credits.** If a Partner remediates 25% or great of HRI discovered on a production workload. (funding is done ONCE per workload. We need to stress that an application could be cut up into multiple workloads, and Partners could help to scope workloads WITH you as the customer.)



The AWS Well-Architected process is a virtuous cycle, much like the Amazon and AWS Flywheels. By running reviews on one workload, teams will learn and be empowered to run reviews with other teams on their workloads and share lessons learned.



If you are an APN Partner are interested in joining the Well-Architected Partner Program, we have some requirements to share with you.
APN Partner - Advanced and Premier APN Consulting Partner in APAC and LATAM, Select and higher in AMER and EMEA.
You need to have a minimum of two Well-Architected leads (champions inside partner org) per geography and these individuals must have an AWS SA Pro cert
You need to deliver a minimum of eight Well-Architected workload reviews per half year

To enroll, you need to watch three on-demand webinars that introduce the program and give training on how it can be integrated into your business. And you need to send your Well-Architected leads to an AWS-run WAPP bootcamp.

If you are interested, please reach out to your partner team (PDM or PSA).

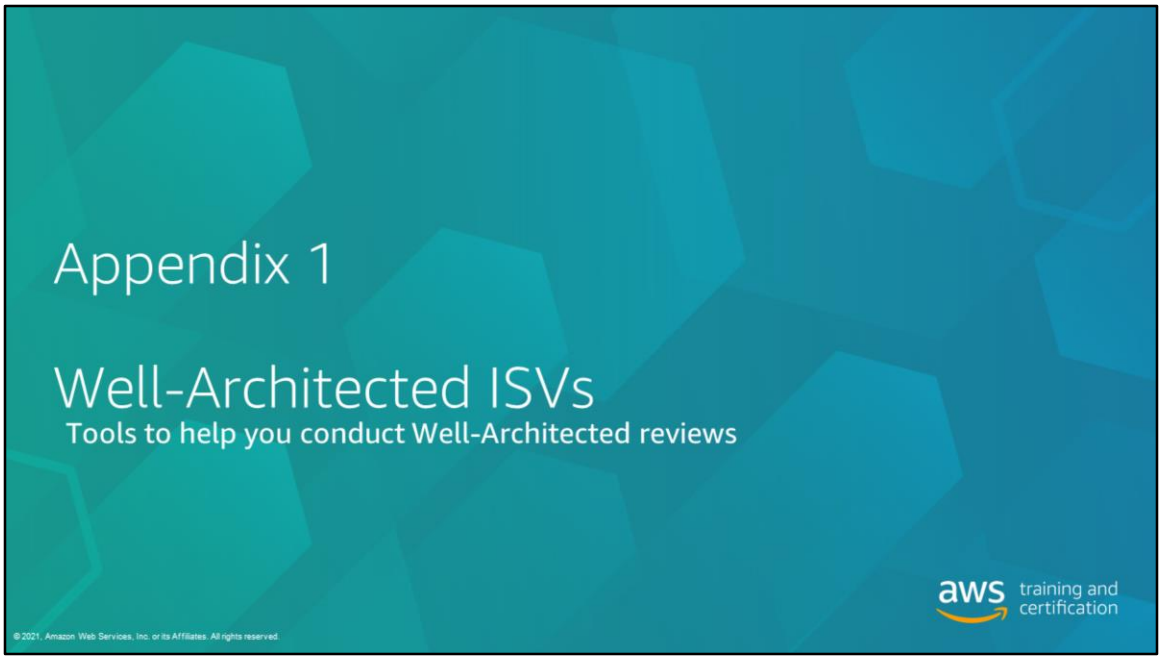
What have we **learned** from doing reviews?
Review **early in the lifecycle. It's quicker and easier to fix things.** – and can **influence design**
Example:
Including while a workload is still on premises.

You could do a Well-Architected Framework review on the designs that you have that are intended to be implemented in the cloud. Then do a review right after the lift and shift or migration.

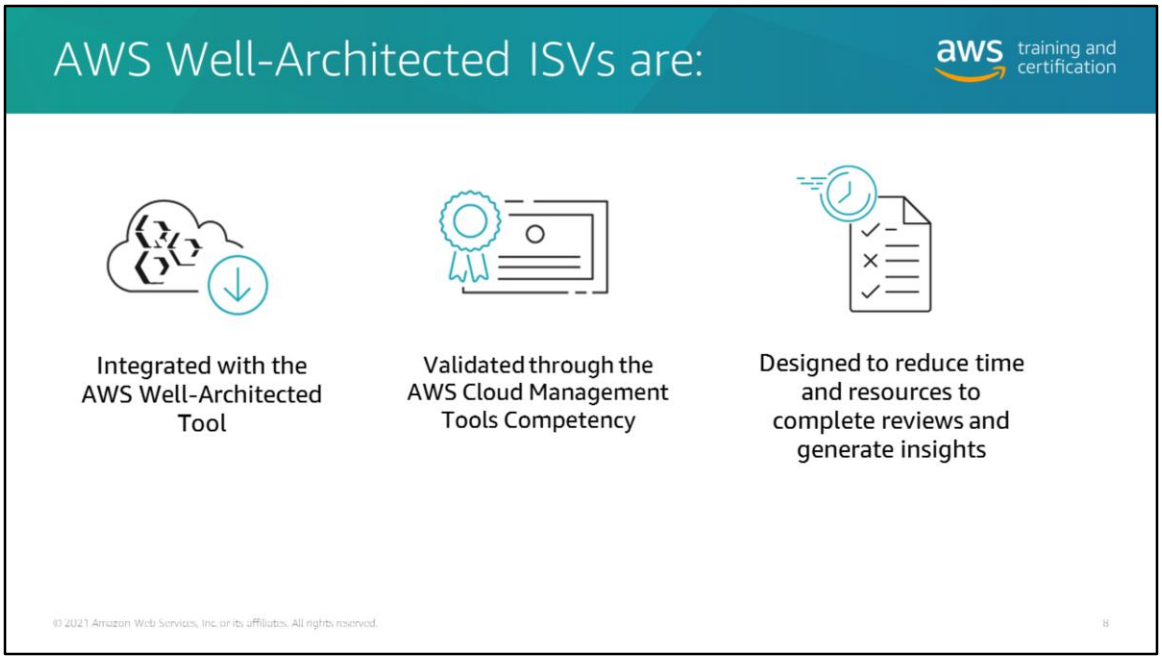
I raise this specifically because the questions of "what if my workload is on prem?" or "Are reviews only for workloads that are in AWS?"
Answer:
no, reviews can be done for workloads not yet in AWS and on prem.
Reviews can even be done on other clouds or multi-cloud. The framework is public and the tool does not cost anything to use.

The most common problem we see is **not bad decisions;** it's people **neglecting a decision**. Don't decide to not back up data, they forget to talk about it.

Most workloads **have high-risk items** that must be addressed. Finding them is **not a bad thing;** they were **always there. If you address them**, that's **one less thing** that can **damage** or **slow your business**.

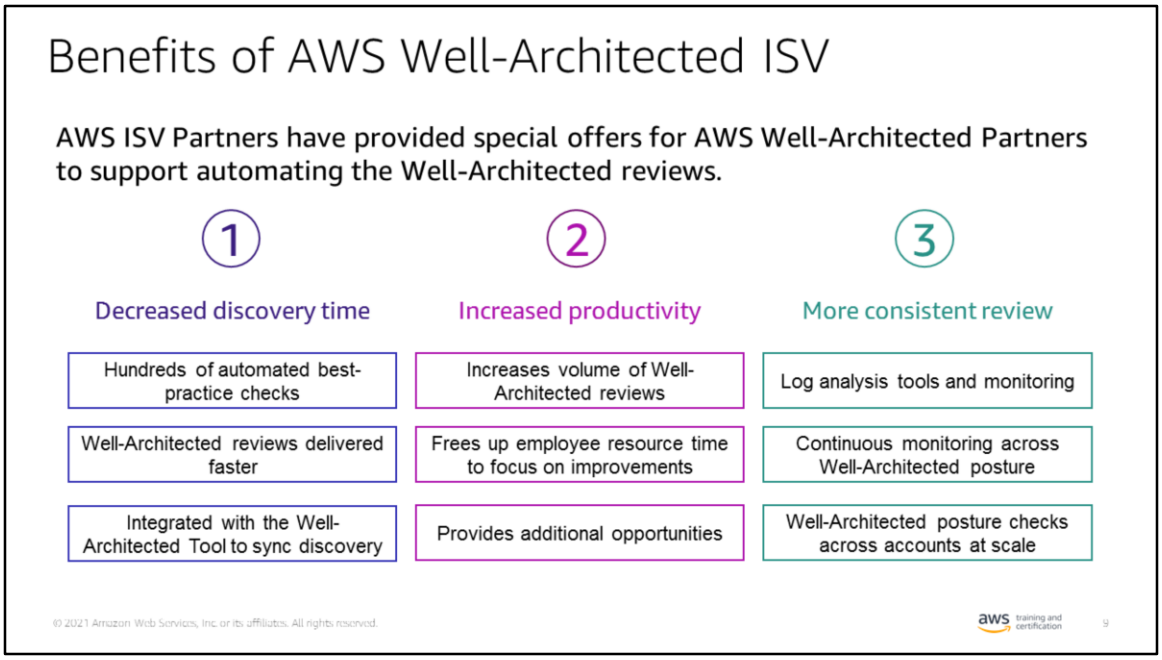


Only use with independent software vendor (ISV) Partners. Otherwise, please hide this section.



In 2018, the AWS Well-Architected Tool was launched to provide a consistent approach for reviewing applications against the latest AWS best practices. At AWS re:Invent 2020, AWS made APIs available for the AWS Well-Architected Tool. This let AWS Partners extend functionality, measurements, and learnings into their existing architecture governance processes and applications. Through custom Well-Architected Tool integrations, ISV Well-Architected Partner Solutions now supports a broad range of use cases to make it faster and easier for AWS customers to adopt Well-Architected best practices across teams and systems.

These AWS Well-Architected ISVs make it easier for you as a Consulting Partner to automate the AWS Well-Architected review. It takes a lot of time to run a Well-Architected review. And we have heard from from Consulting Partners that it is difficult to break even due to the time investment. For example, Well-Architected Consulting Partner iOLAP has seen that the Well-Architected review timeframe is reduced from 2–3 weeks to hours when using one of these Well-Architected ISV Partners. nOps.

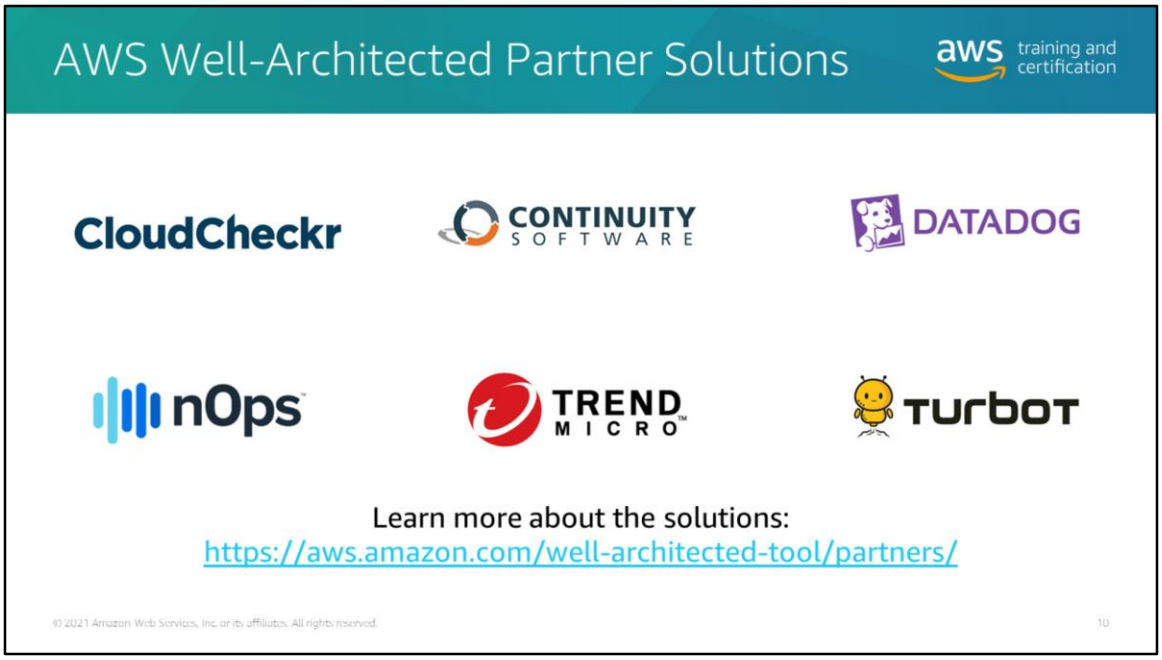


The value for Consulting Partners to use these tools includes decreased discovery time, increased productivity, and a more consistent review.

Across our Partners, there are several hundred best practices that are automated, allowing you to conduct the review faster. Additionally, the integration with the Well-Architected Tool syncs this discovery.

With the additional time, you can increase the volume of Well-Architected reviews you do. This frees up employee resource time to focus on remediations, and therefore providing additional opportunities.

Finally, because these tools are looking at AWS service configurations, you can get more prescriptive, real-time insights. This allows you to review several workloads at scale and in a continuous way. If you are a Managed Service Provider, these tools are incredibly valuable for continuously monitoring the workload health of your environments over time to ensure that the Well-Architected posture is maintained.



Here are our current Well-Architected ISVs who are integrated with the AWS Well-Architected Tool.

These ISVs have Consulting Partner offers available in APN Partner Central. I am going to quickly show you how to access the offer briefs so you can learn more about the tools and integrations. You can learn more about these tools and the offers available exclusively to AWS Well-Architected Partner Program Partners in this folder: https://partnercentral.awspartner.com/ContentFolderPartner?id=a1o0h00000CUFQBAA5

I also wanted to touch on some of the opportunities that are available with these Well-Architected ISVs.

We have already seen a lot of Well-Architected Consulting Partners benefit from using these tools and providing a joint offering to end customers. That is why we are looking to develop Well-Architected Solution Bundles with our Consulting Partners and Well-Architected ISVs. Solution Space is a Partner validation program designed to help you and a Well-Architected ISV promote a joint solution. If you are already working with one of the ISVs we showcased on the previous slide and together you have had great joint success in supporting your customers, let us know if you are interested in participating in the Well-Architected Solution bundles at aws-well-architected-isv@amazon.com.

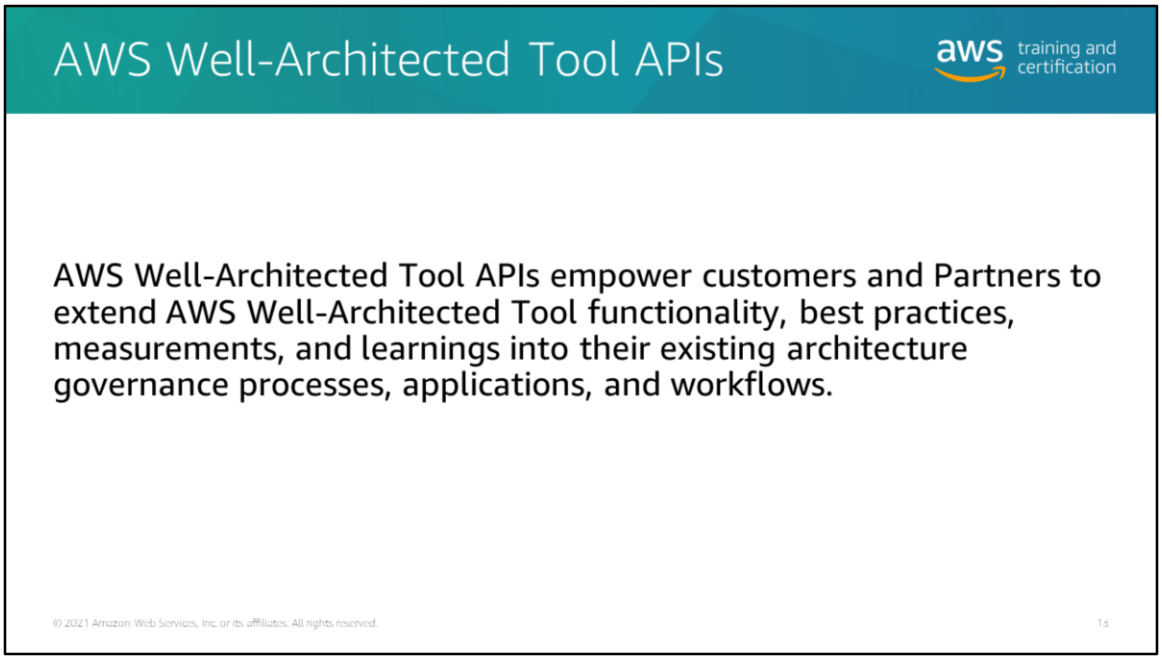
If you don't currently work with an ISV Partner but you are interested, we are hosting a webinar series for you to learn more about each of these ISVs. Once a week, we will bring on one of our Well-Architected ISV partners for 30 minutes so you can see their Well-Architected capabilities in action. Stay tuned for an email over the next few weeks to see the schedule of solutions and to register for each session. As a benefit for attending, you will receive a 3-month unlimited Consulting Partner license to try

out these solutions.

All of the ISVs have Consulting Partner offers available in APN Partner Central. You can learn more about these tools and the offers available exclusively to AWS Well-Architected Partner Program Partners in this folder: https://partnercentral.awspartner.com/ContentFolderPartner?id=a1o0h00000CUFQBAA5

We also know that some Well-Architected Consulting Partners might have their own tools that they use to automate the Well-Architected review. Since the launch of the Well-Architected Tool APIs at re:Invent, you can now integrate your own tools with the Well-Architected Tool, making your processes easier.

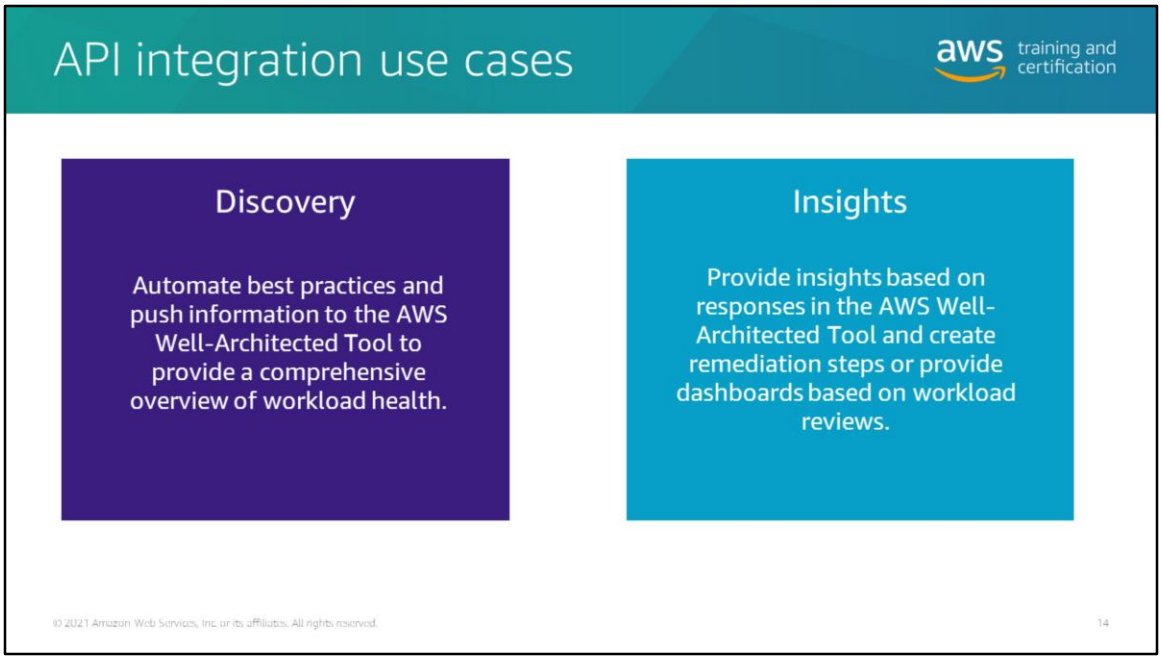
Additionally, if your tools are external facing, we will give you steps on how you can engage in the Well-Architected ISV Program.



The AWS Well-Architected Tool APIs empower ISVs to customize the experience with Well-Architected. APIs help customers and Partners connect with the Well-Architected Tool through 25 different APIs, enabling you to integrate Well-Architected into your tool. These APIs provide the full functionality of the Well-Architected Tool into your tooling.

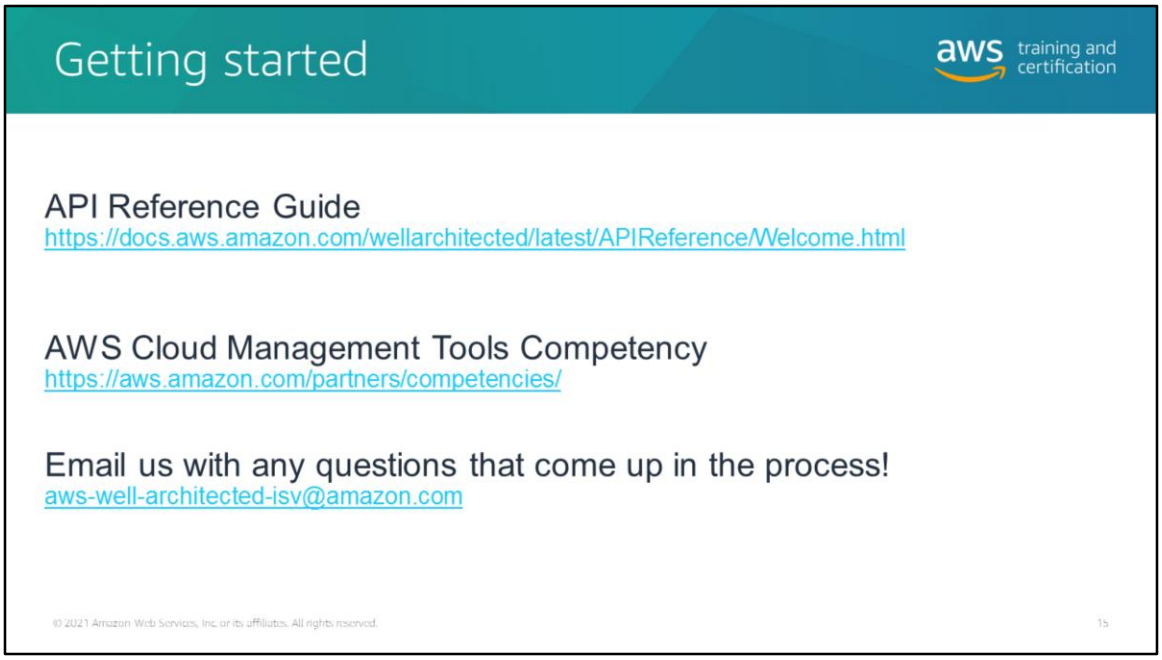
The APIs will help you:

- Prepopulate information in the Well-Architected Tool for your customers based on the information you already know about your customer, making the review more streamlined for your customer
- Pull information from the Well-Architected Tool, and can provide a more holistic picture of a customer's workload health
- Ensure that customers are going to one tool to determine their workload health, they won't need to manually track HRIs offline, and they will use your product for the single source of workload health
- Analyze risks related to workloads and recommend remediation steps and guidance for your customers



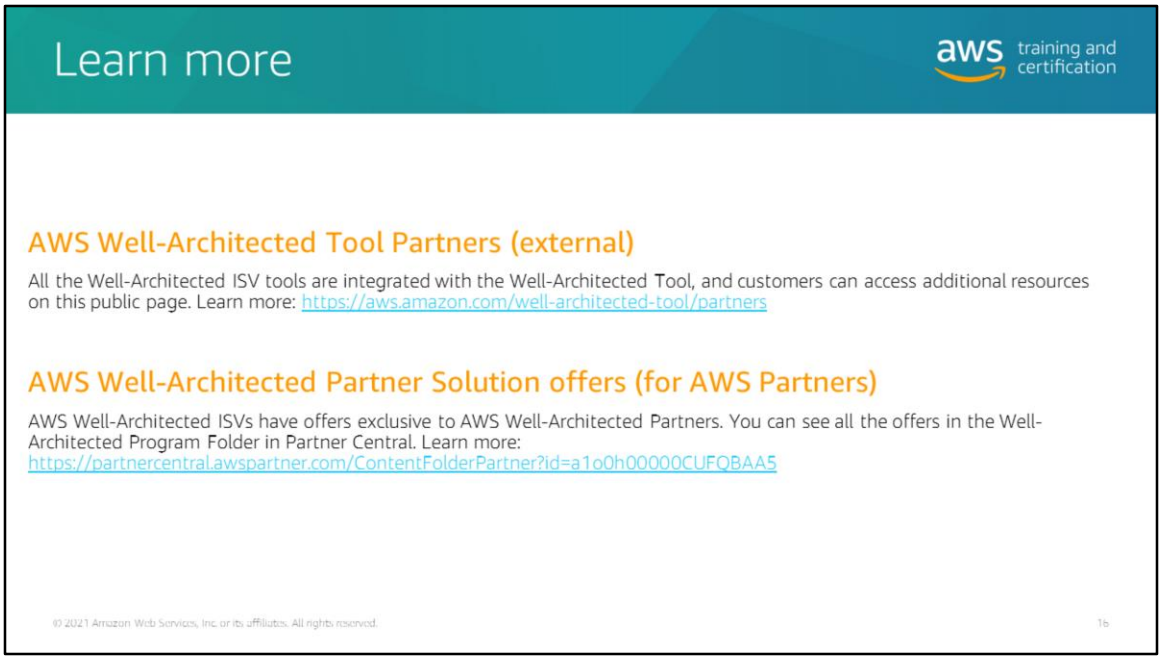
There are two identified Well-Architected API use cases. Because we have 25 APIs available, you can create custom integrations. But the APIs serve two overarching use cases:

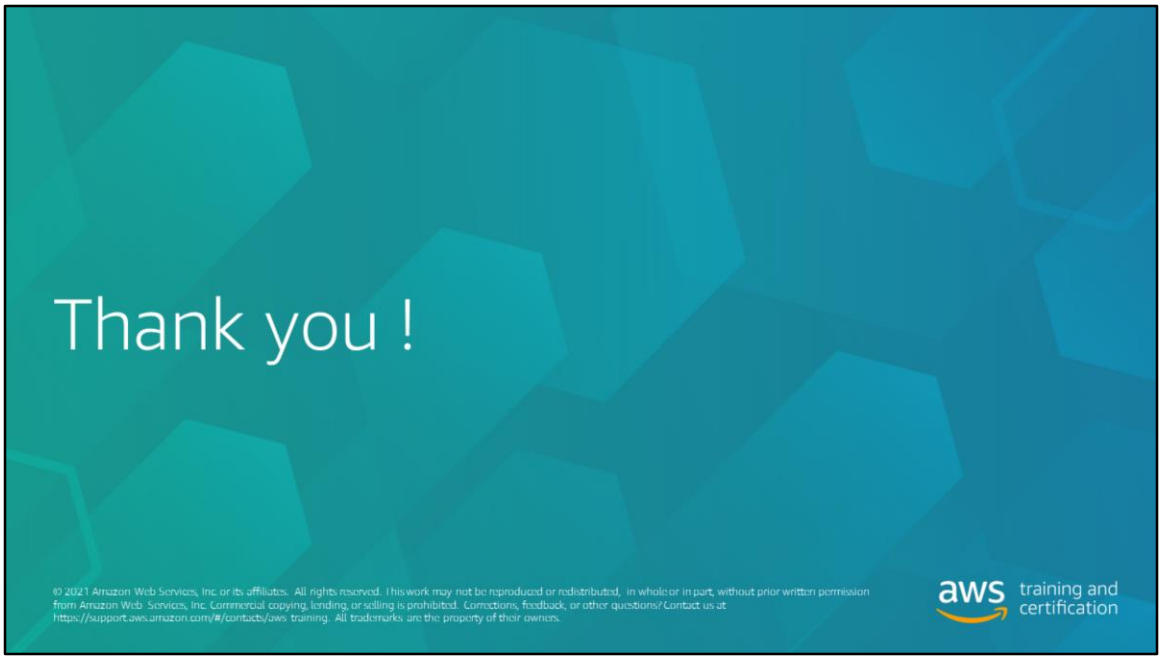
- Discovery Integration Partners: Automate well architected best practices and pull details of the additional best practices that cannot be automated from the Well-Architected Tool to provide a comprehensive view of workload health.
- Insights: ISVs can take the information submitted in the Well-Architected review to provide comprehensive tracking or create remediation steps based on workload reviews.



If your tool is a good fit for the Well-Architected ISV Program, there are a few steps:

1. Read the API Reference Guide linked here, and identify how to integrate with the Well-Architected Tool.
2. Participate in the AWS Cloud Management Tools Competency. Now with the ISV Partner Path, Consulting Partners can register their products as an ISV and apply to programs like the Cloud Management Tools Competency (that recognizes ISVs).
3. Email us at aws-well-architected-isv@amazon.com.





Remind the class of how and when can they submit their feedback.

