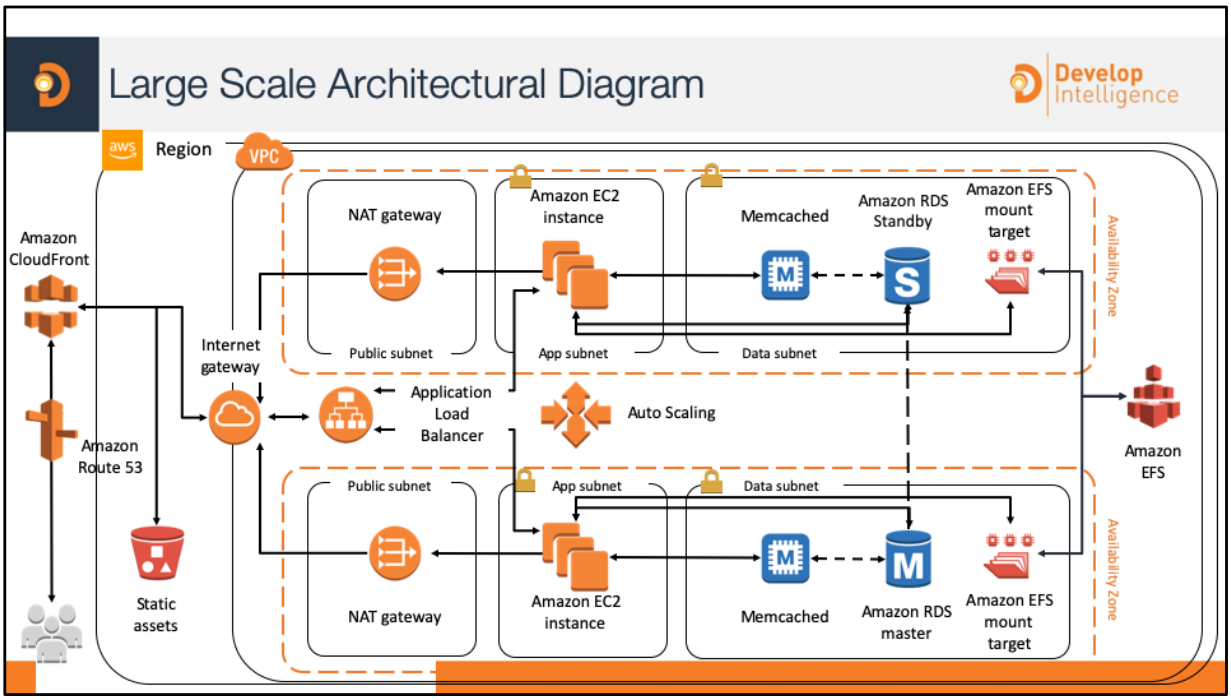
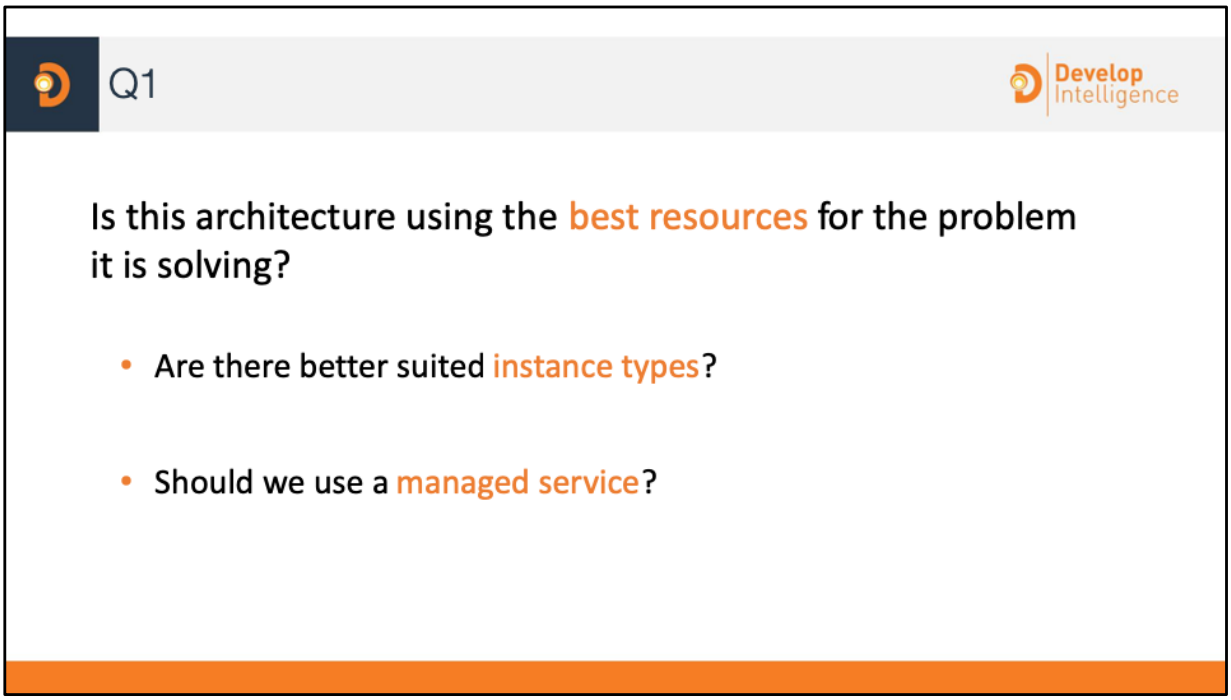
Optimizations and Review

Develop Intelligence

- Module 14



By the end of class, you will be able to understand all of the components of this architectural diagram. You will also be able to construct your own architectural solutions that are just as large and robust.

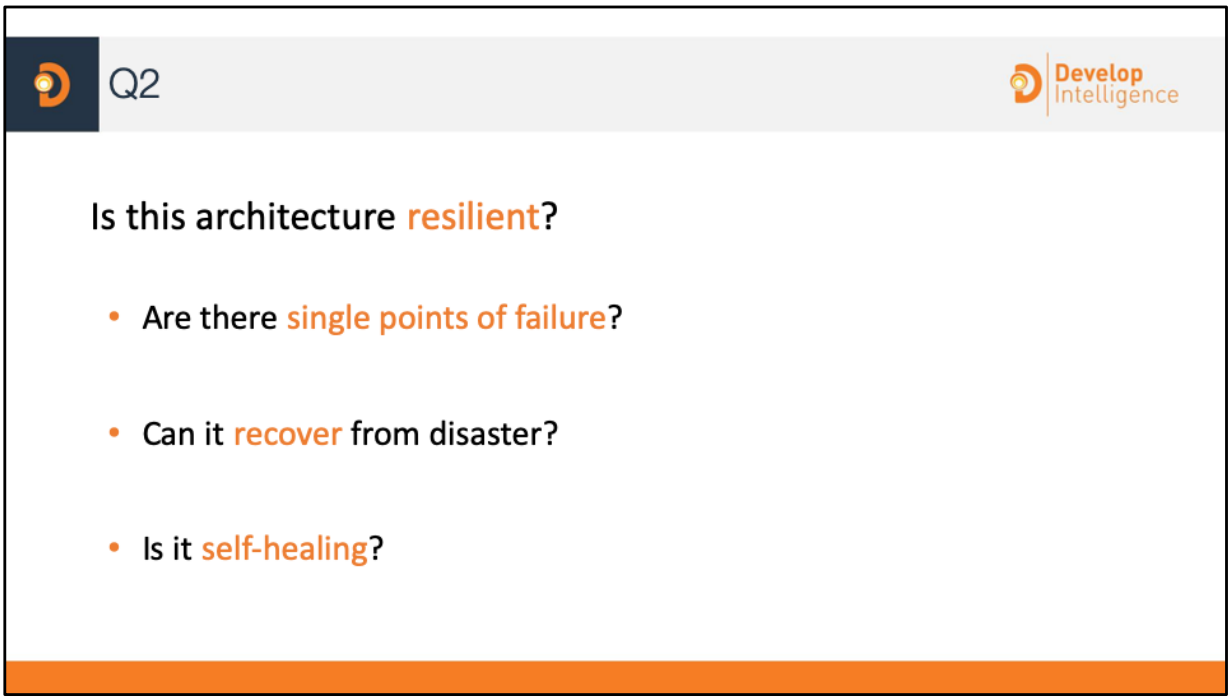


Often, when organizations buy hardware to put in a data center, they're given a budget and buy hardware with an eye to the future. The best hardware today will be obsolete in 18 months (+/-). So, with a one-time purchase, they're forced to buy bigger than they need to stave off future requirements.

So when you're considering building out a data center, ask yourself, "What do I need right now?" If you need to make it larger or smaller, you can.

Another way to think about it is, "Can I really afford to pay for capacity I'm not really using?"

Managed services, such as Amazon RDS, allow you to build resources in AWS without having to figure out all the requirements first. Let's say a developer asks for a Microsoft SQL Server database and you've never installed the software before, it will take considerable time and effort. Instead, find out what kind of database is needed and let AWS build it for you. You have total control of your database, your data, and its access. All AWS does is free you to do the creative work of design and implementation.

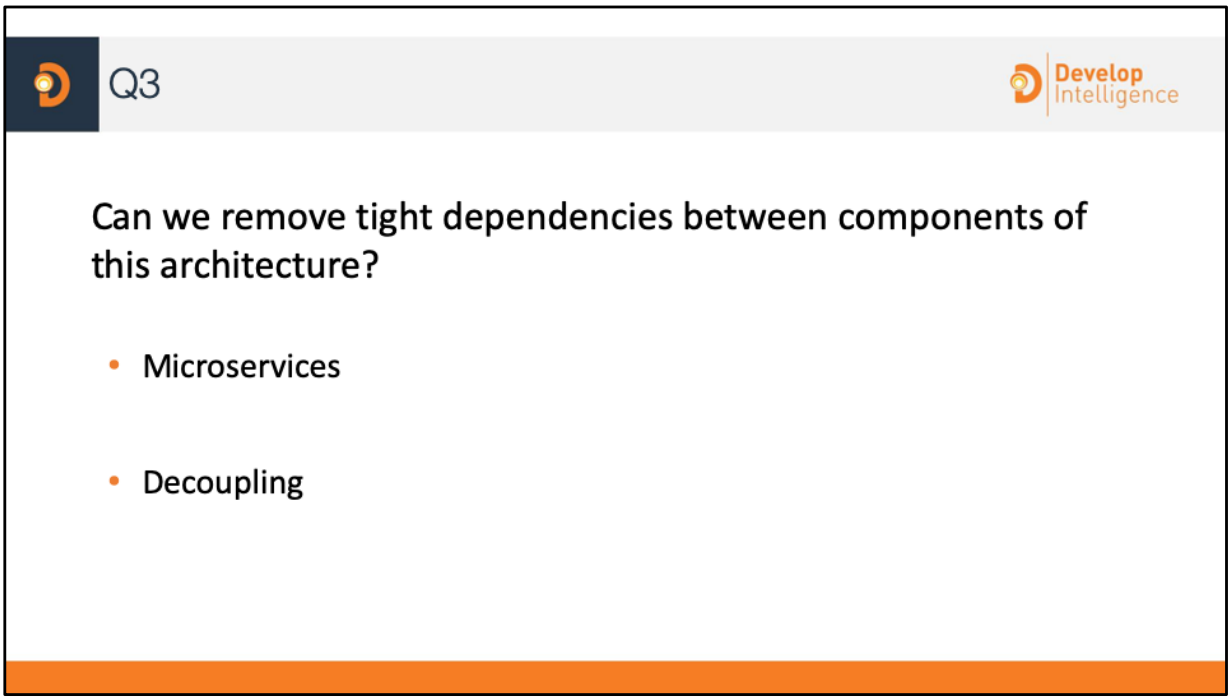


At Amazon, one of our best practices is to start at the end of a process and work backwards. We ask ourselves, What's going to break? If it does, what happens?

Yes, you can fix it, but do you really need to do that? Instead, delete the bits that have gone bad and turn on new ones. You can, and should, do a postmortem report for disasters. However, if an instance, container, or similar thing goes down, replace it first, then figure out why.

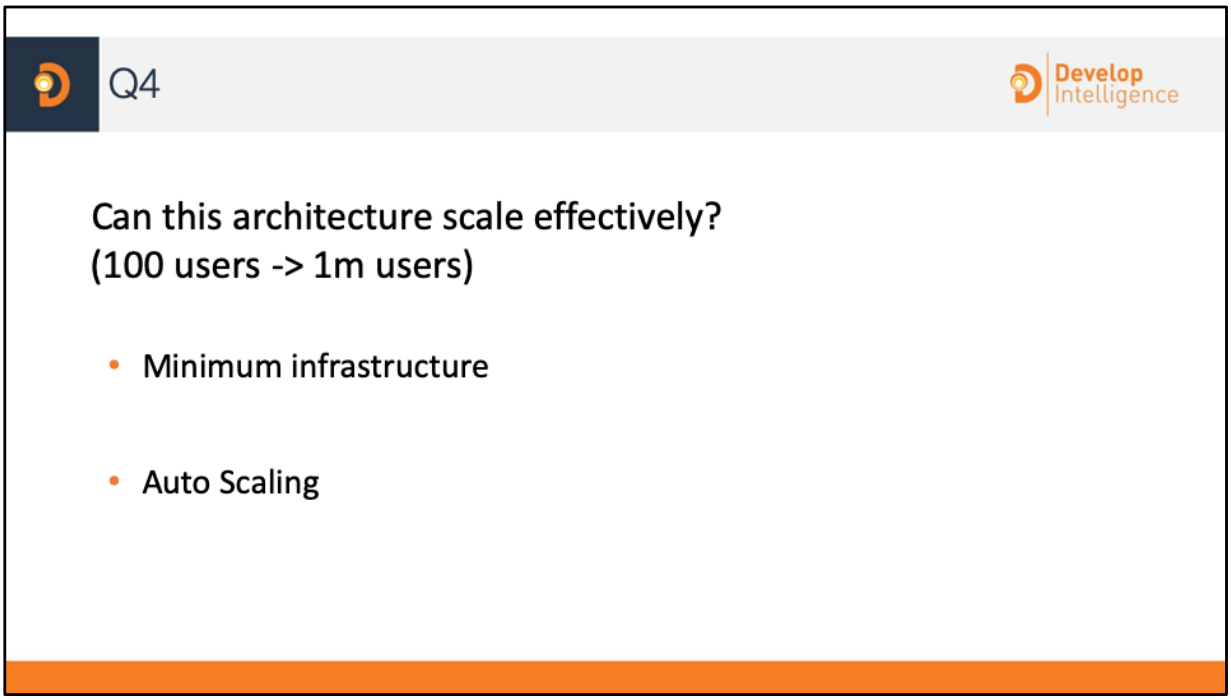
In IT, amateurs run backups—professionals restore. In AWS, you can run a full disaster drill without impacting your production environment. Yes, it might cost a couple of dollars, but it's much less expensive than building a second data center.

You've been hired to do creative work—why spend time on manual work that could be automated? Using AWS, you can set up an environment that can monitor itself, delete broken things as needed and then *tell* you what's happened.



Microservices are one of the best ways to increase reliability and availability of your architecture. Using this approach helps to improve your speed of innovation, by reducing your dependencies between application components. Your teams will be able to act asynchronously to deliver feature updates without affecting the whole system.

Taking this kind of approach may require a new way of thinking about how systems work. Monolithic systems have many sunk costs which can be eliminated in the cloud, if you take the time to pull things apart to see how they really work.



You may want the biggest, fastest, most powerful machine possible—but if you don't use a machine to its fullest capacity, you're wasting money.

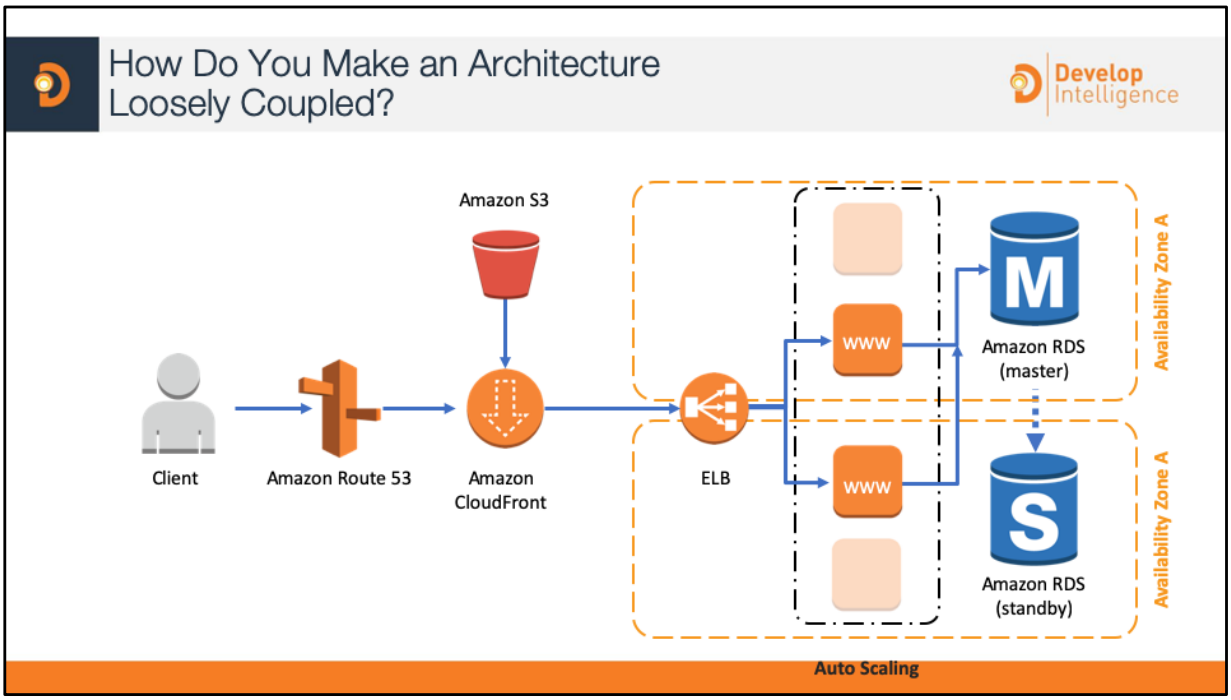
It takes about 3 months to get enough data points to figure out what that number really is. So when you're trying to figure out what capacity you really need, start with a machine that is bigger than you need. Then, once you see what you're using, you can "right-size" your environment.

When you have an on-premises data center, spinning up new resources is absorbed by all the line items involved with the data center itself. When you run a new workload in AWS, there's a cost associated with it. The good news is that in AWS, costs are transparent—and you can see that spinning up a new instance is far less expensive than building out a physical server.

There are two types of failures in the world of IT:

- Something breaks because you asked hardware or software to do something out of its specified tolerances.
- Your product becomes so successful that users ask your hardware and software to exceed its specified tolerances.

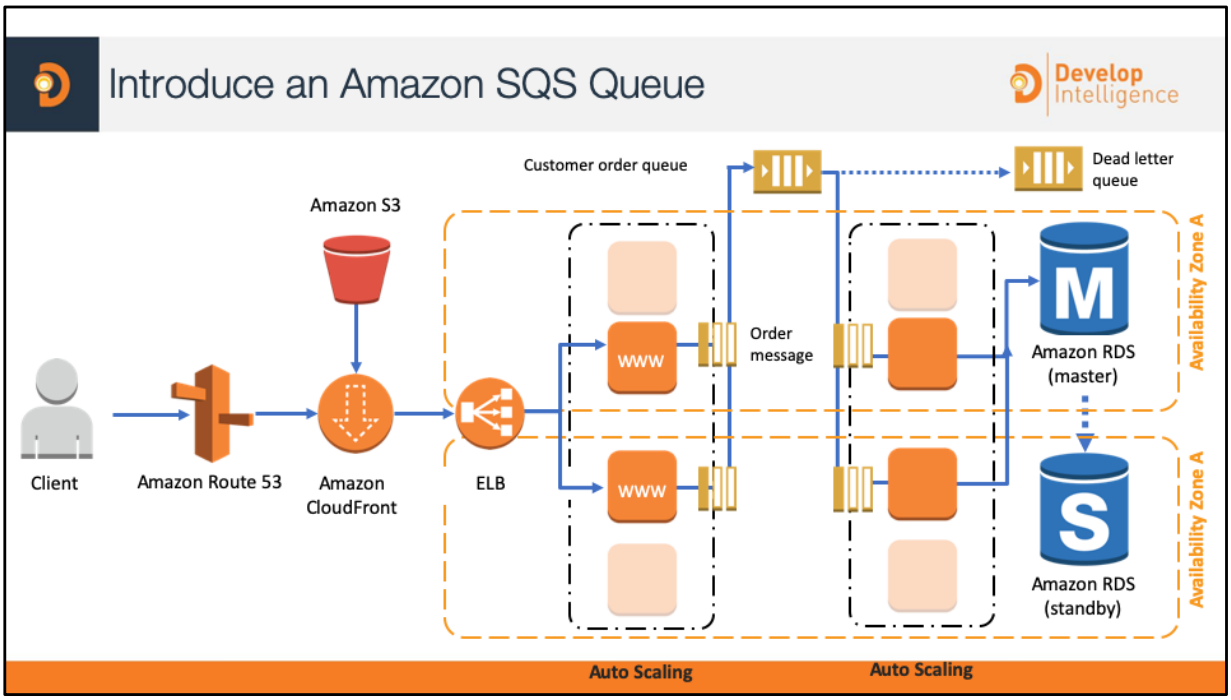
Automatic scaling costs money—but without it, your software will go offline because it can't handle the load. It's worth the investment to keep it running.

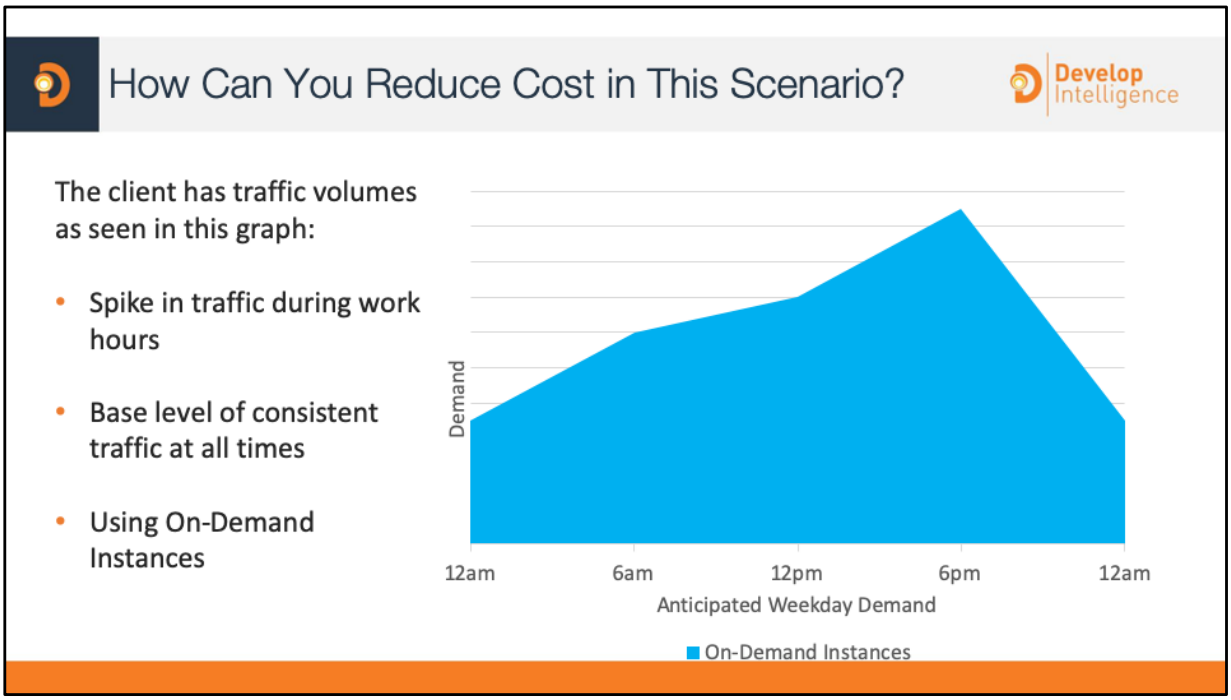


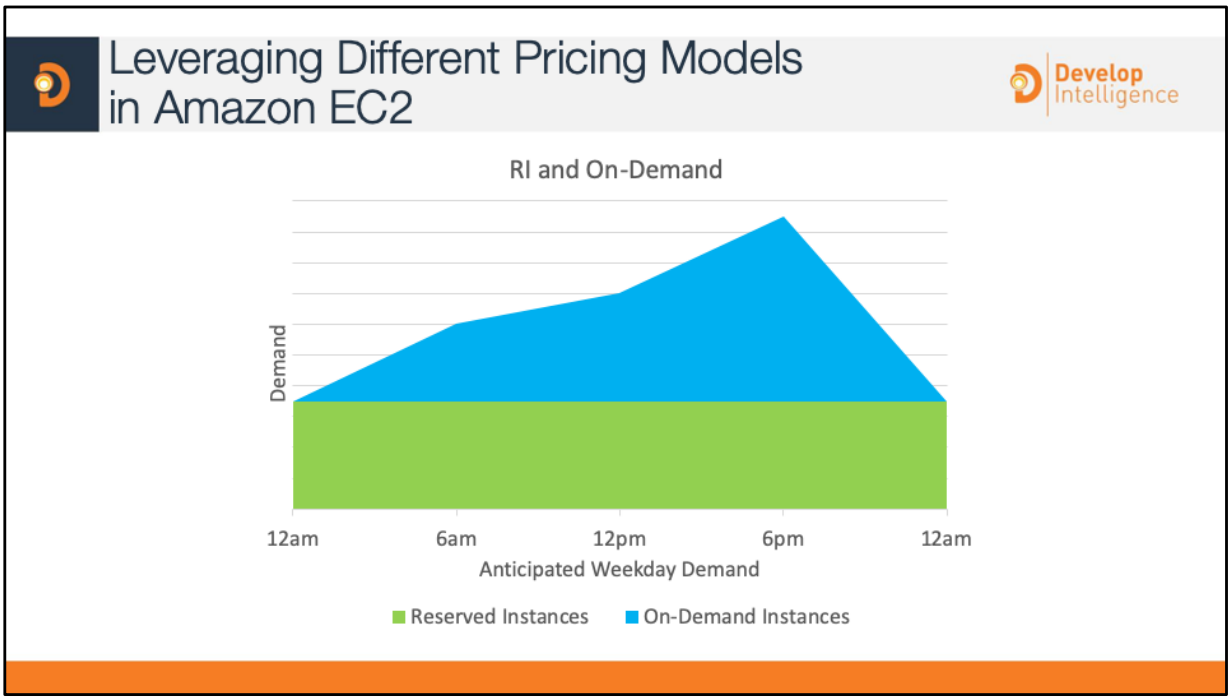
In this example, the application is responsible for handling and persisting the order data, as well as dealing with increases in traffic for popular items.

One potential point of vulnerability in the order processing workflow is in saving the order in the database. The business expects that every order has been persisted into the database. However, any potential deadlock, race condition, or network issue could cause the persistence of the order to fail. Then, the order is lost with no recourse to restore the order.

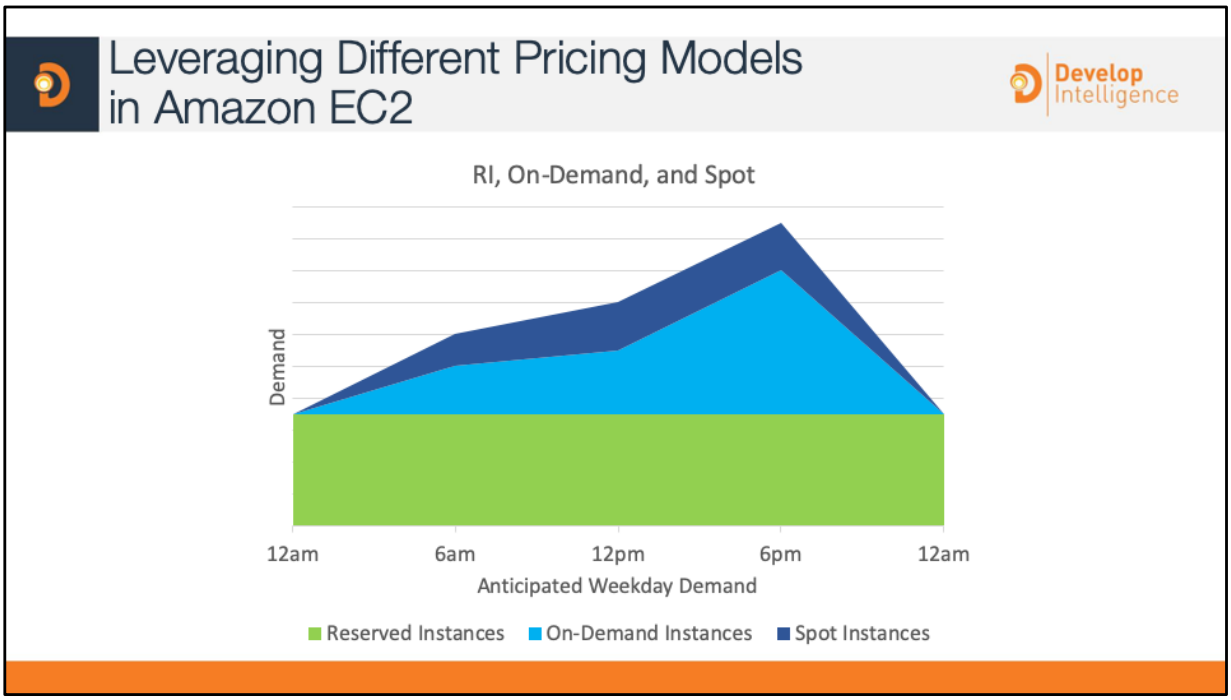
With good logging capability, you may be able to identify when an error occurred and which customer order failed. This wouldn't allow you to "restore" the transaction, and by that stage, your customer is no longer your customer.


Introduce an Amazon SQS Queue
Develop Intelligence
Customer order queue
Dead letter queue
Amazon S3
Client
Amazon Route 53
Amazon CloudFront
ELB
www
www
Order message
Amazon RDS (master)
Amazon RDS (standby)
Availability Zone A
Availability Zone A
Auto Scaling
Auto Scaling


How Can You Reduce Cost in This Scenario?
Develop Intelligence
The client has traffic volumes as seen in this graph:
Spike in traffic during work hours
Base level of consistent traffic at all times
Using On-Demand Instances
Demand
12am
6am
12pm
6pm
12am
Anticipated Weekday Demand
On-Demand Instances



This chart illustrates a very basic way to use Reserved Instances and On-Demand Instances together to accommodate for fluctuations in demand over time. In this example, multiple Reserved Instances have been purchased and are running, but as the day goes on and demand increases, the need for more instances increases as well. This customer supplements their capacity with On-Demand Instances, which can be shut down when they're not needed later in the evening.

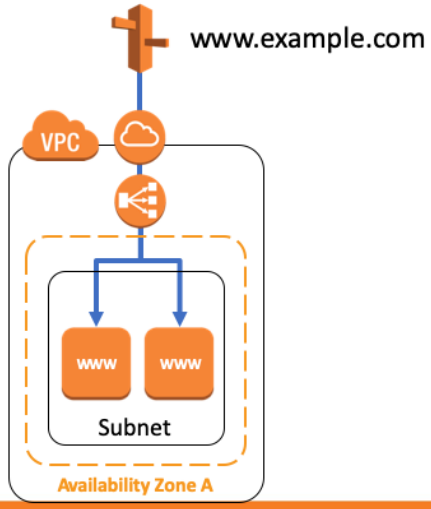


In this second chart, this customer takes a more complex approach, which attempts to leverage all three price models. In this case, some of the supplemental instances come first as On-Demand, but further needs are addressed with Spot Instances. This allows them to save money by using Spot over On-Demand; however it exposes them to unexpected instance termination, because they would lose those instances if they're outbid. This could lead to lost data or insufficient capacity for their customers, which means a model such as this should only be implemented in circumstances where sudden termination of instances is acceptable and handled appropriately.

One AWS customer who has leveraged all three models together is Pinterest. For more information, see http://www.allthingsdistributed.com/2012/08/tco-and-return-on-agility.html.

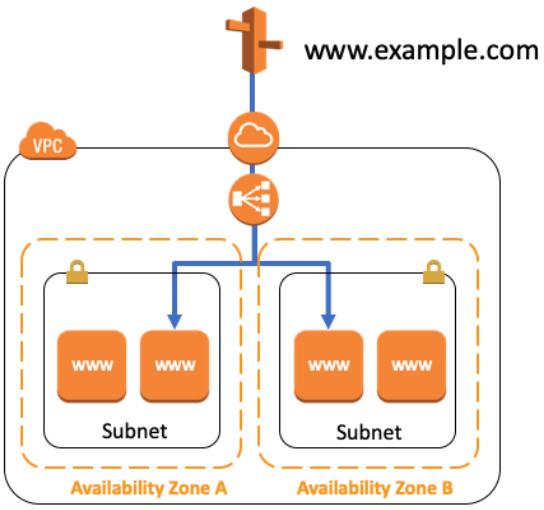
How Can You Make This Architecture More Resilient?

Develop Intelligence



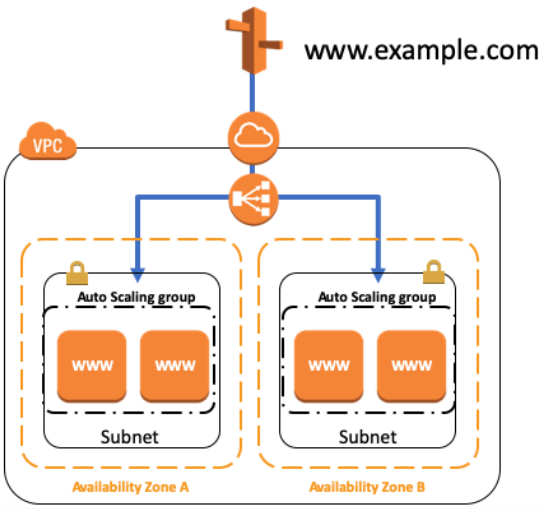
How Can You Make This Architecture More Resilient?

Develop Intelligence

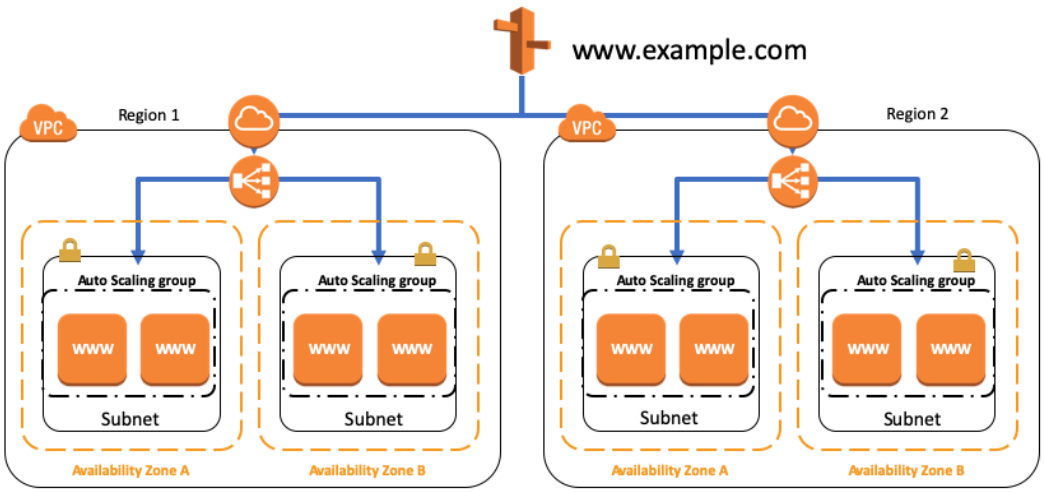


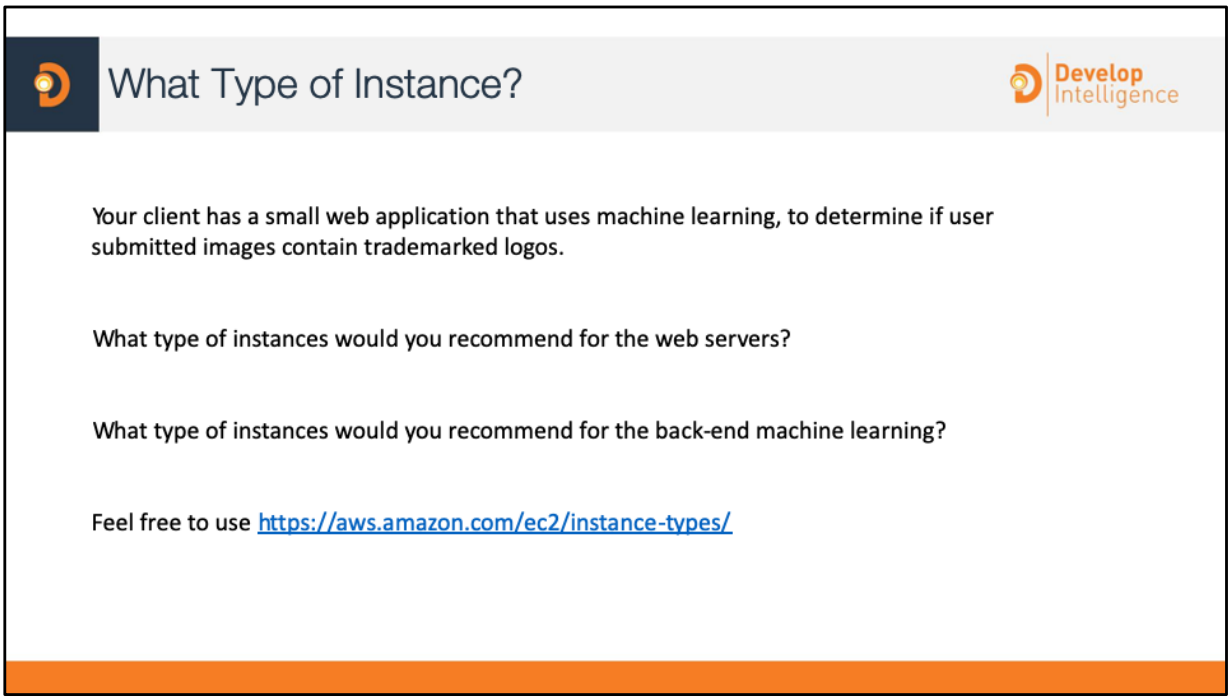
How Can You Make This Architecture More Resilient?

Develop Intelligence

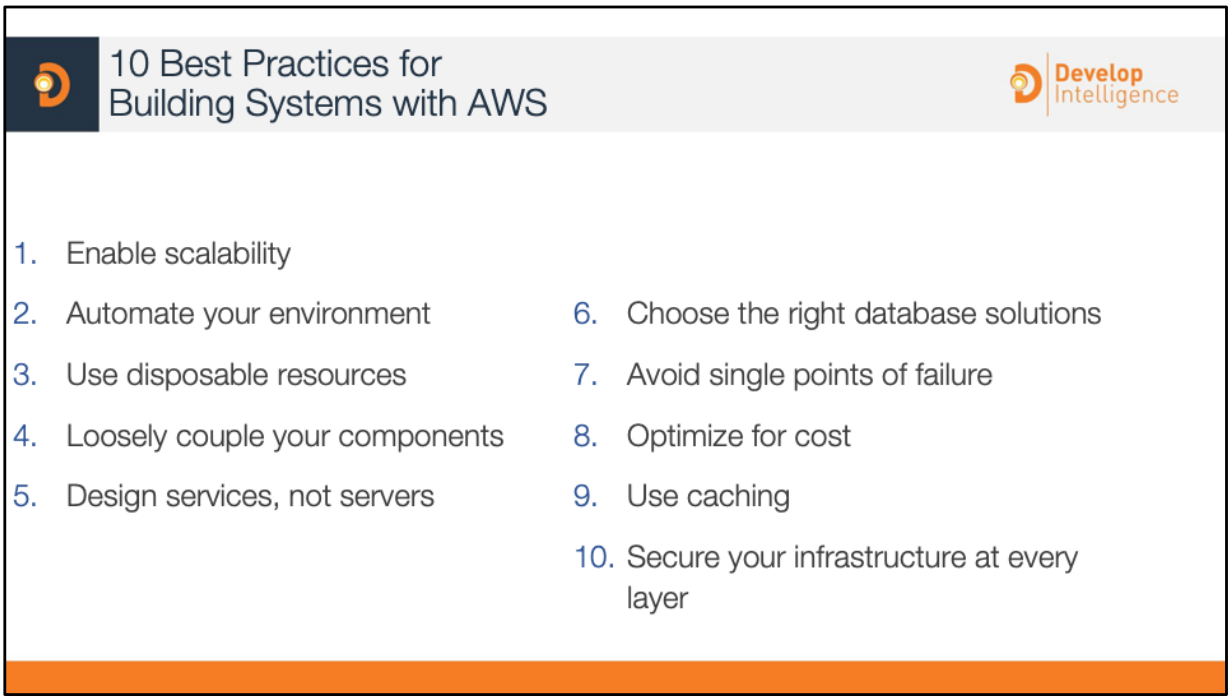


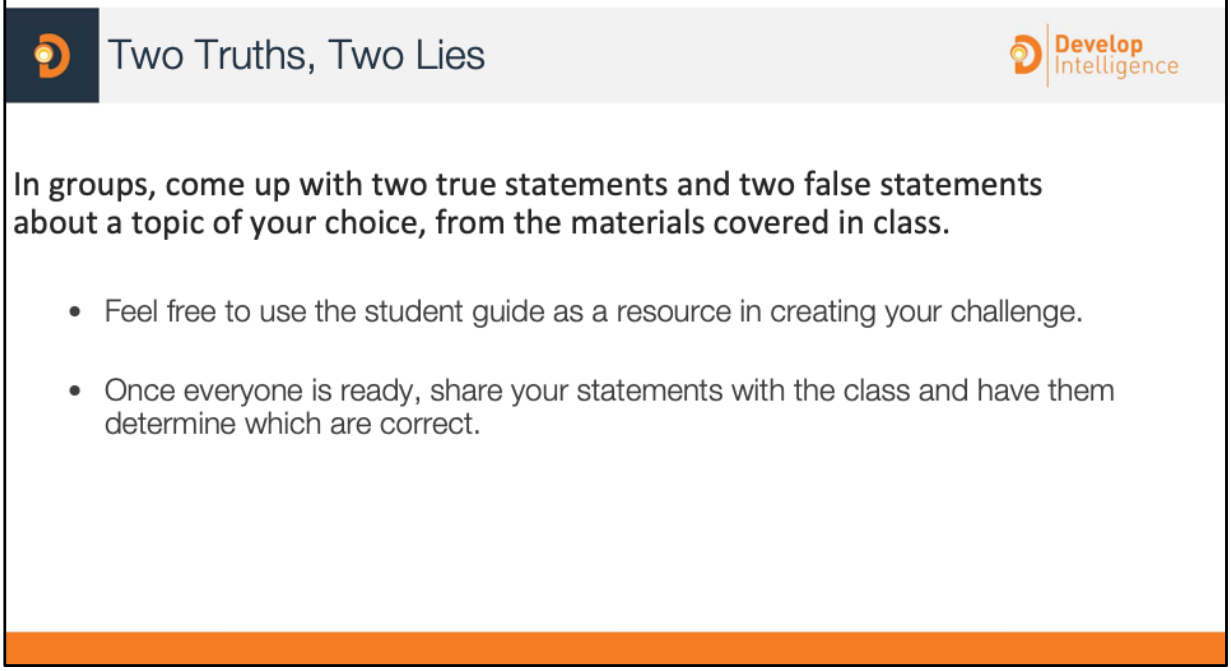
How Can You Make This Architecture More Resilient?

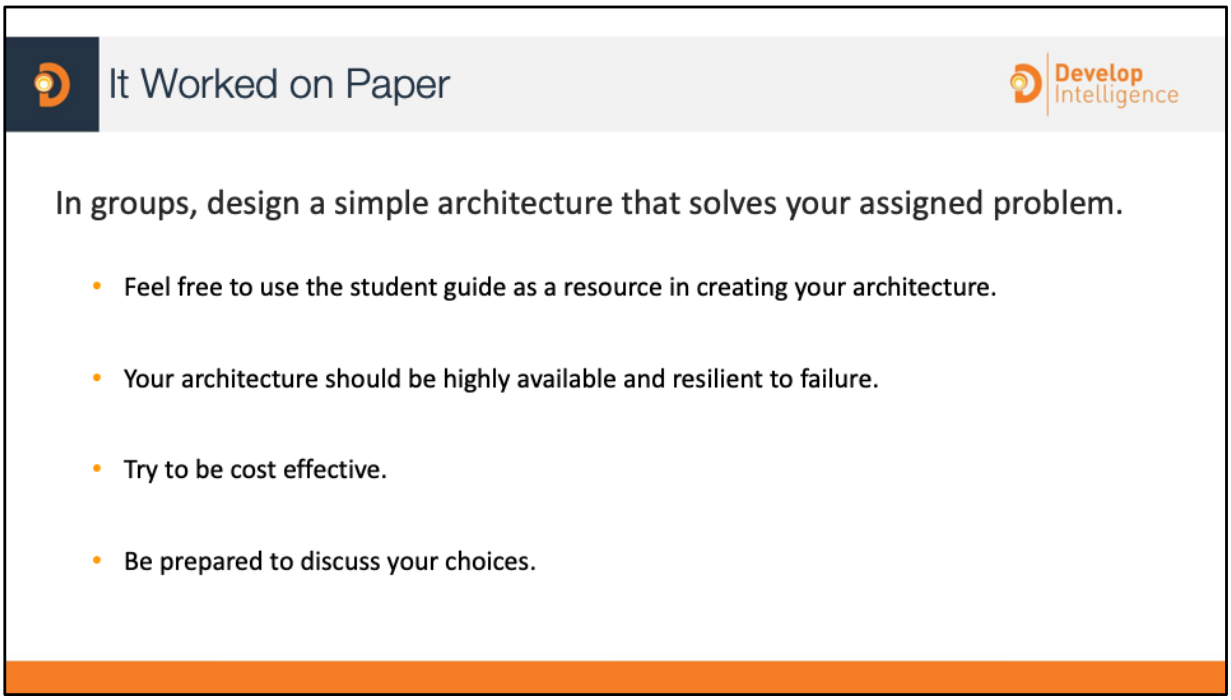




For more information about choosing an instance type, see https://aws.amazon.com/ec2/instance-types/.


10 Best Practices for
Building Systems with AWS
Develop
Intelligence
1. Enable scalability
2. Automate your environment
3. Use disposable resources
4. Loosely couple your components
5. Design services, not servers
6. Choose the right database solutions
7. Avoid single points of failure
8. Optimize for cost
9. Use caching
10. Secure your infrastructure at every layer


Two Truths, Two Lies
Develop Intelligence
In groups, come up with two true statements and two false statements about a topic of your choice, from the materials covered in class.
Feel free to use the student guide as a resource in creating your challenge.
Once everyone is ready, share your statements with the class and have them determine which are correct.



Sample problems:

- Online image resizing app
- Simple online store with order processing
- Video streaming on demand
- Image sharing website with account logins (Facebook/Google/Amazon)
- Online virtual desktop

- Feel free to add your own architectural challenges to this exercise.