|  |  |  |
| --- | --- | --- |
| Title | Click Instructions | Talking Points |
| image.png | 1. Click or tap anywhere to continue. |  |

|  |  |  |
| --- | --- | --- |
| Benefits | Click Instructions | Talking Points |
| image.png | 1. Click or tap anywhere to continue. | With Microsoft Azure Kubernetes Service, you can easily deploy a managed Kubernetes cluster in Azure, reduce the complexity and operational overhead of managing Kubernetes, and handle critical tasks like health monitoring and maintenance. |

|  |  |  |
| --- | --- | --- |
| Reproduce issue in web app | Click Instructions | Talking Points |
| image.png | 1. Click or tap to scroll down. | Let’s look at the example of SmartHotel360, a fictitious hospitality company that uses Azure Kubernetes Service (AKS) to simplify deployment of its containerized apps. We’ll approach the solution from the point of view of a developer assigned a bug to fix in a huge AKS cluster with a variety of microservices written in Node.js, .NET Core, and Java. We’re armed with a few prewritten queries to see logs and CPU usage, and a short description of a defect in the company's web app, which indicates that no results are found when a user searches for hotels in Seattle.  This is the web application’s homepage. Let’s compare results for Seattle and New York to reproduce the problem. |
| image.png | 1. Click or tap in the **Where** field. | To get started, type “New.” |
| image.png | 1. Type “New”and press ENTER. |  |
| image.png | 1. Click or tap the field to enter another query. | You can see there are a couple matches, including New York.  Next, type “Sea” to look for results in Seattle. |
| image.png | 1. Type “Sea” and press ENTER. |  |
| image.png | 1. Click or tap the **Dashboard** tab. | The company has hotels in Seattle, but no results are returned.  We already know the web app is a containerized application and the code is hosted in Azure Kubernetes Service, so let’s head to Microsoft Azure to take a look at the AKS cluster and investigate the issue. |

|  |  |  |
| --- | --- | --- |
| Investigate in Azure | Click Instructions | Talking Points |
| image.png | 1. Click or tap **sh360-cluster**. |  |
| image.png | 1. Click or tap **Health**. | The AKS Overview page for this cluster provides a quick snapshot of the cluster, but we'll need to go a little deeper.  Let’s take a look at the **Health** page. |
| image.png | 1. Click or tap to expand the **aks-agentpool …** agent pool. | To see all the nodes, expand the agent pool. |
| image.png | 1. Click or tap **Containers**. | Here, we have a holistic view of all the microservices running in this AKS cluster. At a glance, we can see how well the app’s containers, Kubernetes components, and Azure Dev Spaces containers are running.  Let’s look at our containers. |
| image.png | 1. Click or tap the **Namespace** drop-down arrow. | The bug is occurring on our production web app, which is running in the default namespace in the AKS Cluster, so let’s look at the containers located there. |
| image.png | 1. Select **default**. |  |
| image.png | 1. Click or tap the **Service** drop-down arrow**.** | We also know that the service that drives the city menu in the web app is the Hotels REST API, so let’s filter to that by using the Service menu. |
| image.png | 1. Select **hotels**. |  |
| image.png | 1. Click or tap **View Logs**. | With AKS, you can drill down into a specific container for a microservice running in production—right from the Azure portal.  Let’s view the logs for this container. |

|  |  |  |
| --- | --- | --- |
| Log Analytics | Click Instructions | Talking Points |
| image.png | 1. Click or tap to expand the **City search Sea …** log entry. | Looking at the logs, we can see that this is where the bug is originating.  Let’s look at a few of the log entries. |
| image.png | 1. Click or tap to collapse the log entry. | This log entry indicates that zero results were returned for the city search on “Sea,” providing further evidence that this container is, indeed, where the bug originated. |
| image.png | 1. Click or tap to expand the third log entry in the list. | Let’s look at another log entry that gives us additional detail. |
| image.png | 1. Click or tap to highlight “CitiesController” in the log entry. | In this entry, you can see that the exact class causing the bug is the CitiesController class. Now, we know where to look in the source code.  Finding the source of this bug took only seconds. |
| image.png | 1. Click or tap the **Query Explorer** icon. | Next, let’s see how you can use queries while investigating bugs. |

|  |  |  |
| --- | --- | --- |
| Query Explorer | Click Instructions | Talking Points |
| image.png | 1. Click or tap to expand **Saved Queries**. | For the purposes of this demonstration, we’ve already saved a few queries. |
| image.png | 1. Click or tap to expand **Demo**. |  |
| image.png | 1. Select **CPU over time**. | First, let’s run the “CPU over time” query. |
| image.png | 1. Click or tap **Run**. |  |
| image.png | 1. Select **Logs**. | This query returns a time chart showing the CPU utilization of the container over a specified time, in this case, the last 24 hours.  Next, let’s run a query to see which log entries indicate “returned 0 results.” |
| image.png | 1. Click or tap **Run**. |  |
| image.png | 1. Select **Log Chart**. | As you know, we've already found the source of the bug. This last query can offer insights into the frequency of the failing search. |
| image.png | 1. Click or tap **Run**. |  |
| image.png | 1. Click or tap anywhere to switch to Microsoft Visual Studio. | The AKS container dashboard and log search capability makes it easy to gain visibility into what's happening in each service in the cluster. Before even looking at the code, we know exactly where to look, what's happening, and how often the issue is occurring—all from the logs. Now we’re ready to review the source code. Let’s switch to Microsoft Visual Studio to do that. |

|  |  |  |
| --- | --- | --- |
| Visual Studio | Click Instructions | Talking Points |
| image.png | 1. Click or tap to expand **SmartHotel.Services.Hotels**. | This is the Solution Explorer, where you can see a one-to-one mapping of projects to services running in a cluster. This only shows the .NET Core services, but there are other services running Node.js, Java, and other languages. That's a lot of code, but luckily, we can use Visual Studio productivity features such as code navigation and search to find what we’re looking for. In this case, we’re looking for the CitiesController class. Let’s continue. |
| image.png | 1. Click or tap to expand **Controllers**. |  |
| image.png | 1. Select **CitiesController.cs**. | Here's the CitiesController class.  Now, let’s review the “Get” method to see what's wrong. |
| image.png | 1. Click or tap scroll down. |  |
| image.png | 1. Click or tap to show double-clicking **GetDefaultCities**. | You can see that the code is making two calls to the GetDefaultCities method.  Additionally, the GetDefaultCities method returns a static list, which doesn't include Seattle. This explains why it isn't showing up. |
| image.png | 1. Click or tap to highlight **Get**. | Looking higher up in the code, you can see that the “Get” method is the one that actually uses Entity Framework to query the database. |
| image.png | 1. Click or tap to return to the **CitiesController.cs** tab. | Let’s change the code. |
| image.png | 1. Click or tap to select **await \_citiesQueries.Get** in the code. |  |
| image.png | 1. Click or tap to open the context menu. |  |
| image.png | 1. Select **Copy**. |  |
| image.png | 1. Click or tap to select **\_citiesQueries …** in the code. |  |
| image.png | 1. Click or tap to open the context menu. |  |
| image.png | 1. Select **Paste**. | To fix this bug, let’s update our code to reference the correct instance of the method. |
| image.png | 1. Click or tap to add a breakpoint in the line of code. | Now that we’ve edited the code, we should debug it to make sure it works.  Let’s set a breakpoint in the relevant line of code in the controller. |
| image.png | 1. Click or tap to show right-clicking **SmartHotel.Services.Hotels**. | Traditionally, developers would need to set up databases and services, or at least get a copy of the AKS cluster on a local machine, to prevent impacting production. This could take hours, or even days, to set up. But now, you can use another Microsoft Azure solution—Azure Dev Spaces—to debug code and fix issues fast. Let’s configure the properties for this Visual Studio project. |
| image.png | 1. Select **Properties.** |  |
| image.png | 1. Click or tap the **Profile** drop-down arrow. | To specify Azure Dev Spaces as the debugging target, simply select the Azure Dev Spaces profile. |
| image.png | 1. Select **Azure Dev Spaces**. |  |
| image.png | 1. Click or tap **Change**. | A copy of the service will be deployed into AKS, and commands will be routed to that copy of the service. That way, we can debug our code changes in our space without impacting either the production environment, or the rest of the team—all of whom may have code running in their own Azure Dev Spaces. |

|  |  |  |
| --- | --- | --- |
| Azure Dev Spaces | Click Instructions | Talking Points |
| image.png | 1. Click or tap the **Space** drop-down arrow. | Next, let’s create our own Azure Dev Space for the code to run in during the debugging session. |
| image.png | 1. Select **<Create New Space…>**. |  |
| image.png | 1. Type “eve” and press ENTER. | Give the space a name. |
| image.png | 1. Click or tap **OK**. |  |
| image.png | 1. Click or tap **OK**. |  |
| image.png | 1. Click or tap to edit the URL. | Next, edit the URL.  When we put the name of the space with “.s.” ahead of the original URL, all HTTP calls to the Hotels REST API will be routed to the container running in our own personal Azure Dev Space. |
| image.png | 1. Type “eve” and press ENTER. |  |
| image.png | 1. Click or tap the **Save All** icon. | Then save the Azure Dev Space. |
| image.png | 1. Click or tap the **CitiesController.cs** tab. | Now, let’s return to the CitiesController class. |
| image.png | 1. Click or tap **Azure Dev Spaces**. | Before running the debugger, ensure that Azure Dev Spaces is selected in the debugging menu. |
| image.png | 1. Click or tap anywhere or use your right arrow key to simulate pressing F5. | Now, we’re ready to run the debugger. When we do, the app will be compiled and then built into a Docker image. That image will be published into AKS and initialized in a namespace with our Azure Dev Space name. Then, the browser will open to the public web app. The URL of the app will include a prefix, however, which will be passed through when REST API calls are made to the Hotels API.  As a result, Azure Dev Spaces will know to route traffic to the Hotels container running in our personal Dev Space, where we’ve got the debugger attached. |
| image.png | 1. Click or tap anywhere or use your right arrow key to animate the debugger. |  |
| image.png | 1. Click or tap anywhere to return to the SmartHotel360 web app. | Once debugging is done, the SmartHotel360 web app will open automatically. Let’s take a look. |

|  |  |  |
| --- | --- | --- |
| Web app | Click Instructions | Talking Points |
| image.png | 1. Click or tap in the **Where** field. | Now when we search for cities, the HTTP call will be sent to an instance of the Hotels service.  First, type “sea.” |
| image.png | 1. Type “sea” and press ENTER. |  |
| image.png | 1. Click or tap anywhere to return to Visual Studio. | Now let’s return to Visual Studio to see the code running in the debugger. |

|  |  |  |
| --- | --- | --- |
| Continue debugging | Click Instructions | Talking Points |
| image.png | 1. Click or tap to expand the node in the **Watch 1** pane. | Visual Studio should obtain focus, and the debugger should stop on the line with the breakpoint we added to the code earlier. |
| image.png | 1. Click or tap anywhere or use your right arrow key to animate the debugger. |  |
| image.png | 1. Click or tap anywhere or use your right arrow key to animate the debugger. |  |
| image.png | 1. Click or tap **Continue**. | Notice that the search scanned the database and found a result for "Seattle."  To let the code continue running beyond the breakpoint, we’ll click **Continue**. |
| image.png | 1. Click or tap anywhere to return to the SmartHotel360 web app. | Now, we should be able to see the results in the web app. |

|  |  |  |
| --- | --- | --- |
| Resolved bug | Click Instructions | Talking Points |
| image.png | 1. Click or tap anywhere to advance to the Microsoft end slide. | In just minutes, we were able to find and fix the bug. The search is working as expected, and we can commit our code. |

|  |  |  |
| --- | --- | --- |
| Call to Action | Click Instructions | Talking Points |
| image.png | 1. **End of demo**.Click or tap **Start Free** to visit the website, or press **ESC** to exit. | You’ve just seen how Microsoft Azure Kubernetes Service can help you deploy a managed Kubernetes cluster in Azure, reduce the complexity and operational overhead of managing Kubernetes, and handle critical tasks like health monitoring and maintenance. |