Cloud-native applications

Whiteboard design session student guide

November 2020

responsible for the contents of any linked site or any link contained in a linked site, or any changes or updates to such sites. Microsoft is not responsible for webcasting or any other form of transmission received from any linked site. Microsoft is providing these links to you only as a convenience, and the inclusion of any link does not imply endorsement of Microsoft of the site or the products contained therein.

Contents

Cloud-native applications whiteboard design session student guide

Abstract and learning objectives

In this whiteboard design session, you will learn about the choices related to building and deploying containerized applications in Azure, critical decisions around this, and

other aspects of the solution, including ways to lift-and-shift parts of the application to reduce applications changes.

By the end of this design session, you will be better able to design solutions that target Azure Kubernetes Service (AKS) and define a DevOps workflow for containerized applications.

Step 1: Review the customer case study

Outcome

Analyze your customer's needs.

Timeframe: 15 minutes

Directions: With all participants in the session, the facilitator/SME presents an overview of the customer case study along with technical tips.

1. Meet your table participants and trainer.
2. Read all of the directions for steps 1-3 in the student guide.
3. As a table team, review the following customer case study.

Customer situation

Fabrikam Medical Conferences provides conference web site services tailored to the medical community. They started out 10 years ago building a few conference sites for a small conference organizer. Since then, word of mouth has spread, and Fabrikam Medical Conferences is now a well-known industry brand. They now handle over 100 conferences per year and growing.

Medical conferences are typically low budget web sites as the conferences are usually between 100 to only 1500 attendees at the high end. At the same time, the conference owners have significant customization and change demands that require turnaround on a dime to the live sites. These changes can impact various aspects of the system from UI through to the back end, including conference registration and payment terms.

The VP of Engineering at Fabrikam, Arthur Block, has a team of 12 developers who handle all aspects of development, testing, deployment, and operational management of their customer sites. Due to customer demands, they have issues with the efficiency and reliability of their development and DevOps workflows.

The conference sites are currently hosted on-premises with the following topology and platform implementation:

- The conference web sites are built with the MEAN stack (Mongo, Express, Angular, Node.js).
- Web sites and APIs are built as microservices hosted on Linux servers.
- The on-prem data backend is MongoDB; also running on a separate cluster of Linux servers machines.
- There is relational data stored in PostgreSQL running on Linux servers.

Customers are considered "tenants", and each tenant is treated as a unique deployment whereby the following happens:

- Each tenant has a database in the MongoDB cluster with its own collections, and a database in PostgreSQL.
- A copy of the most recent functional conference code base is taken and configured to point at the tenant database.
    - This includes a web site code base and an administrative site code base for entering conference content such as speakers, sessions, workshops, sponsors, and session feedback from attendees.
- Modifications to support the customer's styles, graphics, layout, and other custom requests are applied.
- The conference owner is given access to the admin site to enter event details.
    - They will continue to use this admin site each conference, every year.
    - They have the ability to add new events and isolate speakers, sessions, workshops, and other details.
- The tenant's code (conference and admin web site) is deployed to a specific group of load balanced Linux servers dedicated to one or more tenant. Each group of machines hosts a specific set of tenants, and this is distributed according to scale requirements of the tenant.
- Once the conference site is live, the inevitable requests for changes to the web site pages, styles, registration requirements, and any number of custom requests begin.

Arthur is painfully aware that this small business, which evolved into something bigger, has organically grown into what should be a fully multi-tenanted application suite for conferences. However, the team is having difficulty approaching this goal. They are constantly updating the code base for each tenant and doing their best to merge improvements into a core code base they can use to spin up new conferences. The pace of change is fast, the budget is tight, and they simply do not have time to stop and restructure the core code base to support all the flexibility customers require.

Arthur is looking to take a step in this direction with the following goals in mind:

- Reduce regressions introduced in a single tenant when changes are made.
    - One of the issues with the code base is that it has many dependencies across features. Seemingly simple changes to an area of code introduce issues with layout, responsiveness, registration functionality, content refresh, and more.
    - To avoid this, he would like to rework the core code base so that registration, email notifications and templates, content and configuration are cleanly separated from each other and from the front end.
    - Ideally, changes to individual areas will no longer require a full regression test of the site; however, given the number of sites they manage, this is not tenable.
- Improve the DevOps lifecycle.
    - The time it takes to onboard a new tenant, launch a new site for an existing tenant, and manage all the live tenants throughout the lifecycle of the conference is highly inefficient.
    - By reducing the effort to onboard customers, manage deployed sites, and monitor health, the company can contain costs and overhead as they continue to grow. This may allow for time to improve the multi-tenant platform they would like to build for long-term growth.
- Increase visibility into system operations and health.
    - The team has little to no aggregate views of health across the web sites deployed.

While multi-tenancy is a goal for the code base, even with this in place, Arthur believes there will always be the need for custom copies of code for a particular tenant who

requires a one-off custom implementation. Arthur feels that Docker containers may be a good solution to support their short-term DevOps and development agility needs, while also being the right direction once they reach a majority multi-tenant application solution.

Customer needs

1. Reduce the overhead in time, complexity, and cost for deploying new conference tenants.
2. Improve the reliability of conference tenant updates.
3. Choose a suitable platform for their Docker container strategy on Azure. The platform choice should:
    - Make it easy to deploy and manage infrastructure.
    - Provide tooling to help them with monitoring and managing container health and security.
    - Make it easier to manage the variable scale requirements of the different tenants, so that they no longer have to allocate tenants to a specific load balanced set of machines.
    - Provide a vendor neutral solution so that a specific on-premises or cloud environment does not become a new dependency.
4. Migrate data from MongoDB on-premises to Azure Cosmos DB with the least changes possible to the application code.
5. Migrate relational data from PostgreSQL on-premises databases to Microsoft Azure
6. Continue to use Git repositories for source control and integrate into a CI/CD workflow.
7. Prefer a complete suite of operational management tools with:
    - UI for manual deployment and management during development and initial POC work.
    - APIs for integrated CI/CD automation.

    - Container scheduling and orchestration.
    - Health monitoring and alerts, visualizing status.
    - Container image scanning.
8. Complete an implementation of the proposed solution for a single tenant to train the team and perfect the process.
9. Enhance attendee session feedback with AI to prevent inappropriate content from being posted, and real-time language translation to better accommodate growing worldwide conference attendance.

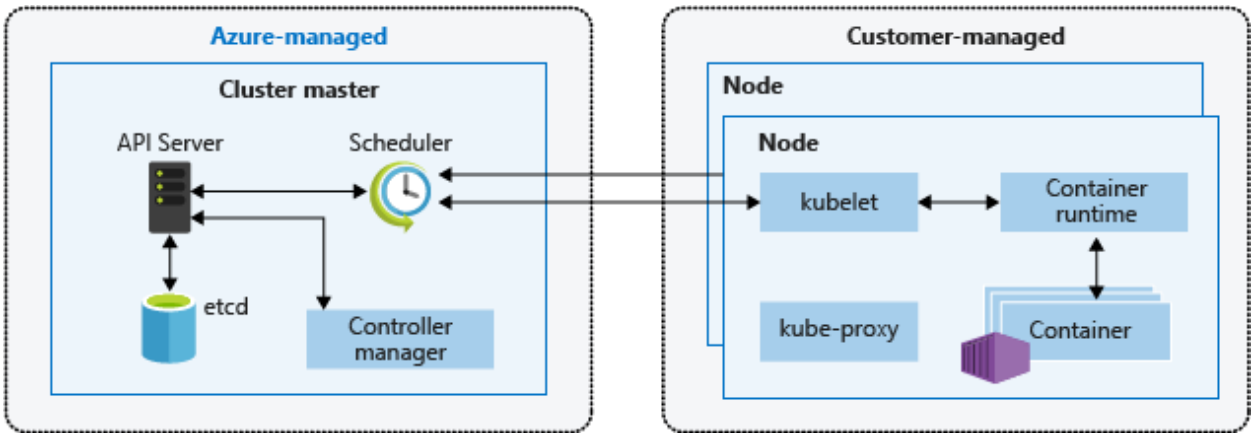
Customer objections

1. There are many ways to deploy Docker containers on Azure. How do those options compare and what are motivations for each?
2. Is there an option in Azure that provides container orchestration platform features that are easy to manage and migrate to, that can also handle our scale and management workflow requirements?
3. We heard Azure Cosmos DB is compatible with MongoDB. Will this provide a migration that minimizes code changes?
4. We know Microsoft offers Cognitive Services with pre-built AI models. What models offer the features we are looking to use for enhancing our conference web site?

Infographic for common scenarios

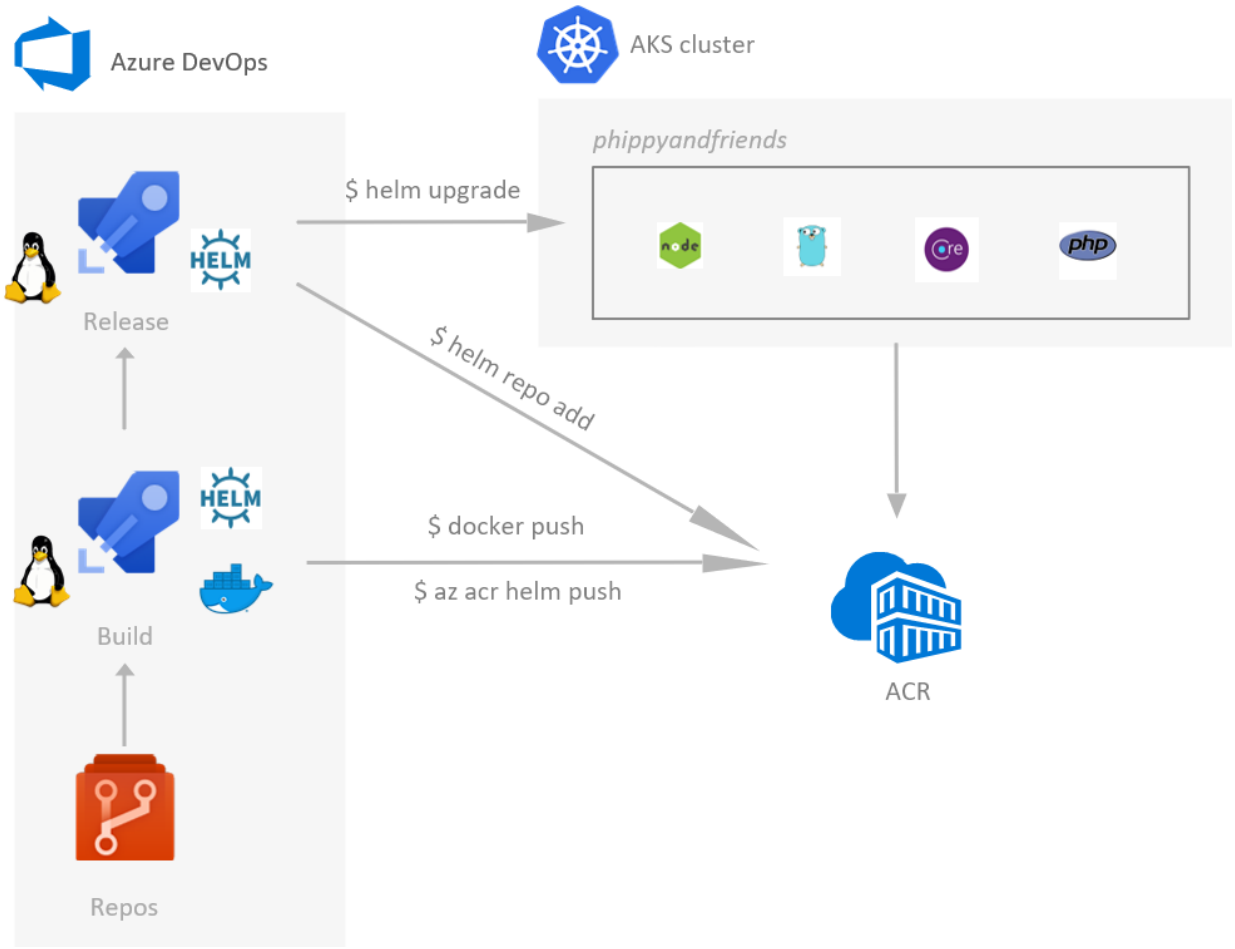
Kubernetes Architecture

Note: This diagram is an illustration of the Kubernetes topology, illustrating the master nodes managed by Azure, and the agent nodes where Customers can integrate and deploy applications.



https://docs.microsoft.com/en-us/azure/aks/intro-kubernetes

CI/CD to Azure Kubernetes Service with Azure DevOps



https://cloudblogs.microsoft.com/opensource/2018/11/27/tutorial-azure-devops-setup-cicd-pipeline-kubernetes-docker-helm/

Step 2: Design a proof of concept solution

Outcome

Design a solution and prepare to present the solution to the target customer audience in a 15-minute chalk-talk format.

Timeframe: 60 minutes

Business needs

Directions: With all participants at your table, answer the following questions and list the answers on a flip chart:

1. Who should you present this solution to? Who is your target customer audience? Who are the decision makers?

2. What customer business needs do you need to address with your solution?

Design

Directions: With all participants at your table, respond to the following questions on a flip chart:

High-level architecture

1. Based on the customer situation, what containers would you propose as part of the new microservices architecture for a single conference tenant?

2. Without getting into the details (the following sections will address the particular details), diagram your initial vision of the container platform, the containers that should be deployed (for a single tenant), and the data tier.

Choosing a container platform on Azure

1. List the potential platform choices for deploying containers to Azure.

2. Which would you recommend and why?

3. Describe how the customer can provision their Azure Kubernetes Service (AKS) environment to get their POC started.

Containers, discovery, and load balancing

1. Describe the high-level manual steps developers will follow for building images and running containers on Azure Kubernetes Service (AKS) as they build their POC. Include the following components in the summary:
    - The Git repository containing their source.
    - Docker image registry.
    - Steps to build Docker images and push to the registry.
    - Run containers using the Kubernetes dashboard.
2. What options does the customer have for a Docker image registry and container scanning, and what would you recommend?
3. How will the customer configure web site containers so that they are reachable publicly at port 80/443 from Azure Kubernetes Service (AKS)?
4. Explain how Azure Kubernetes Service (AKS) can route requests to multiple web site containers hosted on the same node at port 80/443

Scalability considerations

1. Explain to the customer how Azure Kubernetes Service (AKS) and their preconfigured Scale Sets support cluster auto-scaling.

Automating DevOps workflows

1. Describe how GitHub Actions can help the customer automate their continuous integration and deployment workflows and the Azure Kubernetes Service (AKS) infrastructure.
2. Describe the recommended approach for keeping Azure Kubernetes Service (AKS) nodes up to date with the latest security patches or supported Kubernetes versions.

Prepare

Directions: With all participants at your table:

1. Identify any customer needs that are not addressed with the proposed solution.
2. Identify the benefits of your solution.

3. Determine how you will respond to the customer's objections.

Prepare a 15-minute chalk-talk style presentation to the customer.

Step 3: Present the solution

Outcome

Present a solution to the target customer audience in a 15-minute chalk-talk format.

Timeframe: 30 minutes

Presentation

Directions:

1. Pair with another table.
2. One table is the Microsoft team and the other table is the customer.
3. The Microsoft team presents their proposed solution to the customer.
4. The customer makes one of the objections from the list of objections.
5. The Microsoft team responds to the objection.
6. The customer team gives feedback to the Microsoft team.
7. Tables switch roles and repeat Steps 2-6.

Wrap-up

Timeframe: 15 minutes

Directions: Tables reconvene with the larger group to hear the facilitator/SME share the preferred solution for the case study.

Additional references

Description	Links

Azure Kubernetes Services (AKS)	https://docs.microsoft.com/azure/aks/intro-kubernetes/
Kubernetes	https://kubernetes.io/docs/home/
AKS FAQ	https://docs.microsoft.com/azure/aks/faq
Autoscaling AKS	https://github.com/kubernetes/autoscaler
AKS Cluster Autoscaler	https://docs.microsoft.com/azure/aks/cluster-autoscaler
Upgrading an AKS cluster	https://docs.microsoft.com/azure/aks/upgrade-cluster
Azure Pipelines	https://docs.microsoft.com/azure/devops/pipelines/
Container Security	https://docs.microsoft.com/azure/container-instances/container-instances-image-security/
Image Quarantine	https://github.com/Azure/acr/tree/master/docs/preview/quarantine/
Container Monitoring Solution	https://docs.microsoft.com/azure/azure-monitor/insights/containers
Azure Cosmos DB	https://docs.microsoft.com/azure/cosmos-db/introduction
Azure Database for PostgreSQL	https://azure.microsoft.com/services/postgresql/
Azure Cognitive Services	https://azure.microsoft.com/services/cognitive-services/