Microsoft Cloud Workshop

Cloud-native applications
Whiteboard design session trainer guide
November 2019

Contents

Trainer information

Thank you for taking time to support the whiteboard design sessions as a trainer!

Role of the trainer

An amazing trainer:

- Creates a safe environment in which learning can take place.
- Stimulates the participant's thinking.
- Involves the participant in the learning process.
- Manages the learning process (on time, on topic, and adjusting to benefit participants).
- Ensures individual participant accountability.
- Ties it all together for the participant.
- Provides insight and experience to the learning process.
- Effectively leads the whiteboard design session discussion.
- Monitors quality and appropriateness of participant deliverables.
- Effectively leads the feedback process.

Whiteboard design session flow

Each whiteboard design session uses the following flow:

Step 1: Review the customer case study (15 minutes)

Outcome

Analyze your customer's needs.

- Customer's background, situation, needs and technical requirements
- Current customer infrastructure and architecture
- Potential issues, objectives and blockers

Step 2: Design a proof of concept solution (60 minutes)

Outcome

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

- Determine your target customer audience.
- Determine customer's business needs to address your solution.
- Design and diagram your solution.
- Prepare to present your solution.

Step 3: Present the solution (30 minutes)

Outcome

Present solution to your customer:

- Present solution
- Respond to customer objections
- Receive feedback

Wrap-up (15 minutes)

- Review preferred solution

Before the whiteboard design session: How to prepare

Before conducting your first whiteboard design session:

- Read the Student guide (including the case study) and Trainer guide.
- Become familiar with all key points and activities.

- Plan the point you want to stress, which questions you want to drive, transitions, and be ready to answer questions.
- Prior to the whiteboard design session, discuss the case study to pick up more ideas.
- Make notes for later.

During the whiteboard design session: Tips for an effective whiteboard design session

Refer to the Trainer guide to stay on track and observe the timings.

Do not expect to memorize every detail of the whiteboard design session.

When participants are doing activities, you can **look ahead to refresh your memory**.

- **Adjust activity and whiteboard design session pace** as needed to allow time for presenting, feedback, and sharing.
- **Add examples, points, and stories** from your own experience. Think about stories you can share that help you make your points clearly and effectively.
- **Consider creating a "parking lot"** to record issues or questions raised that are outside the scope of the whiteboard design session or can be answered later. Decide how you will address these issues, so you can acknowledge them without being derailed by them.

***Have fun**! Encourage participants to have fun and share!*

Involve your participants. Talk and share your knowledge but always involve your participants, even while you are the one speaking.

Ask questions and get them to share to fully involve your group in the learning process.

Ask first, whenever possible. Before launching into a topic, learn your audience's opinions about it and experiences with it. Asking first enables you to assess their level of knowledge and experience, and leaves them more open to what you are presenting.

Wait for responses. If you ask a question such as, "What's your experience with (fill in the blank)?" then wait. Do not be afraid of a little silence. If you leap into the silence, your participants will feel you are not serious about involving them and will become passive. Give participants a chance to think, and if no one answers, patiently ask again. You will usually get a response.

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 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 hosted on Windows Server machines.
- MongoDB is also running on a separate cluster of Windows Server machines.

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.
- 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, and sponsors.

- 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 Windows Server machines 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 flexibilities 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 CosmosDB with the least change possible to the application code.
5. Continue to use Git repositories for source control and integrate into a CICD workflow.
6. Prefer a complete suite of operational management tools with:
    - UI for manual deployment and management during development and initial POC work.
    - APIs for integrated CICD automation.
    - Container scheduling and orchestration.
    - Health monitoring and alerts, visualizing status.
    - Container image scanning.
7. Complete an implementation of the proposed solution for a single tenant to train the team and perfect the process.

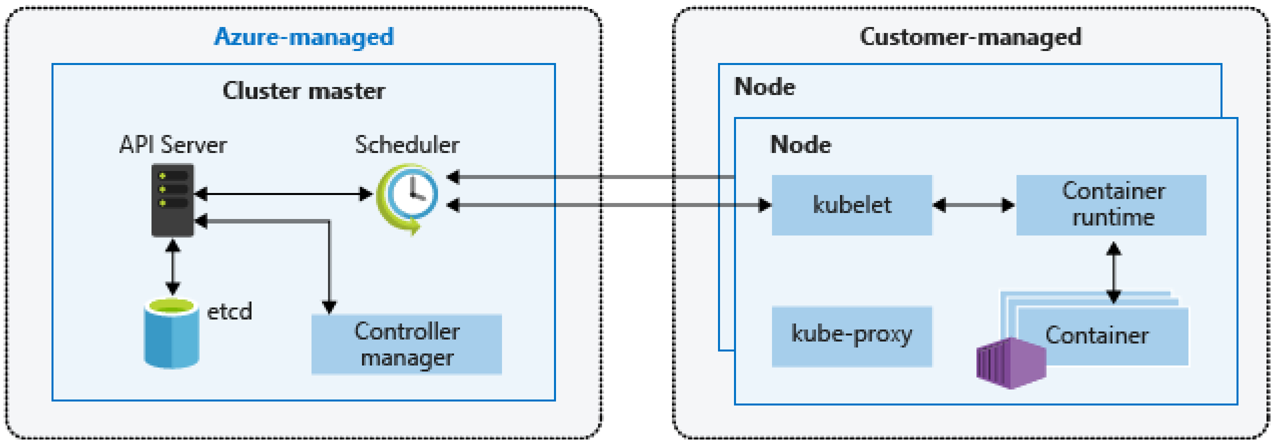
Customer objections

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?

Infographic for common scenarios

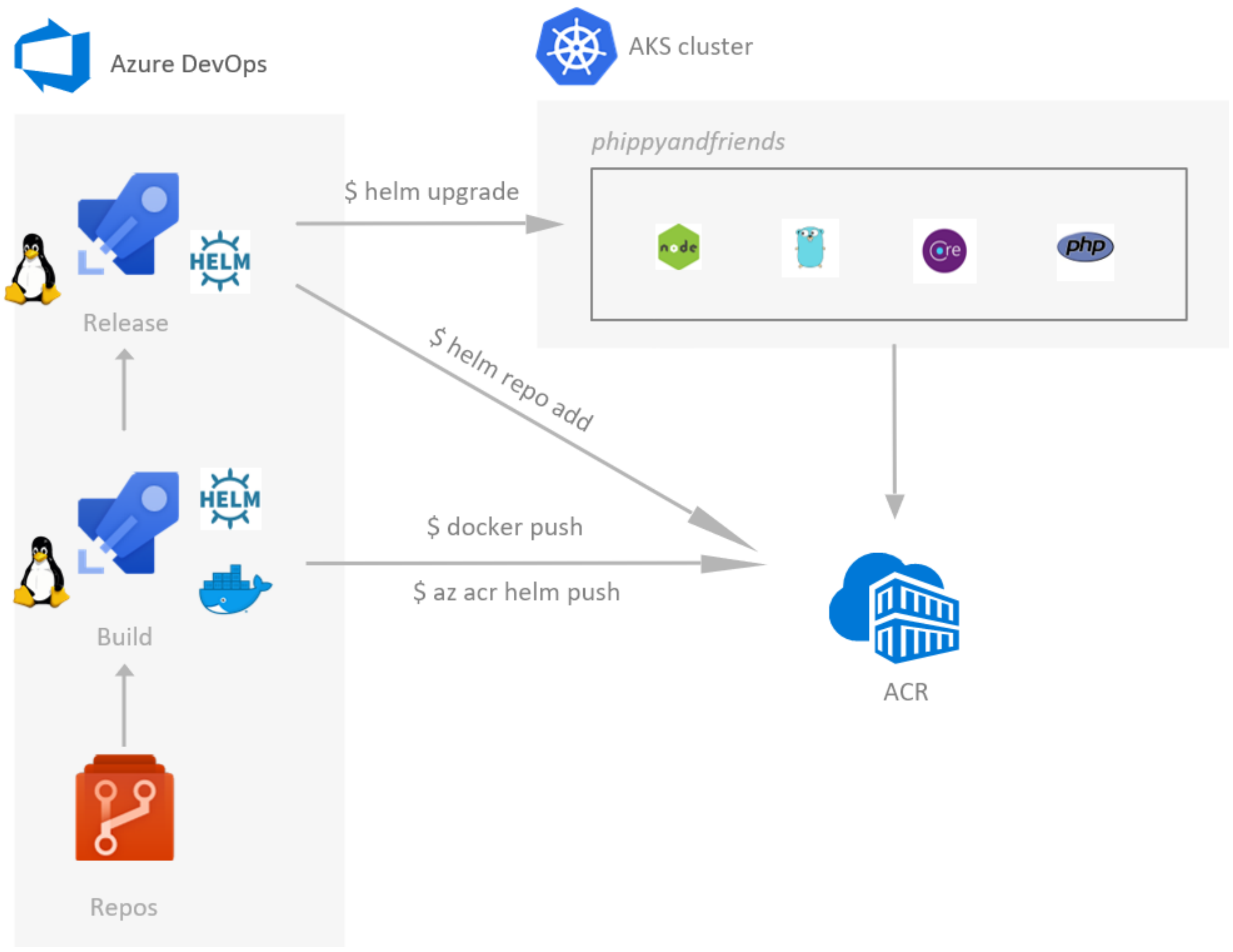
Kubernetes Architecture

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

CICD 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 Azure DevOps 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/en-us/azure/aks/intro-kubernetes/
Kubernetes	https://kubernetes.io/docs/home/
AKS FAQ	https://docs.microsoft.com/en-us/azure/aks/faq
Autoscaling AKS	https://github.com/kubernetes/autoscaler
AKS Cluster Autoscaler	https://docs.microsoft.com/en-us/azure/aks/cluster-autoscaler
Upgrading an AKS cluster	https://docs.microsoft.com/en-us/azure/aks/upgrade-cluster
Azure Pipelines	https://docs.microsoft.com/en-us/azure/devops/pipelines/
Container Security	https://docs.microsoft.com/en-us/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/en-us/azure/azure-monitor/insights/containers

Cloud-native applications whiteboard design session trainer guide

Step 1: Review the customer case study

- Check in with your table participants to introduce yourself as the trainer.
- Ask, "What questions do you have about the customer case study?"
- Briefly review the steps and timeframes of the whiteboard design session.
- Ready, set, go! Let the table participants begin.

Step 2: Design a proof of concept solution

- Check in with your tables to ensure that they are transitioning from step to step on time.
- Provide some feedback on their responses to the business needs and design.
    - Try asking questions first that will lead the participants to discover the answers on their own.
- Provide feedback for their responses to the customer's objections.
    - Try asking questions first that will lead the participants to discover the answers on their own.

Step 3: Present the solution

- Determine which table will be paired with your table before Step 3 begins.
- For the first round, assign one table as the presenting team and the other table as the customer.
- Have the presenting team present their solution to the customer team.
    - Have the customer team provide one objection for the presenting team to respond to.
    - The presentation, objections, and feedback should take no longer than 15 minutes.
    - If needed, the trainer may also provide feedback.

Wrap-up

- Have the table participants reconvene with the larger session group to hear the facilitator/SME share the following preferred solution.

Preferred target audience

Arthur Block, VP Engineering at Fabrikam Medical Conferences

The primary audience is the technical strategic decision-maker with influential solution architects, or lead technical personnel in development or operations. For this example, this could include the VP Engineering and his core team. Usually we talk to the key architects, developers, and infrastructure managers who report to the CIO or equivalent, or to key solution sponsors or those that represent the business unit IT or developers that report to those sponsors.

Preferred solution

After evaluating the options for container platforms on Azure and discussing Azure Kubernetes Service (AKS) features with the team at Microsoft, Fabrikam Medical Conferences decided to move forward with Azure Kubernetes Service (AKS).

They also decided to move forward with Azure DevOps for container DevOps workflows.

High-level architecture

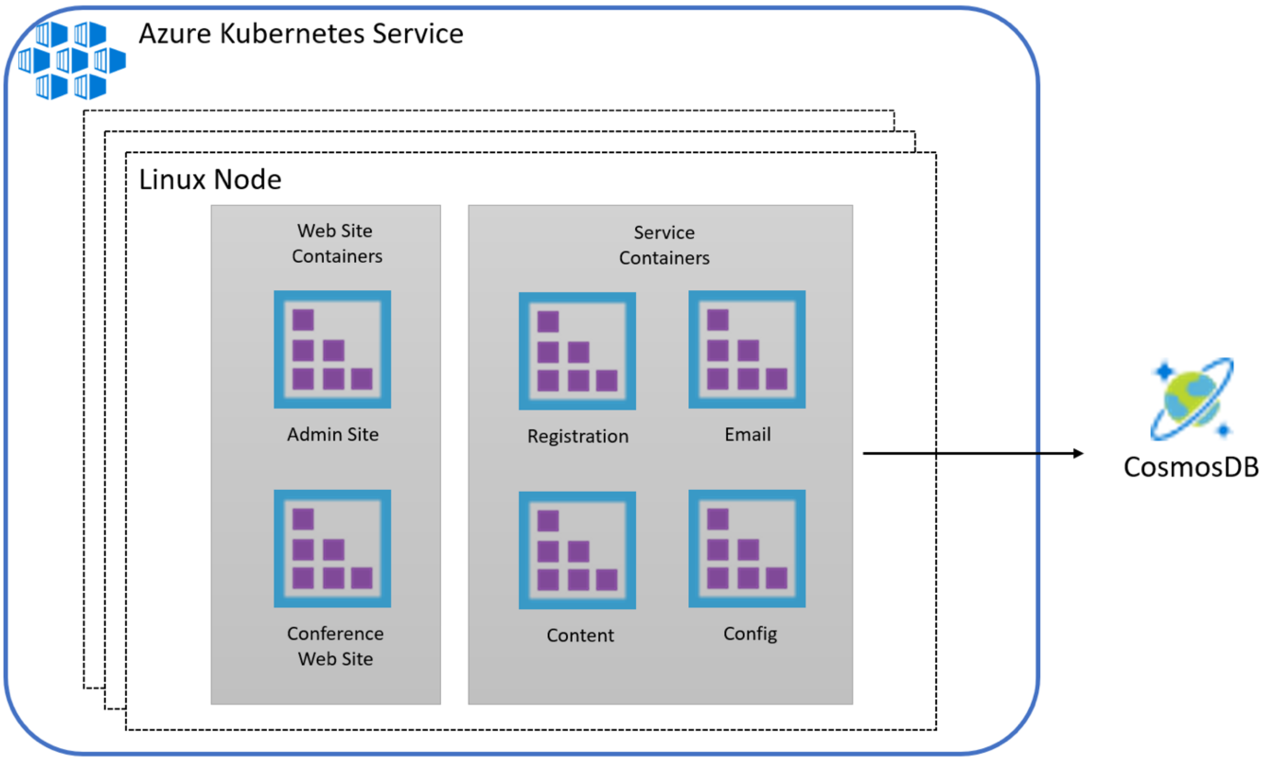
1. Based on the customer situation, what containers would you propose as part of the new microservices architecture for a single conference tenant?

Each tenant will have the following containers:

- **Conference Web site**: The SPA application that will use configuration settings to handle custom styles for the tenant.
- **Admin Web site**: The SPA application that conference owners use to manage conference configuration details, manage attendee registrations, manage campaigns, and communicate with attendees.
- **Registration service**: The API that handles all registration activities, creating new conference registrations with the appropriate package selections, and associated cost.
- **Email service**: The API that handles email notifications to conference attendees during registration, or when the conference owners choose to engage the attendees through their admin site.
- **Config service**: The API that handles conference configuration settings such as dates, locations, pricing tables, early bird specials, countdowns, and related.
- **Content service**: The API that handles content for the conference such as speakers, sessions, workshops, and sponsors.

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.

   The solution will use Azure Kubernetes Service (AKS), which means that the container cluster topology is provisioned according to the number of requested nodes. The proposed containers deployed to the cluster are illustrated below. The data tier is provided by Cosmos DB outside of the container platform:



Choosing a container platform on Azure

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

   Azure Web App for Containers

Azure Web App for Containers specifically targets container deployments, which makes it easy to run containers in a fully managed App Service Plan. This option is ideal for solutions that do not require the features offered by an orchestration platform such as Kubernetes.

Azure Container Instances

Azure Container Instances provide a serverless approach to running containers on demand and at scale enabling additional compute power and elasticity for select workloads - with the security of hypervisor isolation.

Windows Server Containers on Windows Server

Windows Server Containers allow Windows applications to be containerized. Windows Server 2016 or later versions support the installation of Docker Engine to run containers. For orchestration features you can also set up a cluster with an orchestration platform such as Docker Engine (Community or Enterprise), Kubernetes or other platforms - if you want to take responsibility for managing the clustering and related configurations.

Azure Kubernetes Service (AKS)

Azure Kubernetes Service (AKS) is the easiest way to manage a Kubernetes cluster on Azure - providing you with a managed control plane and configurable cluster with automatic updates and easy scaling capabilities. AKS removes the management overhead of container orchestration cluster, allowing teams to focus on the application and core DevOps workflows relevant to the solution.

2. Which would you recommend and why?

   Azure Kubernetes Service (AKS) is the recommended platform for the following reasons:

   - It has the necessary orchestration features without the management overhead of the control plane.
   - Ability to monitor and manage applications using a Management UI. This will also make it easier to view the overall state of all tenant applications in a single pane, and drill down into the health of an individual tenant easily.
   - Integration with Container Monitoring Solution in Azure for additional visibility into containers running in the AKS cluster from the Azure Portal, without connecting to the Kubernetes control plane.
   - Full set of integrated features, working out of the box including load balancing, service discovery, self-healing capabilities, scheduling, orchestration, task monitoring, and more.
   - Simple REST API supporting automation with DevOps workflows.
   - Open source, mature, and production tested platform.

   Generally, if the customer has experience with one of the supported orchestrators, you can apply that experience in Azure Kubernetes Service (AKS). There is a great deal of momentum in the community behind Kubernetes, and with Microsoft providing a fully managed solution based on this platform, it is the natural choice.

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

    - The Azure Kubernetes Service (AKS) environment is deployed using a few simple Azure CLI commands.

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*

    The basic workflow is to build an image from the service source repository, push the image to a registry from which it is deployed, and run as a container.

    A Dockerfile describing each container can reside in the Git repository together with the source. Using command line tools, the developers can build Docker images and push to the registry. A CI process can also automate building images and push to the registry when changes are checked in using Azure DevOps build pipelines.

    To deploy and run a container, the developer can:

    - Securely access the Kubernetes dashboard and create a deployment specifying an image from the repository manually
    - POST a service definition file (JSON) to the REST API using kubectl from the command line. This process can also be automated as part of a CD process using Azure DevOps release pipelines.
    - Create Azure DevOps CICD build and release pipelines to automate building images and deploying them to run in the cluster.

2. What options does the customer have for a Docker image registry, and what would you recommend?

    The image registry is core to the CICD workflow and must be a production worthy implementation as it is the source of container images, versioning, deployment, upgrade, and rollback strategies. Registry images can also be used for cross-environment promotion (between development, test, staging, and production for example).

    The following are a few natural options for image registries that could support Azure container deployments:

    - Azure Container Registry is a natural fit with Azure deployments, and it integrates well with deployment options previously mentioned for Docker containers in Azure. This includes an integrated experience in the Azure portal to view the repositories, images, tags, and the contents of manifests associated with an image. In addition, Azure Container Registry has new security features including image quarantine (currently in preview).

   - For development, you can also consider a public Docker Hub account. As all images in the public Docker Hub repository are public; however, this is not typically viable for corporate assets.

   - You can optionally pay for a private repository on Docker Hub, which enables you to control who can access your repository. This comes at a reasonable cost and is fully managed.

   - You can deploy and manage your own Docker Registry in Azure VMs---which would have to be clustered for high availability and this is not trivial to set up. This is not a recommended option when a hosted repository can fit solution requirements.

   Deploying and configuring a Docker Registry, clustered or not, is a complex and time-consuming task. We recommend the use of Azure Container Registry where possible for Azure solutions.

3. How will the customer configure web site containers so that they are reachable publicly at port 80/443 from Azure Kubernetes Service (AKS)?

   When you configure services for a Kubernetes deployment, you can choose to use the public load balancer such that each service instance will be accessible through the Azure load balancer. So long as the required ports are openly accessible, the Azure load balancer will be able to route traffic to all available service instances associated with the endpoint.

   Kubernetes also seamlessly supports load balanced services without making them publicly accessible. Requests from within the cluster can reach internal services and will load balanced across all service instances.

4. Explain how Azure Kubernetes Service (AKS) can route requests to multiple web site containers hosted on the same node at port 80/443

   The location of a container across all nodes in the Azure Kubernetes Service (AKS) cluster should not matter to calling clients. A client application will send a request to a particular endpoint (URL) and expect it to find the correct container instance to service the request. Container routing is an important part of this.

   Web application and api service containers bind to random ports on their host node allowing multiple instances per node. Kubernetes supports dynamic service port discovery and will choose between all instances across nodes to route requests.

Scalability considerations

1. Explain to the customer how Azure Kubernetes Service (AKS) supports cluster auto-scaling.

   You can scale the agent nodes in the cluster with the Kubernetes Autoscaler as of Kubernetes 1.10.

Automating DevOps workflows

1. Describe how Azure DevOps can help the customer automate their continuous integration and deployment workflows and the Azure Kubernetes Service (AKS) infrastructure.

With Azure DevOps you can create build definitions that, on commit or check-in can produce build artifacts from the latest source (for example) and build Docker images, then push them to a Docker image repository such as Azure Container Registry. This build definition can be configured to respond to specific folder changes, can build one or more Docker image based on different project folders, and tag images with build number, required image repository tags and other information useful to your image promotion workflows.

To trigger deployment, you can also use Azure DevOps to produce release definitions that can create or update services in AKS. You may, for example, want your development cluster to always deploy the latest images as code is committed. On the other hand, for test, UAT or production clusters you may want to manually run release jobs based on a specific image tag of the environment in order to control when a new version of a service or services are released.

2. Describe the recommended approach for keeping Azure Kubernetes Service (AKS) nodes up to date with the latest security patches or supported Kubernetes versions.

   Azure applies security patches on a nightly schedule however you must reboot the servers to apply the update. This can be done through the Azure portal or Azure CLI, for example.

   You can upgrade the cluster to a later version of Kubernetes using Azure CLI commands.

Checklist of preferred objection handling

1. There are many ways to deploy Docker containers on Azure. How do those options compare and what are motivations for each?

   The best of all worlds is to go with a managed orchestration platform like AKS -- native to Azure. It reduces the cost and management overhead of the cluster, while still providing a solution that supports growth, scale, and native management tooling.

   With Kubernetes you will have additional features at your fingertips beyond the pure Docker approach including:

   - The Kubernetes dashboard includes web interface and remote APIs for managing, running, and scaling containers, including CICD integration options.

   - The kubectl command line tool for engaging remote Kubernetes APIs and assisting with automation.

   - Built-in dynamic service discovery simplifies the deployment of new container instances to a load balanced environment.

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?

   The easiest way to move to containers on Azure is to deploy containers to the Linux variant of App Service. However, this option does not provide a full-featured container orchestration platform with highly customizable load balancing, dynamic service discovery, and a holistic approach to container monitoring.

Azure Container Instances also provide a simple way to manage individual containers without management tooling.

Azure Kubernetes Service (AKS) provides a fully managed service with the full set of orchestration and management tools. This is the best possible choice for reduced management overhead while still having access to the features provided with orchestration platforms like Kubernetes.

Customer quote (to be read back to the attendees at the end)

"With Azure Kubernetes Service (AKS) we feel confident we can make the move to a container-based platform with the right DevOps support in place to be successful with a small team."

- Arthur Block, VP of Engineering at Fabrikam Medical Conferences