Microsoft Cloud Workshop

Containers and DevOps

Whiteboard Design Session

Learner Guide

September 2017

Contents

Containers and DevOps

Step 1: Review the customer case study

Outcome

Analyze your customer's needs.

Facilitator/subject matter expert (SME) presentation of customer case study
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 leader.
2. Read all of the directions for Steps 1–3 in this attendee 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 conferences 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 in Azure 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 in Azure App Services
- MongoDB is a managed service provided by mLab on Azure

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, each 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 Web Apps in an App Service Plan
- 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 that evolved to 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 would no longer require a full regression test of the site 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
    - Be affordable, if possible with no additional licensing
4. Continue to use its managed MongoDB cluster for data storage.
5. Continue to use Git repositories for source control and desires this to be integrated into a CICD workflow.
6. As there are team members who already have familiarity with Chef they are hoping to use this with the container solution.
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 CICD automation

    - Container scheduling and orchestration
    - Health monitoring and alerts, visualizing status
8. Complete an implementation of the proposed solution for a single tenant to train the team and perfect the process.

Customer objections

1. The Docker ecosystem has a large number of open source tooling and options for things such as networking, discovery, configuration, scheduling and orchestration. We are concerned about the ability to make decisions about these tools with best fit for Azure in mind.
2. We are accustomed to working in a PaaS environment with simple DevOps workflows. In moving to Docker we know we are adding complexity but we are concerned that all Azure container offerings are IaaS-based. Will any of those offerings provide manageable DevOps solutions?

Infographic for common scenarios

Azure Container Service

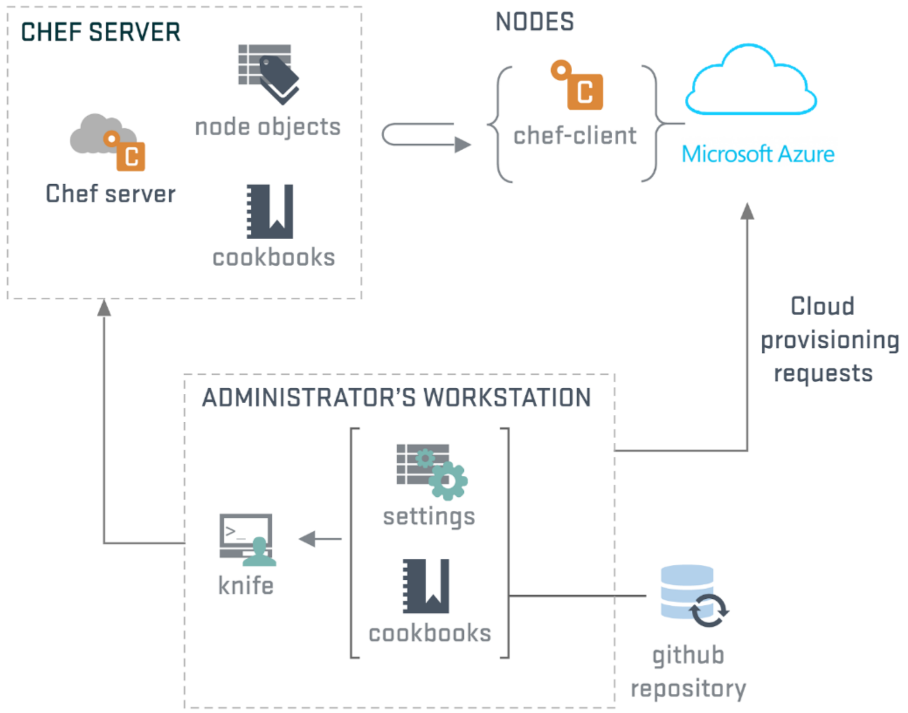
https://azure.microsoft.com/en-us/documentation/articles/container-service-intro/

Automating Azure VM deployments with chef

https://azure.microsoft.com/en-us/blog/automating-azure-virtual-machine-deployment-with-chef/

Chef has three main architectural components: Chef Server, Chef Client (node), and Chef Workstation.

- The Chef Server is our management point and there are two options for the Chef Server: a hosted solution or an on-premises solution.
- The Chef Client (node) is the agent that sits on the servers you are managing. In Azure, your server nodes will all be running this agent so they can be controlled.
- The Chef Workstation is our admin workstation where we create our policies and execute our management commands. We run the knife command from the Chef Workstation to manage our infrastructure.
- There is also the concept of “cookbooks” and “recipes.” These are effectively the policies we define and apply to our servers. These assets are typically stored with your infrastructure source control repository for version control.



Step 2: Call to action: Design a proof of concept solution

Outcome

Prepare to present a solution to the target customer audience in a 10-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 Container Service 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 Container Service as they build their POC. Include the following components in the summary:
2. What options does the customer have for a Docker image registry, 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 Container Service?
4. Explain how Azure Container Service 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 Container Service and their preconfigured Scale Sets support cluster auto-scaling.

Automating DevOps workflows

1. Describe how Chef can help the customer automate their continuous integration and deployment workflows and the Azure Container Service infrastructure.

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 for a 10-minute presentation to the customer.

Step 3: Call to action: Present the solution

Outcome

Present a solution to the target customer audience in a 10-minute chalk-talk format.

Presentation

Timeframe: 30 minutes

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

- Tables reconvene with the larger group to hear a SME share the preferred solution for the case study.

Additional references

Item	Description	Links
Marathon LB	Preferred load balancing and service discovery component for DCOS	https://github.com/mesosphere/marathon-lb
DC/OS	DC/OS overview documentation	https://docs.mesosphere.com/1.9/overview/
Chef Delivery	An overview of Chef Delivery	https://docs.chef.io/delivery_overview.html
UCP	Universal Control Plane, the enterprise grade container cluster management solution from Docker	https://docs.docker.com/datacenter/ucp/2.0/guides/
Mesos-DNS	Service discovery	http://mesosphere.github.io/mesos-dns/
Chef Containers	Documentation describing support for containers in Chef	https://docs.chef.io/containers.html