Cloud Workshops

Containers and DevOps

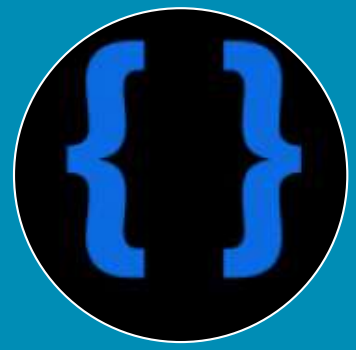
Agenda

9am	Topic Introduction – containers and devops
10am	Architecture Design Session – case study
10.30am	Morning Tea
10.45am	Architecture Design Session – continue & present
12.30pm	Lunch
1pm	Modernize apps with containers & mobile DevOps
2pm	Hands-on lab / hackathon
2.30pm	Afternoon Tea
4pm	Share learnings
4.30	Wrap-up

Microservices

Containers

Serverless



APIs

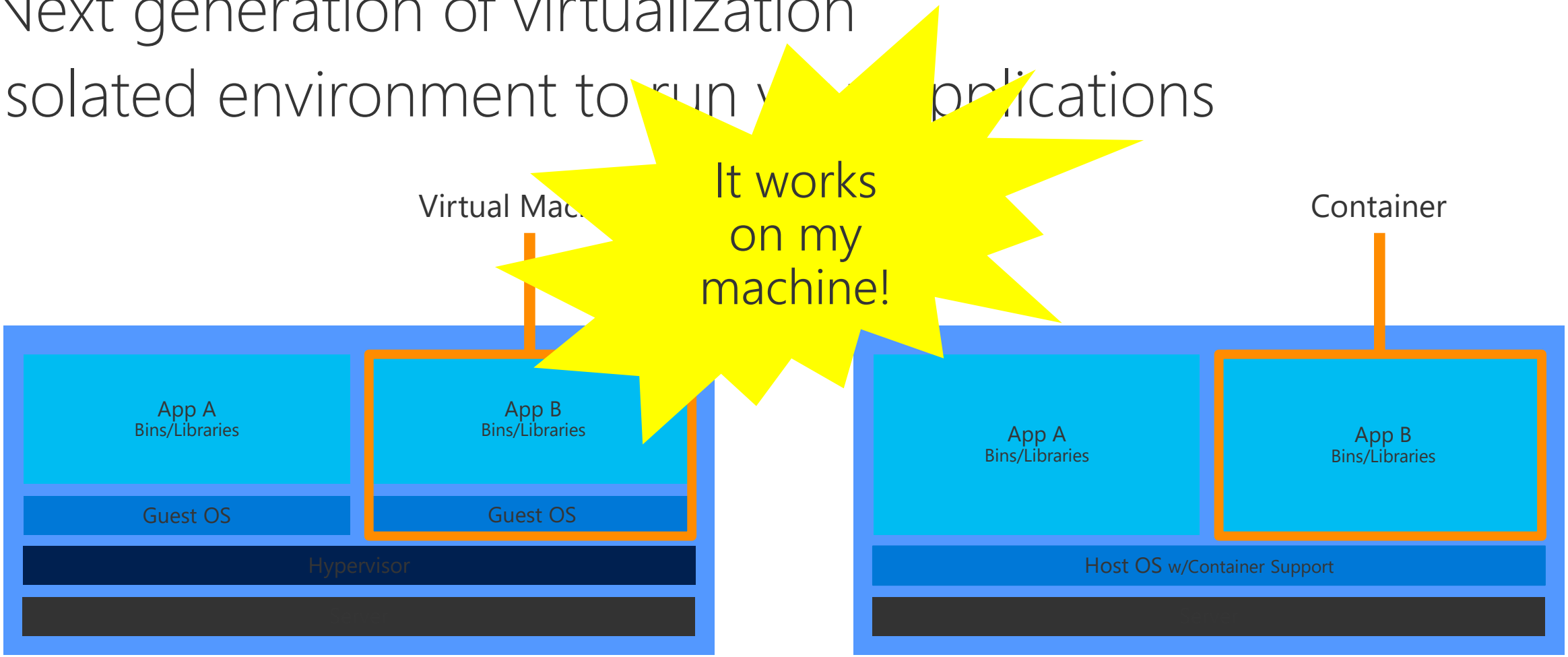
DevOps

Microservices

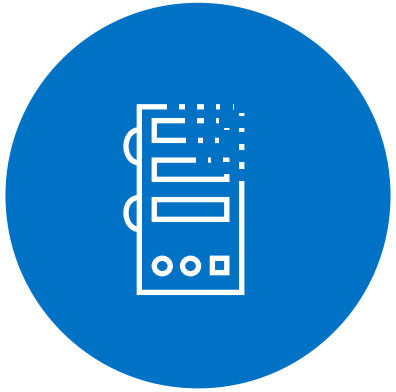
What is a container?

Next generation of virtualization

Isolated environment to run v[illegible]pplications



What is Serverless?



Manage Apps,
Not Servers

Instant, automatic
scale

Micro-billing

The rise of APIs

"In the age of the customer, every business is a digital business. This means software is central for today's enterprises, and **APIs are central for today's software**."

"... APIs play on the frontlines of business opportunity, creating new sources of revenue and market presence; they play behind the scenes, unlocking data and transactions buried in back-office systems; and they play anywhere in between."

"Sizing The Market For API Management Solutions"
Randy Heffner and Michael Yamnitsky
Forrester Research Inc.
April 2, 2015

Microservices

+

Containers

+

Serverless

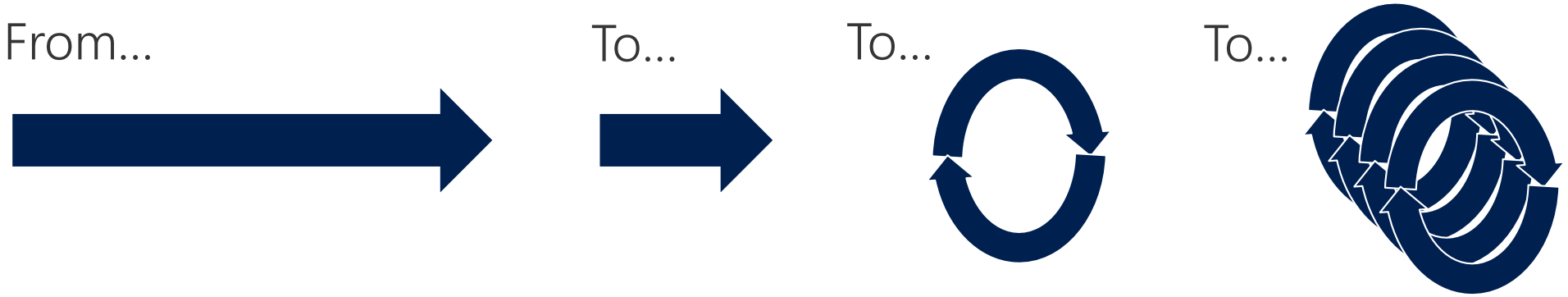
+

APIs

=

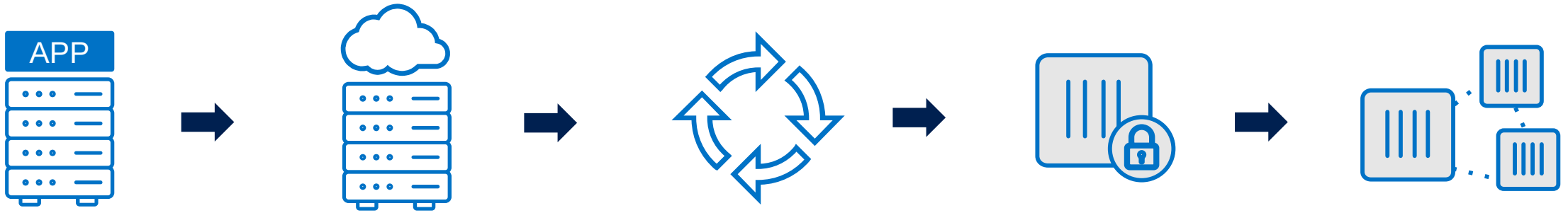
Faster Innovation

DevOps = How to manage the pace



Cultural change, organizational change
... and some tools

From traditional app to modern app



Existing Application

Modern Infrastructure

Move to the cloud as VMs or Containers or refresh HW.

Modern Methodologies

Implement CI/CD and automation.

Containerize Applications

Re-architect apps for scale with containers.

Modern Microservices

Add new services or start peeling off services from monolithic code.

Azure Microservices Options

Xamarin

Web Apps

PowerApps

API Management

ACS
Azure Container Service

Service Fabric

AKS
Azure Container Service

ACI
Azure Container Instances

App Service

Functions

Logic Apps

IaaS

Low Level PaaS

High Level PaaS

Containers

Serverless

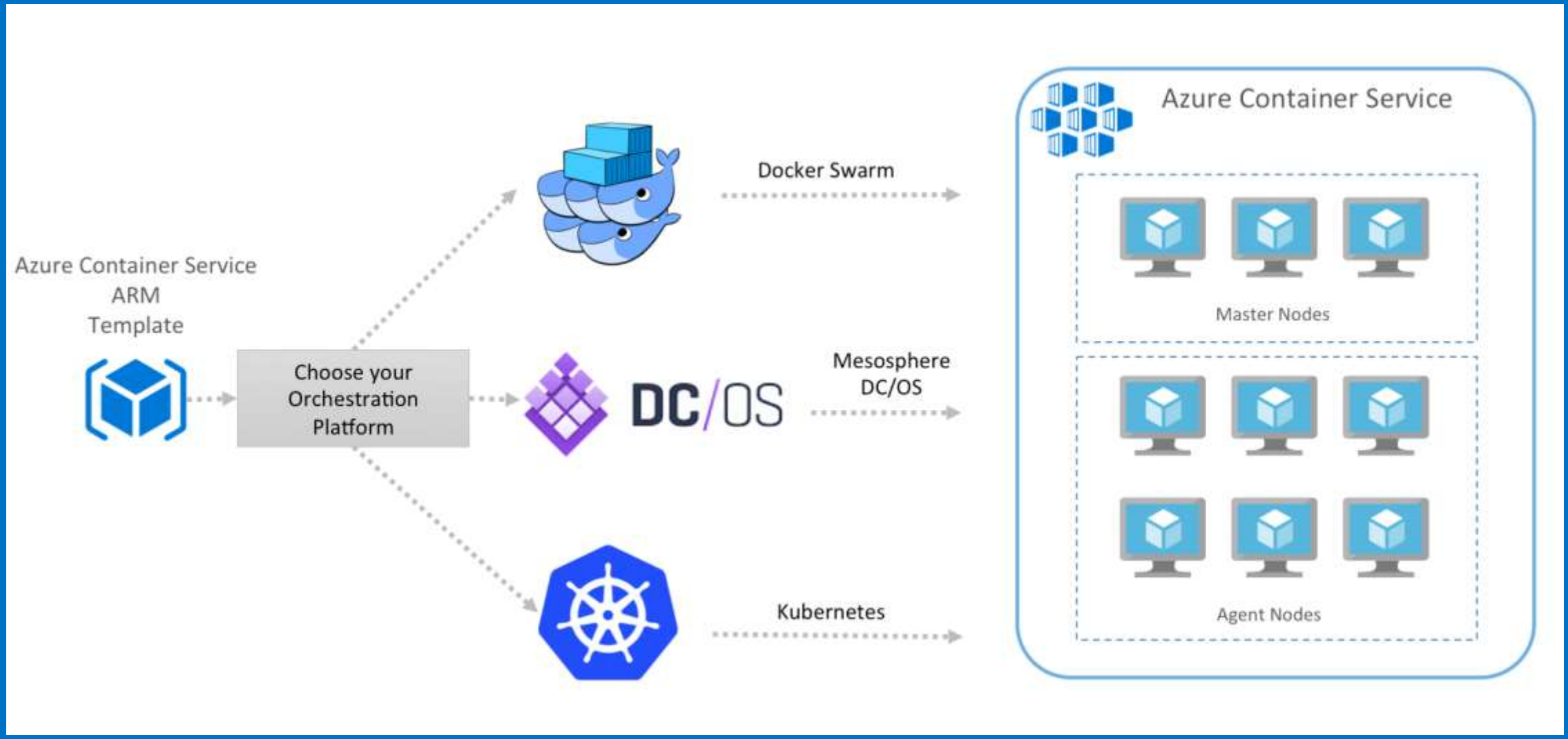
MICROSOFT CONFIDENTIAL – INTERNAL ONLY

Azure Container Service (AKS)

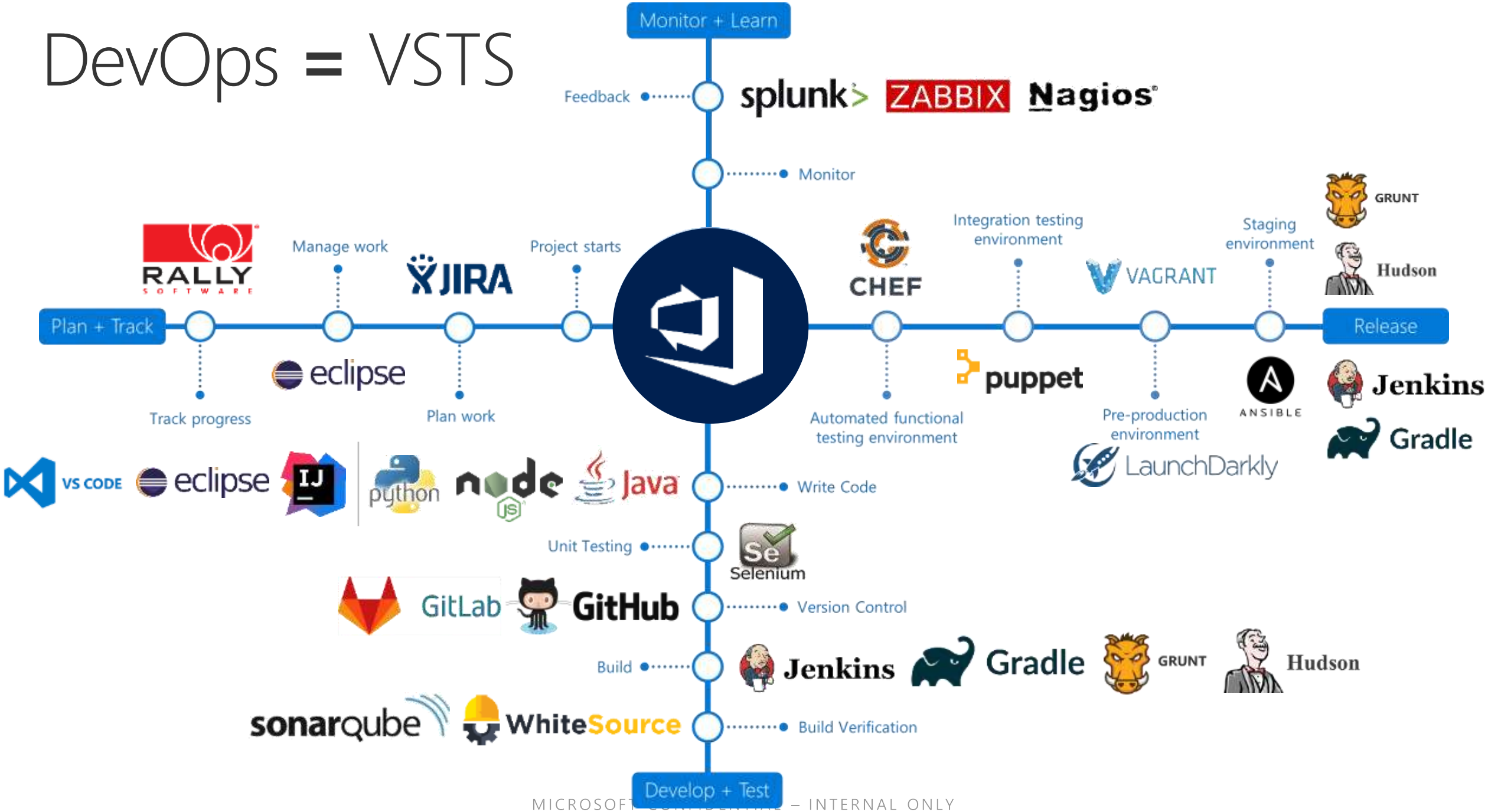
- Managed kubernetes clusters
  - Automated Kubernetes version upgrades and patching
  - Easy cluster scaling
  - Self-healing hosted control plane (masters)
  - Cost savings - pay only for running agent pool nodes
- https://docs.microsoft.com/en-us/azure/aks/intro-kubernetes
- https://docs.microsoft.com/en-us/azure/aks/tutorial-kubernetes-prepare-app

Common scenarios

Azure Container Services



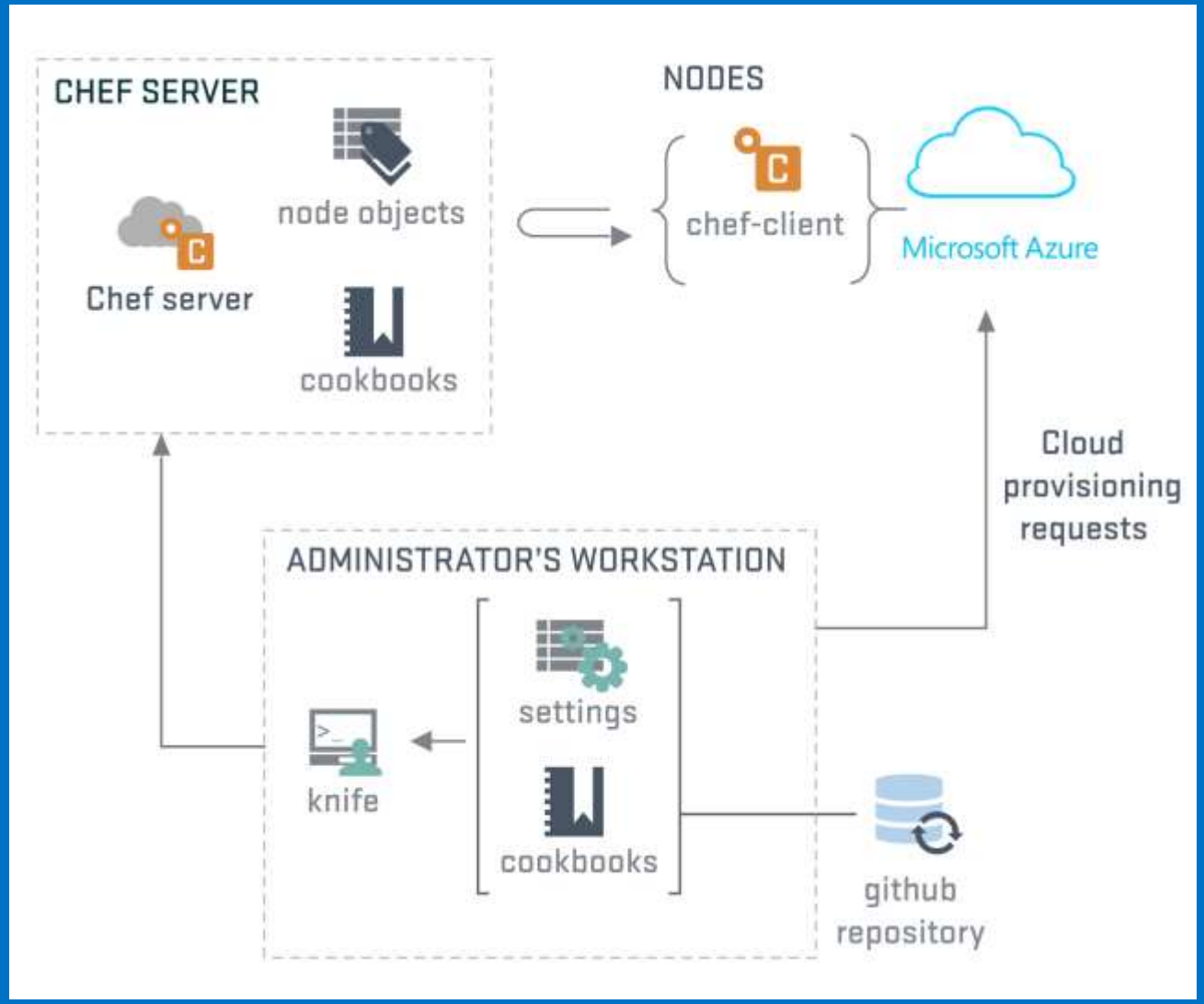
DevOps = VSTS



MICROSOFT CONFIDENTIAL – INTERNAL ONLY

Common scenarios

Chef automation with Azure



Microsoft

Step 1:
Review the customer case study

Outcome

Analyze your customer's needs

Timeframe

30 minutes

Customer situation

Fabrikam Medical Conferences

Fabrikam Medical Conferences provides conference web site services tailored to the medical community.

After starting with a few small conferences, they now have evolved into a well-known brand and handle over 100 conferences per year, and growing.

Each conference site has limited budget but the conference owners have significant customization and change demands. These changes can impact various aspects of the system from UI through to back end including conference registration and payment terms.

Customer situation

- **12 developers 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.**

Customer situation

- **The technology used is the MEAN stack (Mongo, Express, Angular, Node.js)**

- **Conference sites are currently hosted in Azure**
- **Web sites and APIs are hosted in Azure App Services**
- **MongoDB is a managed service provided by MongoLabs on Azure**

Customer situation

- Conference owners (“customers”) are considered “tenants” and each tenant is treated as a unique deployment including:
  - 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
- Modifications are then made to support the customer’s needs
- The tenant’s code is deployed to the App Service Plan (VM)
- Once the conference site is live, the inevitable requests for customizations to the deployment begins

Customer situation

- **They are looking to achieve the following:**
  - **Reduce potential regressions introduced to functional tenant code when changes are made**
  - **Ideally, changes to individual areas should not require a full regression test of the site functionality**
  - **Reduce the time to onboard new tenants**
  - **Reduce overhead managing changes, and related deployments**
  - **Improve ability to roll back and recover post change**
  - **Increase visibility into system operations and health**

Customer needs

- **Reduce the overhead in time, complexity and cost for deploying new conference tenants.**
- **Improve the reliability of conference tenant updates.**
- **Choose a suitable platform for their Docker container strategy on Azure.**
- **Continue to use its managed MongoDB cluster for data storage.**
- **Continue to use Git repositories for source control.**
- **Look at Chef as the CICD tool of choice.**
- **Leverage management tools for deployment, CICD integration, container scheduling and orchestration, health monitoring and alerts.**

- **They wish to complete an implementation of the proposed solution for a single tenant to train the team and perfect the process.**

Customer objections

- **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.**

- **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?**

http://aka.ms/container-workshop

Step 2:
Call to action: Design the 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 (10 minutes)	• Respond to questions outlined in your guide and list the answers on a flipchart.
Design (35 minutes)	• Design a solution for as many of the stated requirements as time allows. Show the solution on a flipchart.
Prepare (15 minutes)	• Identify any customer needs that are not addressed with the proposed solution. • Identify the benefits of your solution. • 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.

Timeframe
30 minutes (10 minutes each team, to present and receive feedback.)

Directions
- Pair with another table.
- One table is the Microsoft team and the other table is the customer.
- The Microsoft team presents their proposed solution to the customer.
- The customer asks one of the objections from the list of objections in the case-study.
- The Microsoft team responds to the objection.
- The customer team gives feedback to the Microsoft team. .

Wrap-up

Outcomes

- Identify the preferred solution for the case-study.
- Identify solutions designed by other teams.

Timeframe

15 minutes

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 Container Services features with the team at Microsoft, Fabrikam Medical Conferences decided to move forward with Azure Container Services and DC/OS.

They also decided to move forward with Chef for infrastructure and container DevOps workflows.

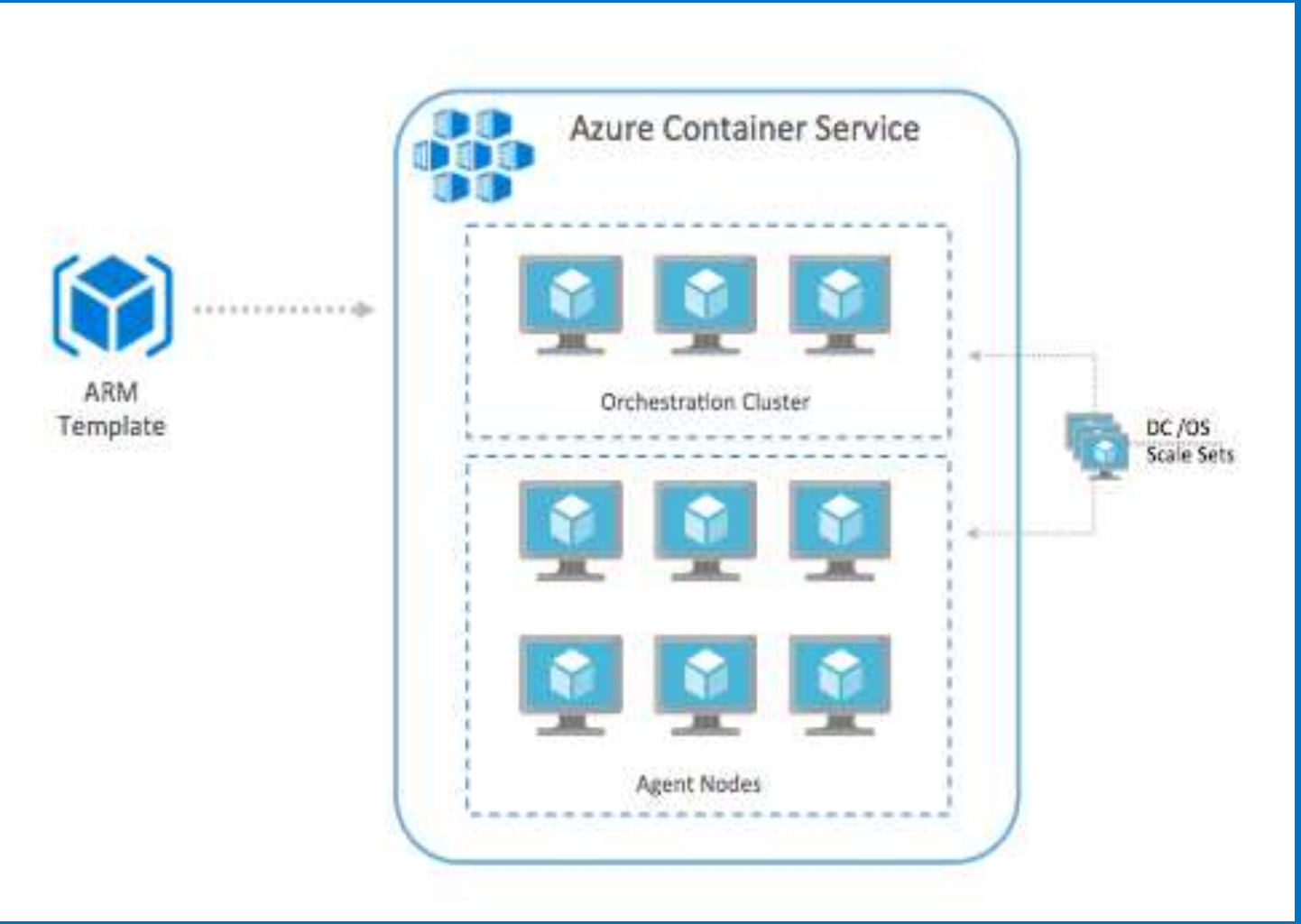
Preferred solution

High-level architecture

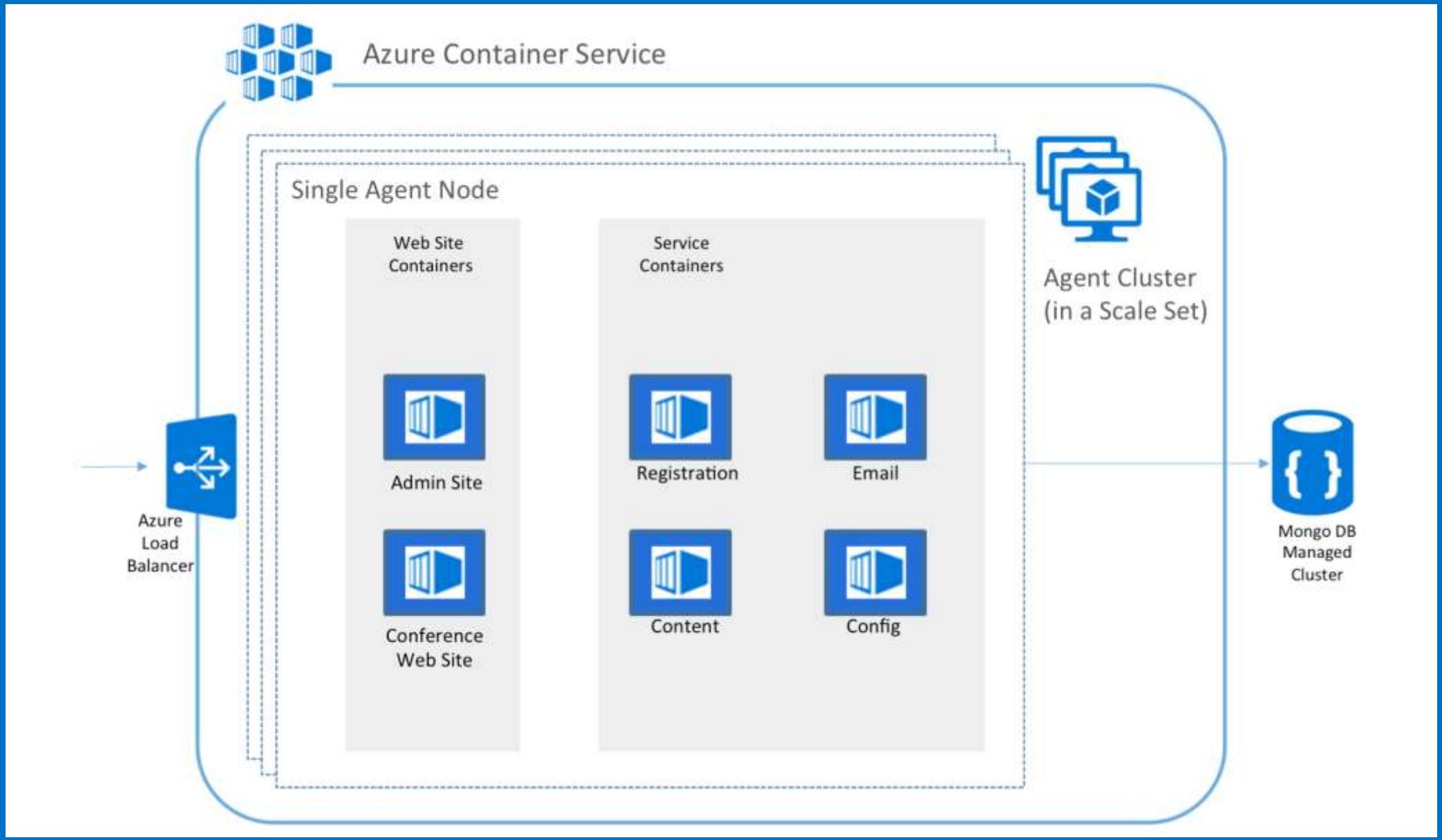
Based on the customer situation, what containers would you propose as part of the new microservices architecture for a single conference tenant?

- Each tenant will have the following containers:
  - **Conference Web site**
  - **Admin Web site**
  - **Registration service**
  - **Email service**
  - **Config service**
  - **Content service**

Preferred solution



Preferred solution



Preferred solution

Choosing a container platform on Azure

List the potential platform choices for deploying containers to Azure.

- Docker Engine with Swarm cluster running on Azure VM's
- Docker Universal Control Plan (UCP) on Azure
- Windows Containers on Windows Server 2016
- Azure Container Service with Docker Swarm Orchestration
- Azure Container Service with DC/OS Orchestration
- Azure Container Service with Kubernetes Orchestration

Preferred solution

Which would you recommend and why?

Azure Container Service with DC/OS is the recommended platform for the following reasons:

- Ability to monitor and manage applications using a Management UI
- 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
- Close working relationship between Mesosphere and Microsoft

Describe how the customer can provision their Azure Container Services environment to get their POC started.

- The Azure Container Service environment is deployed using an ARM template available in the Azure Portal.

Preferred solution

Containers, discovery, and load balancing

Describe the high level manual steps developers will follow for building images and running containers on Azure Container Services as they build their POC. Include the following components in the summary:

- *The Git repository containing source code*
- *Docker image registry*
- *Steps to build docker images and push to the registry*
- *Run containers using the Marathon UI*

- 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.

- To deploy and run a container the developer can either securely access the DC/OS Console and launch a service from the repository manually or POST a service definition file (JSON) to the REST API using a command line tool such as curl. This process can also be automated as part of a CICD process.

Preferred solution

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

Options:

- Public Docker Hub account
- Private repository hosted on Docker Hub
- Azure Container Registry
- Run your own Docker Registry cluster in Azure VMs
- Deploy Docker Registry containers to your DC/OS cluster

Deploying and configuring a Docker Registry, clustered or not, is a complex and time consuming task. We recommend the use of Azure Registry now that it is available in Preview, to allow for a fully managed image registry.

Preferred solution

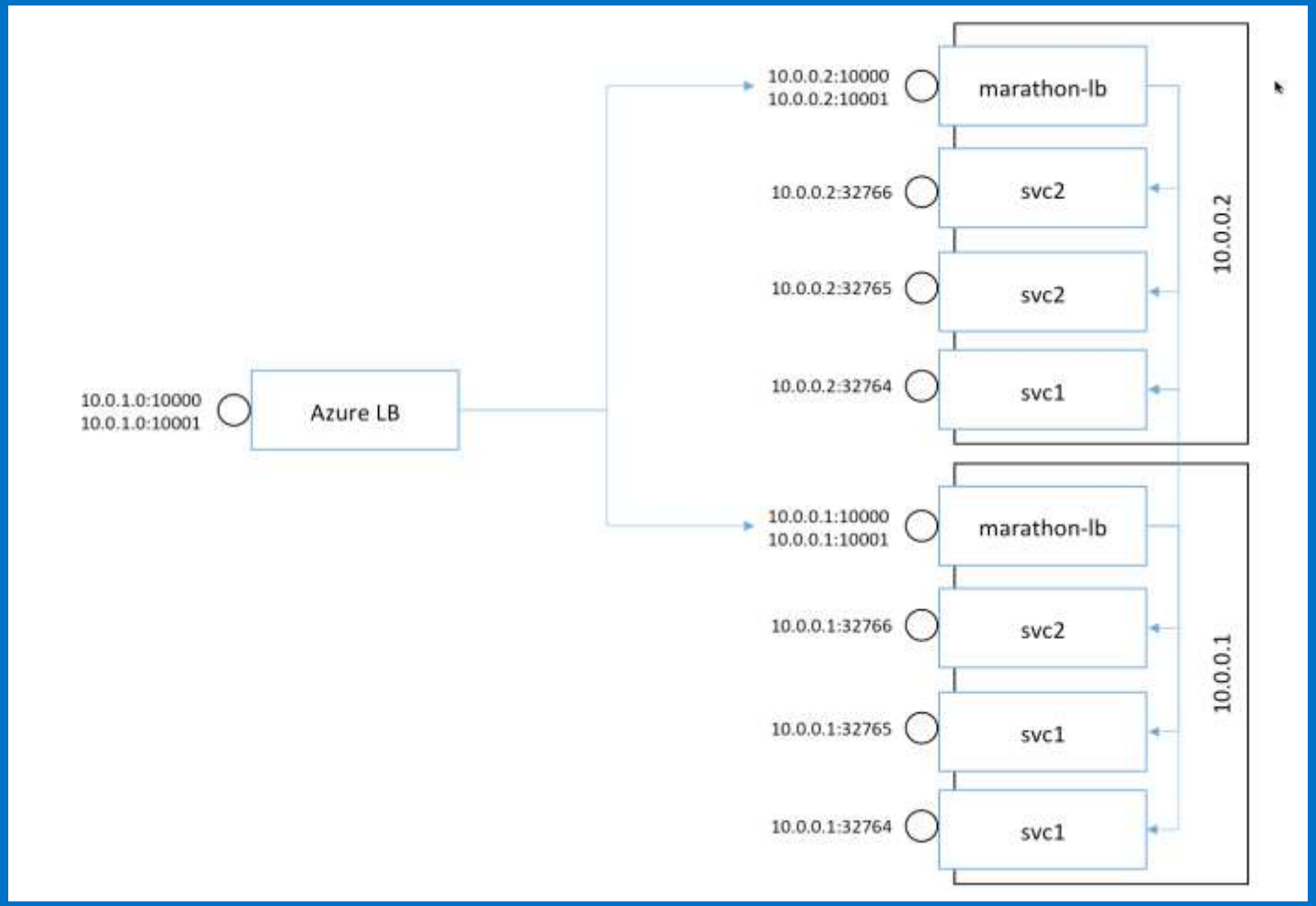
How will the customer configure website containers so that they are reachable publicly at port 80/443 from Azure Container Service?

- The Marathon configuration file for each web site will include the public DNS name. Incoming traffic will pass through an Azure Load Balancer and route to Marathon Load Balancer (LB) which binds to port 80/443. Marathon LB in turn will route requests to the web containers based on the HTTP host header.

Explain how Azure Container Services can route requests to multiple web site containers hosted on the same node at port 80/443.

- Website and service containers bind to random ports on their host node, allowing multiple instances per node. Marathon Load Balancer (marathon-lb) dynamically updates its routing table in response to changes to the set of running containers. The Azure Load Balancer sends all requests to Marathon Load Balancer.

Preferred solution



Preferred solution

Scalability considerations

Explain to the customer how Azure Container Services and their preconfigured Scale Sets support cluster auto-scaling.

The agents in the Azure Container Service are defined with a VM scale set. You can scale in a few ways:

- Reapply the ARM template used to create the Azure Container Service cluster and specify a new target value for the number of agents.

- You can auto-scale if you configure the VM scale set nodes with the Azure Diagnostics Extension. Auto-scale can be configured to react to metrics collected by the nodes in the cluster.

Preferred solution

Automating DevOps workflows

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

Chef Automate contains a powerful and flexible continuous integration and deployment workflow engine. Workflows can be created for applications and the infrastructure itself. Some examples:

- Continuous integration and deployment of the application.
- Automate updates to the Azure Container Service virtual machines.
- Scale the number of agent nodes in the Azure Container Service cluster.

Preferred objections handling

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.

Azure Container Service provides core preconfigured options for container-based solutions:

- Docker Swarm and Compose
- Marathon and DC/OS
- Kubernetes

Although you can optionally add or modify this open source tooling on the Scale Sets provisioned for Azure Container Service, the combination of tools supplied with the preconfigured VMs provide you with a baseline that you can work with immediately.

Preferred objections handling

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?

- Azure Container Service provides a set of preconfigured Virtual Machines ready for you to deploy container-based solutions—simplifying DevOps.
- Beyond simplifying cluster management, there are also options for simplifying how you manage your container-based solution.
  - You can choose the container scheduling and orchestration tooling of your choice when you select Azure Container Service;
  - The Mesosphere DC/OS options provides you with additional features beyond the pure Docker approach, including a rich management interface for configuration, management and visibility

Customer quote

"With Azure Container Services and DC/OS 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

Hackathon – Your choice

- Choose a project you're passionate about, and go do it!
- Follow step-by-step through a lab implementing DC/OS on Azure using ACS
  - http://aka.ms/container-workshop
- Change-up the lab, by implementing using AKS instead
  - http://aka.ms/aks-tutorial