Microsoft

Microsoft Build

May 7–9, 2018 // Seattle, WA

Microsoft

Best Practices with Azure & Kubernetes

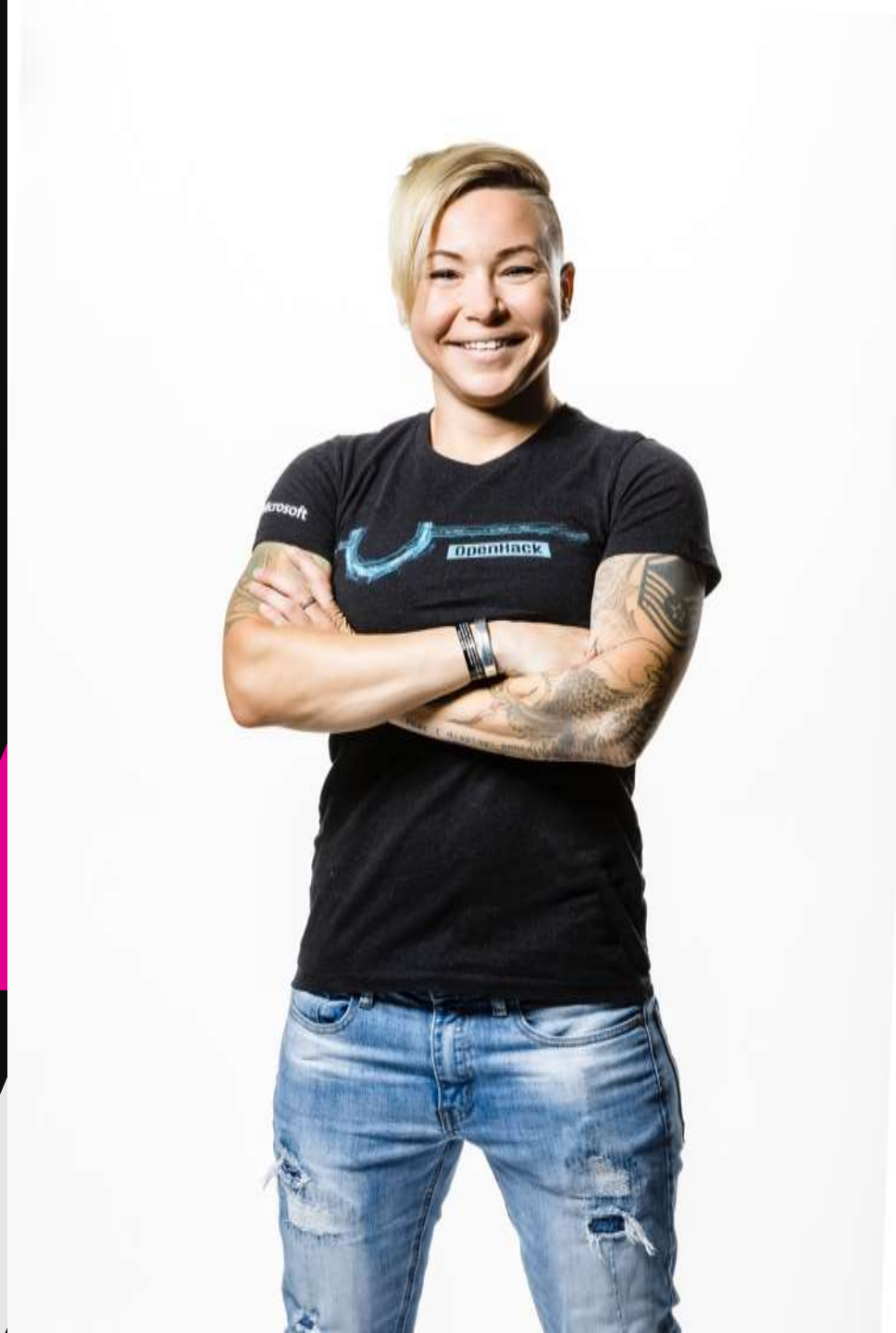
Jessica Deen
Abel Wang

BRK3701

Who am I?

- Linux, Open Source, Containers, IT/Ops
- CrossFit
- HUGE Disney and Star Wars fan
- League of Extraordinary Cloud DevOps Advocates

@jldeen
GitHub | Twitter | Instagram

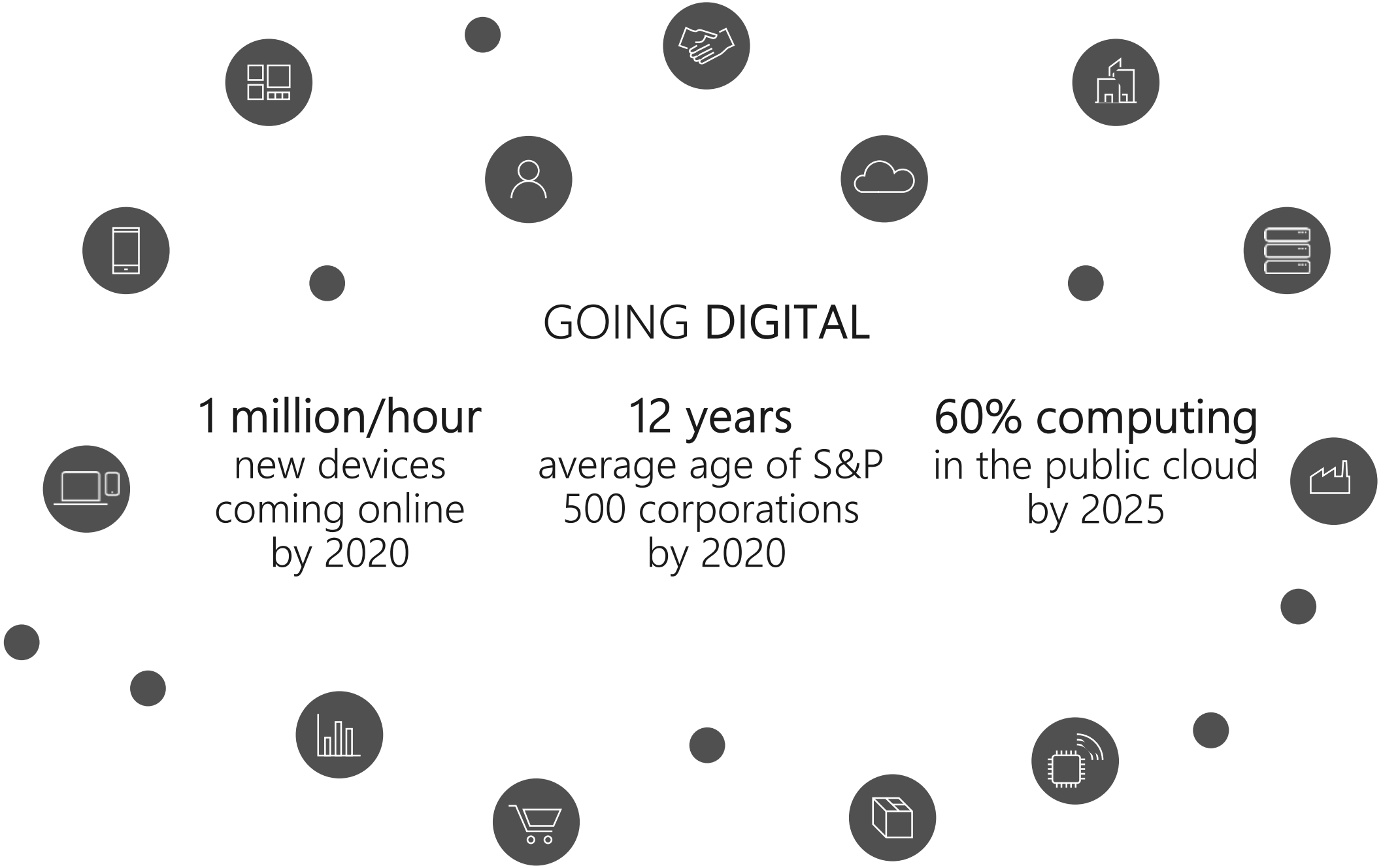


#LoECDA

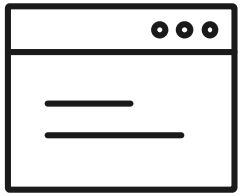
Cloud Developer Advocates
#azureavengers | @azureadvocates

Agenda

- DevOps Overview
- Container Orchestration
- Kubernetes
- Helm
- Demo

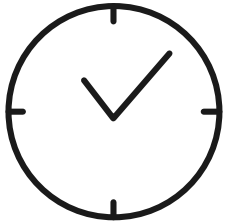

GOING DIGITAL
1 million/hour
new devices
coming online
by 2020
12 years
average age of S&P
500 corporations
by 2020
60% computing
in the public cloud
by 2025

What we hear from developers



I need to create applications at a competitive rate without worrying about IT

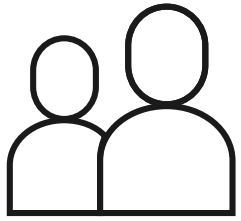
New applications run smoothly on my machine but malfunction on traditional IT servers



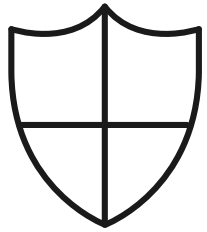
My productivity and application innovation become suspended when I have to wait on IT

What we hear from IT

I need to manage servers and maintain compliance with little disruption



I’m unsure of how to integrate unfamiliar applications, and I require help from developers

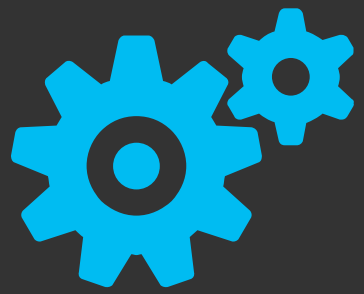


I’m unable to focus on both server protection and application compliance

IT stress points

Security threats

Datacenter efficiency



Supporting innovation

Cloud is a new way to think about a datacenter

Traditional model

Dedicated infrastructure for each application

Purpose-built hardware

Distinct infrastructure and operations teams

Customized processes and configurations

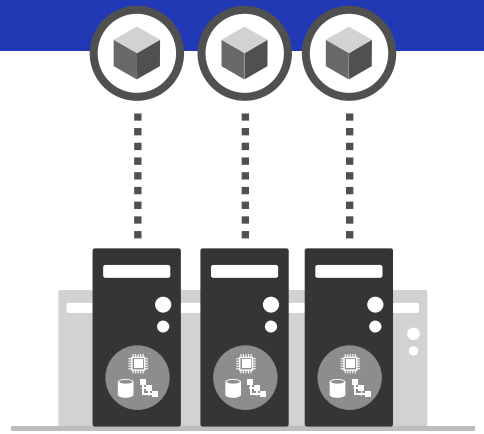
Cloud model

Loosely coupled apps and micro-services

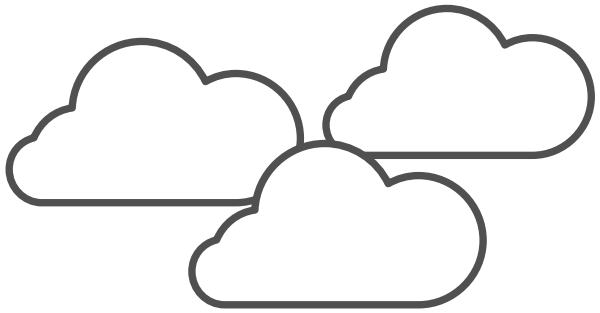
Industry-standard hardware

Service-focused DevOps teams

Standardized processes and configurations



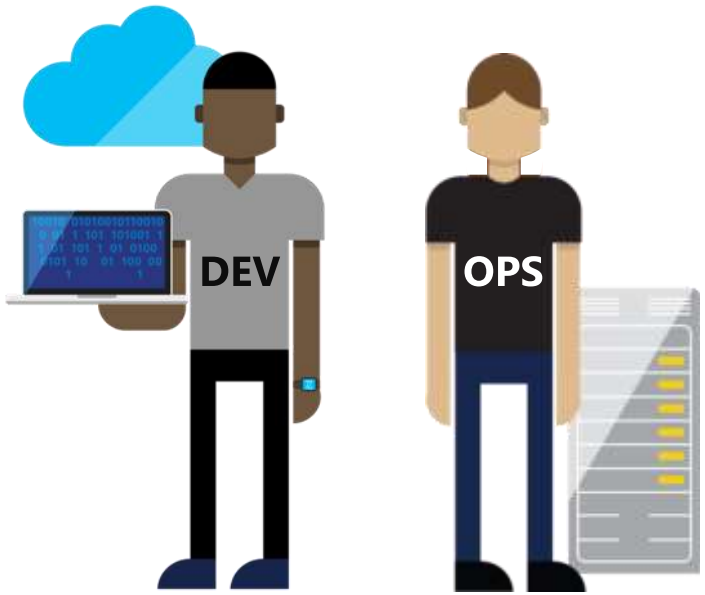
Servers

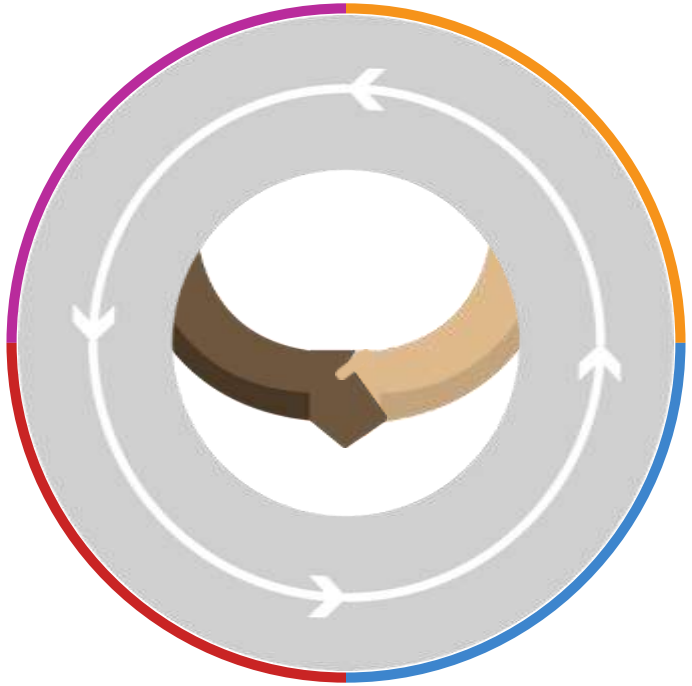


Services

Microsoft

DevOps: the three stage conversation





1	People	2	Process	3	Products

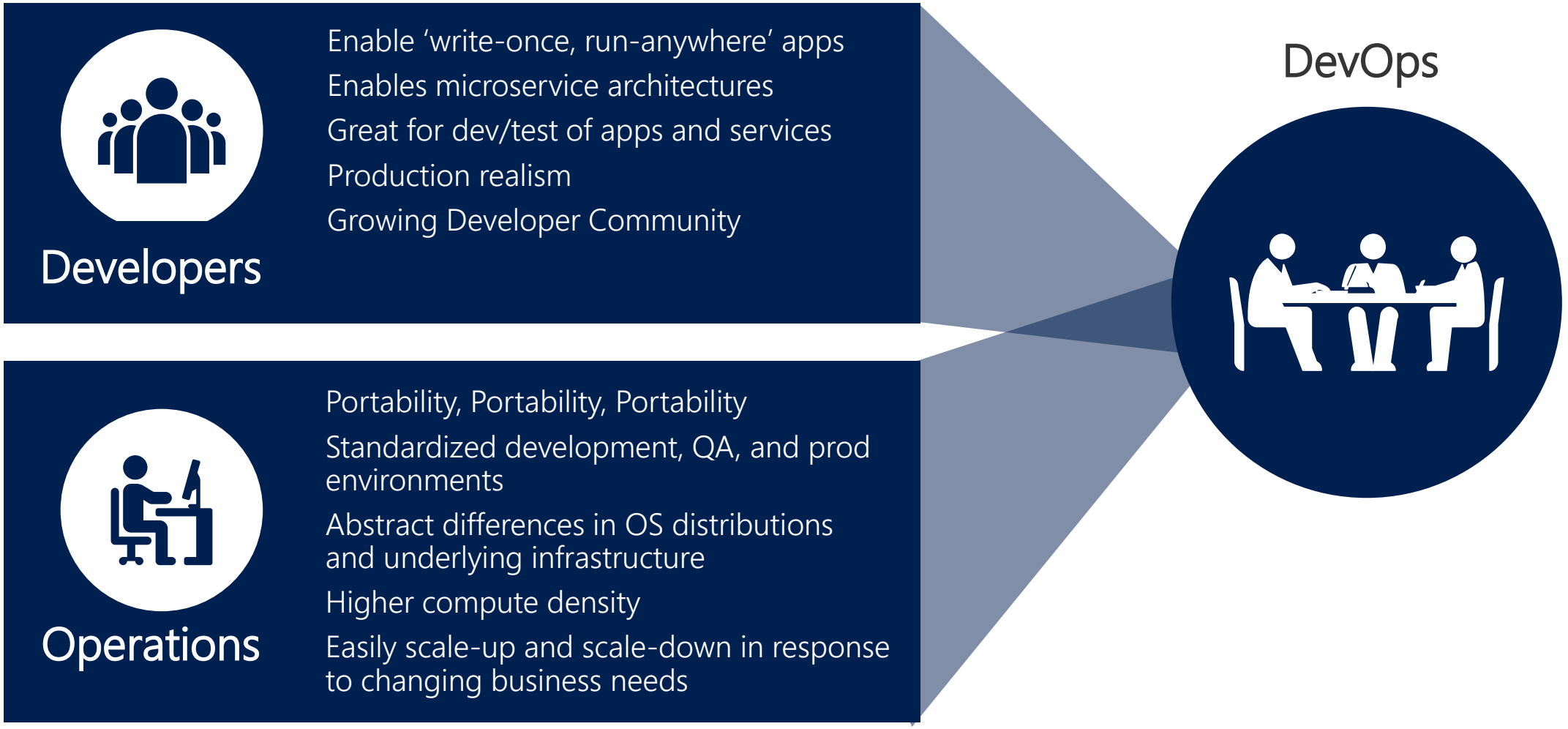
DevOps is the union of people, process, and products to enable continuous delivery of value to our end users.

- Donovan Brown

http://bit.ly/WhatIs-DevOps


The Open Container Certification Platform
anchore
Azure
HELM
Azure
Azure
Azure
Azure

Why Containers?



Container Orchestration

The elements of orchestration

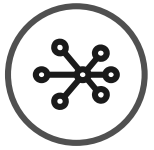
Scheduling

Affinity/anti-affinity

Health monitoring

Failover

Scaling



Networking

Service discovery

Coordinated app upgrades

Kubernetes: the de-facto orchestrator

Portable

Public, private, hybrid, multi-cloud

Extensible

Modular, pluggable, hookable, composable

Self-healing

Auto-placement, auto-restart, auto-replication, auto-scaling

Kubernetes: empowering you to do more

- Deploy your applications quickly and predictably
- Scale your applications on the fly
- Roll out new features seamlessly
- Limit hardware usage to required resources only

Container Orchestration: Kubernetes

Azure Container Service (AKS)

Your Kubernetes cluster, managed by Azure

Why AKS?

- Easy to use
  - Fastest path to Kubernetes on Azure
  - Up and running with 3 simple commands
- Easy to manage
  - Automated upgrades and patching
  - Easily scale the cluster up and down
  - Self-healing control plane
- Uses Open APIs
  - 100% upstream Kubernetes

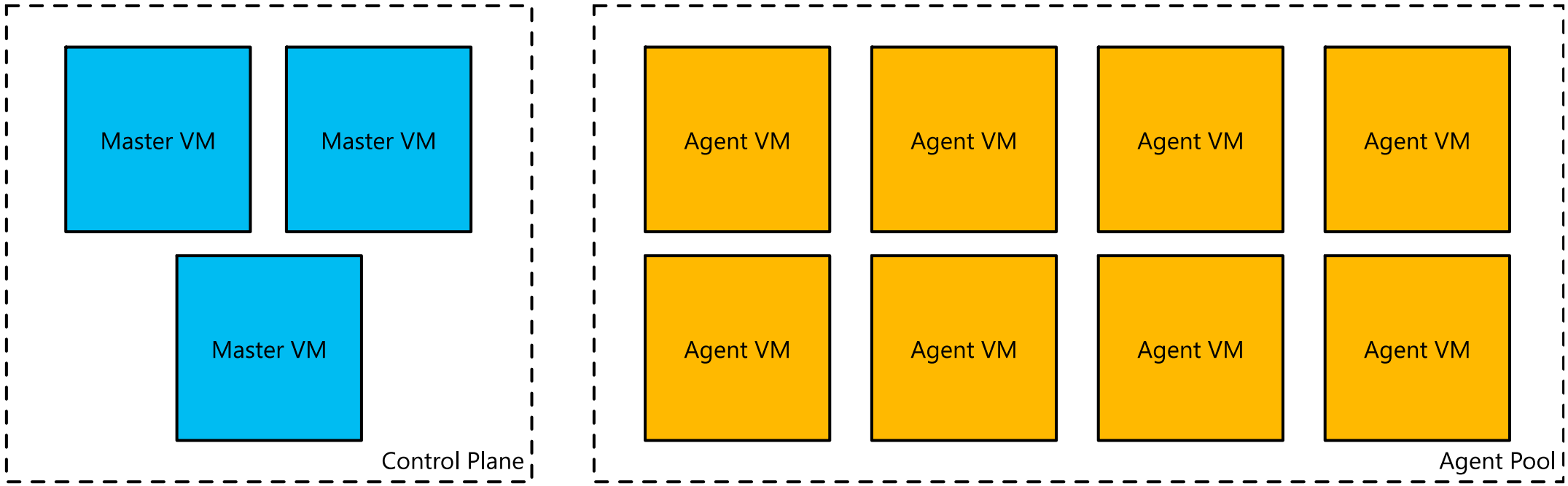
AKS in Action – Creating a K8s Cluster

```
$ az aks create –g <resourceGroupName> -n
<k8sClusterName> --generate-ssh-keys
\ Running ..

$ az aks install-cli
Downloading client to /usr/local/bin/kubectl ..

$ az aks get-credentials –g <resourceGroupName> -n
<k8sClusterName>
Merged “<k8sClusteName>” as current context ..
```

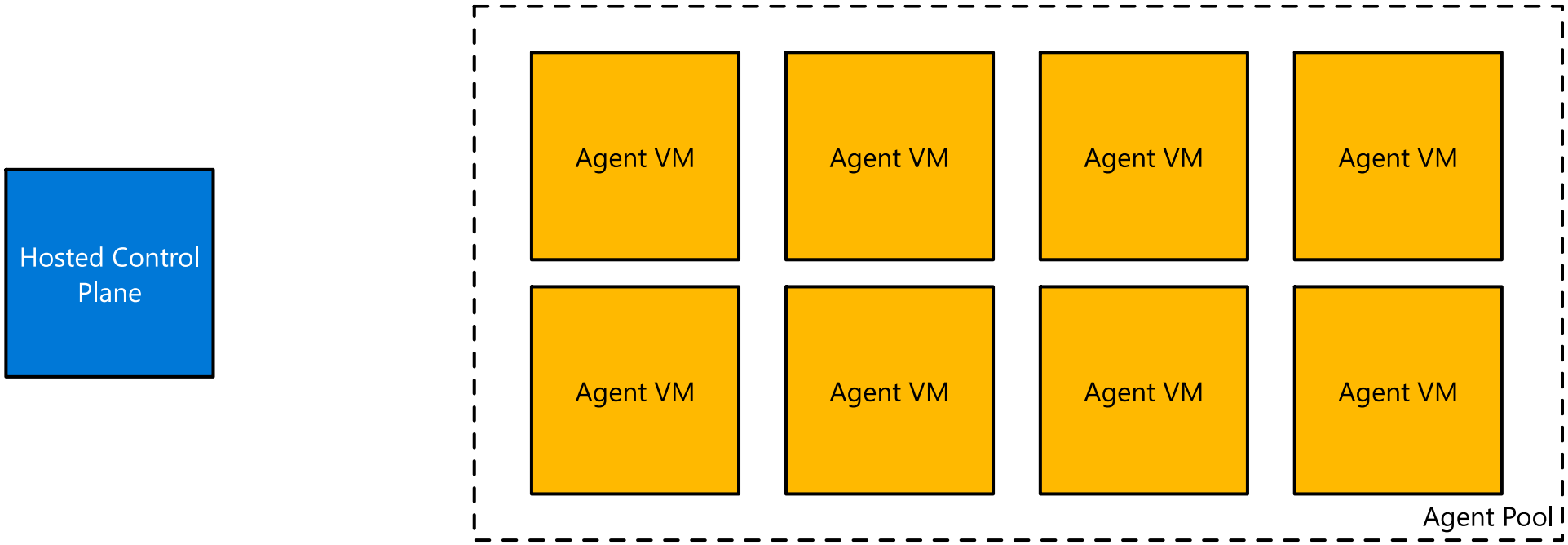
AKS in Action – Managing an AKS Cluster

```
$ az aks upgrade –g <resourceGroupName> -n
<k8sClusterName> --kubernetes-version 1.8.1
\ Running ..

$ az aks scale –g <resourceGroupName> -n <k8sClusterName>
--agent-count 6
| Running ..
```

Kubernetes without AKS



Kubernetes with AKS



Azure Container Service (AKS)

Azure Container Instances (ACI)

Azure Container Registry

Open Service Broker API (OSBA)

Release Automation Tools

Release automation tools

Simplifying the Kubernetes experience

Streamlined Kubernetes development

The package manager for Kubernetes

Event-driven scripting for Kubernetes

Visualization dashboard for Brigade

Azure Container Service (AKS)

Azure Container Instances (ACI)

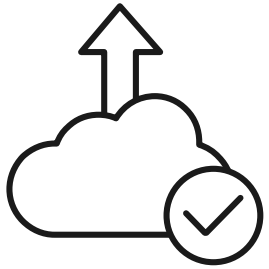
Azure Container Registry

Open Service Broker API (OSBA)

Release Automation Tools

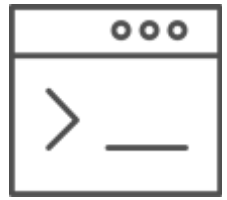
Draft

Simple app development and deployment – into any Kubernetes cluster



Simplified development

Using two simple commands, developers can now begin hacking on container-based applications without requiring Docker or even installing Kubernetes themselves



Language support

Draft detects which language your app is written in, and then uses packs to generate a Dockerfile and Helm Chart with the best practices for that language

Demo – Draft & Helm Collaboration

Azure Container Service (AKS)

Azure Container Instances (ACI)

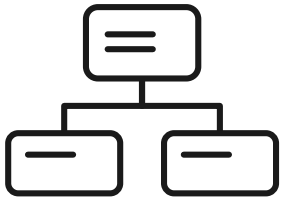
Azure Container Registry

Open Service Broker API (OSBA)

Release Automation Tools

Helm

The best way to find, share, and use software built for Kubernetes



Manage complexity

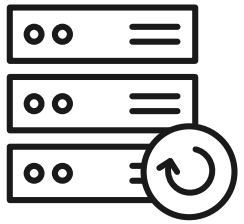
Charts can describe complex apps; provide repeatable app installs, and serve as a single point of authority

Easy updates

Take the pain out of updates with in-place upgrades and custom hooks

Simple sharing

Charts are easy to version, share, and host on public or private servers



Rollbacks

Use `helm rollback` to roll back to an older version of a release with ease

Azure Container Service (AKS)

Azure Container Instances (ACI)

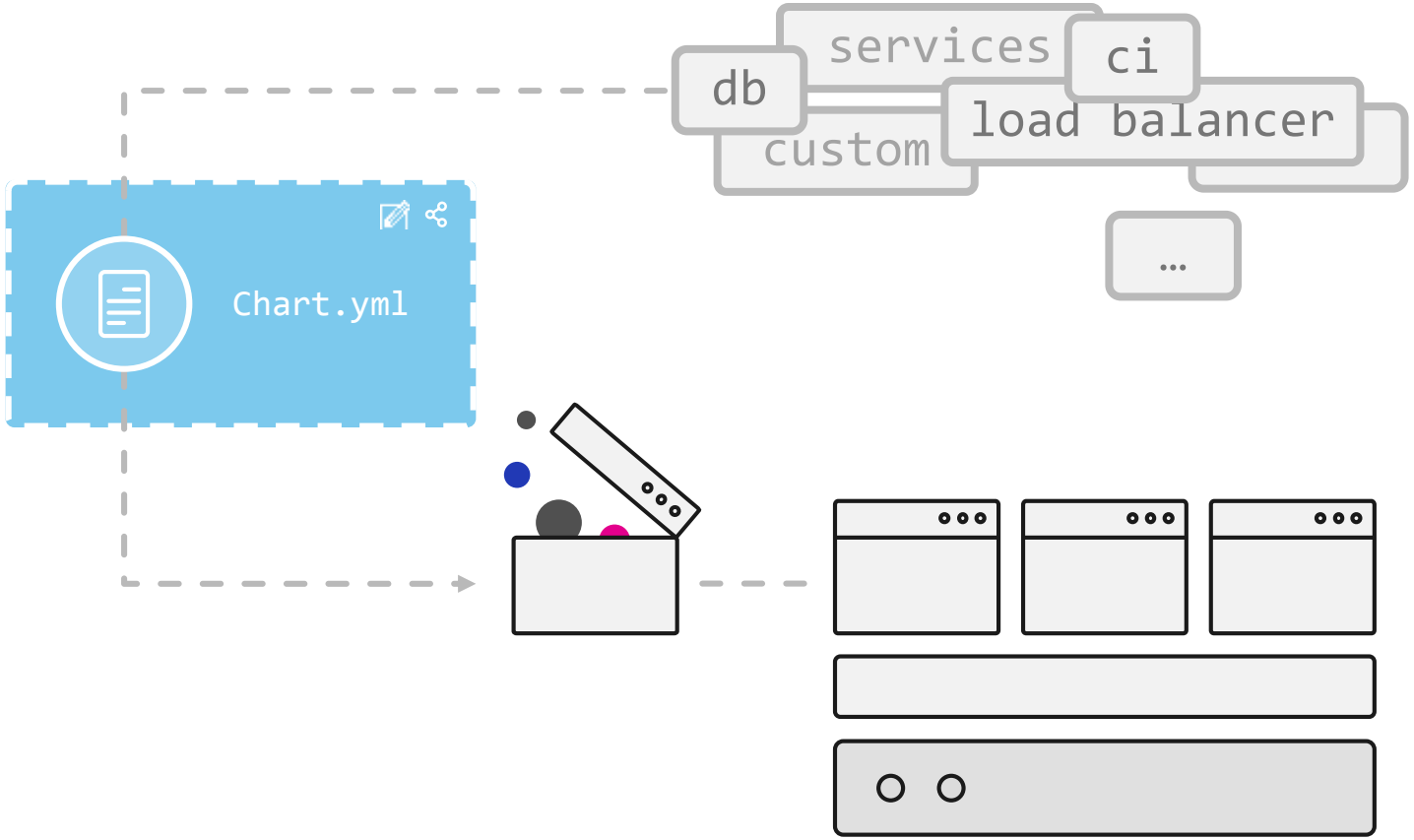
Azure Container Registry

Open Service Broker API (OSBA)

Release Automation Tools

Helm

Helm Charts helps you define, install, and upgrade even the most complex Kubernetes application



What is Helm?

What is Helm?

DEATH TO COPY PASTE

What is Helm?

No more indent errors

- The Kubernetes package manager
  - Similar to brew/apt/pip/nuget but for Kubernetes
- Combines multiple Kubernetes resources in one versioned unit (a *chart*)
- Templates provide default behavior and ability to override
- Enables lifecycle operations like upgrade
- Helm repos enable easy deployment of pre-built apps

Birth of Helm

- On October 15th, 2015
- Hackathon project at company offsite
  - Could we take the ideas behind npm and Homebrew and build something for deploying apps into Kubernetes?
- Installation tool for Deis Workflow
- Announced at the first KubeCon in San Francisco 2015

Architecture

Key concepts: **Helm**, **Tiller**, **Charts**

- Helm has two major components:
  - **Helm client**
  - **Tiller server**
- Helm client is the CLI for end users
  - Written in Go, uses gRPC to interact with the server
  - Sends charts and values to Tiller for install, upgrade, etc.

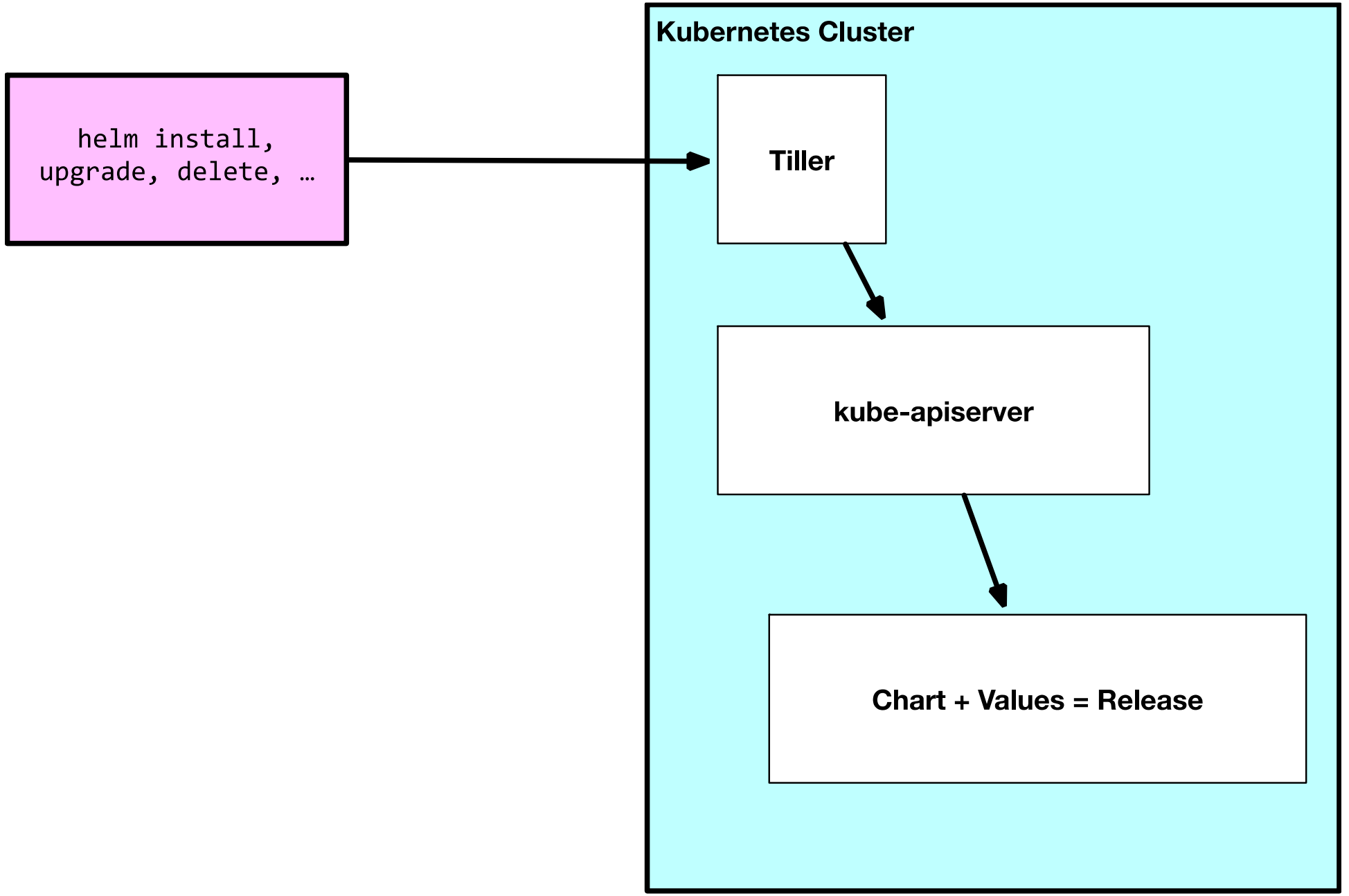
Architecture

Tiller server

- In-cluster server that interacts with the client and interfaces with the Kubernetes API server

Chart

- Collection of Kubernetes resources


helm install,
upgrade, delete, …
Kubernetes Cluster
Tiller
kube-apiserver
Chart + Values = Release

Demo – Helm and Crochunter

Helm Charts

Chart structure

- Charts have structure
  - Set of conventions, including file and directory names
  - Charts can be packaged into tarballs for distribution

Chart structure

- Layout
  - Helm expects a strict chart structure

```
wordpress/
  Chart.yaml          # A YAML file containing information about the chart
  LICENSE             # OPTIONAL: A plain text file containing the license for the chart
  README.md           # OPTIONAL: A human-readable README file
  values.yaml         # The default configuration values for this chart
  charts/             # OPTIONAL: A directory containing any charts upon which this chart depends.
  templates/          # OPTIONAL: A directory of templates that, when combined with values,
                      # will generate valid Kubernetes manifest files.
  templates/NOTES.txt # OPTIONAL: A plain text file containing short usage notes
```

Chart.yaml

```
name: The name of the chart (required)
version: A SemVer 2 version (required)
description: A single-sentence description of this project (optional)
keywords:
  - A list of keywords about this project (optional)
home: The URL of this project's home page (optional)
sources:
  - A list of URLs to source code for this project (optional)
maintainers: # (optional)
  - name: The maintainer's name (required for each maintainer)
    email: The maintainer's email (optional for each maintainer)
engine: gotpl # The name of the template engine (optional, defaults to gotpl)
icon: A URL to an SVG or PNG image to be used as an icon (optional).
```

Helm values.yaml

- The knobs and dials:
  - A `values.yaml` file provided with the chart that contains **default** values
  - Use -f to provide your own values overrides
  - Use --set to override individual values

Helm Templates

- Built on Go's template language w/addition of 50 or so add-on template functions
- Almost anything goes! ;)
- Also useful in generating random values (e.g. passwords)
  - Provides flow control (if/else, with, range, etc)
  - Named templates (partials)

Helm Templates

- Built-in predefined values
  - Release-related
    - `Release.Name, Release.Namespace, Release.IsInstall, Release.IsUpgrade`, etc.
  - Chart:
    - Values from `chart.yaml`: **Chart.Version**, **Chart.Maintainers**, etc.

Chart versions

- Helm uses version numbers as release markers
  - All charts must specify SemVer2 numbers

Helm charts (the other bits)

Chart LICENSE, README and NOTES

- **`README.md`**: description of application or service that the chart provides, prereqs to run, description of options in `values.yaml` and their default values
- **`templates/NOTES.txt`**: NOTES are printed after install, upgrade, or when viewing the status of a release. Meant for chart developers to display usage notes, next steps, etc. for chart users.

Dependencies

- A chart can depend on any number of other charts.
- Dependencies are expressed by
  - copying the dependency charts into the `charts/` subdirectory
  - `Requirements.yaml` allows you to declare dependencies

Helm’s other hats

- Lifecycle Management
  - Update
  - Rollback
  - Config management
  - Testing
  - Repeatability

Demo – DevOps Project with AKS

Installing Helm

Helm.sh

Installing Draft

Draft.sh

Copy from the best!

- https://github.com/kubernetes/charts
- https://kubeapps.com

- Stable charts: we are pretty sure these work well.
- Incubator charts: good luck, hard fight!

- Interested in writing a chart? Reach out! We only bite on the weekends.

Connect with me!

github.com/jldeen/croc-hunter
hub.kubeapps.com

Email: jessica.deen@microsoft.com
Twitter/GitHub/Instagram: @jldeen
Blog: jessicadeen.com
LoECDA Website: loecda.com

Complete your evaluations to **win prizes!**

Please contact build@eventcore.com for more information

Daily Prizes

Xbox One X

DJI Spark Drone

Starbucks gift cards

Be eligible to win!

Make sure you are scanned into your session and complete your session eval to be eligible to win.

Seven winners each day!

Winners will be notified via email.

Pick up your prize before leaving!

Prizes can be claimed at the info counter located on WSCC level 4, Galleria level.

Monday: pick up 2:15pm–6:30pm.

Tuesday: pick up 7:00am–6:00pm.

Wednesday: pick up 7:00am–5:30pm.

Microsoft