



[Features](#) [Forum](#) [Jobs](#) [Workplace](#)[Home](#) » [Blogs](#) » [mk's blog](#)

Kubernetes คืออะไร ทำไม Orchestration จึงเป็นหัวใจของ Infrastructure ยุคนี้

By: mk   on 17 November 2018 - 15:59Tags: [Kubernetes](#) [Container](#) [CNCF](#) [Google](#) [Cloud Computing](#) [In-Depth](#)

ในรอบ 1-2 ปีนี้ ชื่อเทคโนโลยีฝั่งเซิร์ฟเวอร์ที่พบเห็นได้บ่อยมากคือคำว่า **Kubernetes** ที่แค่เห็นก็อ่านไม่ออกแล้วว่าออกเสียงอย่างไร (อ่านว่า "คูเบอร์เนตส์") แต่ความรุ่มร่าแรงของมันถึงขั้นพลิกโฉมสถาปัตยกรรมของระบบโครงสร้างพื้นฐานทางไอที (infrastructure) ไปอย่างสิ้นเชิง

Kubernetes เป็นซอฟต์แวร์สำหรับ **Container Orchestration** (คำเดียวกับ "วงออร์เคสตรา") ถ้าให้อธิบายแบบสั้นๆ มันคือซอฟต์แวร์ที่ใช้จัดการและควบคุม "วงดนตรีคอนเทนเนอร์" นั่นเอง ส่วนคำ

อธิบายแบบยาวๆ สามารถอ่านได้จากบทความนี้ครับ



Orchestration = Container + Cluster

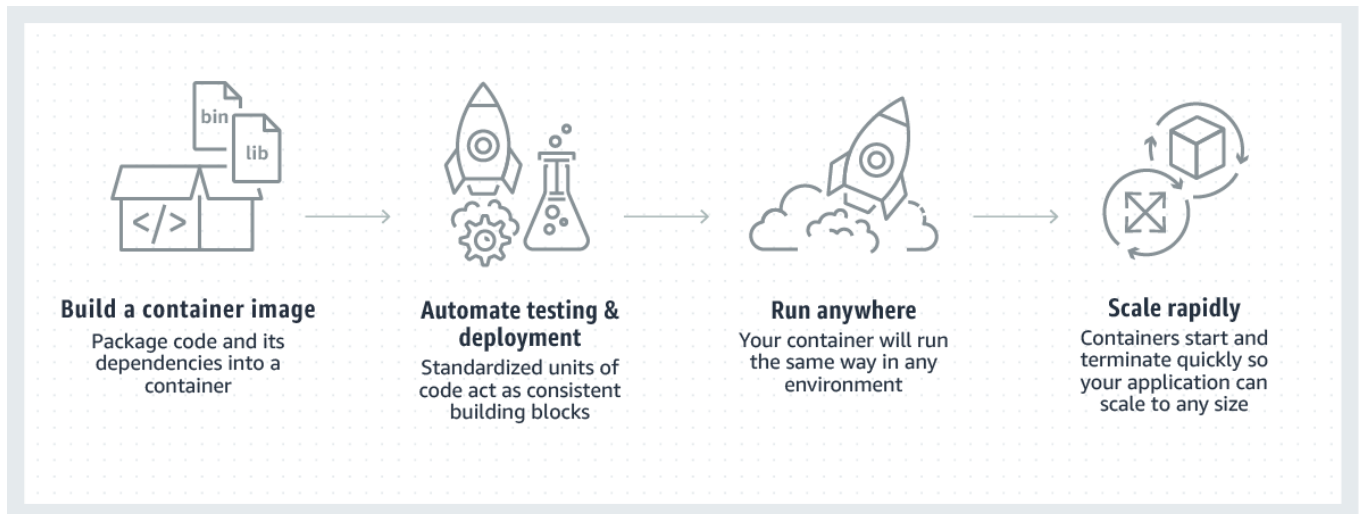
ในบทความตอนที่แล้ว เราอธิบายถึง "คอนเทนเนอร์" ว่ามันคืออะไร (รู้จัก [Container](#) มันคืออะไร แตกต่างจาก [Virtualization](#) อย่างไร?)

เทคโนโลยีคอนเทนเนอร์ถูกสร้างขึ้นมาเพื่อให้เราจัดการแพ็คเกจซอฟต์แวร์ที่จำเป็น และขนย้ายมันไปใช้ตามที่ต่างๆ ได้ง่ายขึ้น เหมือนกับจับของใส่ตู้คอนเทนเนอร์แล้วขนขึ้นเรือหรือรถบรรทุกก็ได้

แต่การจะนำคอนเทนเนอร์ไปใช้งานอย่างไร ถือเป็นอิสระเต็มที่ของผู้ใช้งาน ซึ่งเราสามารถรันคอนเทนเนอร์ในเครื่องโลคอล (นักพัฒนาทดสอบโค้ดตัวเองบนเครื่องตัวเอง) หรือรันบนเซิร์ฟเวอร์แบบดั้งเดิม (bare metal) ก็ได้ทั้งนั้น

ในโลกยุคคลาวด์ เราสามารถเช่าเซิร์ฟเวอร์ใหม่ขึ้นมาได้ทันทีตามความต้องการของผู้ใช้ การเช่าเครื่องเสมือนจำนวนมากๆ แล้วมาต่อเชื่อมกันเป็นคลัสเตอร์เพื่อขยายตัว รองรับโหลดจำนวนมาก เป็นเรื่องปกติที่ทำได้ทุกวันนี้

เมื่อเรานำเทคโนโลยี "คอนเทนเนอร์" ในฝั่งแอปพลิเคชัน มาจับคู่กับ "คลัสเตอร์" ในฝั่งของโครงสร้างพื้นฐานที่รันงาน เหล่านั้น หน้าที่บริหารจัดการตรงนี้คือคำว่า Orchestration นั่นเอง (บางครั้งก็ใช้คำว่า container management)



ภาพจาก [AWS](#)

ศึกชิงความเป็นเจ้า Orchestration

ในอดีต การนำแอปพลิเคชันที่พัฒนาเสร็จแล้วไปรันบนเครื่องจริง (deployment) เป็นกระบวนการที่ซับซ้อน และต้องพึ่งพาการทำงานแบบ manual ชะเยอะ ผู้ดูแลระบบมักต้องเขียนสคริปต์กันเอง ซึ่งมีข้อผิดพลาดได้ง่าย ยิ่งการรันบนคลัสเตอร์ที่มีเครื่องจำนวนมาก ต้องย้ายงานไปมา ต้องขยายตัวหรือหดตัวตามความต้องการของลูกค้า ยิ่งก่อให้เกิดความซับซ้อนมากขึ้นเป็นทวีคูณ

หลายปีที่ผ่านมามีความพยายามแก้ปัญหาจากหลายฝ่าย รายที่โดดเด่นมีด้วยกัน 3 ค่ายคือ

- **Docker Swarm** จาก Docker Inc.
- **Kubernetes** จากกูเกิล
- **Apache Mesos/Marathon** ที่ริเริ่มโดยทีมงานจาก University of California, Berkeley และภายหลังก่อตั้งบริษัท Mesosphere, Inc. มาสนับสนุน

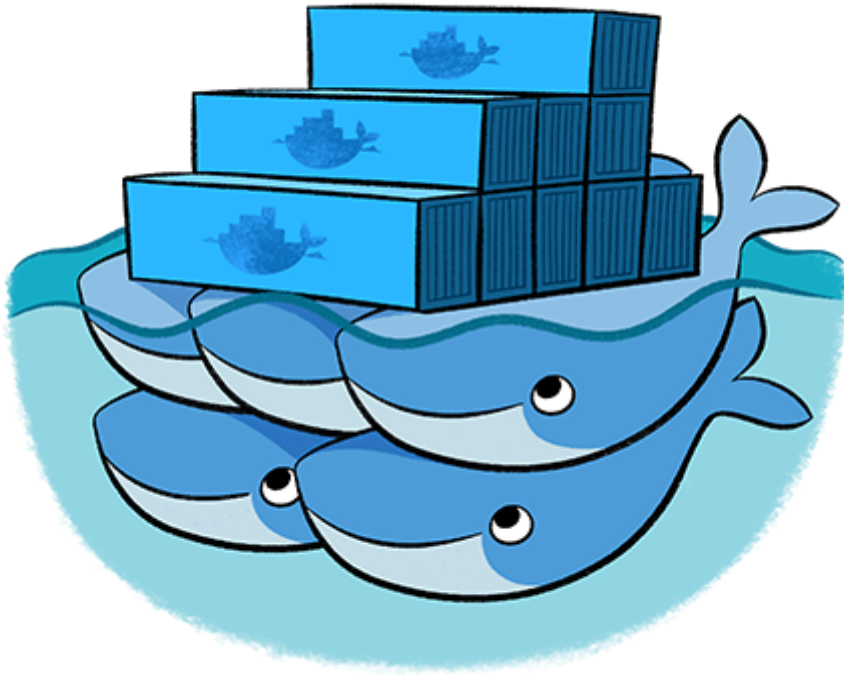
แนวคิดทางเทคนิคของซอฟต์แวร์ orchestration แต่ละรายมีความแตกต่างกันอยู่บ้าง

กรณีของ Apache Mesos สร้างขึ้นก่อนเป็นรายแรก (เริ่มในปี 2009) และออกแบบมาจัดการคลัสเตอร์อะไรก็ได้ ไม่จำเป็นต้องเป็นคอนเทนเนอร์อย่างเดียว ภายหลังจึงมีซอฟต์แวร์ Marathon มาเสริมเรื่องการจัดการคอนเทนเนอร์ ในขณะที่บริษัท Mesosphere นำ Mesos ไปปรับแต่งเป็นซอฟต์แวร์เชิงพาณิชย์ และเรียกชื่อว่า DC/OS (ย่อมาจาก Datacenter OS) แสดงให้เห็นถึงการเน้นเรื่องการจัดการศูนย์ข้อมูลเป็นหลัก



MESOS

ส่วน Docker Swarm เริ่มออกเวอร์ชันแรกๆ ในปี 2015 โดยเป็นส่วนต่อขยายตามธรรมชาติของ Docker Container ของบริษัท Docker Inc. ที่เอาชนะสงครามคอนเทนเนอร์ได้อย่างเบ็ดเสร็จ (อีกฝ่ายคือ [Rocket](#) หรือ [rkt](#) ของ CoreOS)



แน่นอนว่ามาถึงตอนนี้ เรารู้แล้วว่าผู้ชนะสงคราม orchestration คือ Kubernetes ของกูเกิล ดังนั้นมาย้อนประวัติของมันแบบละเอียดกันสักหน่อย

ประวัติความเป็นมาของ Kubernetes: จาก Borg สู่โลกภายนอก

ชื่อ Kubernetes มาจากภาษากรีก κυβερνήτης ซึ่งแปลเป็นภาษาอังกฤษได้ว่า helmsman หรือ "กัปตันเรือ" จึงเป็นที่มาของโลโก้รูปพวงมาลัยเรือ (คำว่า Kubernetes มีรากศัพท์มาจากคำเดียวกับคำว่า Governor ด้วย) แต่เนื่องจากชื่อโครงการค่อนข้างยาว เรียกยาก ในวงการเลยมักเรียกกันย่อๆ ว่า K8s หรือบ้างก็เรียก Kube

Kubernetes เปิดตัวสาธารณะครั้งแรกในเดือนมิถุนายน 2014 (เป็นข่าวใน [Blognone](#) ครั้งแรกเดือนกรกฎาคม 2014) แต่ประวัติของมันยาวนานกว่านั้น

กูเกิลมีธรรมเนียมการสร้างซอฟต์แวร์เพื่อใช้เองภายในอยู่แล้ว และมีระบบจัดการคอนเทนเนอร์ของตัวเองเรียกว่า **Borg** (มาจากเผ่าเอเลี่ยนกึ่งหุ่นยนต์ในซีรีส์ Star Trek) ซึ่งทำงานอยู่บน [cgrouops](#) พีเจเออร์ของเคอร์เนลลินุกซ์ และเป็นต้นแบบของคอนเทนเนอร์ในยุคปัจจุบัน

Borg วิวัฒนาการมาเรื่อยๆ โดยไม่ได้คำนึงถึงการออกแบบสถาปัตยกรรมมาตั้งแต่ต้น เพราะเน้นสร้างพีเจเออร์ใหม่ตามความต้องการของทีมงานภายในแต่ละทีม สุดท้าย Borg จึงไปต่อไม่ไหว กูเกิลจึงสร้างซอฟต์แวร์ตัวใหม่ขึ้นมาแทนโดยใช้ชื่อว่า **Omega**

Omega นำแนวคิดหลายอย่างจาก Borg มาใช้งาน แล้วเพิ่มระบบการเก็บสถานะของคลัสเตอร์ไว้ที่ส่วนกลางเพื่อป้องกันข้อมูลขัดแย้งกันเอง

เมื่อเทคโนโลยีคอนเทนเนอร์เริ่มได้รับความนิยม และกูเกิลเองก็มีบริการลักษณะนี้ต่อลูกค้าภายนอกบริษัท (Google Cloud Platform) วิศวกรกลุ่มหนึ่งของกูเกิลจึงตัดสินใจสร้างโครงการแบบนี้ขึ้นมาใหม่อีกครั้ง ซึ่งก็คือ Kubernetes

เนื่องจาก Kubernetes สืบทอดแนวคิดหลายอย่างมาจาก Borg ในช่วงแรก โครงการจึงมีโค้ดเนมว่า **Project Seven** ซึ่งมาจากตัวละคร Seven of Nine ขอรักสาวจากเรื่อง Star Trek เช่นกัน (และเป็นเหตุผลว่าทำไมไอคอนพวงมาลัยเรือถึงมี 7 แขน)



Seven of Nine ภาพจาก StarTrek.com

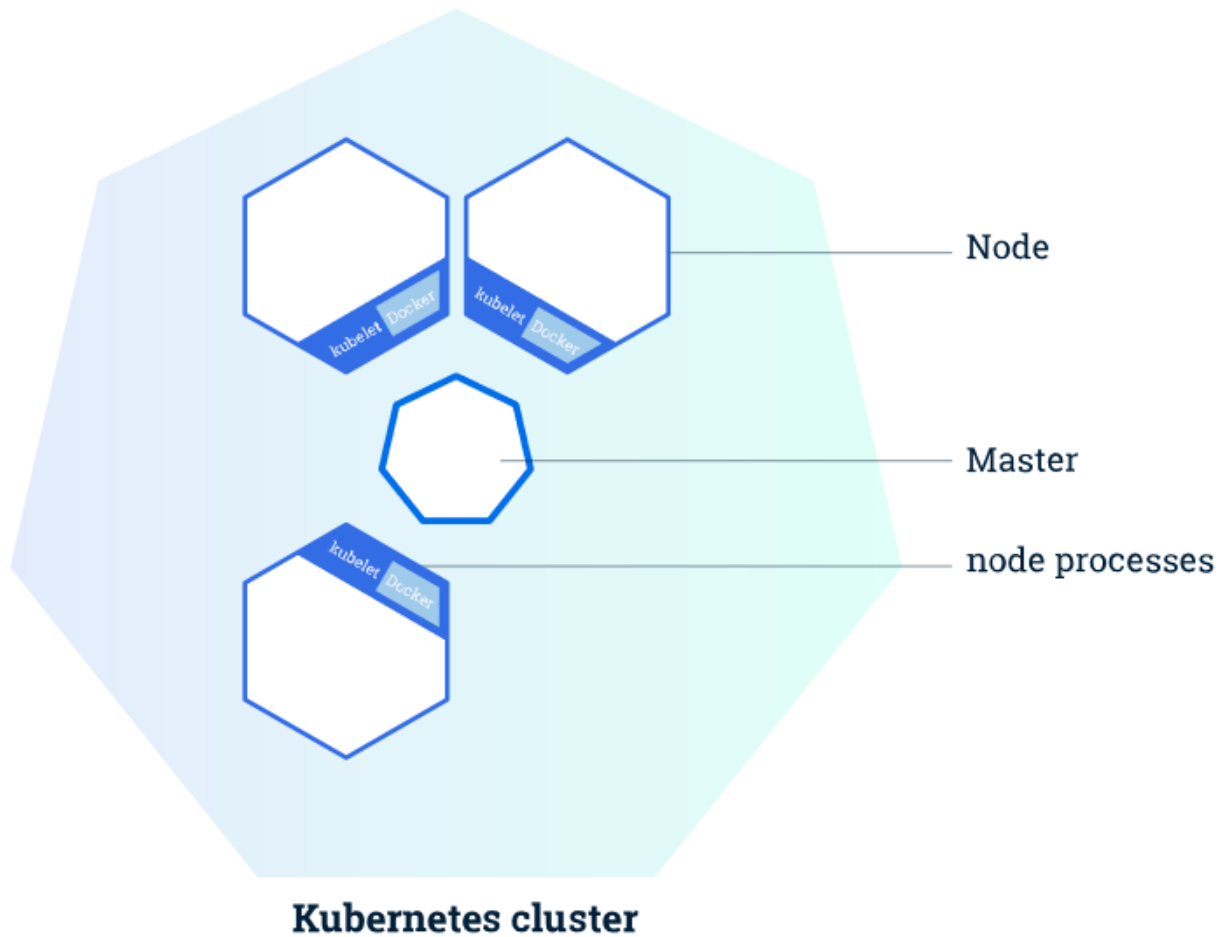
Kubernetes ได้รับความนิยมอย่างรวดเร็ว และได้รับการสนับสนุนจากบริษัทขนาดใหญ่อย่าง Microsoft, Red Hat, IBM จนเมื่อถูกปล่อยออก [Kubernetes 1.0](#) ในเดือนกรกฎาคม 2015 ก็ตัดสินใจตั้งมูลนิธิ **Cloud Native Computing Foundation (CNCF)** มาดูแลโครงการต่อแทน

หมายเหตุ: ประวัติอย่างละเอียดของ Borg, Omega, Kubernetes สามารถอ่านได้จากบทความ [Borg, Omega, and Kubernetes](#) ที่เขียนโดยทีมงานของกุเกิลผู้ร่วมก่อตั้ง Kubernetes

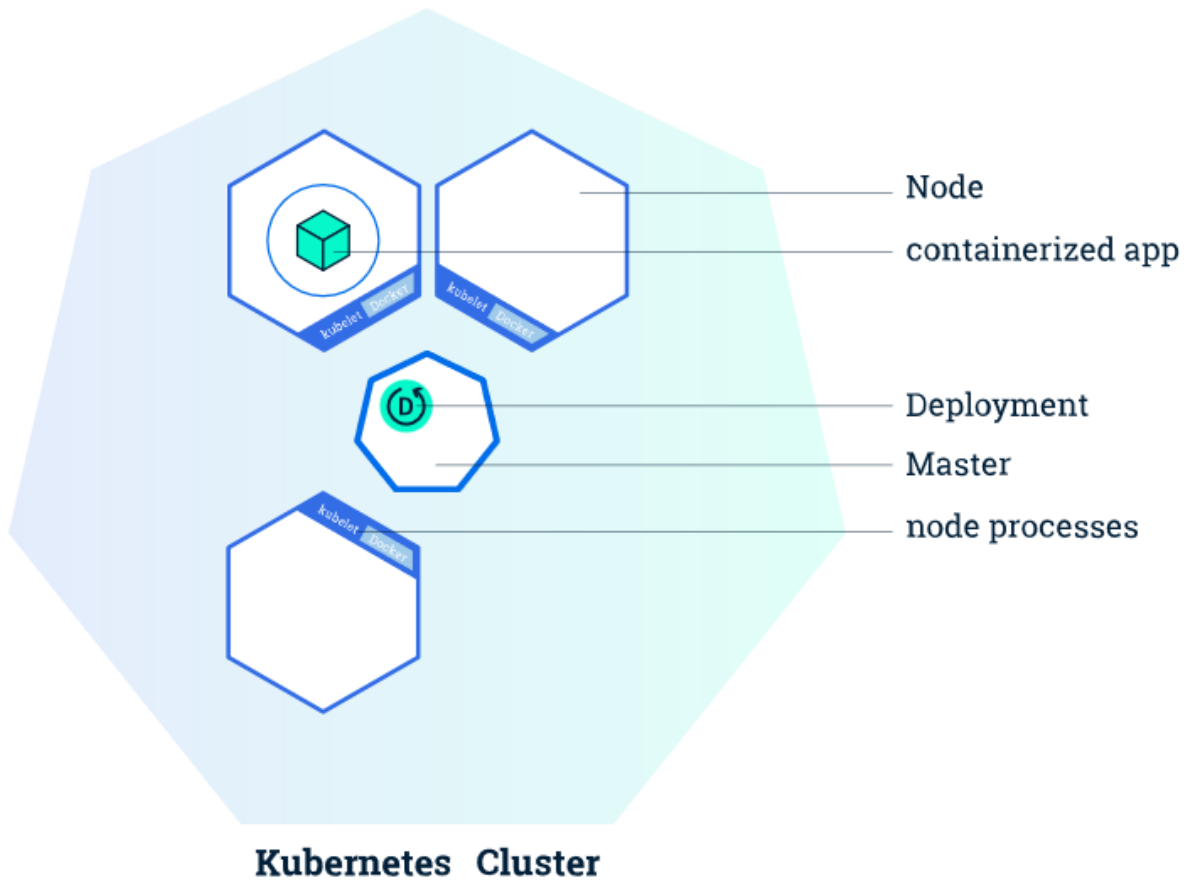
แนวคิดของ Kubernetes

เนื่องจากบทความนี้ไม่ใช่บทความสอนการใช้งานเชิงเทคนิค จึงอธิบายแนวคิดของ Kubernetes เพียงคร่าวๆ เพื่อให้เห็นภาพเท่านั้น รายละเอียดอื่นๆ สามารถอ่านได้จาก [เว็บไซต์ Kubernetes](#)

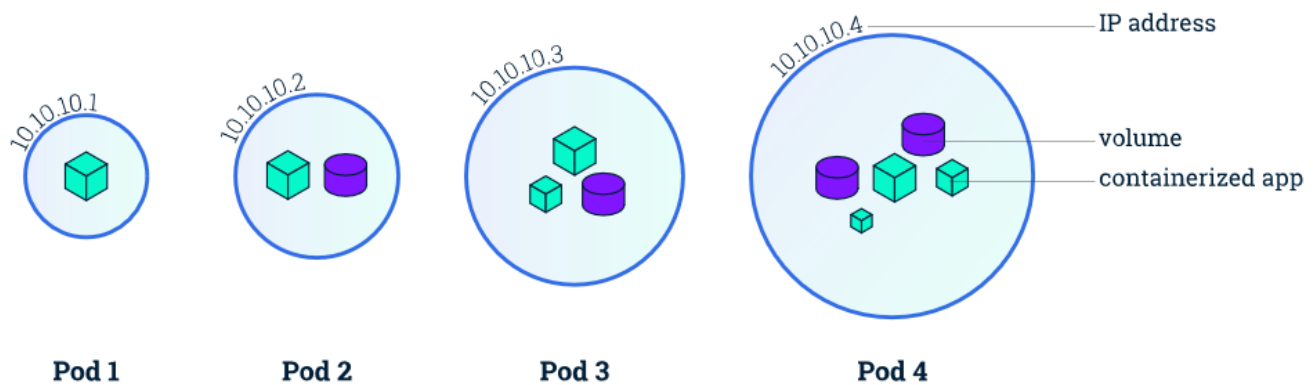
สถาปัตยกรรมคลัสเตอร์ของ Kubernetes ไม่ต่างอะไรจากคลัสเตอร์ทั่วๆ ไป นั่นคือมีเครื่องที่เป็น Master และ Slave (ในโลกของ Kubernetes ใช้คำว่า **Node**) โดยในแต่ละโหนดติดตั้งซอฟต์แวร์ Kubelet เพื่อบริหารจัดการ



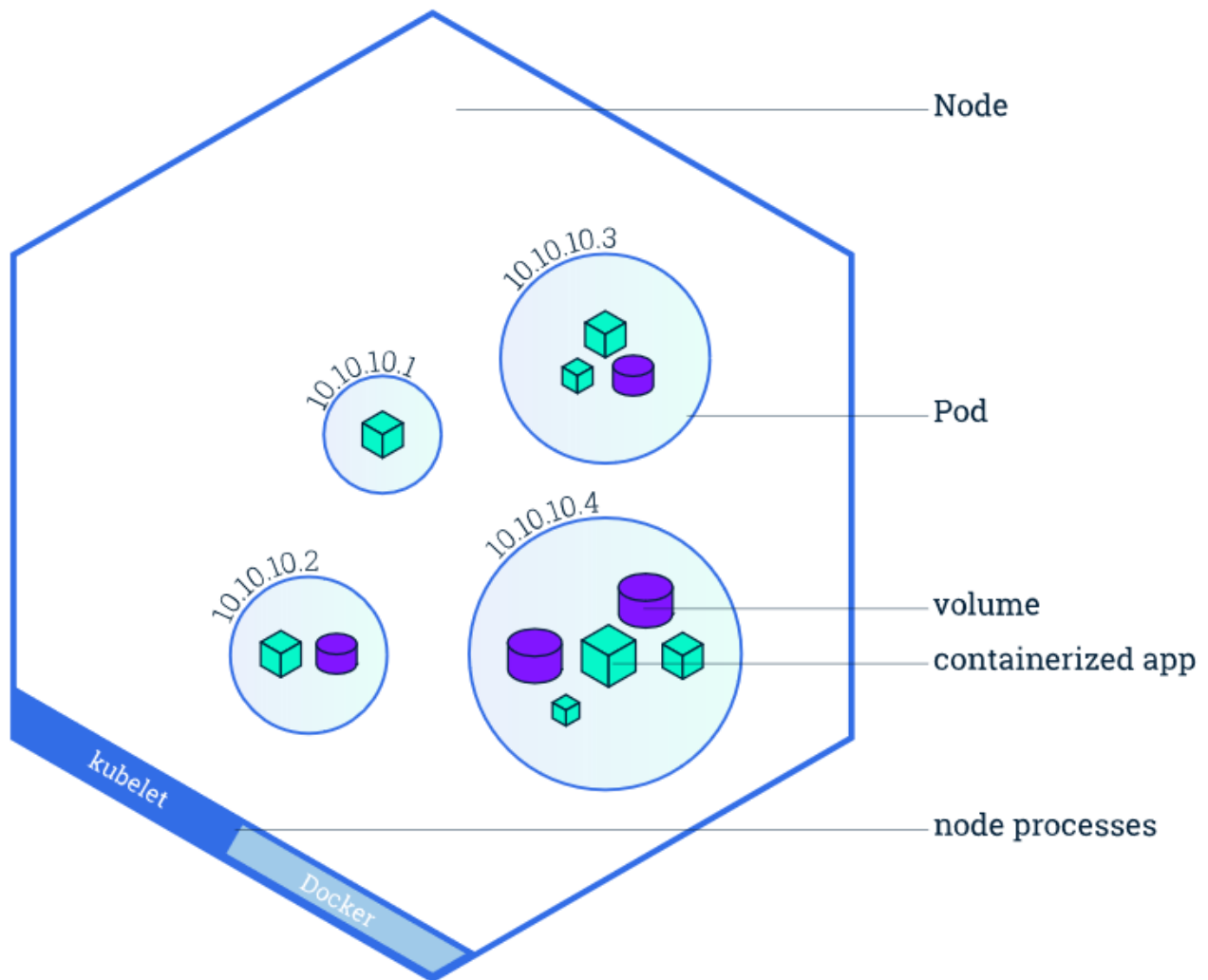
จากนั้นเมื่อเรานำแอปในคอนเทนเนอร์ไปรันบนคลัสเตอร์ (deploy) ระบบของ Kubernetes จะสื่อสารกันระหว่าง Master/Node เพื่อนำงานไปรันบนเครื่องให้เราเอง



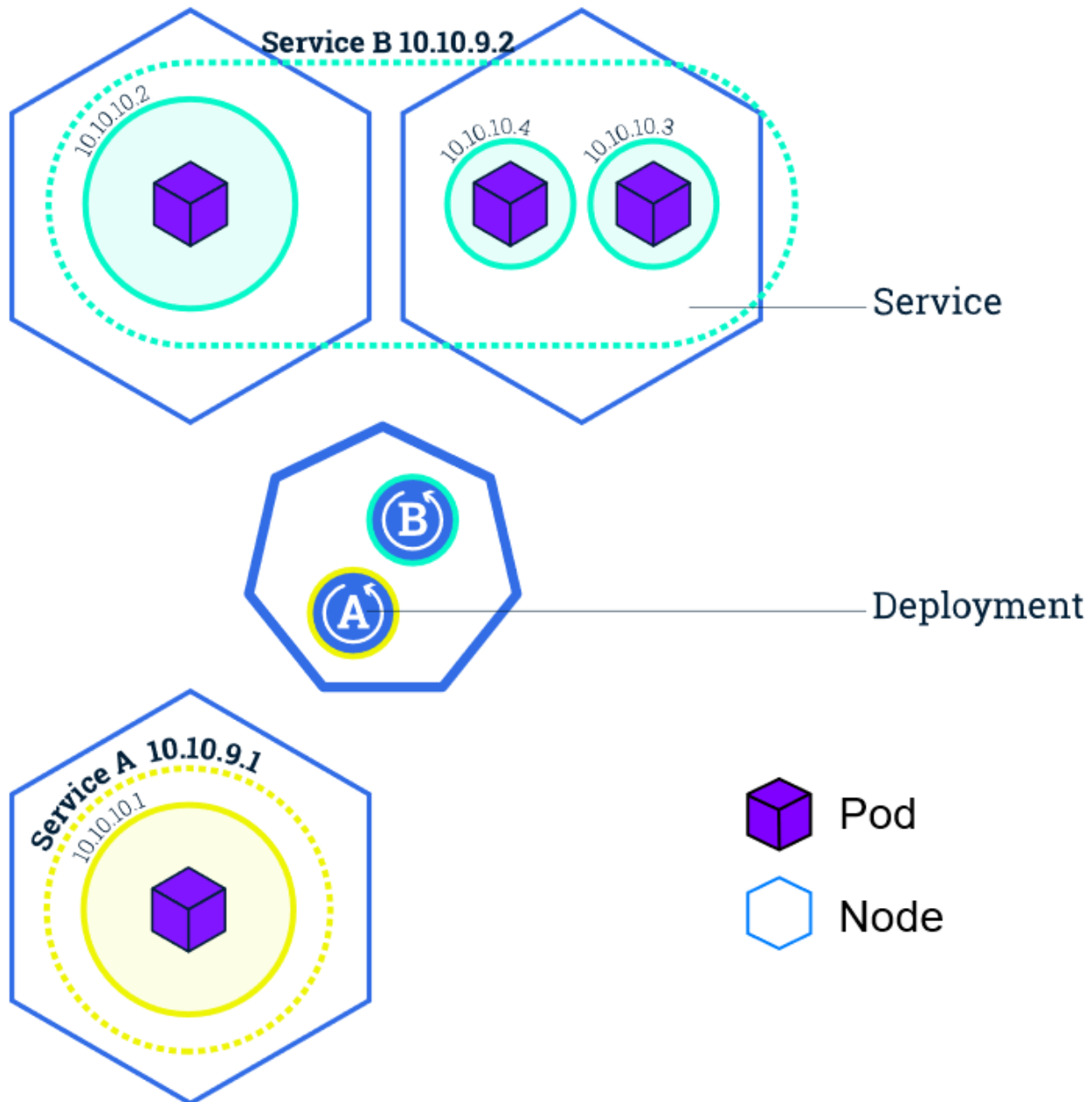
แนวคิดสำคัญของ Kubernetes คือคำว่า **Pod** เปรียบเสมือนเป็นก้อนรวมของสิ่งที่จำเป็นทั้งหมดในการรันแอปพลิเคชัน ประกอบด้วยตัวแอปเอง (ที่อยู่ในคอนเทนเนอร์) และสโตเรจ (Volume) โดย Pod แต่ละก้อนมีหมายเลขไอพีของตัวเอง



ในเครื่องแต่ละ Node จึงสามารถมิงงานไปรันได้หลาย Pod ตามภาพ



นอกจาก Pod แล้ว โลกของ Kubernetes ยังมีแนวคิดเรื่อง **Services** หรือการจัดกลุ่ม Node/Pod จำนวนมากๆ เข้าด้วยกัน แอปพลิเคชันหนึ่งตัวสามารถแบ่งได้เป็นหลาย Pod และรันอยู่บนหลาย Node ดังนั้น การสเกลให้รองรับโหลดที่เพิ่มขึ้นจึงทำได้ง่าย เพราะมีแนวคิดเชิง abstraction คือ Node/Pod/Service มาคอยช่วยนั่นเอง



นอกจากนี้ Kubernetes ยังมีฟีเจอร์อื่นๆ อีกมาก เช่น การอัปเดตข้อมูลในคอนเทนเนอร์ (เปลี่ยนเวอร์ชันของไลบรารีหรือโค้ด), การ rollback กลับไปเวอร์ชันเดิมเมื่อเกิดปัญหา และการทำ Continuous Integration and Continuous Delivery (CI/CD) ซึ่งจะไม่กล่าวถึงในบทความนี้

ชัยชนะของ Kubernetes

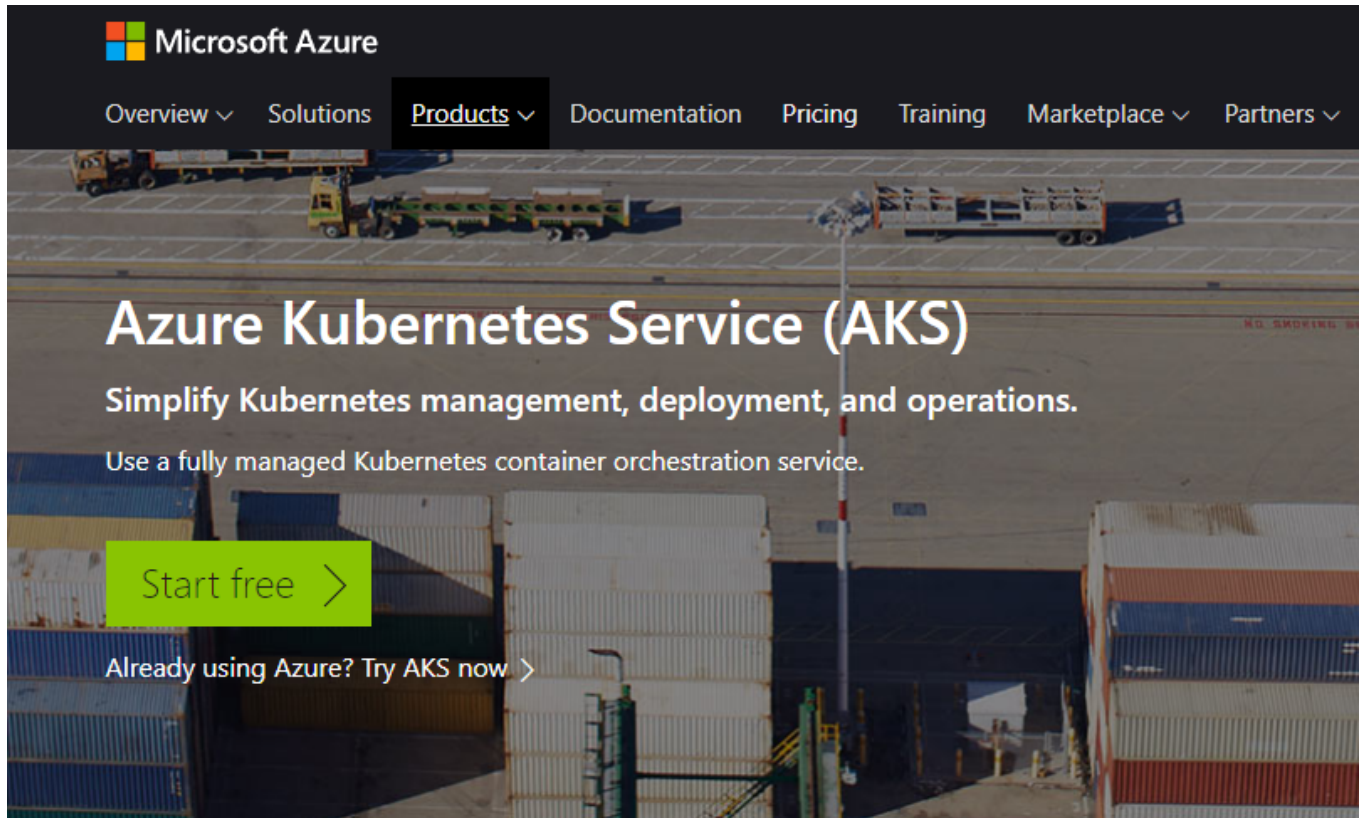
ชัยชนะของ Kubernetes มาจากปัจจัยหลายประการ ในแง่เทคนิค มันสืบทอดสถาปัตยกรรมภายในของกูเกิลที่ผ่านการทดสอบงานจริง รองรับโหลดหนักๆ ระดับกูเกิลมาได้แล้ว (Pokemon Go ก็รันอยู่บน Kubernetes) บวกกับมีจุดสมดุลที่พอดีระหว่างฟีเจอร์และความซับซ้อนในการดูแล เมื่อเปรียบเทียบกับคู่แข่งอย่าง Docker Swarm ที่อาจใช้ง่ายกว่า แต่ความสามารถด้อยกว่า ส่วน Apache Mesos ก็มีความซับซ้อนสูงมาก

ปัจจัยเรื่องพันธมิตรร่วมรบก็มีความสำคัญไม่แพ้กัน Kubernetes เป็นโครงการของกูเกิล ย่อมได้รับการสนับสนุนจากกูเกิลอย่างเต็มที่มาตั้งแต่แรก แคมป์ยังได้พันธมิตรระดับบิ๊กอย่าง IBM หรือ Red Hat เข้าร่วมด้วยตั้งแต่ยุคแรกๆ

พันธมิตรของ Kubernetes ที่น่าสนใจคือไมโครซอฟท์ ช่วงแรกที่กระแส container orchestration เริ่มมาแรง แต่ยังไม่ปรากฏผู้ชนะที่ชัดเจน ไมโครซอฟท์ก็ใช้วิธีสนับสนุนทุกฝ่ายแล้วรอดูว่าใครจะชนะ โดยการเข้าลงทุนในบริษัท Mesosphere เมื่อปี 2016 และเปิดตัวบริการ Azure Container Service (ACS) ที่รองรับทั้ง Mesos และ Docker Swarm

แต่พอมาถึงปี 2017 เมื่อชัยชนะของ Kubernetes เริ่มปรากฏชัด ไมโครซอฟท์ก็เปลี่ยนใจกันง่ายๆ เปลี่ยนชื่อ **Azure Container Service (ACS)** เป็น **Azure Kubernetes Service (AKS)** และหันมาเลือกข้าง Kubernetes เต็มตัว (ปัจจุบันถ้าเข้าหน้าเว็บ ACS จะถูก redirect มายัง AKS แทน และ ACS ยังมีให้ใช้งานอยู่บน Azure Marketplace แต่ก็ไม่เน้นแล้ว)

ไมโครซอฟท์ยังจ้าง [Brendan Burns](#) หนึ่งในผู้ร่วมก่อตั้ง Kubernetes มาทำงานด้วย เรียกได้ว่าดึงตัวมาจากกูเกิลกันตรงๆ เพื่อมารับผิดชอบงานด้าน Kubernetes โดยเฉพาะ แสดงให้เห็นถึงความสำคัญของ Kubernetes ในสายตาไมโครซอฟท์ได้เป็นอย่างดี



อีกตัวอย่างที่น่าสนใจไม่แพ้กันคือ AWS ที่ช่วงแรกๆ อินดี้ไม่สนใจใคร สร้างระบบ orchestration ของตัวเองขึ้นมาโดยไม่ยุ่งกับใคร โดยใช้ชื่อว่า **Amazon Elastic Container Service (ECS)** และผูกกับเทคโนโลยีของ AWS อย่างแนบแน่น

แต่เมื่อ Kubernetes เริ่มดังขึ้นมา AWS ก็ออกบริการใหม่อีกตัวคือ **Amazon Elastic Container Service for Kubernetes (EKS)** ขึ้นมาให้เลือกใช้งานได้ตามชอบ (อย่างไรก็ตาม ECS ใช้ฟรี แต่ EKS คิดเงิน)

Orchestration

Amazon Elastic Container Service

Fully managed container orchestration, seamlessly integrated with other AWS services.

[Learn more »](#)

Amazon Elastic Container Service for Kubernetes

Easily deploy, orchestrate, and scale containerized applications using Kubernetes.

[Learn more »](#)

ทั้งไมโครซอฟท์และ AWS ต่างก็สมัครเข้าร่วมเป็นสมาชิก Cloud Native Computing Foundation (CNCF) ในช่วงกลางปี 2017 เป็นสัญญาณว่าผู้ให้บริการคลาวด์รายใหญ่ต่างยอมสยบให้กับ Kubernetes แล้ว

แต่จุดสิ้นสุดสงครามคือเดือนตุลาคม 2017 ที่ Docker ประกาศความพ่ายแพ้ ทานกระแสนี้ไหว ต้องยอมซัพพอร์ต Kubernetes นอกเหนือจาก Docker Swarm ของตัวเอง (หลังจากดิ่งตัน พยายามต่อสู้มานาน) ถึงแม้ในแถลงการณ์ของ

Docker จะบอกว่า "ผู้ใช้มีสิทธิเลือก"ว่าจะใช้ตัวไหน แต่ก็สะท้อนว่าถ้าสู้เขาไม่ได้ เข้าร่วมดีกว่านั่นเอง



ฝั่งของ Apache Mesos ก็ไม่ใช่คู่แข่งตรงๆ ของ Kubernetes และสามารถทำงานด้วยกันได้ โดยบริษัท Mesosphere เองก็มีผลิตภัณฑ์ Mesosphere Kubernetes Engine (MKE) ออกมาจับตลาดเช่นกัน

มาถึงตอนนี้โลก container orchestration ได้คำตอบชัดเจนแล้วว่า ถ้าต้องการ container ให้ใช้ Docker และถ้าต้องการ orchestration ให้ใช้ Kubernetes

Kubernetes Distributions

ในการใช้งานจริง Kubernetes ประกอบด้วยโครงการย่อยๆ มากมาย การใช้ซอฟต์แวร์จากโครงการโอเพนซอร์สด้านนี้ (เช่น ระบบมอนิเตอร์ Prometheus) จึงต้องใช้ความรู้ความเชี่ยวชาญสูง

ตรงนี้จึงเกิด Kubernetes Distribution หรือเทียบได้กับ "ดิสโทร" ของลินุกซ์ ที่เป็นการรวบรวมซอฟต์แวร์ที่จำเป็น และเครื่องมืออำนวยความสะดวกอื่นๆ มาให้พร้อมสรรพ ติดตั้งแล้วใช้งานได้ทันที

ตัวอย่าง Kubernetes Distribution แบบง่ายๆ ได้แก่ [Minikube](#) ที่เน้นการใช้งานบนเครื่องตัวเอง (คลัสเตอร์มีเครื่องเราเครื่องเดียว) เหมาะสำหรับการทดสอบและลองใช้งาน

ปัจจุบัน บริษัทไอทียักษ์ใหญ่ที่มีธุรกิจด้านศูนย์ข้อมูล-คลาวด์ ต่างก็มี Kubernetes Distribution เวอร์ชันของตัวเองไว้บริการลูกค้า โดยใช้ชื่อเชิงพาณิชย์แตกต่างกันไป (บางครั้งถ้าไม่บอกก็ไม่ว่ามันคือ Kubernetes) เช่น





















- IBM Cloud Private
- Red Hat OpenShift ([บทความใน Blognone](#))
- Pivotal Container Service (PKS)
- Oracle Linux Container Services
- Cisco Container Platform
- Canonical Distribution of Kubernetes (CDK)
- Rancher

โซลูชันเหล่านี้สามารถรันได้ทั้งบนคลาวด์แบบสาธารณะ (public cloud) หรือเครื่องขององค์กรเอง (private cloud/on premise)

ส่วนผู้ให้บริการคลาวด์แบบสาธารณะ (public cloud) รายใหญ่ ต่างก็มีบริการ Kubernetes ให้บริการเช่นกัน

- Google Kubernetes Engine (GKE)
- Amazon Elastic Container for Kubernetes (EKS)
- Azure Kubernetes Serve (AKS)

รายชื่อทั้งหมดดูได้จาก [Kubernetes Partners](#)

 Alibaba Cloud Alibaba Cloud Container Service MCap: \$402B Alibaba Cloud	 AmazonEKS Amazon Elastic Container Service for Kubernetes (EKS) MCap: \$792B Amazon Web Services	 Azure Container Service Azure (ACS) Engine ★ 993 MCap: \$824B Microsoft	 Microsoft Azure Azure Kubernetes Service (AKS) MCap: \$824B Microsoft	 Microsoft Azure Azure Stack (ACS) Engine ★ 4 MCap: \$824B Microsoft
 Baidu Cloud Baidu Cloud Container Engine MCap: \$65.7B Baidu	 博云 BoCloud BoCloud BeyondcentContainer BoCloud	 catalyst cloud Catalyst Kubernetes Service ★ 2 Catalyst Cloud	 CISCO Cisco Container Platform MCap: \$214B Cisco	 EasyStack open cloud computing EasyStack Kubernetes Service (EKS) Funding: \$110M EasyStack
 eBaoCloud® enable connected insurance eBaoCloud eBaoTech Corporation	 eKing Technology 易 建 科 技 eKing Cloud Container Platform Hainan eKing Technology	 Google Kubernetes Engine Google Kubernetes Engine (GKE) MCap: \$744B Google	 谐云科技 HARMONY CLOUD HarmonyCloud Container Platform Hangzhou Harmony Technology	 HASURA Hasura Funding: \$1.6M Hasura
 HUAWEI Huawei Cloud Container Engine (CCE) Huawei	 IBM Cloud Kubernetes Service IBM Cloud Kubernetes Service MCap: \$110B IBM	 nirmata Nirmata Managed Kubernetes Nirmata	 NUTANIX™ Nutanix Karbon MCap: \$7.62B Nutanix	 ORACLE Oracle Container Engine MCap: \$192B Oracle

ตัวอย่างพาร์ทเนอร์ของ Kubernetes บางส่วน

เมื่อไรที่ควรใช้ Kubernetes

แน่นอนว่า Kubernetes (หรือ container orchestration) ไม่ใช่โซลูชันที่เหมาะสมสำหรับทุกคน แคมการใช้งานยังต้องแลกมาด้วยความซับซ้อนของระบบที่เพิ่มจากเดิมมาก

รูปแบบงานที่เหมาะสมกับการทำ container orchestration จึงควรมีปัจจัยเหล่านี้

- แอปพลิเคชันที่ใช้งาน ต้องถูกปรับให้อยู่ในสภาพแวดล้อมแบบ container มาก่อนแล้ว ถึงค่อยมาทำ orchestration ในภายหลัง หากเป็นแอปพลิเคชันยุคเก่า (legacy) ที่ไม่ได้ปรับให้เป็น container มาก่อน ก็ต้องไปทำให้เสร็จก่อน
- งานมีความต้องการ scalable ขยายขีดความสามารถให้รองรับจำนวนเวิร์คโหนดที่แปรเปลี่ยนไป หากไม่จำเป็นขนาดนั้น ก็ใช้วิธีรันงานใน container เพียงอย่างเดียว (docker-compose) ไปก่อน แล้วค่อยขยับไปใช้ Kubernetes ในภายหลังได้ ซึ่งตรงนี้มีเครื่องมืออย่าง [Kompose](#) ช่วยแปลงอยู่แล้ว
- ตัวบริการมีขนาดใหญ่เกินกำลังของเครื่องเดียว (หรือ VM ตัวเดียว) จึงถูกบังคับให้ต้องต่อเครื่องเป็นคลัสเตอร์เพื่อรองรับผู้ใช้งาน กรณีแบบนี้จึงเหมาะกับการทำ orchestration

สรุปง่ายๆ ว่าถ้างานของเรา "ไม่ใหญ่พอ" ที่จะต้องใช้ Kubernetes ก็ไม่จำเป็นต้องใช้ Kubernetes นั่นเอง



Kubernetes Features

Service discovery and load balancing

No need to modify your application to use an unfamiliar service discovery mechanism. Kubernetes gives containers their own IP addresses and a single DNS name for a set of containers, and can load-balance across them.

Storage orchestration

Automatically mount the storage system of your choice, whether from local storage, a public cloud provider such as [GCP](#) or [AWS](#), or a network storage system such as NFS, iSCSI, Gluster, Ceph, Cinder, or Flocker.

Automated rollouts and rollbacks

Kubernetes progressively rolls out changes to your application or its configuration, while monitoring application health to ensure it doesn't kill all your instances at the same time. If something goes wrong, Kubernetes will rollback the change for you. Take advantage of a growing ecosystem of deployment solutions.

Batch execution

In addition to services, Kubernetes can manage your batch and CI workloads, replacing containers that fail, if desired.

Automatic binpacking

Automatically places containers based on their resource requirements and other constraints, while not sacrificing availability. Mix critical and best-effort workloads in order to drive up utilization and save even more resources.

Self-healing

Restarts containers that fail, replaces and reschedules containers when nodes die, kills containers that don't respond to your user-defined health check, and doesn't advertise them to clients until they are ready to serve.

Secret and configuration management

Deploy and update secrets and application configuration without rebuilding your image and without exposing secrets in your stack configuration.

Horizontal scaling

Scale your application up and down with a simple command, with a UI, or automatically based on CPU usage.

สื่อนาคต Cloud Native

ซอฟต์แวร์อย่าง Kubernetes เกิดขึ้นมาสำหรับยุคของคลาวด์ จึงเกิดคำเรียกซอฟต์แวร์กลุ่มนี้ว่า Cloud Native

กูเกิลเองก็ใช้คำนี้ไปตั้งชื่อมูลนิธิ Cloud Native Computing Foundation (CNCF) ซึ่งนอกจาก Kubernetes แล้ว ยังมีซอฟต์แวร์ตัวอื่นๆ ที่เกี่ยวข้องกันอีกหลายตัว เช่น

- Prometheus (monitoring)
- OpenTracing (tracing)
- Fluentd (logging)
- Containerd (container runtime) จาก Docker
- Rkt (container runtime) จาก CoreOS
- Envoy (service proxy)
- CoreDNS (service discovery)
- Linkerd (service mesh)

นอกจากนี้ยังมีซอฟต์แวร์ตัวอื่นๆ ที่อยู่นอก CNCF แต่พัฒนาโดยกูเกิลและพันธมิตร เช่น Istio ที่ร่วมกับ IBM และ Red Hat โดยเรียกใช้ซอฟต์แวร์บางตัวจาก CNCF เช่น Envoy อีกทีหนึ่ง

ซอฟต์แวร์เหล่านี้นำมาใช้ร่วมกันเพื่อให้การสร้าง container orchestration สมบูรณ์แบบมากขึ้น เพิ่มส่วนการบริหารจัดการ, monitoring, logging เข้ามา และการวางโครงสร้างพื้นฐานด้านการประมวลผลยุคคลาวด์ในอนาคต ซึ่งถ้ามีโอกาสเราจะหยิบมันมาเล่าถึงต่อไป



Kubernetes
Orchestration



Prometheus
Monitoring

Incubating



OpenTracing
Distributed Tracing
API



Fluentd
Logging



gRPC
Remote Procedure
Call



containerd
Container Runtime



rkt
Container Runtime



CNI
Networking API



Envoy
Service Proxy



Jaeger
Distributed Tracing



Notary
Security



TUF
Software Update
Spec



Vitess
Storage

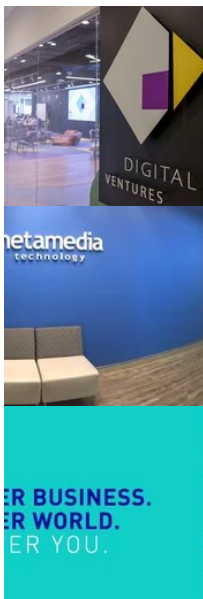


CoreDNS
Service Discovery



Get latest news from Blognone [Follow @twitterapi](#)

Hiring! บริษัทที่น่าสนใจ



Digital Ventures Co., Ltd.

Founded in 2016, a subsidiary of Siam Commercial Bank to promote innovation in FinTech

Metamedia Technology

บริษัทซอฟต์แวร์สัญชาติไทย เจ้าของเว็บ Longdo.com พจนานุกรม แผนที่ ออนไลน์

Unilever Thai Trading

Unilever Thai Group of Companies is a fast-moving consumer goods company.

Comments

By: jimmyis on 17 November 2018 - 17:10 #1082460

ออกเสียงว่า คูเบอร์เนเตส เหรอครับ

ดูจากวิดีโอมาหลายอัน เห็นฝรั่งออกเสียง คูเบอร์เนทิส กัน

By: iamz   on 19 November 2018 - 08:54 #1082617 Reply to:1082460

เคยเห็นคนกรีกมาตอบครับว่าอ่านไปเถอะแบบไหนก็ได้ ถ้าไม่ใช่คนกรีกผิดทุกคน 55

By: Mario on 20 November 2018 - 13:18 #1082890 Reply to:1082460

tes ออกเสียงสั้นมากๆ จะฟังออกเป็นดิสครับ
ผมว่าจั้น

By: deaw on 17 November 2018 - 18:25 #1082463

ตอนนี้ อ่านมันส์มากครับ
ผมชอบอ่าน 2 Tags นี้ครับ Special Report กับ In-Depth

By: 9rockky   on 17 November 2018 - 18:51 #1082466

บางงานก็เหมาะกับ serverless ข้าม k8s ไปเลย

By: Neroroms  on 17 November 2018 - 18:57 #1082467

ผมติดเรียกชื่อสั้นๆว่า คับ (Kube) ไปเลยละ

By: sammyskywalker     on 17 November 2018 - 19:02 #1082469

Pokemon Go ก็รันอยู่บน Kubernetes -> Community Day ล่มติดกันมาสองครั้งละ มันไม่ auto scale หรือ Nia มัน
งกไม่ยอม scale รอ

By: TW   on 19 November 2018 - 16:01 #1082689

"...และขนย้ายมันไปใช้ตามที่ต่างๆ ได้งานขึ้น"
ได้งานขึ้น?

By: Mypandacm on 7 January 2019 - 11:24 #1090612 Reply to:1082689

ถ้าได้งานขึ้น นี่คิดหนักเลย

By: Kittichok  on 21 November 2018 - 00:27 #1083013

ขอบคุณที่เขียนบทความนี้ให้อ่าน เข้าใจภาพรวมมากขึ้น ตอนหาข้อมูลเอง เจอแต่ข้อมูลเป็นรายตัวเลยไม่เข้าใจภาพ
รวมเท่าไร

ผมมีอ่านสะดุดช่วงหนึ่งจากการใช้เครื่องหมายคือ "(ในโลกของ Kubernetes ใช้คำว่า Node)" ซึ่งข้อความในวงเล็บจะใช้ในการขยายหรืออธิบายข้อความเพิ่มเติม โดยที่ไม่มีผลต่อเนื้อความนอกวงเล็บ แต่หลังจากวงเล็บนี้ก็มีการใช้คำว่า Node ถึง 5 ครั้งและโหนดอีก 1 ครั้ง ผมจึงแนะนำว่าเราวงเล็บออกจะเป็นลักษณะข้อความที่ถูกต้อง

By: Nawawishkid on 23 November 2018 - 00:15 #1083386

ขอบคุณครับ เข้าใจง่ายดีครับ

By: btoy  on 12 December 2018 - 09:22 #1086638

ขอบคุณมากๆเลยครับ

...: เรื่อยไป

sign in

ลงทะเบียน ลืมรหัสผ่าน

Username: *

Password: *

Log in

Blognone Jobs Premium



Senior Back-end Developer

฿40,000 - ฿80,000 Conicle Co., Ltd. - พญาไท กรุงเทพมหานคร



AI Specialist

฿38,000 - ฿80,000 ConvoLab - พญาไท กรุงเทพมหานคร



Backend Developer (Junior/Senior)

฿30,000 - ฿80,000 Bluebik Group Co.,Ltd. - บางรัก กรุงเทพมหานคร



Team Lead (Laravel or Python)

฿50,000 - ฿100,000 PeerPower - บางรัก กรุงเทพมหานคร