

Azure Data Pipeline Engineering

Framing the Journey

1



Develop a passion for
learning.

Overview



This course will teach you the fundamentals of running data engineering in Azure. You will learn best practices for deploying containerized applications using Docker and Azure Kubernetes Service (AKS). This course also dives deep into automating deployment of applications using CI/CD tools like Jenkins, and Helm.

Overview

In this course you will learn the following:

- Azure foundation
- Azure data engineering services
- Deploy applications on Azure
- Docker & Containerization
- Manage containerized applications
- DevOps automation
- Data pipeline observability
- Azure Kubernetes Service
- CI/CD
- DevSecOps
- Performance monitoring
- PaaS vs FaaS
- Disaster Recovery



LOGISTICS



Class Hours:

- Start time is X:XXam
- End time is X:XXpm
- Class times may vary slightly for specific classes
- Breaks mid-morning and afternoon (15 minutes)



Lunch:

- Lunch is noon-ish for 75 minutes
- Yes, 1 hour and 15 minutes
- Extra time for email, phone calls, or simply a walk.



Telecommunication:

- Turn off or set electronic devices to vibrate
- Reading or attending to devices can be distracting to other students
- Try to delay until breaks or after class

Miscellaneous

- Courseware
- Bathroom
- Fire drills



Azure & Data Engineering

- Azure infrastructure
- Azure data engineering services
- Deploy applications on Azure
- Observability



Containers & Orchestration

- Docker
- Containerization
- Kubernetes
- AKS
- Deploying Containers



CI/CD, DevSecOps & SRE

- DevSecOps
- SRE
- Deploy applications on Azure
- CI/CD
- Jenkins
- Resiliency
- Helm

Meet The Instructor

George Niece AKA geo

Digital Transformation, DevSecOps, IoT Consultant with a Software Engineering, Digital Commerce, and IT operations background. Focused on cloud-native application modernization, datacenter divestiture, and cloud adoption.



Twitter
[@georgeniece](https://twitter.com/georgeniece)

LinkedIn
[Linkedin.com/in/GeorgeNiece](https://www.linkedin.com/in/GeorgeNiece)

mail
George.Niece@DigitalTransformationStrategies.net

Expertise

- Cloud
- XaaS
- AppDev
- IoT
- Automation
- CI/CD
- Microservices
- Agile

GTKY

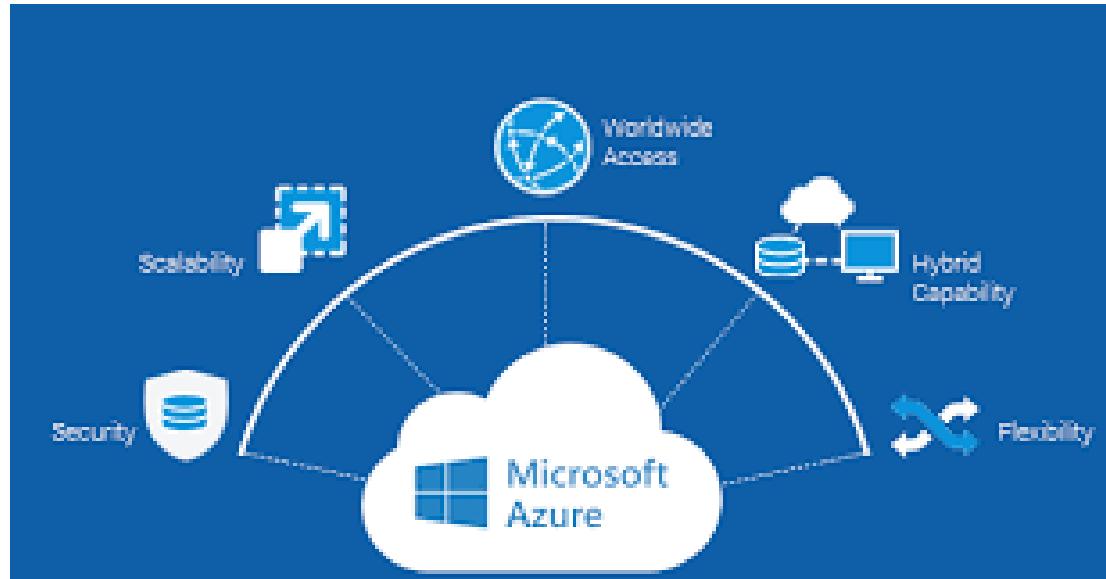
Please let me know your

- Name
- Where you work from?
- Experience with
 - Azure
 - Data Engineering
 - Kubernetes
 - CI/CD

Please rate yourself (1-5) 1 = new joiner, 5 = SME

Lastly, what is your favorite dessert?

Azure



The Azure cloud platform is more than 200 products and cloud services designed to help you bring new solutions to life—to solve today's challenges and create the future. Build, run, and manage applications across multiple clouds, on-premises, and at the edge, with the tools and frameworks of your choice.

Azure Infrastructure

Azure global infrastructure is made up of two key components—physical infrastructure and connective network components. The physical component is comprised of 200+ physical datacenters, arranged into regions, and linked by one of the largest interconnected networks on the planet.

With the connectivity of the global Azure network, each of the Azure datacenters provides high availability, low latency, scalability, and the latest advancements in cloud infrastructure—all running on the Azure platform.

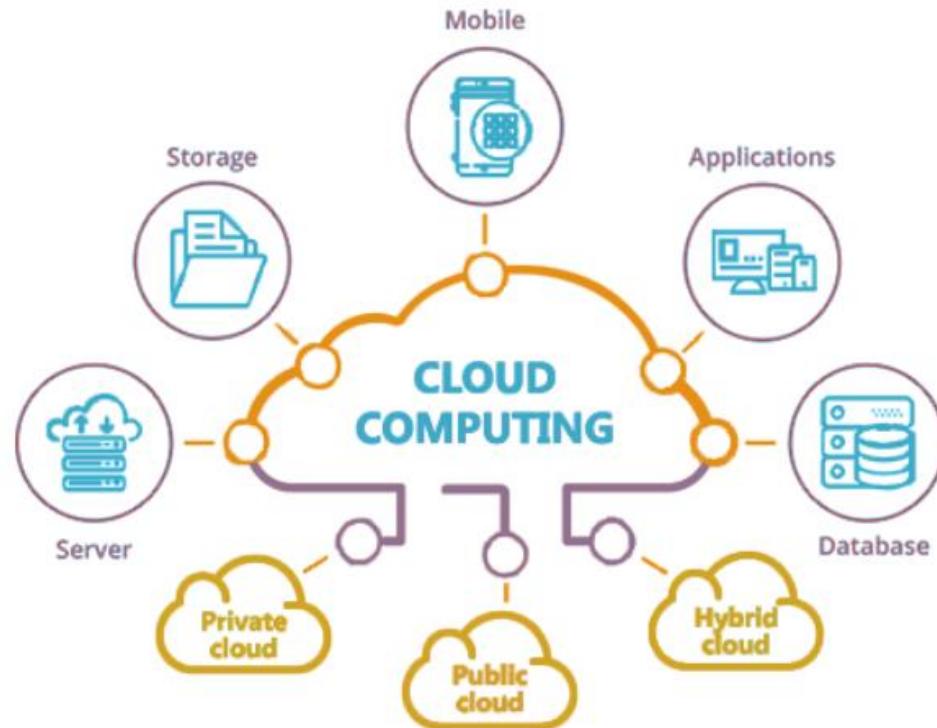
Services

- Application development
- AI
- Cloud migration and modernization
- Data and analytics
- Hybrid cloud and infrastructure
- Internet of Things
- Security and governance
- Industry solutions

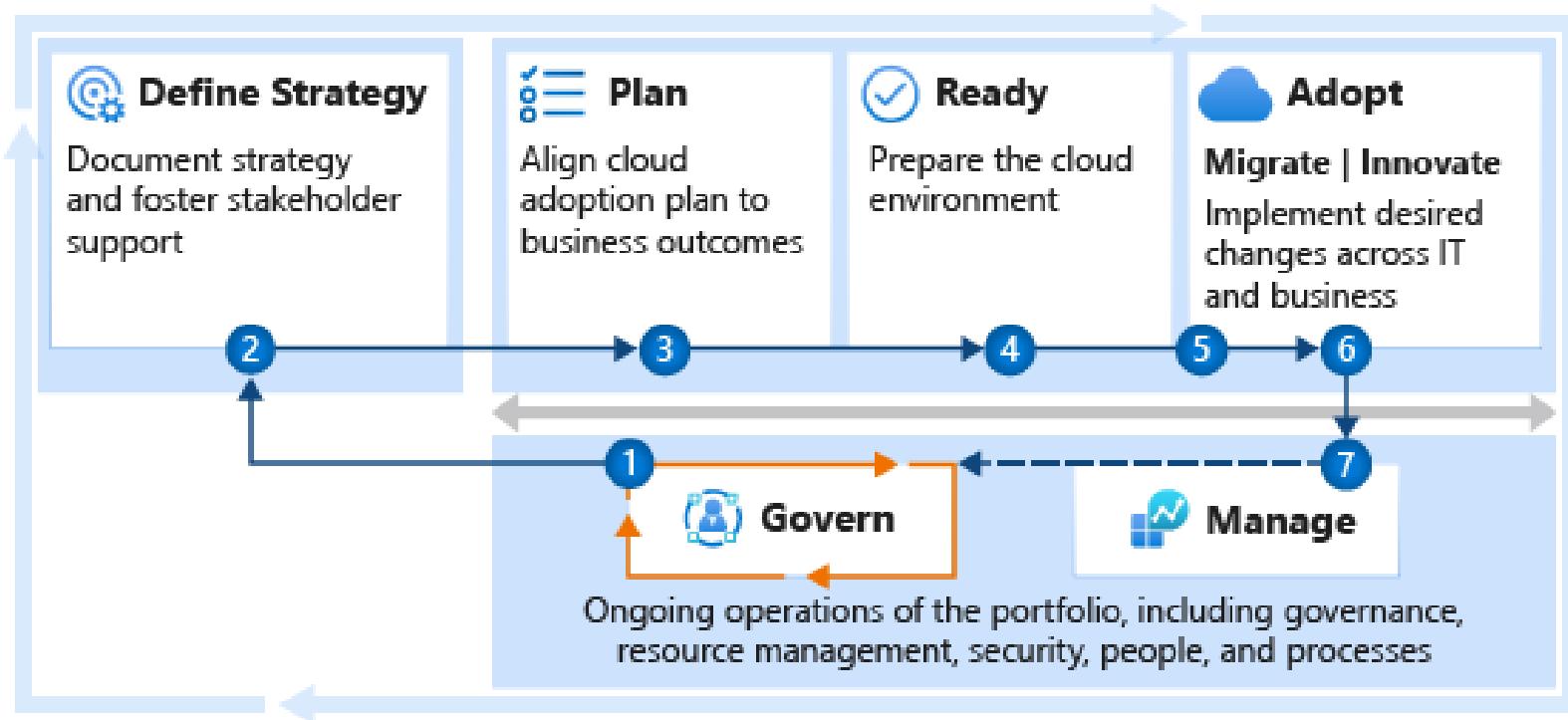
What is Azure Cloud Computing?

These services were broadly divided into three categories by NIST: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS). The name cloud computing was inspired by the cloud symbol that's often used to represent the Internet in flowcharts and diagrams.

Much has changed from the Cloud early days but still many organizations are only beginning their Cloud Adoption Journeys



Azure Cloud Adoption Framework



Azure Cloud Governance

Governance models for Cloud focus on auditing and assessing the system specific controls (security in the cloud).

CSCs can leverage a CSP's third-party attestations as evidence or proof that security requirements associated with CSP have been satisfied. Depending on the CSP, CSCs can achieve continuous compliance through security services from their CSP instead of point in time assessments.

Applying Cloud focus reduces Duplicative audits that have already been completed by independent third-party auditors. Additionally, there are a wide variety of scopes when auditing cloud, from just a specific workload to enterprise cloud environments.

Azure Cloud Financial Management



Azure Infrastructure Foundation

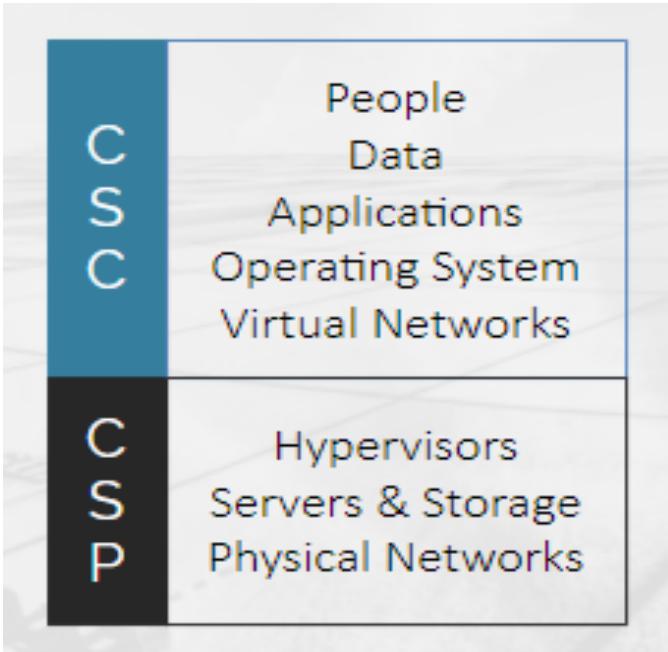


Infrastructure as a Services (IaaS)

Infrastructure as a service run like on-premise IT servers and personal machines, storage, networks, etc.



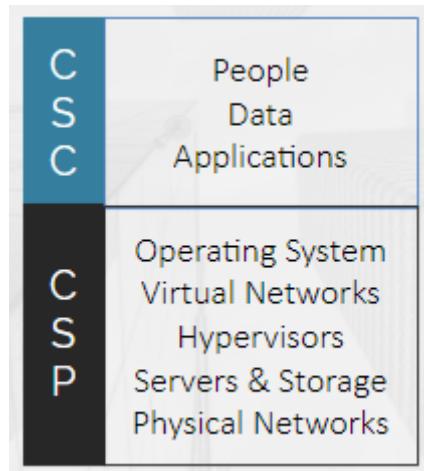
IaaS – Lift and Shift



Internet-based access to storage and computing power. Cloud computing most like on-premise, IaaS lets you rent IT infrastructure - servers and virtual machines, storage, networks, and operating systems - from a cloud provider on a pay-as-you-go basis.

Platform as a Service (PaaS)

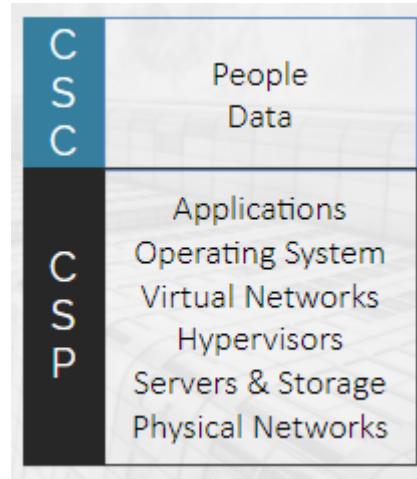
Supplies on-demand environments for developing, testing and delivering software applications.



gives developers the tools to build and host web applications. PaaS is designed to give users access to the components they require to quickly develop and operate web or mobile applications over the Internet, without worrying about setting up or managing the underlying infrastructure of servers, storage, networks, and databases.

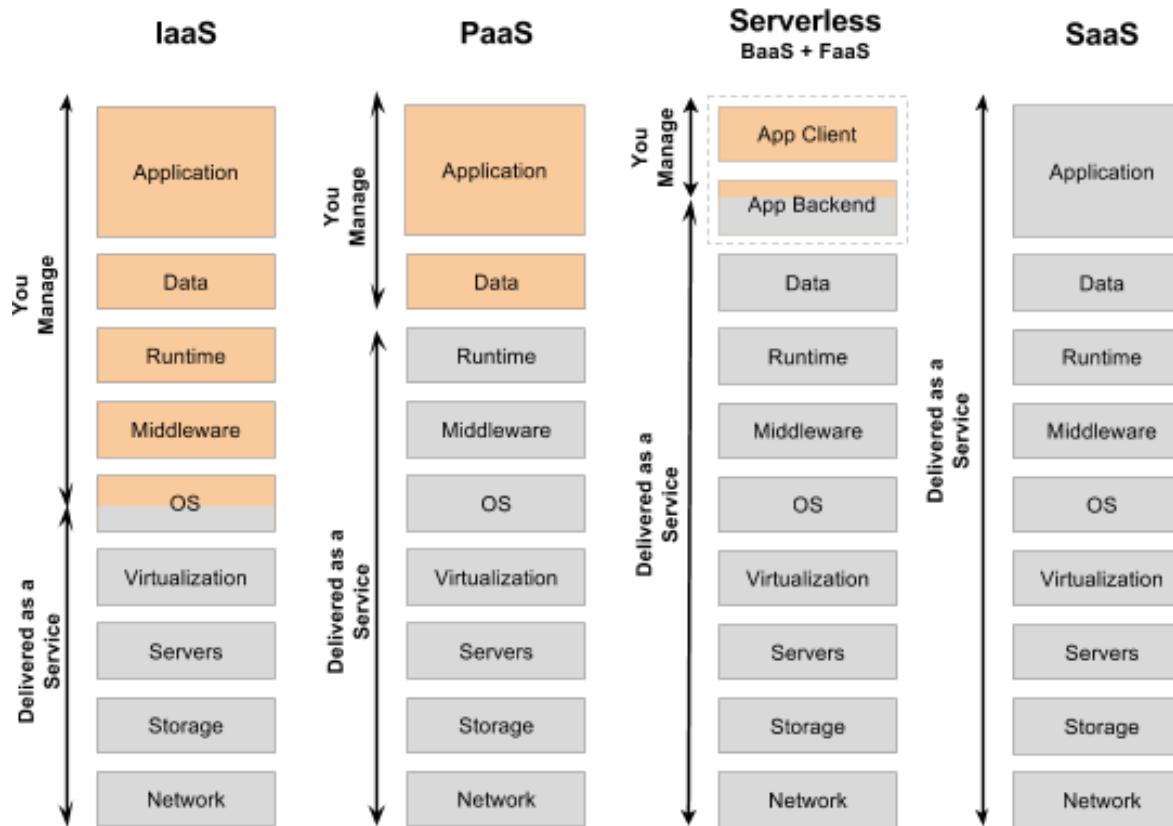
Software as a Service (SaaS)

Offers on-demand pay-per use of application software. Users connect to application over the internet instead of their personal machines.



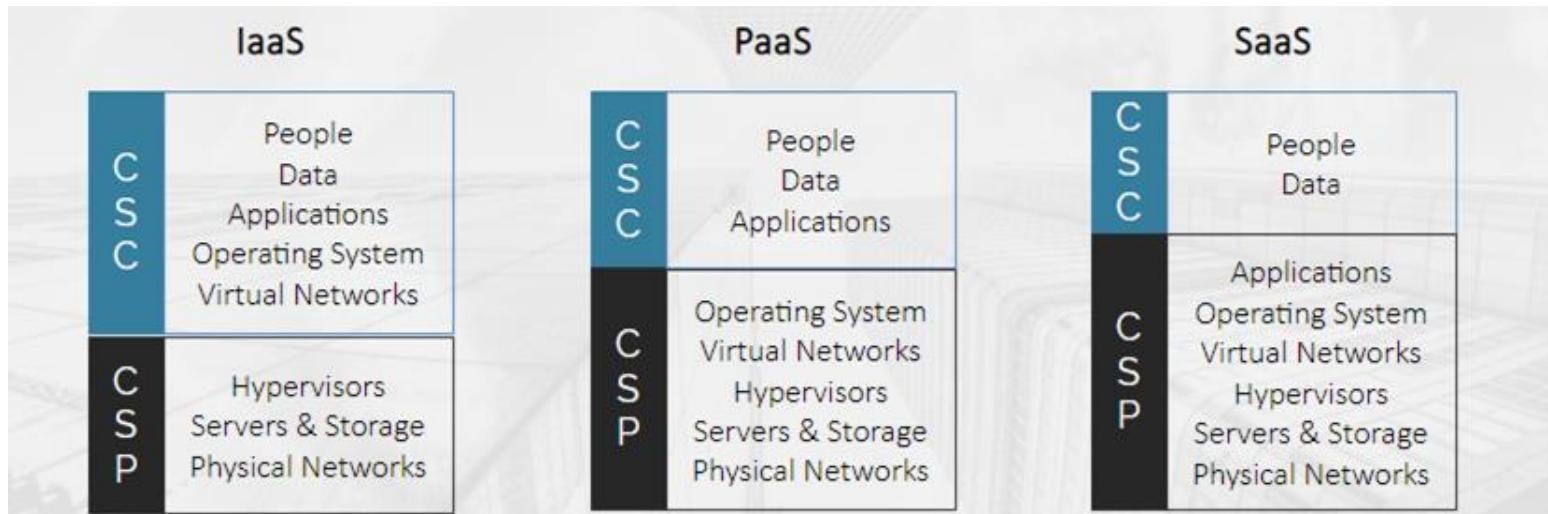
is used for web-based applications. SaaS is a method for delivering software applications over the Internet where cloud providers host and manage the software applications making it easier to have the same application on all of your devices at once by accessing it in the cloud.

IaaS vs PaaS vs FaaS vs SaaS



Shared Responsibility

The difference in auditing cloud versus on-premises access management is the access management is a shared responsibility in the cloud



Shared Responsibility Azure - aaS

Responsibility	SaaS	PaaS	IaaS	U/I- prem
Information and data				
Devices (Mobile and PCs)				
Accounts and identities				
Identity and directory infrastructure				
Applications				
Network controls				
Operating system				
Physical hosts				
Physical network				
Physical datacenter				

RESPONSIBILITY ALWAYS RETAINED BY CUSTOMER

RESPONSIBILITY VARIES BY SERVICE TYPE

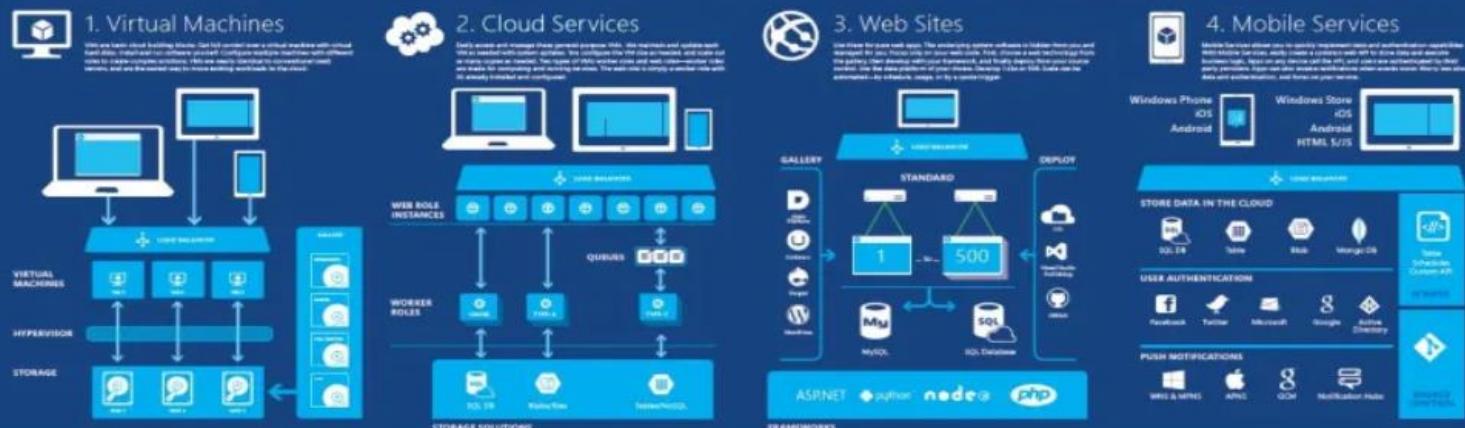
RESPONSIBILITY TRANSFERS TO CLOUD PROVIDER

Microsoft

Customer

Four primary models for building and running apps

Microsoft Azure



Here's just one way to get your first web app with a database running on Azure

Choose from an extensive service catalog

Compute		Data Services		Cloud Services		Storage		SQL Database		HDIgnostic		SQL Data Sync		Hyper-V Recovery Manager		Backup		Cache					
 Virtual Machines	 Web Sites	 Mobile Services	 Cloud Services	 Storage	 SQL Database	 HDIgnostic	 SQL Data Sync	 Hyper-V Recovery Manager	 Backup	 Cache	 Store	 Notification Hubs	 Service Bus	 Visual Studio Online	 Media Services	 BizTalk Services	 Active Directory	 Scheduler	 CDN	 Multi-Factor Authentication	 Virtual Network	 Traffic Manager	 ExpressRoute
Get full control over your virtual machines and manage them from the cloud, and monitor all your instances.	Deliver content with web apps for free. Scale up or down. Build using a range of tools and frameworks.	Add local and mobile capabilities to mobile apps with native cloud support for most mobile devices.	Create and easily make modifications to web sites and mobile cloud services. Perform real-time monitoring and performance management across multiple regions.	Manage data more securely using a range of storage options. Accelerate read and write speeds by replicating your data across multiple regions.	Provision and integrate big data easily with tools including HDFS, Apache Hadoop, and Apache Storm.	Sync data regularly or on-demand between instances of SQL Database and Azure SQL Database.	Protect important servers by monitoring the replication and status of System Center 2012 R2 witness servers for your failover clusters.	Schedule backups of your local server databases using PowerShell cmdlets.	Make your applications available and fast responsive, and help them scale by keeping data close to application logic.	Print, access, and store large volumes of data directly from the cloud.	Deliver millions of push notifications, pull notifications, and real-time updates from your application backend, or push or pull to your application frontends.	Get the messaging, storage, and integration services you need to deliver native mobile, web, and enterprise applications.	Host code, jobs, and batch analysis, and distribute them across multiple regions to deliver better results.	Build B2B connections and facilitate hybrid scenarios by integrating data from the cloud.	Use this set of services to provide an identity and management system for your organization, including directory services, identity governance, security, and application access management.	Create interconnected and isolated regions, and have them share users, and their own unique public keys.	Improve performance by scaling traffic and workloads across multiple regions or different vendor providers.	Provision and manage off-premises data centers and securely connect to on-premises infrastructure.	Load balance incoming global traffic across multiple regions using a single edge provider.	Connect to on-premises infrastructure via a direct connection to Azure Network.	Connect to the public cloud and improve connectivity speed and reliability.		

Azure Portal

The screenshot shows the Microsoft Azure Portal homepage. At the top, there's a blue header bar with the "Microsoft Azure" logo, a search bar containing "Search resources, services, and docs (G+)", and various navigation icons. Below the header is a "DEFAULT DIRECTORY" dropdown and a user profile icon.

The main content area has several sections:

- Azure services:** A row of icons for different Azure services: Create a resource, Service Fabric clusters, Virtual networks, Batch accounts, Azure Cosmos DB, Virtual machines, App Services, Storage accounts, SQL databases, and More services.
- Search:** A large yellow-highlighted search bar.
- Explore:** A yellow-highlighted section with a "Create a resource" button and links for Subscriptions, Resource groups, All resources, and Dashboard.
- Tools:** A row of tools: Microsoft Learn (101), Azure Monitor, Security Center, and Cost Management.
- Useful links:** Links to Technical Documentation, Azure Services, Recent Azure Updates, Azure Migration Tools, Find an Azure expert, Quickstart Center, and more.
- Azure mobile app:** Buttons to download the app from the App Store and Google Play.

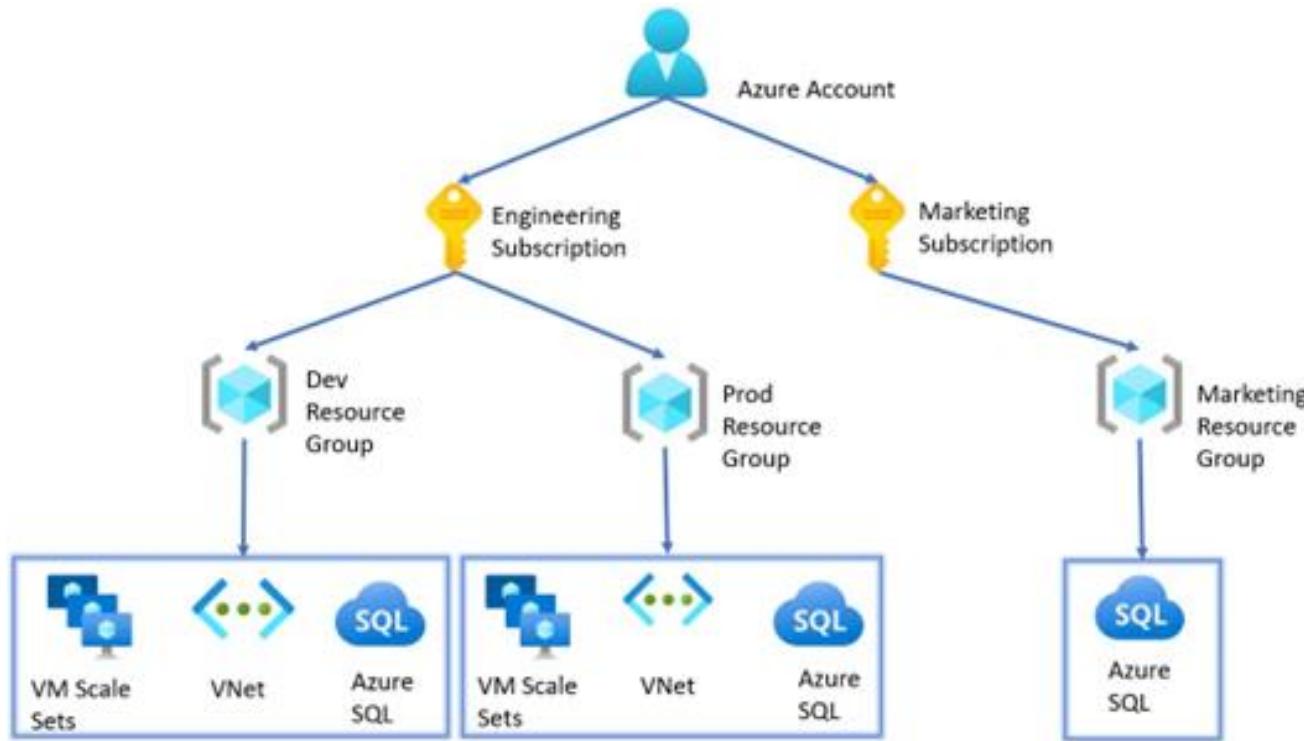
Azure Foundation Components

An **Azure account** refers to the Azure Billing account. It is mapped to the email id that you used to sign up for Azure. An account can contain multiple subscriptions; each of these subscriptions can have multiple resource groups and the resource groups, in turn, can have multiple resources.

Subscriptions are containers for all the resources that are created for applications and solutions under that subscription. A subscription contains the details of all the VMs, networks, storage, and other services that were used during that month that will be used for billing purposes.

Resource groups are logical groups of resources belonging to an application or a team.

Azure Foundation Example



Azure Blade Example

Home > Virtual machines >

Create a virtual machine

...

Basics

Disks

Networking

Management

Advanced

Tags

Review + create

Create a virtual machine that runs Linux or Windows. Select an image from Azure marketplace or use your own customized image. Complete the Basics tab then Review + create to provision a virtual machine with default parameters or review each tab for full customization. [Learn more ↗](#)

Project details

Select the subscription to manage deployed your resources.

Subscription * ⓘ



Resource group * ⓘ

Name *

AzureDataEngineeringPipeline

OK

Cancel

size and manage all

Resource group

My Resource group

Create new

Instance details

Virtual machine name * ⓘ

Review + create

< Previous

Next : Disks >

Managed Services- Azure

Design, Architecture & Migration Services

Cloud Readiness Assessment,
Design & Planning

Cloud Migration

Data Resilience Planning &
Implementation

Cloud Optimization

Azure Managed Services

Cloud Managed Services Platform - Azure Cloud Governance & Service Management

Proactive Monitoring & Incident
Response

IaaS & PaaS Management: Patching,
HA

Data Resilience:
Replication, Backups, DR etc.

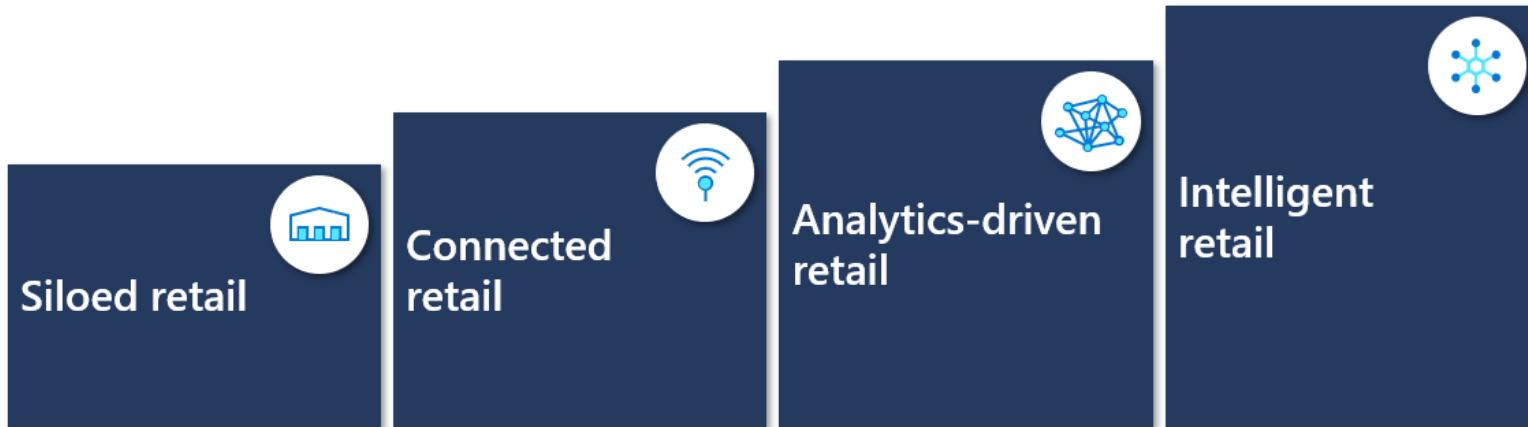
Networking, Load Balancer
and Firewall Management

Comprehensive Security Management:
AV, IDS, IPS, Log Management & More

HIPAA-Compliant Solutions

Cloud Maturity – Azure in Retail

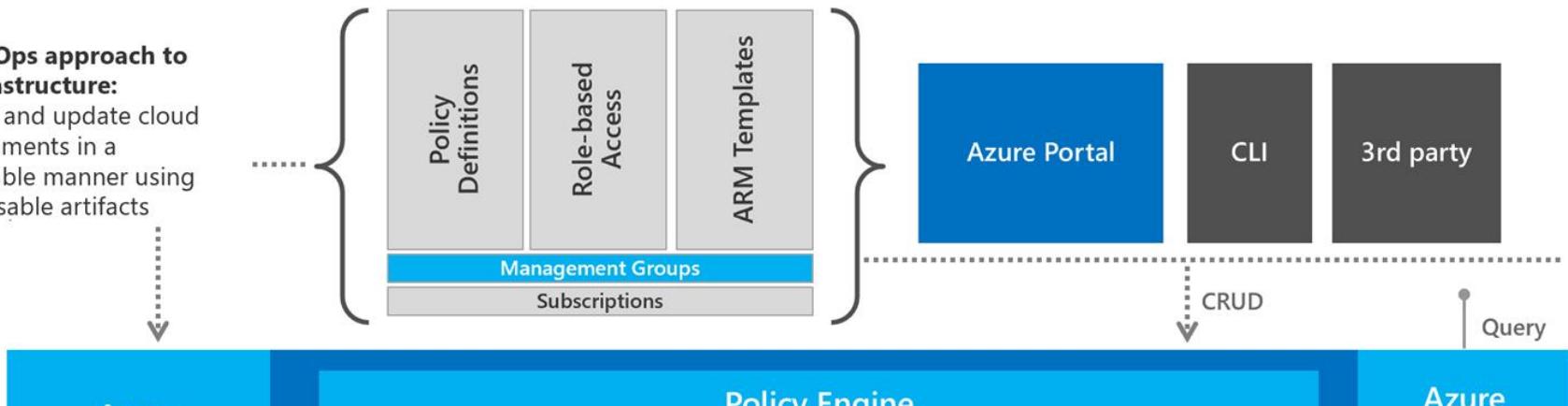
The stages of digital retail maturity



Cloud Policy Management - Azure

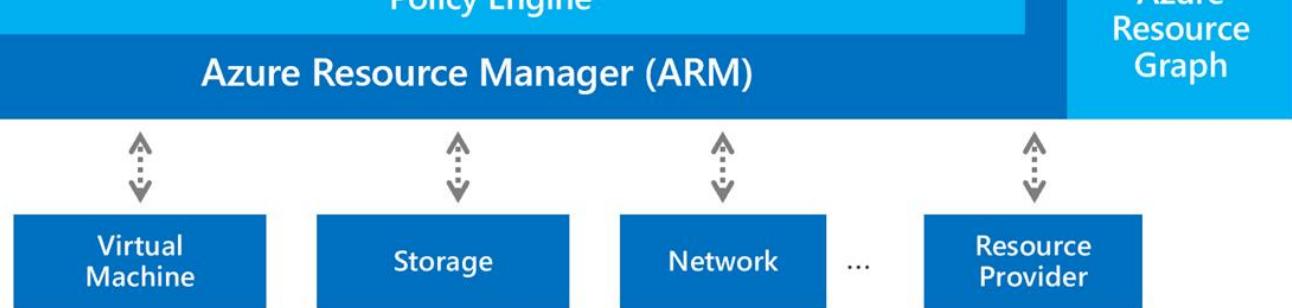
1. DevOps approach to Infrastructure:

Deploy and update cloud environments in a repeatable manner using composable artifacts



2. Policy-based Control: Real-time enforcement, compliance assessment and remediation at scale

3. Resource Visibility: Query, explore & analyze cloud resources at scale



Compute

Whether you're building new applications or deploying existing ones, Azure compute provides the infrastructure you need to run your apps. Tap in to compute capacity in the cloud and scale on demand. Containerize your applications, deploy Windows and Linux virtual machines (VMs), and take advantage of flexible options for migrating VMs to Azure. With comprehensive support for hybrid environments, deploy how and where you want to. Azure compute also includes a full-fledged identity solution, so you gain managed end-point protection, and Active Directory support that helps secure access to on-premises and cloud apps.

Azure Compute Products

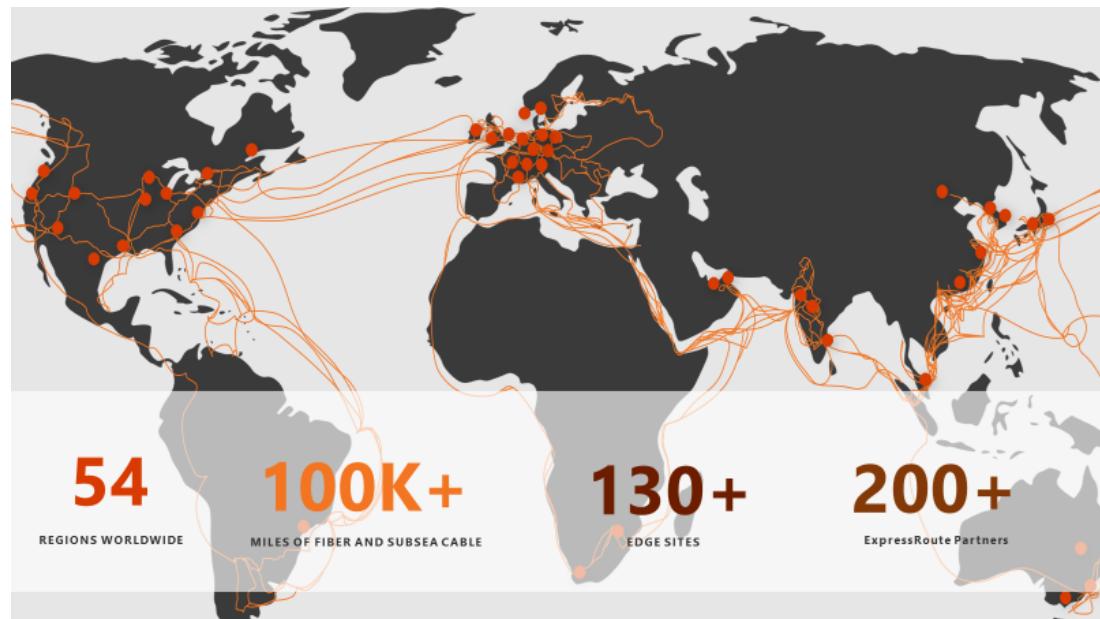
- Virtual Machines
- Virtual machine Scale Sets
- Azure Spot Virtual Machines
- Azure Kubernetes Service (AKS)
- Azure Functions
- Azure Service Fabric
- App Service
- Azure Container Instances
- Batch
- Cloud Services
- Azure Dedicated Host

Secure network infrastructure:

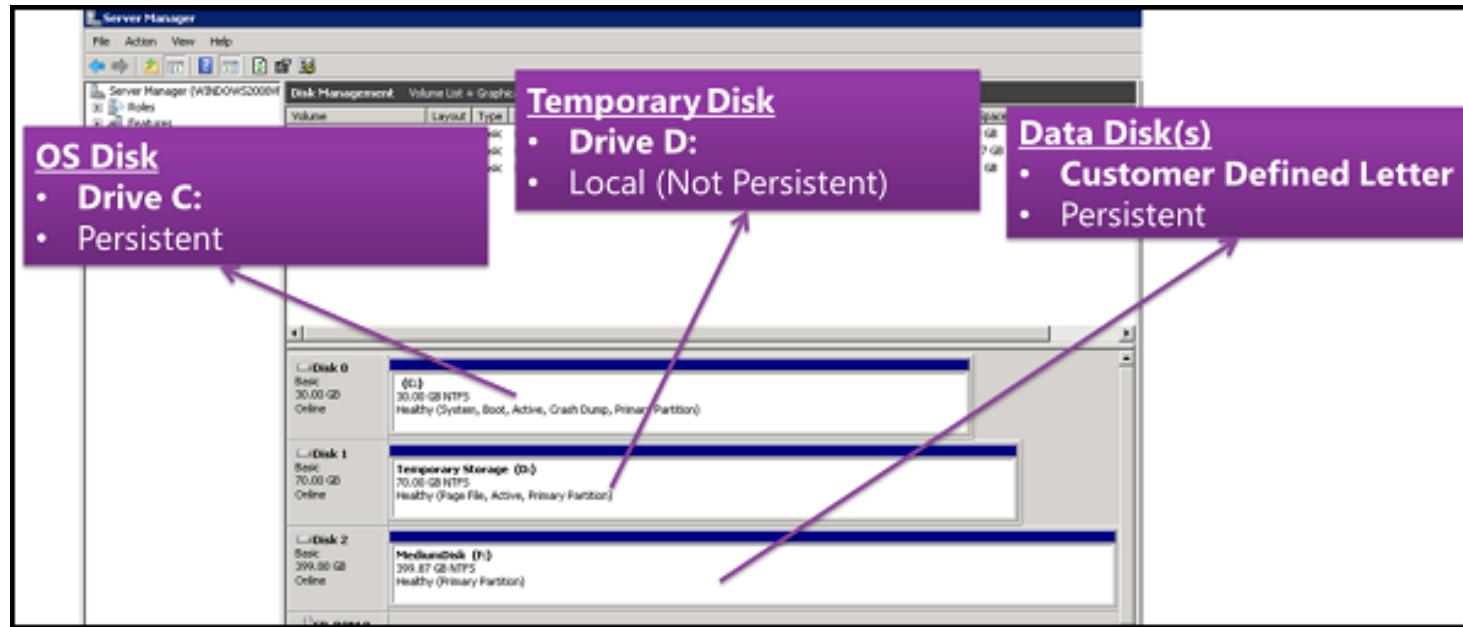
Build and protect your virtual network infrastructure.

If you want to	Use this
Connect everything from virtual machines to incoming VPN connections	Azure Virtual Network
Balance inbound and outbound connections and requests to applications	Azure Load Balancer
Protect your applications from DDoS attacks	Azure DDoS Protection
Native firewall capabilities with built-in high availability and zero maintenance	Azure Firewall
Manage network security policy and routing centrally	Azure Firewall Manager
Private and fully managed RDP and SSH access to your virtual machines	Azure Bastion
Private access to services hosted on the Azure platform	Azure Private Link
Route incoming traffic for better performance and availability	Traffic Manager
Monitor and diagnose network issues	Network Watcher
Extend Azure management for deploying 5G and SD-WAN network functions on edge devices	Azure Network Function Manager

Networking as a Service (NaaS)



Azure Managed Disks



Azure Virtual Network (VNET)



Build a hybrid infrastructure that you control



Establish sophisticated network topologies



Bring your own IP addresses and DNS servers

Azure VM Scale Sets

Scenario	Manual group of VMs	Virtual machine scale set
Add extra VM instances	Manual process to create, configure, and ensure compliance	Automatically create from central configuration
Traffic balancing and distribution	Manual process to create and configure Azure load balancer or Application Gateway	Can automatically create and integrate with Azure load balancer or Application Gateway
High availability and redundancy	Manually create Availability Set or distribute and track VMs across Availability Zones	Automatic distribution of VM instances across Availability Zones or Availability Sets
Scaling of VMs	Manual monitoring and Azure Automation	Autoscale based on host metrics, in-guest metrics, Application Insights, or schedule

Azure Functions



Automated and flexible scaling



Integrated programming model



End-to-end development
experience



Variety of programming languages
and hosting options

Azure Kubernetes Service



Elastic provisioning of capacity without the need to manage the infrastructure and with the ability to add event-driven autoscaling and triggers through [KEDA](#)



Faster end-to-end development experience through [Visual Studio Code](#) [Kubernetes tools](#), [Azure DevOps](#), and [Azure Monitor](#)



Most comprehensive authentication and authorization capabilities using [Azure Active Directory](#), and dynamic rules enforcement across multiple clusters with [Azure Policy](#).



Availability in more regions than any other cloud provider

Azure Data Foundation



—3

Azure Blob Storage



Scalable, durable, and available



Secured



Optimized for data lakes



Comprehensive data management

Azure Files



Serverless file shares



Built for Hybrid with File Sync



Optimized TCO

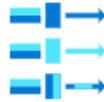


Multiple protocols support

Azure Queues



Decouple components



Build in resilience



Scale for bursts

Azure Data Lake



Limitless scale and 16 9s of data durability
with automatic geo-replication



Single storage platform for ingestion,
processing, and visualization that
supports the most common analytics
frameworks



Highly secure with flexible mechanisms
for protection across data access,
encryption, and network-level control



Cost optimization via independent scaling
of storage and compute, lifecycle policy
management, and object-level tiering

Azure Tables



Store petabytes of structured data



Made for enterprise



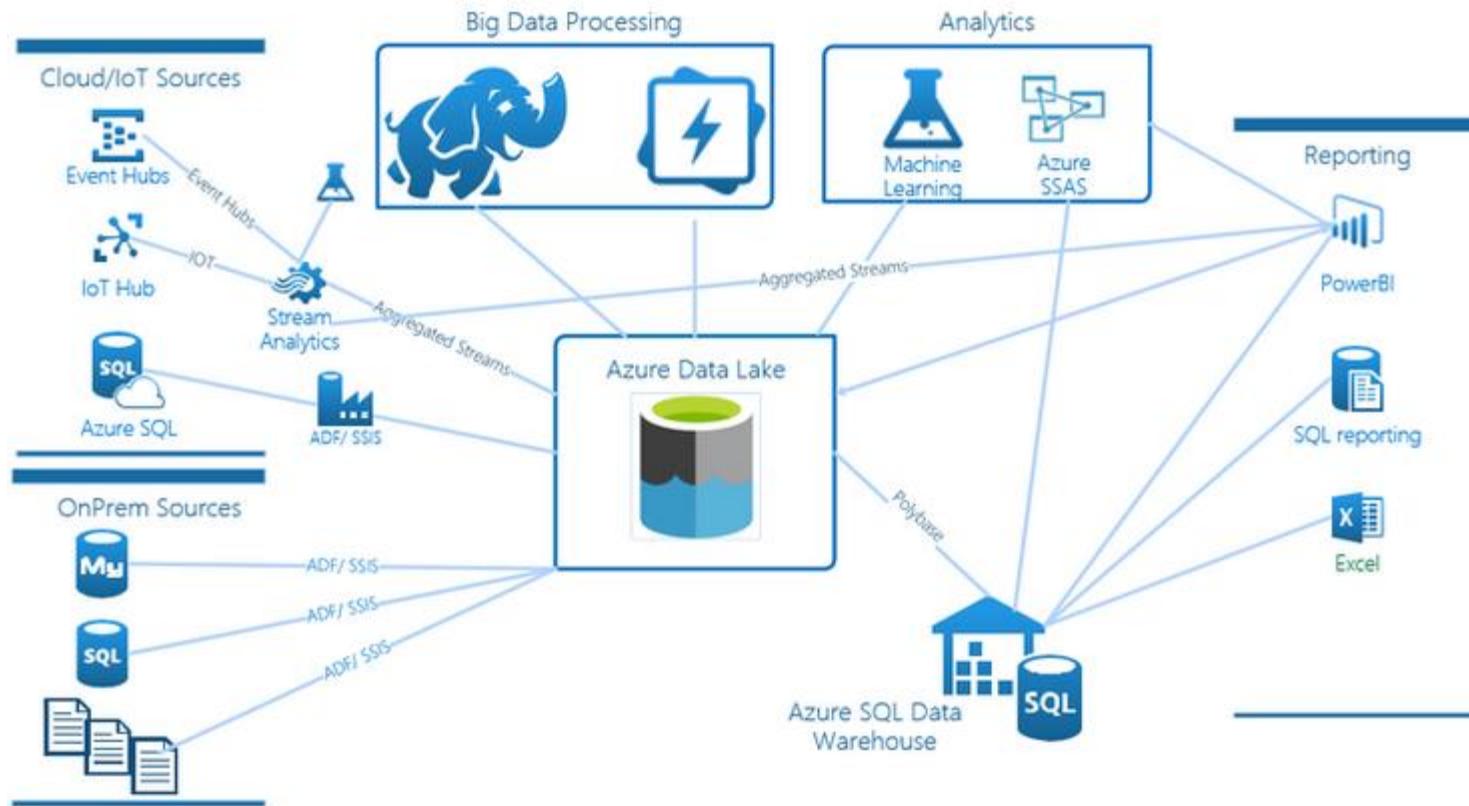
Supports flexible data schema



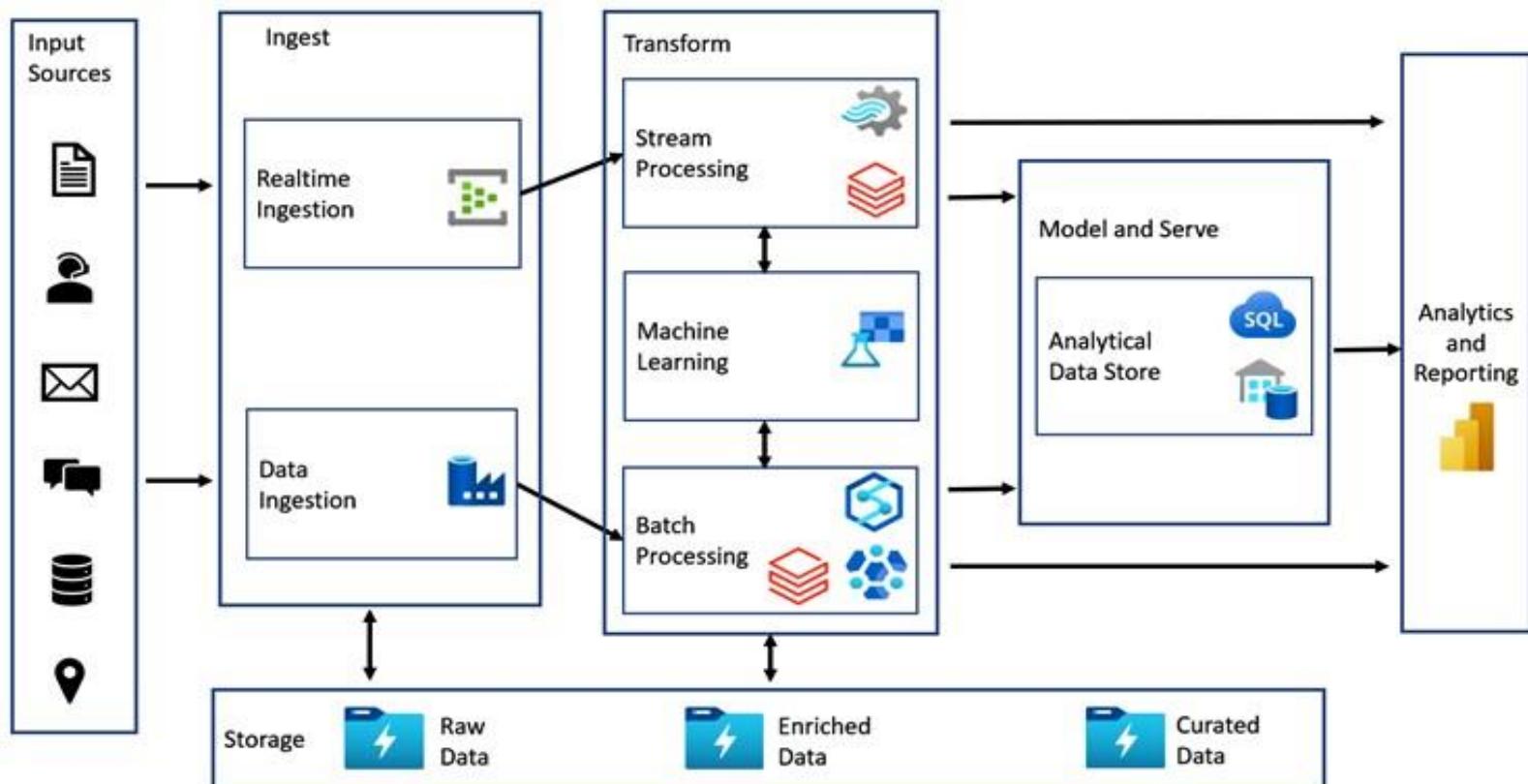
Designed for developers



Data Lake Functions



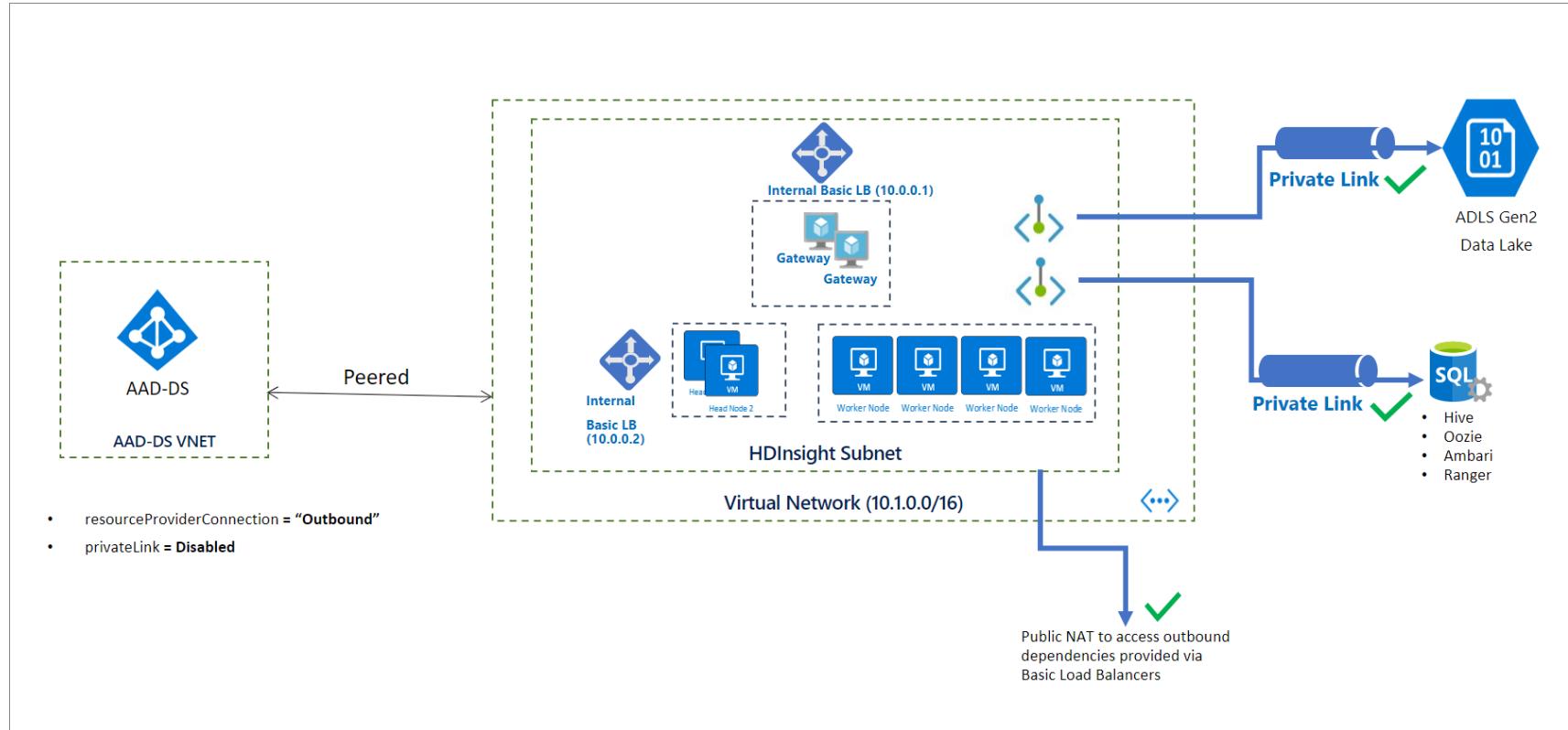
Data Lake Components



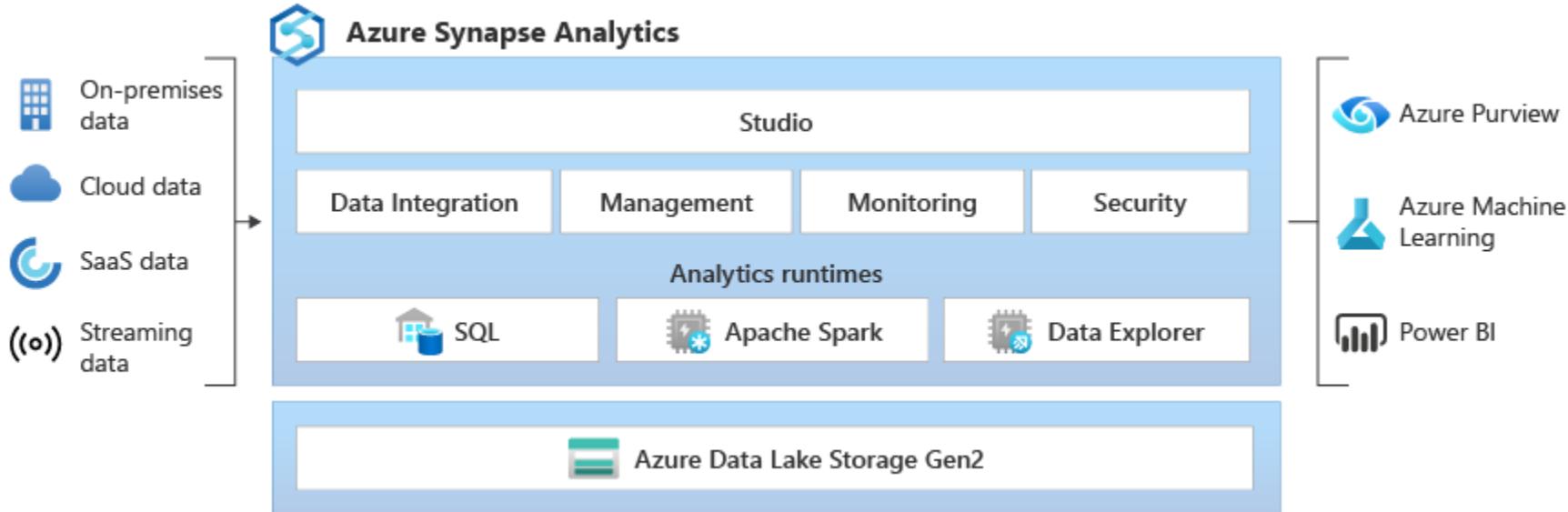
HDInsight



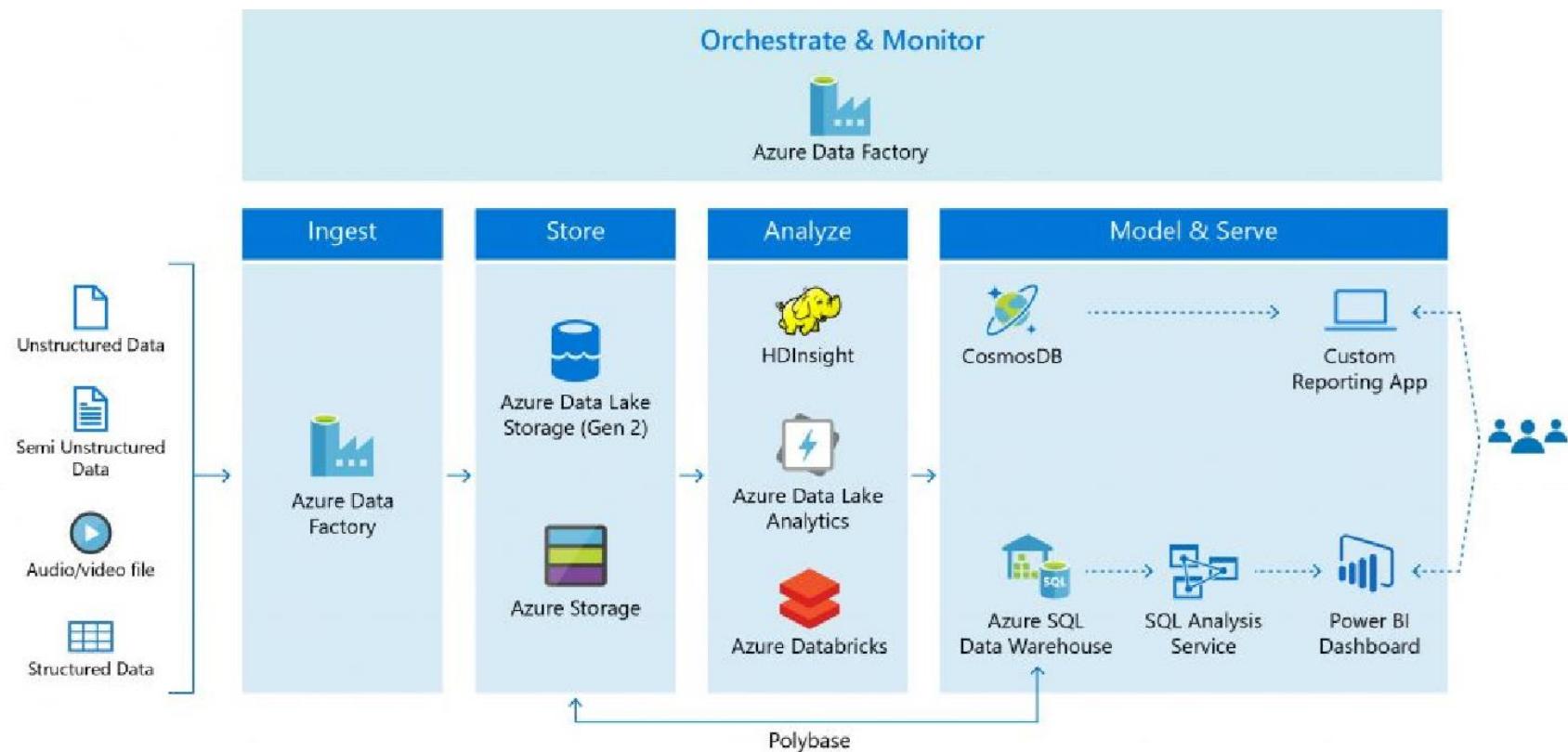
Cluster HDINSIGHT



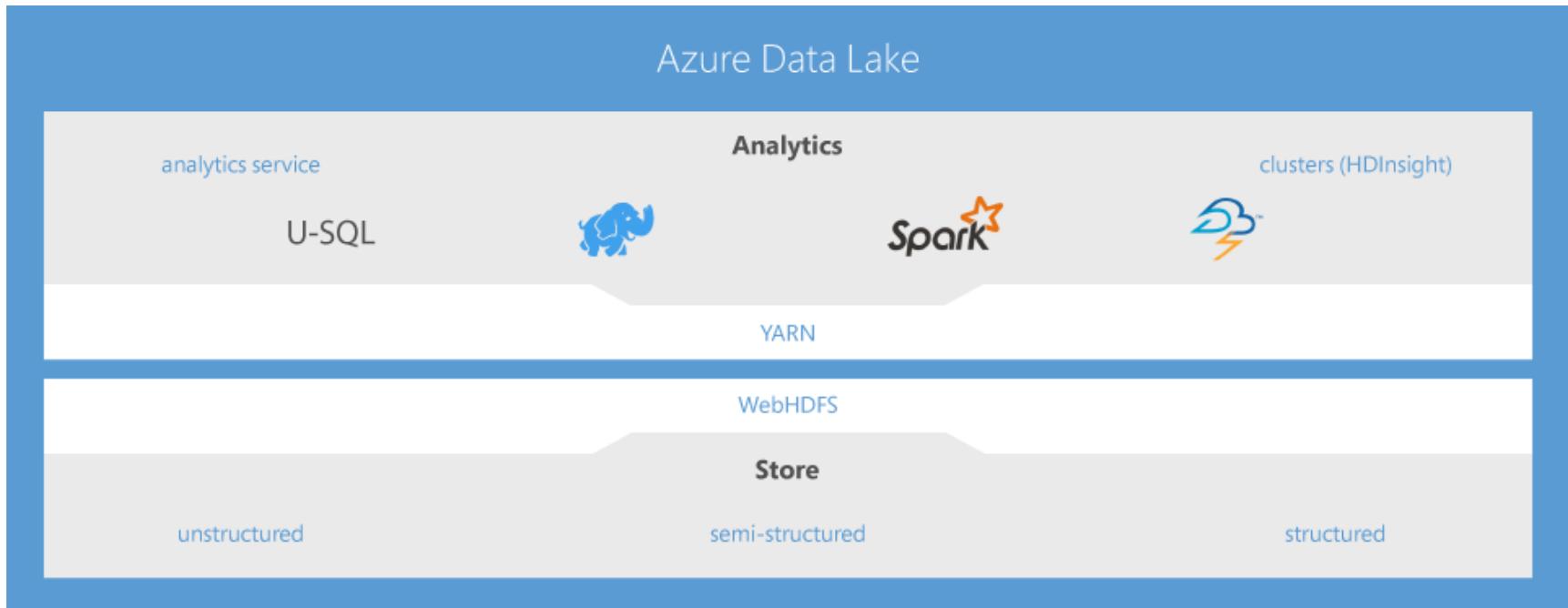
Azure Synapse Analytics



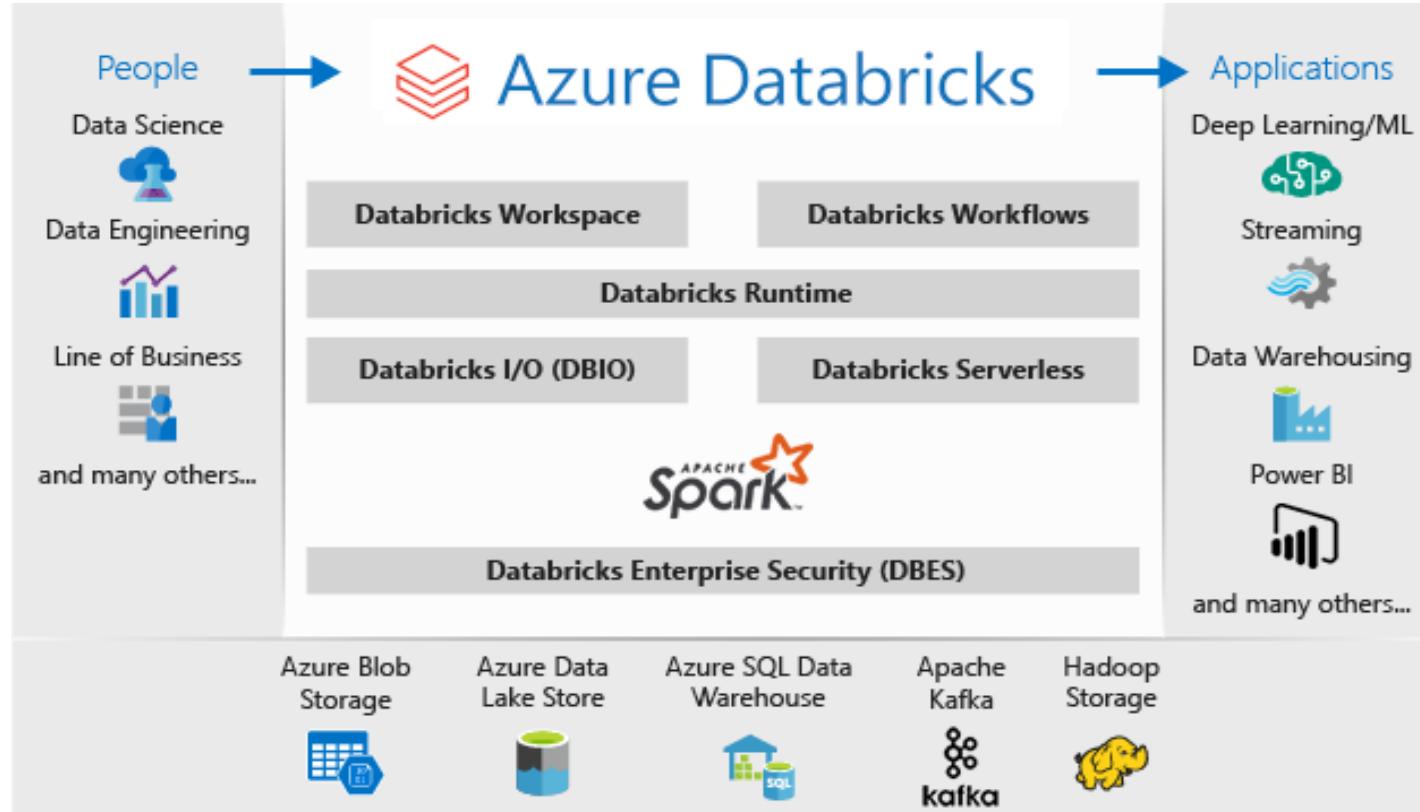
Data Lake Analytics



Data Lakes Storage



Azure Databricks



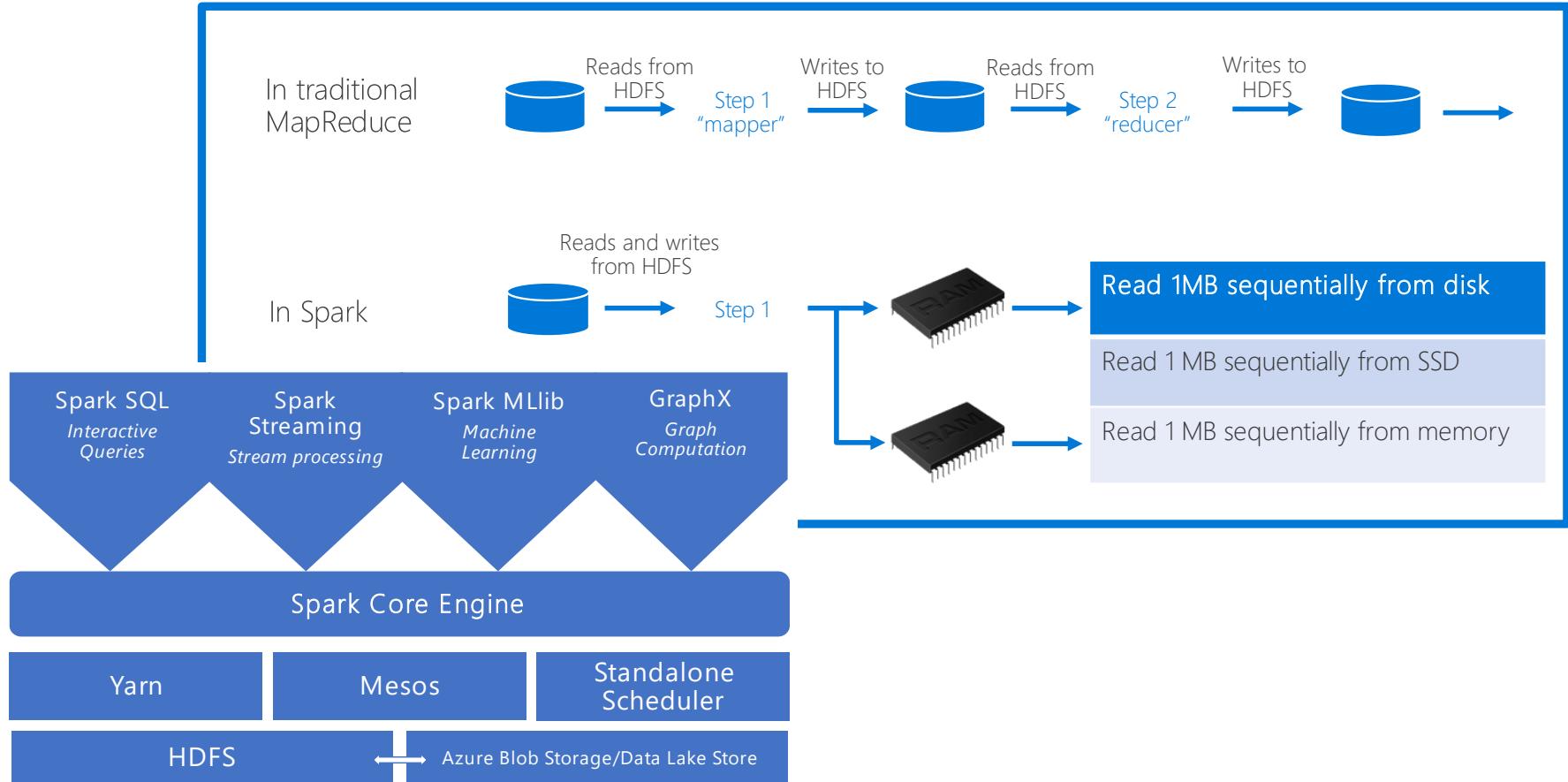
Azure Data Lake Gen2

Azure Data Lake Storage

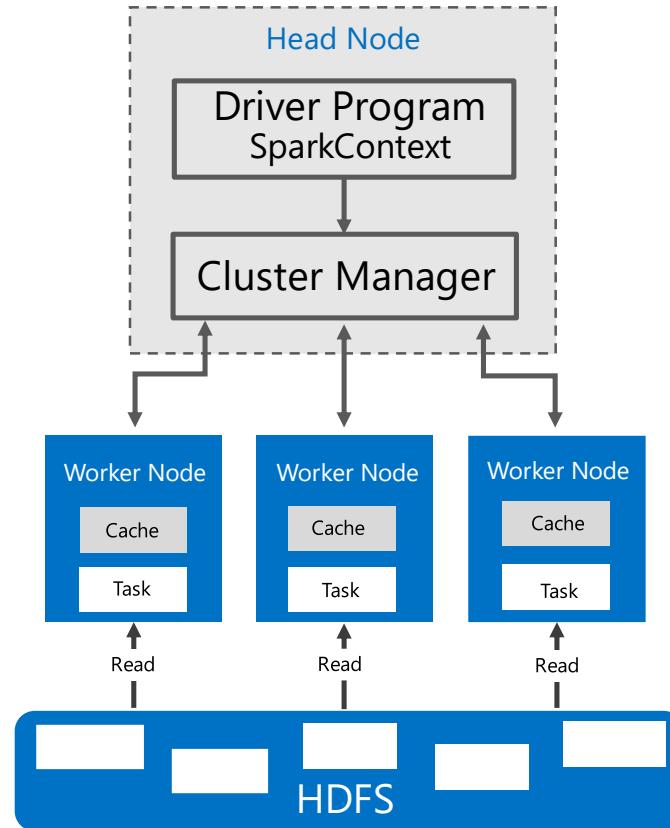
Gen 2 (ADLS Gen2) is a superset of Blob storage that is optimized for **big data analytics**. ADLS Gen2 is the preferred option for data lake solutions in Azure. It provides hierarchical namespace support on top of Blob storage.

The screenshot shows the Azure portal interface for creating a new storage account. The title is "Create a storage account". Below it, there are tabs: Basics (selected), Advanced (underlined), Networking, Data protection, and Tags. A "Review + create" button is visible. The main section is titled "Data Lake Storage Gen2". A descriptive text states: "The Data Lake Storage Gen2 hierarchical namespace accelerates big data analytics workloads and enables file-level access control lists (ACLs). Learn more". At the bottom, there is a checkbox labeled "Enable hierarchical namespace" which is checked, indicated by a green border around the input field and the word "checked" next to it.

Apache Spark



Spark Cluster Architecture

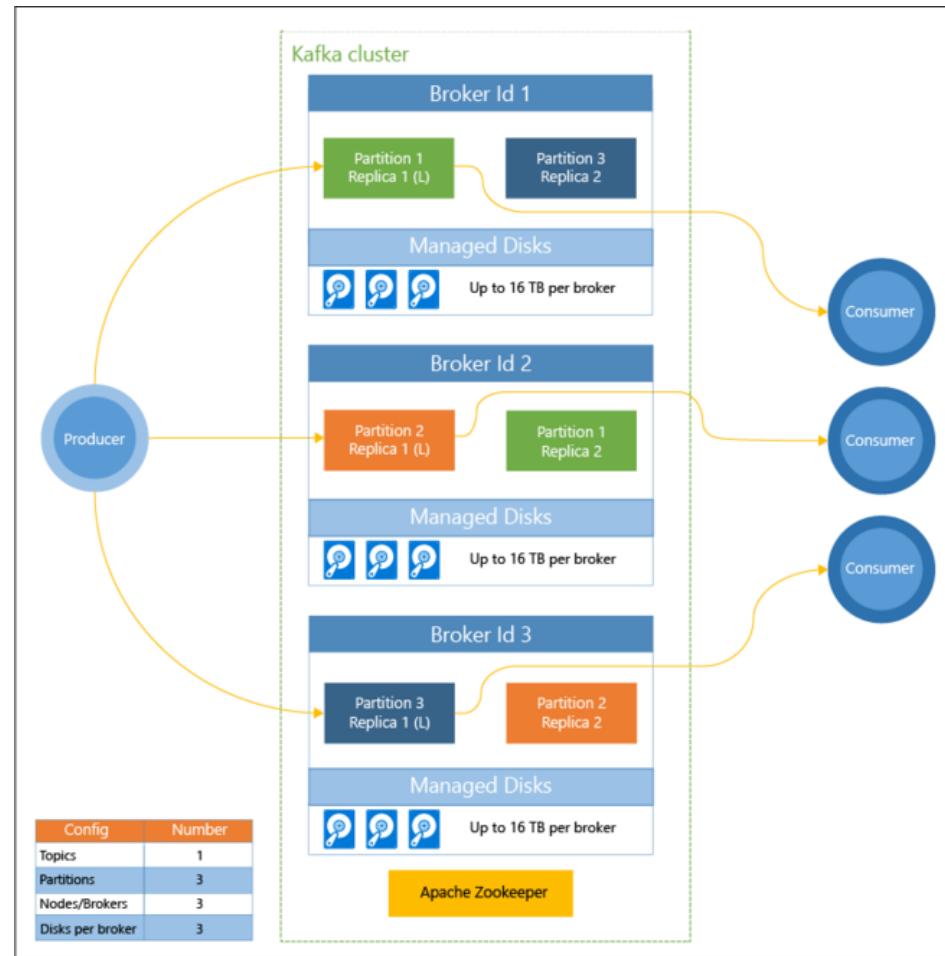


Hadoop



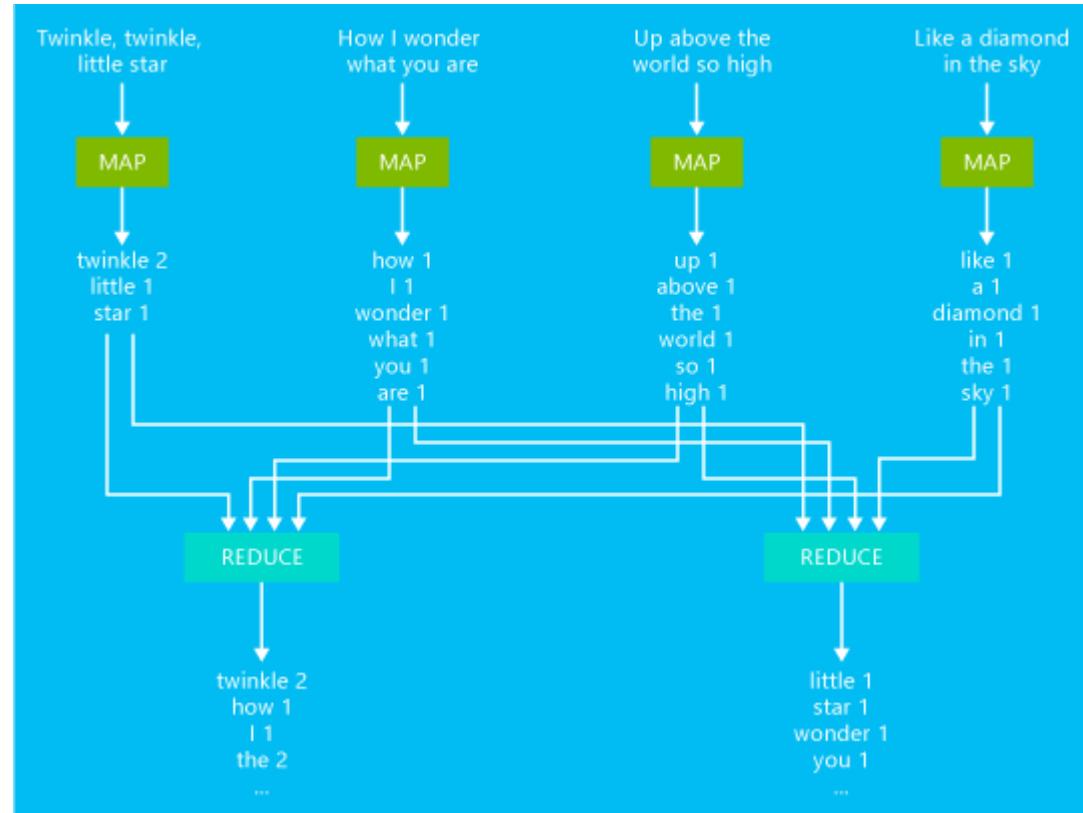
Apache Hadoop software is an open source framework that allows for the distributed storage and processing of large datasets across clusters of computers using simple programming models. Hadoop is designed to scale up from a single computer to thousands of clustered computers, with each machine offering local computation and storage. In this way, Hadoop can efficiently store and process large datasets ranging in size from gigabytes to petabytes of data.

Apache Kafka



MapReduce

Apache Hadoop MapReduce is a software framework for writing jobs that process vast amounts of data. Input data is split into independent chunks. Each chunk is processed in parallel across the nodes in your cluster.



Azure Operations



Monitoring

Monitoring is the collection of tools, processes, and techniques we use to increase the observability of the system.

- Consume information passively
- Ask questions based on dashboards
- Built to maintain based on dashboards

Built to maintain static environments with little variation

Used by developers of systems with little change and known permutations



Observability

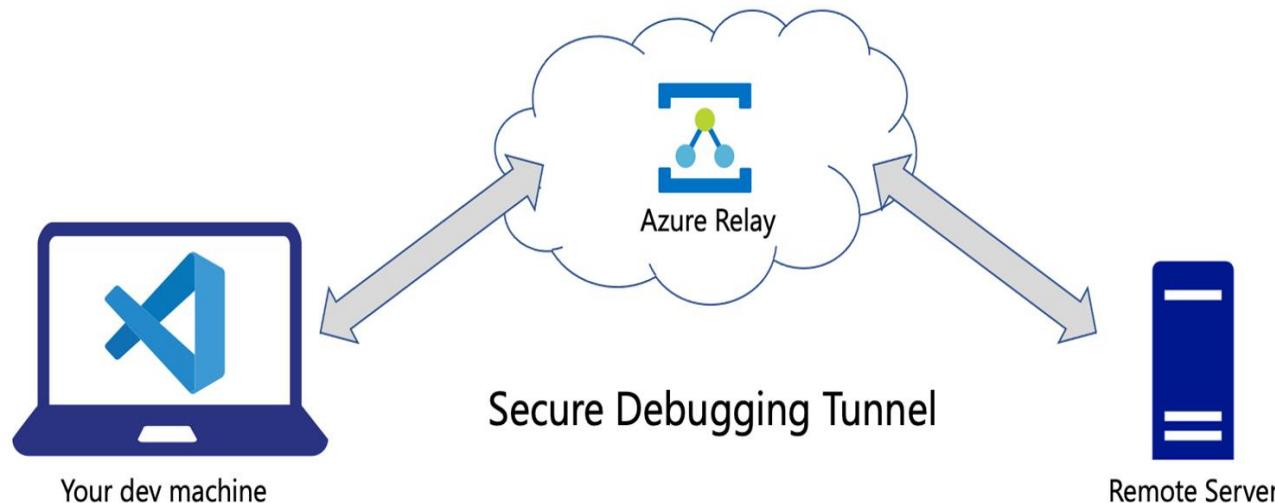
Observability is the property of the system that defines what we can tell about the state and behavior of the system, right now and historically.

- Gain understanding actively
- Ask questions based on hypotheses
- Build to tame dynamic environments with changing complexity
- Preferred by developers of systems with variability and unknown permutations

Three pillars of observability



Debugging



Error Reporting

The screenshot shows the Azure Log Analytics workspace interface. On the left, a navigation sidebar lists various monitoring and management options. The 'Logs' option is currently selected, indicated by a grey background. The main area is titled 'New Query 1' and displays a list of pre-defined queries under the 'API MANAGEMENT SERVICES' category.

Queries

Category: API Management services

★ Favorites

All Queries

API Management services

App Services

Application Gateways

Application Insights

Automation account

Azure Active Directory

Azure AD Domain Services

Azure Cosmos DB

Azure Database for MySQL

Azure Database for PostgreSQL

Azure Database for SQL

Azure Monitor agents

Azure Spring Cloud

Batch Accounts

CDN Profiles

API MANAGEMENT SERVICES

Number of requests
Count the total number of calls across all APIs in the last 24 hours.
Run Example query

Logs of the last 100 calls
Get the logs of the most recent 100 calls in the last 24 hours.
Run Example query

Number of calls by APIs
View the number of calls per API in the last 24 hours.
Run Example query

Bandwidth consumed
Total bandwidth consumed in the last 24 hours.
Run Example query

Request sizes
Statistics of request sizes in the last 24 hours.
Run Example query

Response sizes
Statistics of response sizes in the last 24 hours.
Run Example query

Feedback: Always show Queries | Community Guide

Trace

GET /api/entity Trace ID: b13673ec2c546297

Download X

Trace Start Time 2019-03-27 06:59:33	Sub Calls 4	Errors in Calls 1	Latency 4ms
Timeline			
			
Call Details			
SELECT [t].[ID], [t].[Name] FROM [tblDingens] AS [t] WHERE [t].[I... DATABASE			
Service To {tblDingens} of PerfDataStuff			
Caller Details			
Connection Server=localhost;Database=PerfDataStuff;User ID=squser;			
Statement			
1 SELECT [t].[ID], [t].[Name] 2 FROM [tblDingens] AS [t] 3 WHERE [t].[ID] > CAST(2 AS bigint)			
Caller Stack Trace			
ExecuteAsync in Microsoft.EntityFrameworkCore.Storage.Internal.RelationalCommand ExecuteReaderAsync in Microsoft.EntityFrameworkCore.Storage.Internal.RelationalCommand MoveNextCore in System.Linq.AsyncEnumerable+MoveNextCore<+AsyncEnumerator+<BufferlessStart>+System.Runtime.CompilerServices.AsyncMethodBuilderCore BufferlessMoveNext in Microsoft.EntityFrameworkCore.Query.Internal.AsyncQueryingEnumerable+<AsyncEnumerator+MoveNextCore>+System.Runtime.CompilerServices.AsyncMethodBuilderCore<+SqlServerExecutionStrategy>+ExecuteAsync_d_0_Start<+System.Runtime.CompilerServices.AsyncMethodBuilderCore> ExecuteAsync in Microsoft.EntityFrameworkCore.SqlServer.Storage.Internal.SqlServerExecutionStrategy MoveNextCore in System.Linq.AsyncEnumerable+MoveNextCore<+AsyncEnumerator+<MoveNextCore>+MoveNextStart<+System.Runtime.CompilerServices.AsyncMethodBuilderCore> ExecuteAsync in Microsoft.EntityFrameworkCore.Query.Internal.AsyncQueryingEnumerable+<AsyncEnumerator+MoveNextCore>+System.Runtime.CompilerServices.AsyncMethodBuilderCore<+MoveNextCore_d_7_Start<+System.Runtime.CompilerServices.AsyncMethodBuilderCore> MoveNextCore in System.Linq.AsyncEnumerable+SelectEnumerableAsyncIterator<+MoveNextCore_d_8_Start<+System.Runtime.CompilerServices.AsyncMethodBuilderCore> MoveNextCore in System.Linq.AsyncEnumerable+SelectEnumerableAsyncIterator<+MoveNextCore_d_10_Start<+System.Runtime.CompilerServices.AsyncMethodBuilderCore> MoveNextCore in System.Linq.AsyncEnumerable+SelectEnumerableAsyncIterator<+MoveNextCore_d_11_Start<+System.Runtime.CompilerServices.AsyncMethodBuilderCore> MoveNextCore in System.Linq.AsyncEnumerable+SelectEnumerableAsyncIterator<+MoveNextCore_d_12_Start<+System.Runtime.CompilerServices.AsyncMethodBuilderCore> Aggregate in System.Linq.AsyncEnumerable Aggregate in System.Linq.AsyncEnumerable ToListAsync in System.Linq.AsyncEnumerable ToListAsync in System.Linq.AsyncEnumerable GetNamesFromDbAsync in CoreMvcApp.Controllers.EntityController<+GetNamesFromDbSync_d_3_Start<+System.Runtime.CompilerServices.AsyncMethodBuilderCore> GetNamesFromDbSync in CoreMvcApp.Controllers.EntityController<+GetNamesFromDbSync_d_2_Start<+System.Runtime.CompilerServices.AsyncMethodBuilderCore> Get in CoreMvcApp.Controllers.EntityController<+GetNamesFromDbSync_d_1_Start<+System.Runtime.CompilerServices.AsyncMethodBuilderCore> MoveNextCore in System.Linq.AsyncEnumerable+ActionMethodExecutor+AwaitableObjectResultExecutor+<Execute_d_0_Start<+System.Runtime.CompilerServices.AsyncMethodBuilderCore> MoveNextCore in System.Linq.AsyncEnumerable+ActionMethodExecutor+AwaitableObjectResultExecutor<+MoveNextCore_d_12_Start<+System.Runtime.CompilerServices.AsyncMethodBuilderCore> InvokeActionMethodAsync in Microsoft.AspNetCore.Mvc.Internal.ControllerActionInvoker<+InvokeActionMethodAsync_d_10_Start<+Microsoft.AspNetCore.Mvc.Internal.ControllerActionInvoker> MoveNextCore in Microsoft.AspNetCore.Mvc.Internal.ControllerActionInvoker<+MoveNextCore_d_11_Start<+System.Runtime.CompilerServices.AsyncMethodBuilderCore> InvokeActionMethodAsync in Microsoft.AspNetCore.Mvc.Internal.ControllerActionInvoker<+InvokeActionMethodAsync_d_12_Start<+Microsoft.AspNetCore.Mvc.Internal.ControllerActionInvoker> MoveNextCore in Microsoft.AspNetCore.Mvc.Internal.ControllerActionInvoker<+MoveNextCore_d_13_Start<+System.Runtime.CompilerServices.AsyncMethodBuilderCore>			

Colorize by Endpoint Technology

Service	Endpoint	Calls	Aggregated Time ↓	Errors
localhost	GET /api	1	4ms	0
PerfDataStuff	{tblDingens}	1	1ms	0
Unspecified	Unspecified	1	< 1ms	1
PerfDataStuff	CONNECT	1	< 1ms	0

Colorize by Endpoint Technology

Calls
 GET /api/entity HTTP To GET /api of localhost EntityController INTERNAL  GetNamesFromDbAsync INTERNAL Inherited from GET /api/entity

Monitoring

GET /api/entity Trace ID: b13673ec2c546297

Download X

Trace Start Time
2019-03-27
06:59:33

Sub Calls 4 Errors in Calls 1 Latency 4ms

Timeline Colorize by Endpoint Technology

Started: 06:59:33

GET /api/entity EntityController GetNamesFromDbAsync SELECT [t].[ID], [t].[Name] FROM [tblDingens] AS [t] WHERE [t].[I... DATABASE

Service To {tblDingens} of PerfDataStuff

Caller Details

Connection Server=localhost;Database=PerfDataStuff;User ID=squser;

Statement

```
1 | SELECT [t].[ID], [t].[Name]
2 | FROM [tblDingens] AS [t]
3 | WHERE [t].[ID] > CAST(2 AS bigint)
```

Caller Stack Trace

```
ExecuteAsync< in Microsoft.EntityFrameworkCore.Storage.Internal.RelationalCommand
ExecuteReaderAsync< in Microsoft.EntityFrameworkCore.Storage.Internal.RelationalCommand
MoveNextCore< in System.Runtime.CompilerServices.AsyncMethodBuilderCore
BufferlessMoveNext< Microsoft.EntityFrameworkCore.Query.Internal.AsyncQueryingEnumerable`1+AsyncEnumerator
MoveNextCore< in System.Runtime.CompilerServices.AsyncMethodBuilderCore
SqlServerExecutionStrategy+>ExecuteAsync<_d>
Start< in System.Runtime.CompilerServices.AsyncMethodBuilderCore
ExecuteAsync< in Microsoft.EntityFrameworkCore.SqlServer.Storage.Internal.SqlServerExecutionStrategy
MoveNextCore< in System.Runtime.CompilerServices.AsyncMethodBuilderCore
BufferlessMoveNext< Microsoft.EntityFrameworkCore.Query.Internal.AsyncQueryingEnumerable`1+AsyncEnumerator
MoveNextCore< in System.Runtime.CompilerServices.AsyncMethodBuilderCore
MoveNextCore< in Microsoft.EntityFrameworkCore.Query.Internal.AsyncQueryingEnumerable`1+AsyncEnumerator
MoveNextCore< in System.Runtime.CompilerServices.AsyncMethodBuilderCore
MoveNextCore< in System.Linq.AsyncEnumerable+SelectEnumerableAsyncIterator`2+MoveNextCore`d_7
Start< in System.Runtime.CompilerServices.AsyncMethodBuilderCore
MoveNextCore< in System.Linq.AsyncEnumerable+SelectEnumerableAsyncIterator`2+MoveNextCore`d_8
MoveNextCore< in System.Linq.AsyncEnumerable+SelectEnumerableAsyncIterator`2+MoveNextCore`d_9
Start< in System.Runtime.CompilerServices.AsyncMethodBuilderCore
MoveNextCore< in System.Linq.AsyncEnumerable+SelectEnumerableAsyncIterator`2+MoveNextCore`d_10
Start< in System.Runtime.CompilerServices.AsyncMethodBuilderCore
MoveNextCore< in System.Linq.AsyncEnumerable+SelectEnumerableAsyncIterator`2+MoveNextCore`d_11
Start< in System.Runtime.CompilerServices.AsyncMethodBuilderCore
MoveNextCore< in Microsoft.EntityFrameworkCore.Query.Internal.AsyncLinqOperatorProvider+ExceptionInterceptor`1+Enumerator
MoveNextCore< in System.Linq.AsyncEnumerable+SelectEnumerableAsyncIterator`2+MoveNextCore`d_12
Start< in System.Runtime.CompilerServices.AsyncMethodBuilderCore
Aggregate< in System.Linq.AsyncEnumerable
Aggregate< in System.Linq.AsyncEnumerable
ToListAsync< in System.Linq.AsyncEnumerable
ToListAsync< in Microsoft.EntityFrameworkCore.Storage.Internal.RelationalCommand+>GetNamesFromDbAsync<_d_3
Start< in System.Runtime.CompilerServices.AsyncMethodBuilderCore
GetNamesFromDbAsync< CoreMvcApp.Controllers.EntityController+>GetNamesFromDbAsync<_d_4
MoveNextCore< in System.Runtime.CompilerServices.AsyncMethodBuilderCore
Get< in CoreMvcApp.Controllers.EntityController
MoveNextCore< in System.Runtime.CompilerServices.AsyncMethodBuilderCore
InvokeActionMethodAsync< Microsoft.AspNetCore.Mvc.Internal.ActionMethodExecutor+AwaitableObjectResultExecutor+>Execute<_d_0
Start< in System.Runtime.CompilerServices.AsyncMethodBuilderCore
Execute< in Microsoft.AspNetCore.Mvc.Internal.ActionMethodExecutor+AwaitableObjectResultExecutor
MoveNextCore< in System.Runtime.CompilerServices.AsyncMethodBuilderCore
Get< in CoreMvcApp.Controllers.EntityController
InvokeActionMethodAsync< Microsoft.AspNetCore.Mvc.Internal.ControllerActionInvoker
MoveNextCore< in System.Runtime.CompilerServices.AsyncMethodBuilderCore
MoveNextCore< in Microsoft.AspNetCore.Mvc.Internal.ControllerActionInvoker+>InvokeNextActionFilterAsync<_d_10
Start< in System.Runtime.CompilerServices.AsyncMethodBuilderCore
InvokeActionMethodAsync< Microsoft.AspNetCore.Mvc.Internal.ControllerActionInvoker
MoveNextCore< in Microsoft.AspNetCore.Mvc.Internal.ControllerActionInvoker+>InvokeNextActionFilterAsync<_d_11
Start< in System.Runtime.CompilerServices.AsyncMethodBuilderCore
MoveNextCore< in Microsoft.AspNetCore.Mvc.Internal.ControllerActionInvoker
Next< in Microsoft.AspNetCore.Mvc.Internal.ControllerActionInvoker
MoveNextCore< in Microsoft.AspNetCore.Mvc.Internal.ControllerActionInvoker+>InvokeNextActionFilterAsync<_d_12
Start< in System.Runtime.CompilerServices.AsyncMethodBuilderCore
MoveNextCore< in Microsoft.AspNetCore.Mvc.Internal.ControllerActionInvoker+>InvokeNextActionFilterAsync<_d_13
Start< in System.Runtime.CompilerServices.AsyncMethodBuilderCore
```

Service Endpoint Calls Aggregated Time ↓ Errors

Service	Endpoint	Calls	Aggregated Time	Errors
localhost	GET /api	1	4ms	0
PerfDataStuff	{tblDingens}	1	1ms	0
Unspecified	Unspecified	1	< 1ms	1
PerfDataStuff	CONNECT	1	< 1ms	0

Azure Data Engineering



Designing an Azure Data Lake

If you have been following the big data technologies domain, you would have definitely come across the term *data lake*. Data lakes are distributed data stores that can hold very large volumes of diverse data. They can be used to store different types of data such as structured, semi-structured, unstructured, streaming data, and so on.

Designing an Azure Data Lake

A data lake solution usually comprises a storage layer, a compute layer, and a serving layer. The compute layers could include **Extract, Transform, Load (ETL)**; **Batch**; or **Stream** processing. There are no fixed templates for creating data lakes. Every data lake could be unique and optimized as per the owning organization's requirements.

Data Lake Zones

A data lake can be broadly segregated into three zones where different stages of the processing take place, outlined as follows:

1. **Landing Zone or Raw Zone:** This is where the raw data is ingested from different input sources.
2. **Transformation Zone:** This is where the batch or stream processing happens. The raw data gets converted into a more structured and **business intelligence (BI)**-friendly format.
3. **Serving Zone:** This is where the curated data that can be used to generate insights and reports are stored and served to BI tools. The data in this zone usually adheres to well-defined schemas.

Data Lake Azure Services



Azure Event Hub



Azure Synapse
Analytics



Azure HDInsight



Azure Streaming
Analytics



Azure Databricks



Azure ML



Azure Data Factory



Azure SQL



SQL Data Warehouse



Azure Streaming Analytics



Power BI

Data Lake Batch Services

Storage Technologies	Azure Data Lake Gen2 Azure Blob Storage Azure CosmosDB Azure SQL Database
Data Transformation Technologies	Spark (via Azure Synapse, Azure HDInsight or Azure Databricks) Apache Hive (via Azure HDInsight) Apache Pig (via Azure HDInsight)
Analytical Datastore	Synapse SQL Warehouse (via Azure HDInsight) Apache HBase (via Azure HDInsight) Apache Hive (via Azure HDInsight)

Data Lake Stream Services

Ingestion Tools	Azure Event Hub Azure IoT Hub Apache Kafka (via Azure HDInsight)
Stream Processing Tools	Azure Stream Analytics Spark Streaming (via Azure HDInsight or Azure Databricks) Apache Storm (via Azure HDInsight)
Analytical Datastore	Synapse SQL Warehouse Apache HBase (via Azure HDInsight) Apache Hive (via Azure HDInsight)

Cloud-native Microservices



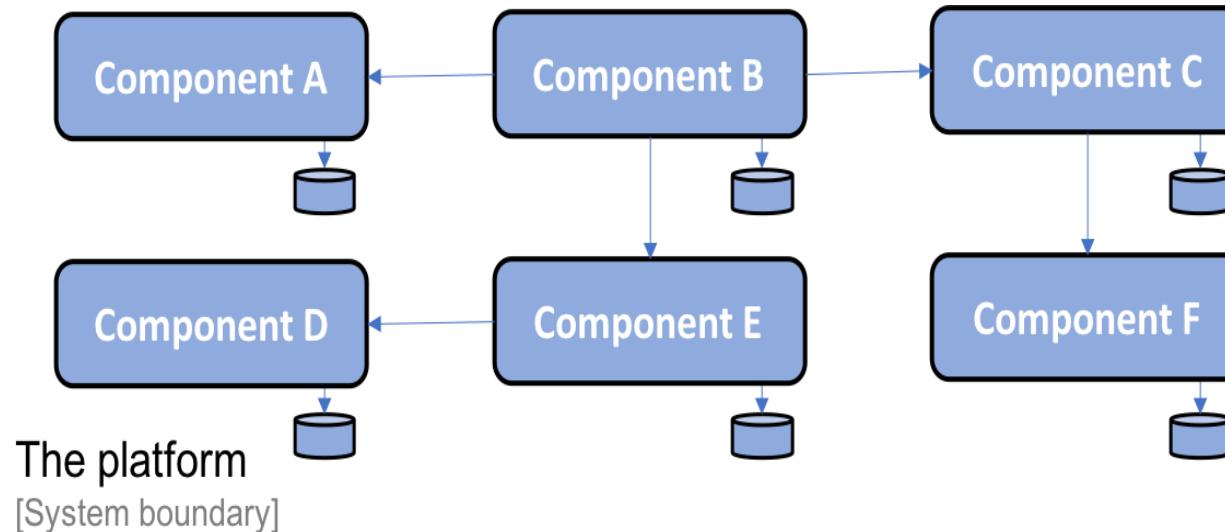
MICROSERVICE OVERVIEW

“The Microservice architectural style is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API.”

- Martin Fowler



Microservices



WHY MICROSERVICES?

The driving force behind microservices is to get the highest quality software possible delivered to customers as fast as possible, at web scale.

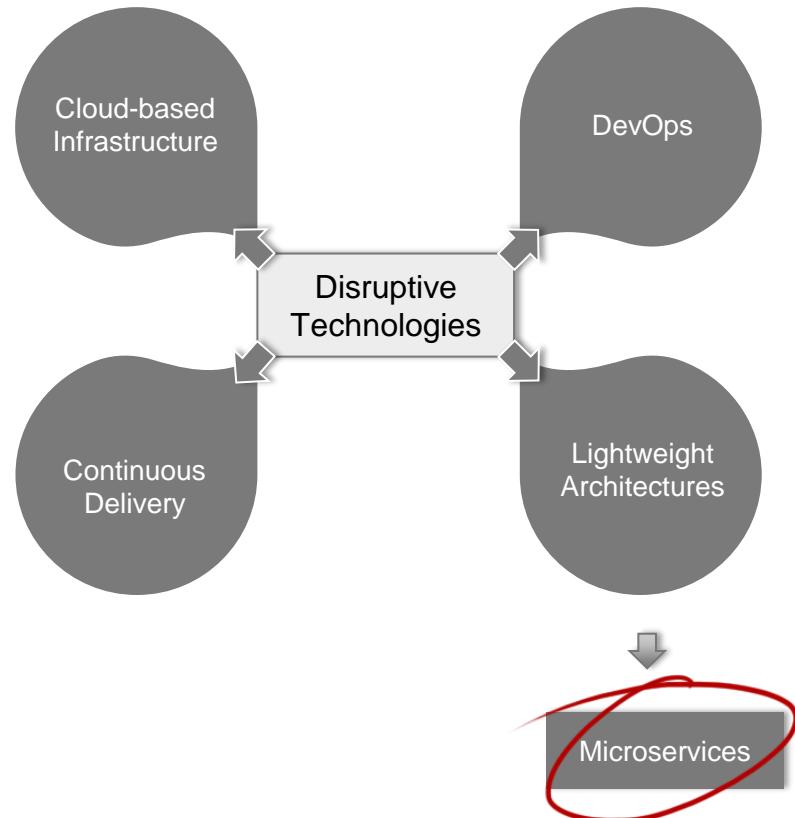
WHY MICROSERVICES?

As new disruptive technologies are introduced, it is critical for DevSecOps teams to evaluate their potential and facilitate experimentation together with the business.

These new disruptive technologies, force IT into a dramatic change and are designed to drive time to market and value for our business teams.

Enterprises who ignore them, will soon or later fall short and lose market share for their competitors.

When combined with Cloud-based Infrastructure, DevOps and Continuous Delivery, Microservices, as a lightweight Architecture approach, enables rapid Enterprise IT transformation and innovation.



WHY MICROSERVICES ARCHITECTURE?

Adopting this architecture increases the potential benefits of complex IT initiatives

- | | | |
|--|------------------------------|--|
|  | Agility | Agility allows organizations to deliver new products, functions, and features quicker |
|  | Reduce time to market | Independent components move new features into production quicker and provide more flexibility for piloting and prototyping, thus reducing time to market |
|  | Composability | Composability reduces development time, provides a compound benefit through reusability over time, and guarantees system scalability |
|  | Comprehensibility | Comprehensibility of the software system simplifies development planning , increases accuracy, and accelerates new resource integration |
|  | Polyglotism | Polyglotism permits the use of the right tools for the right task , thus accelerating technology introduction and increasing solution options |
|  | Cost reduction | Reduce cost at operating speed by reducing the overall cost of designing, implementing, and deploying services |

Microservices Principles

- Microservices are a set of small, independent, composable services.
- Each service can be accessed by way of a well-known API format such as REST, GraphQL, gRPC or in response to event notifications.
- This architectural style de-constructs business processes to their most basic level by creating small, separate processes that then replace large, single applications.

- Breaking a system into smaller parts allows each service to focus on a single business capability.
- Microservices do not live alone.
- They are a part of the larger system.

- They work alongside other services to accomplish the tasks normally handled by one large standalone monolith application.
- They communicate with each other synchronously or asynchronously.
- Microservices make applications easier to develop and maintain.
- BUT A LOT HARDER TO MANAGE

Microservice Reviewed



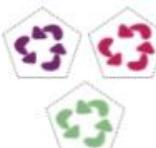
Independent services increase the speed of experimentation



Long standing autonomous teams responsible for each service



Services can be built with diverse set of technologies



Services have independent lifecycle



Ecosystem enables innovation through composing services



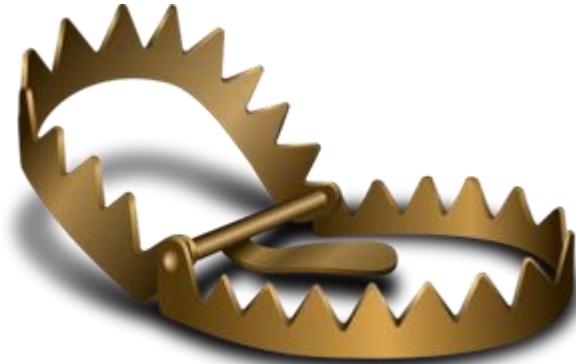
Building small and independent services lowers the developer's cognitive load



Ecosystem enables growth and scale of operation

Microservice Traps

- Message loss
- Duplicate messages
- Data corruption
- Resiliency
- Reliability
- Variability
- Authentication
- Monitoring
- Hardening



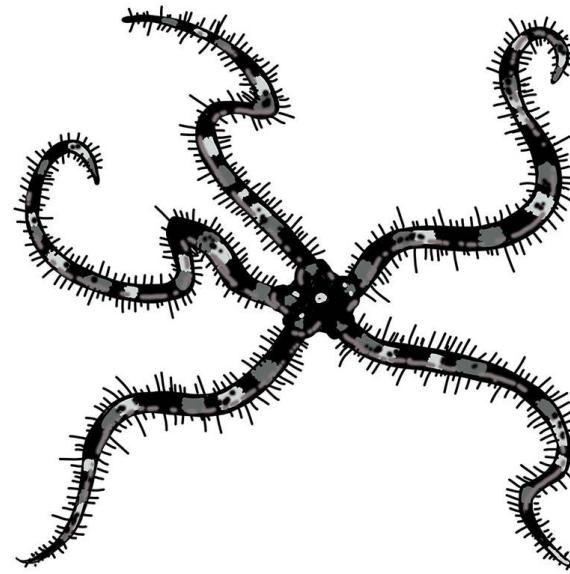
Nanoservices

The basic difference between the two frameworks is that nanoservices are considered smaller siblings of microservices. Nanoservices are designed to perform a single function, whose output is exposed through a specific API endpoint (command).

Nanoservices are fully discoverable among each other. Each one can link with other services to perform additional actions and extend functionality.

Brittle Services

- Brittle services are those that composite so many features and functions that they begin to become mini-monoliths.



12-Factor Application Building Blocks

 Codebase	 Port Binding
 Dependencies	 Concurrency
 Configuration	 Disposability
 Backing Services	 Dev/Prod Parity
 Build, Release, Run	 Logs
 Processes	 Admin Processes

Why 12 Factor?

The methodology was drafted by developers at Heroku and was first presented by Adam Wiggins circa 2011.

The twelve-factor app is a methodology for building software-as-a-service apps that:

Use **declarative formats** for setup automation, to minimize time and cost for new developers joining the project;

Have a **clean contract** with the underlying operating system, offering **maximum portability** between execution environments;

Are suitable for **deployment on modern cloud platforms**, obviating the need for servers and systems administration;

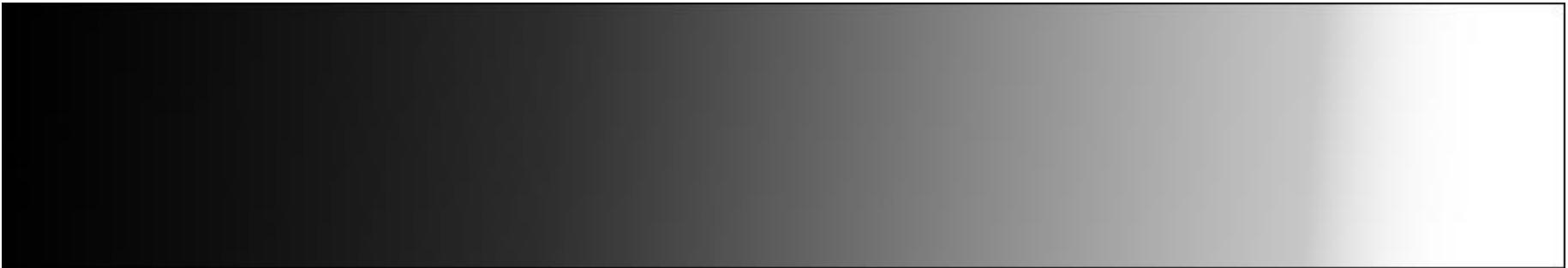
Minimize divergence between development and production, **enabling continuous deployment** for maximum agility;

And can **scale up** without significant changes to tooling, architecture, or development practices.

- Adam Wiggins, 12factor.net

A Word about Implementation

It's shades of gray.



Adopting 12 Factor App is not an all or nothing thing.
Adopt what you can, when you can.

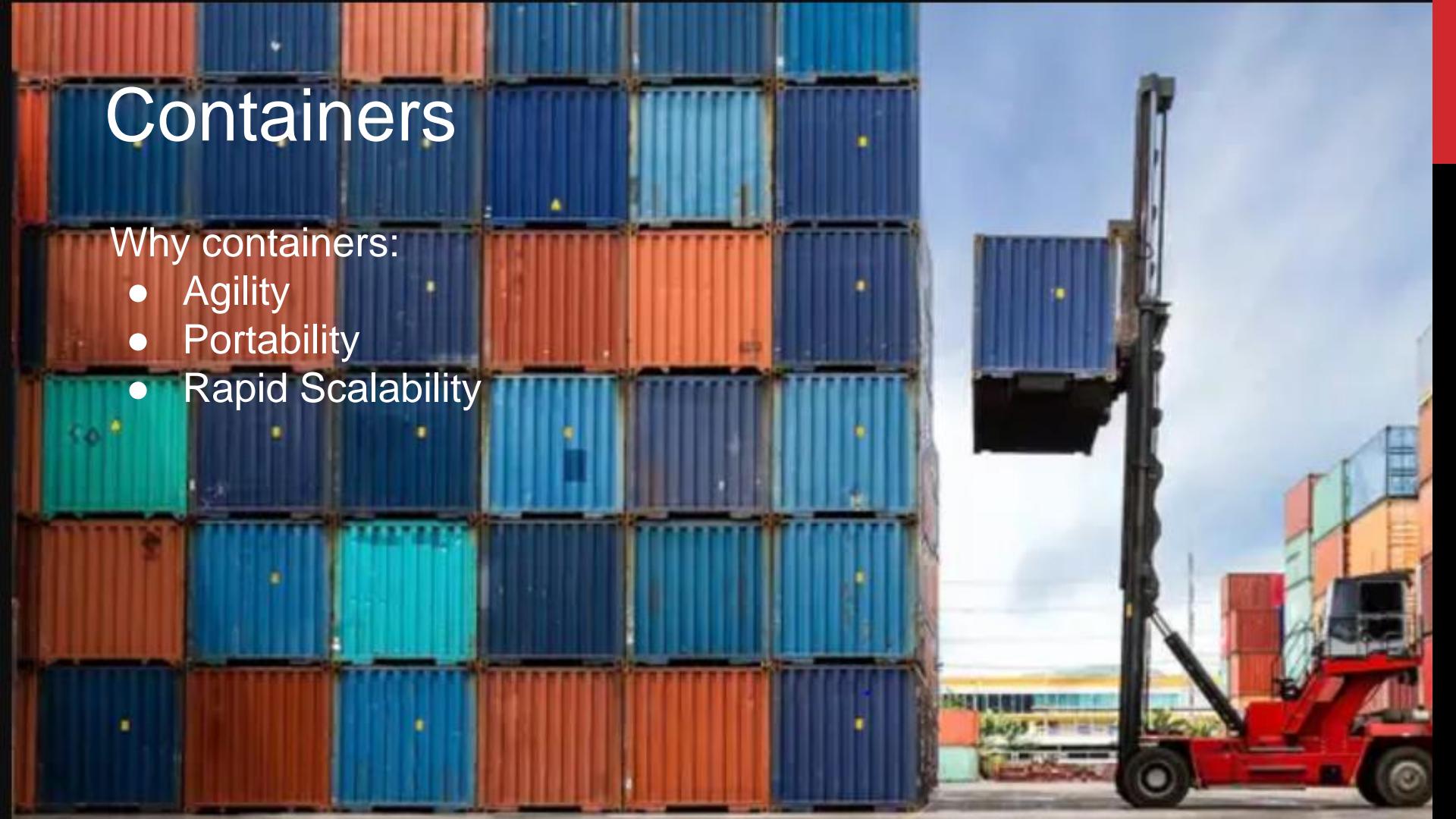
Containers



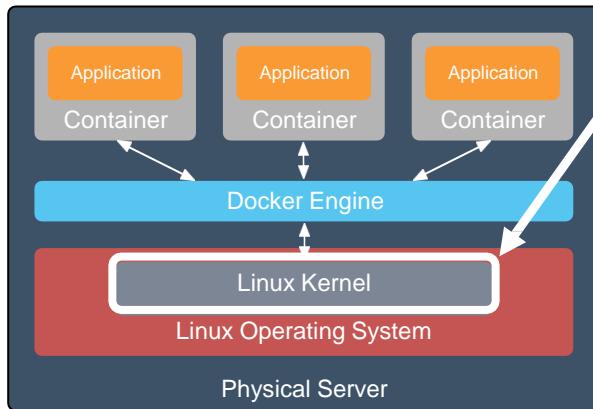
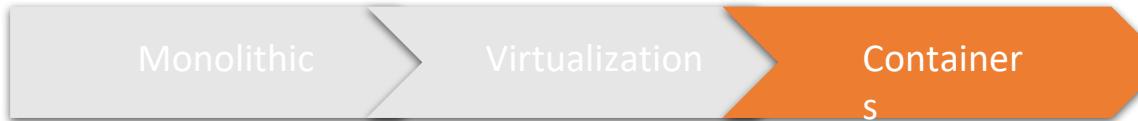
Containers

Why containers:

- Agility
- Portability
- Rapid Scalability

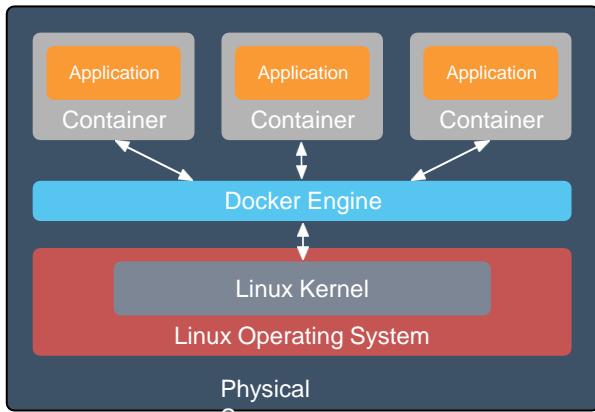


Containers



Shared kernel on the host to run multiple guest applications

Containers - Advantages

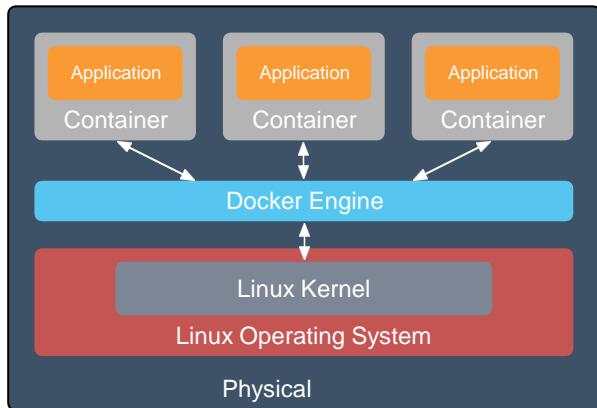
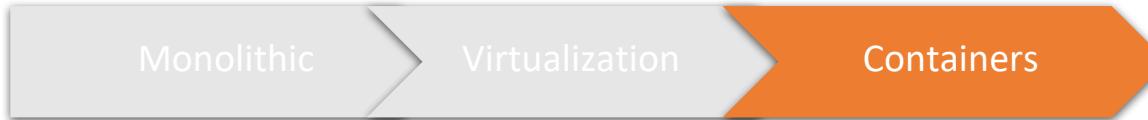


Shared kernel on the host to run multiple guest applications

Advantages over VMs

- Containers are more lightweight
- No need to install a guest Operating System
- Less CPU, RAM, storage overhead
- More containers per machine
- Greater portability

Containers - Challenges

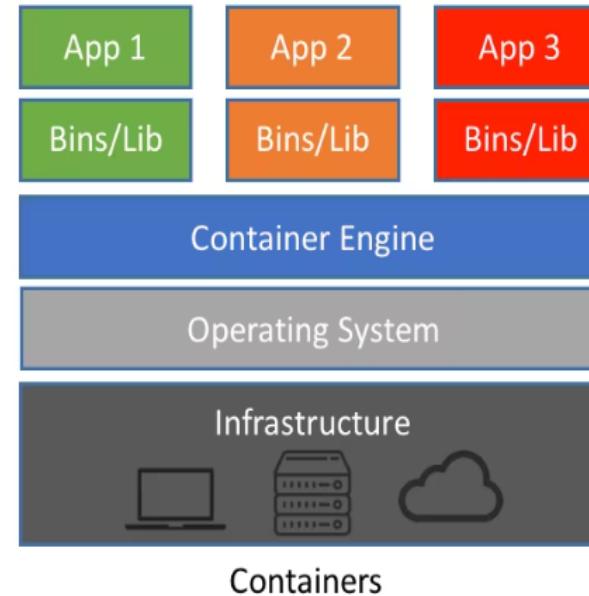
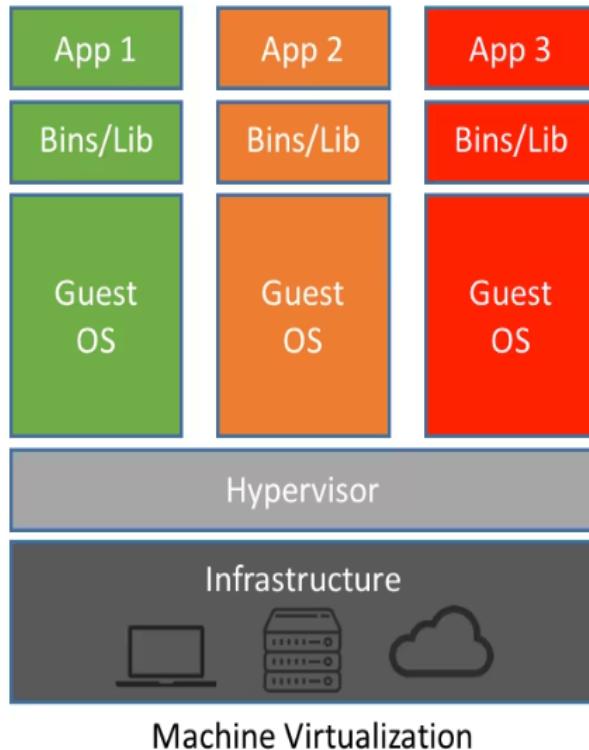


Shared kernel on the host to run multiple guest applications

Container Challenges

- Early Docker focused on single-node operations
- Up to user to cluster Docker hosts and manage deployment of containers on cluster
- User solves for automatic scale out of applications
- User solves for service discovery between application components (microservices)

Containers vs. Virtual Machine



Container based virtualization

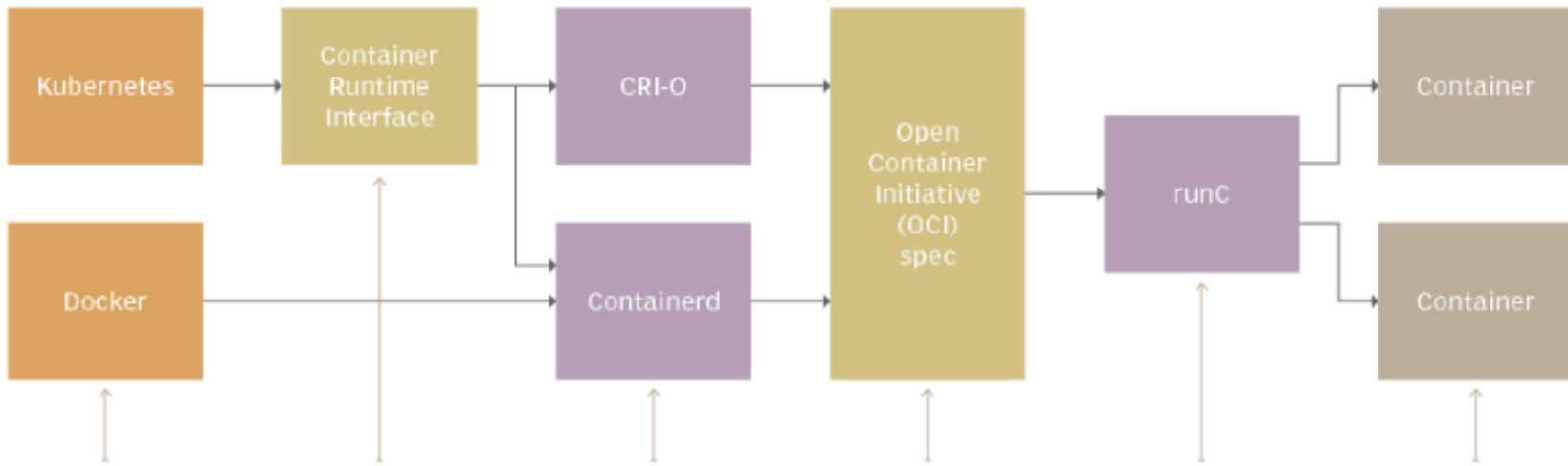
Uses the kernel on the host operating system to run multiple guest instances

- Each guest instance is a container
- **Each container has its own**

- Root filesystem
- Processes
- Memory
- Devices
- Network Ports



Runtimes



These tools run containers in development or production.

CRI is a standardized Kubernetes API that defines how Kubernetes interacts with container runtimes.

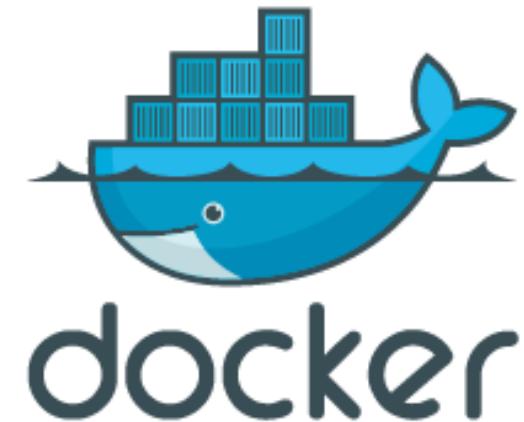
Containerd and CRI-O are CRI-compliant container runtimes that implement the CRI spec.

OCI provides instructions for container images, running containers and an implementation of runC.

RunC is an OCI-compliant container runtime that creates and runs container processes.

A containerized process runs in production.

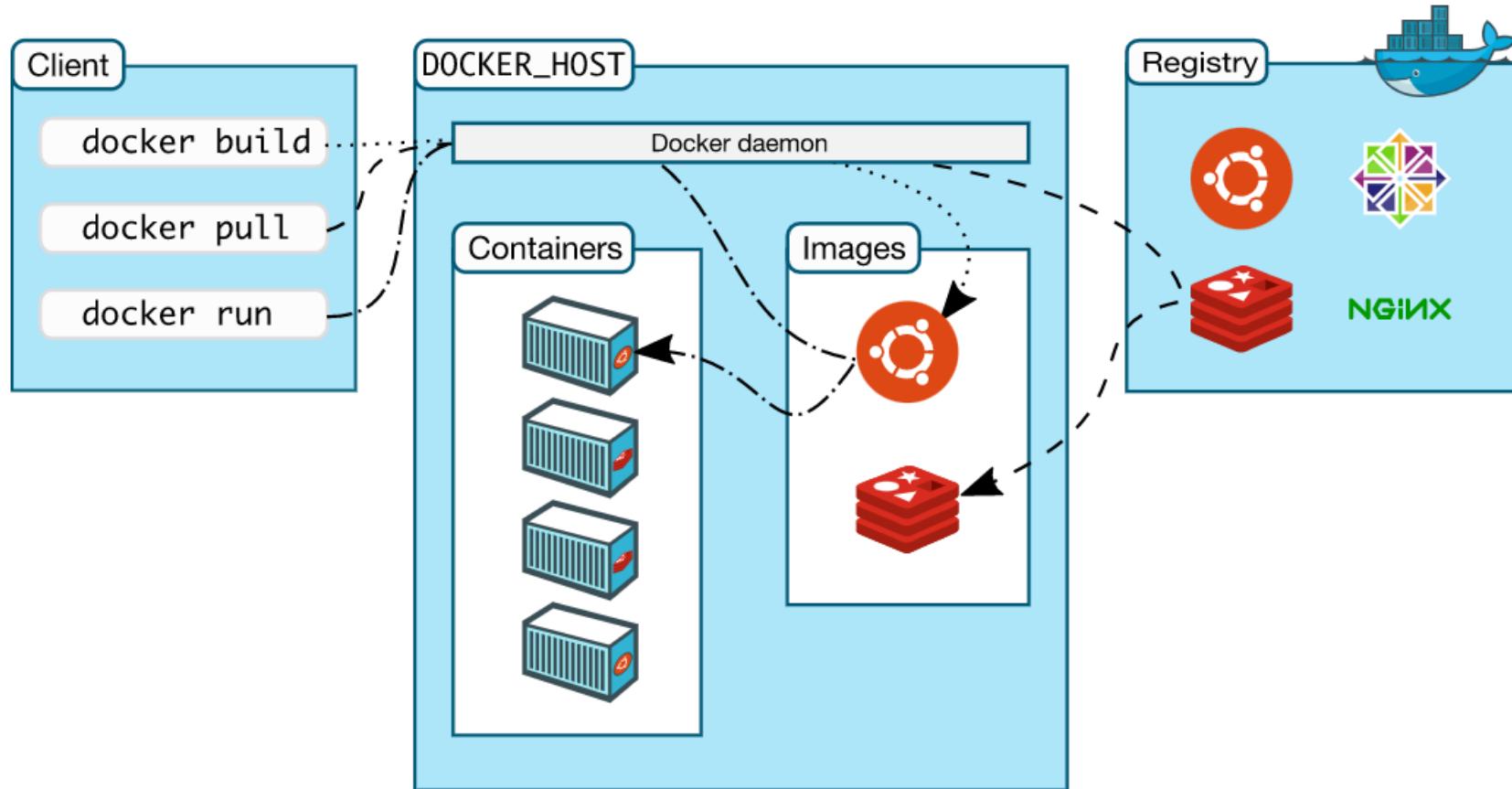
Docker



Docker Ecosystem



Docker Flow



Docker Architecture

Docker uses a client-server architecture. The Docker *client* talks to the Docker *daemon*, which does the heavy lifting of building, running, and distributing your Docker containers.

The Docker client and daemon *can* run on the same system, or you can connect a Docker client to a remote Docker daemon. The Docker client and daemon communicate using a REST API, over UNIX sockets or a network interface. Another Docker client is Docker Compose, that lets you work with applications consisting of a set of containers.

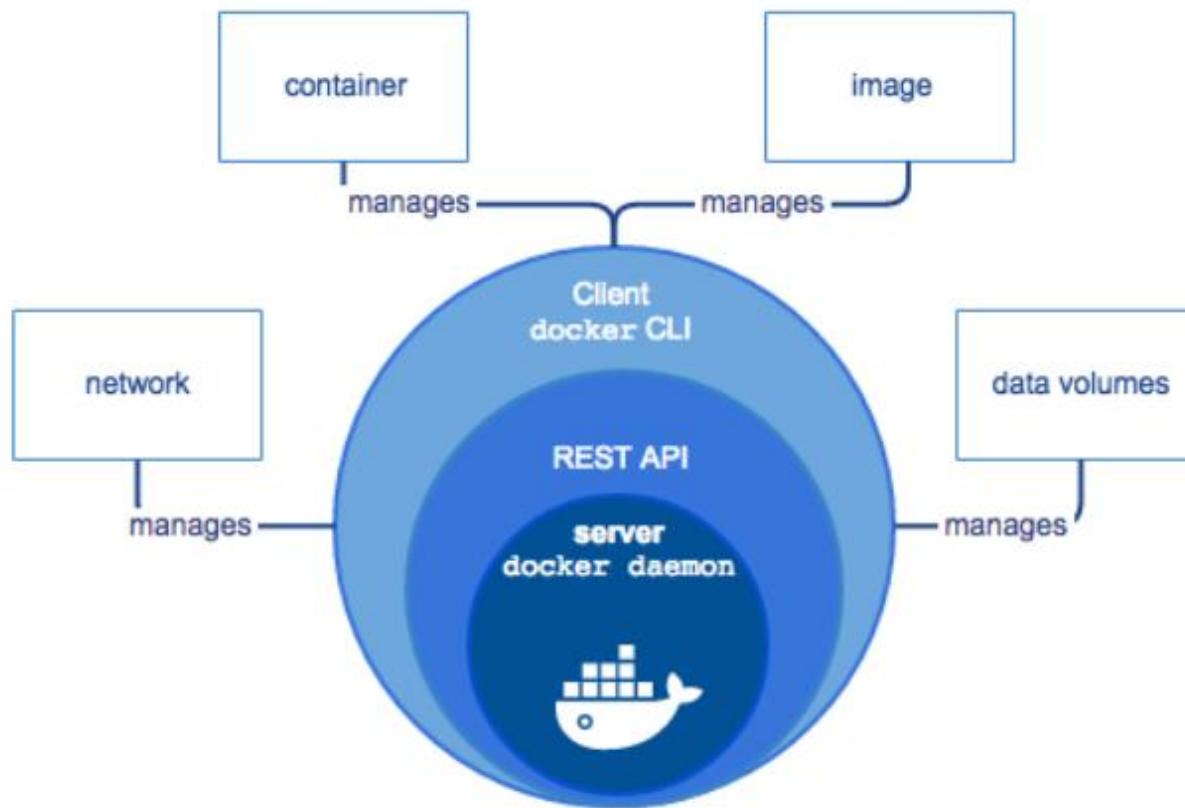
Docker Containers

To deploy Docker containers on Azure, you must meet the following requirements:

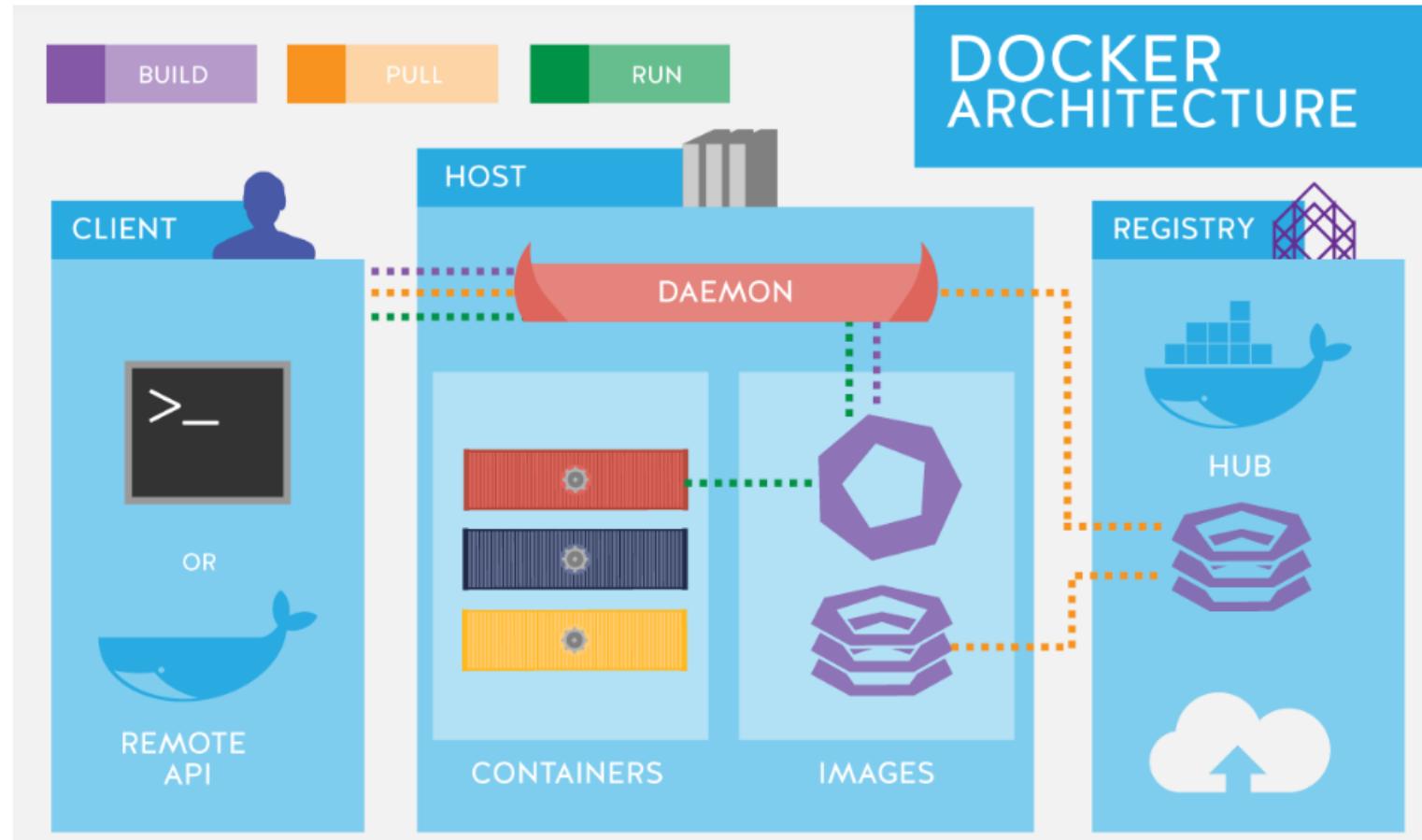
1. Download and install the latest version of Docker Desktop.
 - Download for Mac
 - Download for Windows
2. Alternatively, install the Docker Compose CLI for Linux.
3. Ensure you have an Azure subscription. You can get started with an Azure free account.



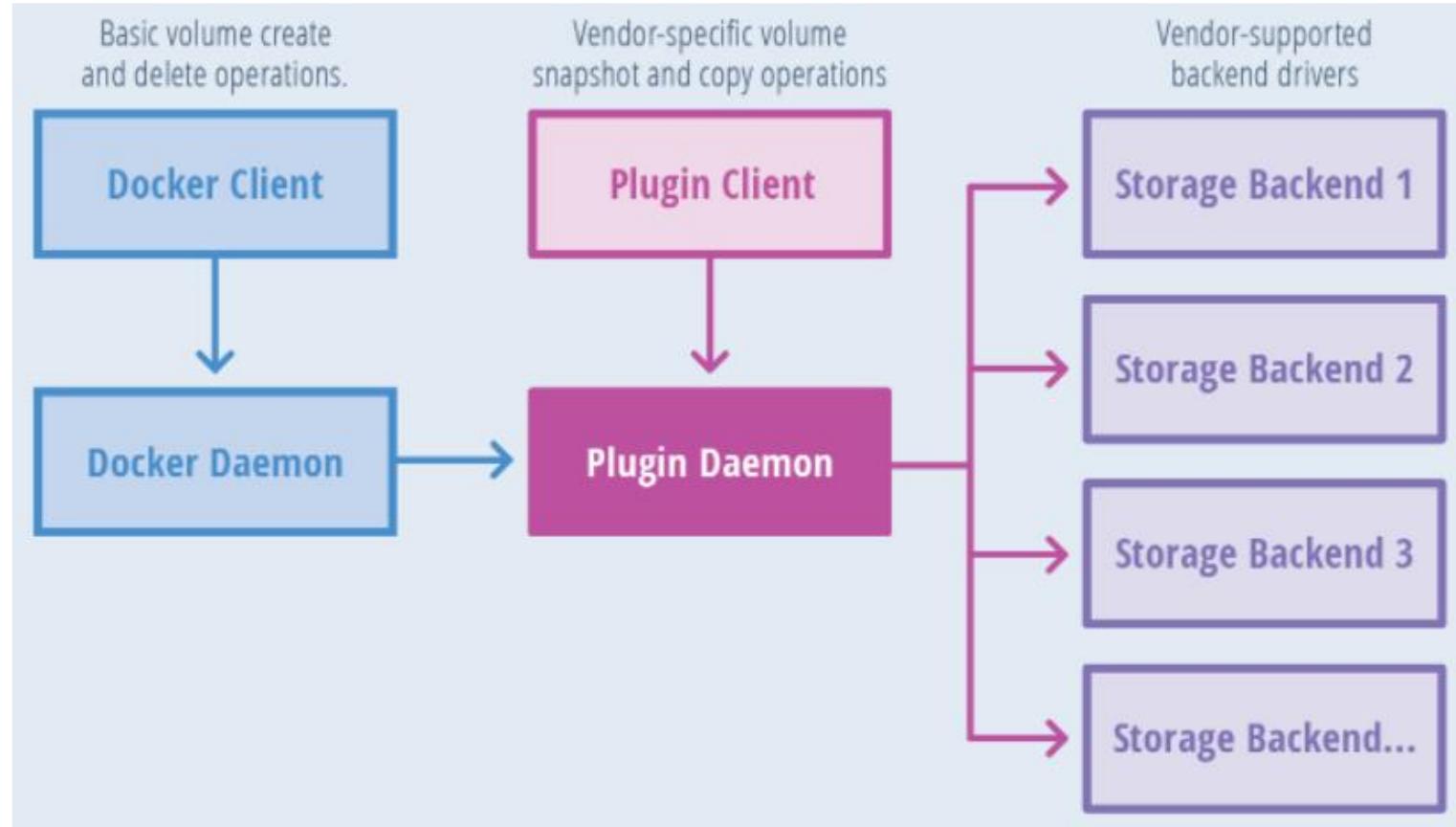
Docker Engine



Docker Architecture



Docker Objects



Docker Engine

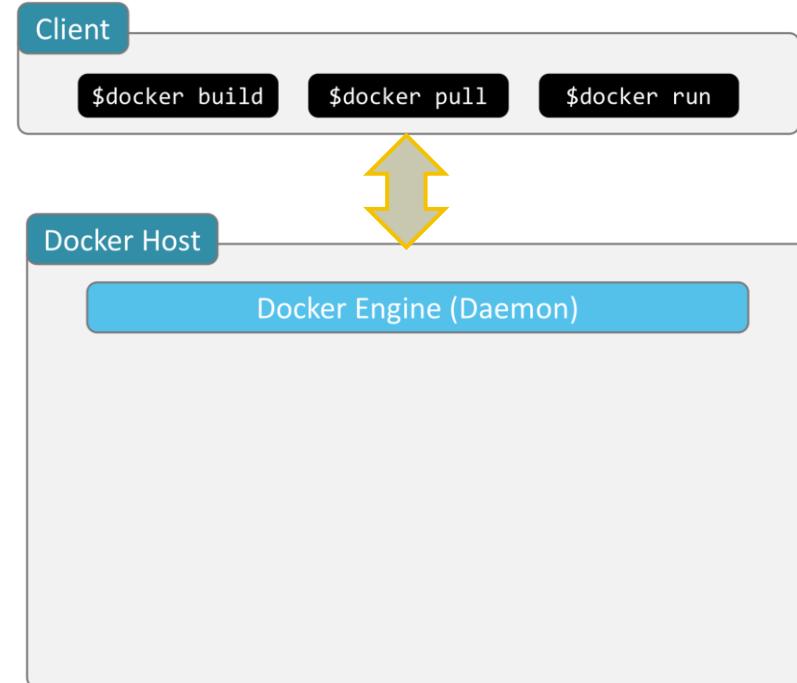
Docker Engine

Docker Registry

Docker Compose

Docker Swarm

- The Docker Client is the `docker` binary
 - Primary interface to the Docker Host
- Accepts commands and communicates with the Docker Engine (Daemon)



Docker Engine

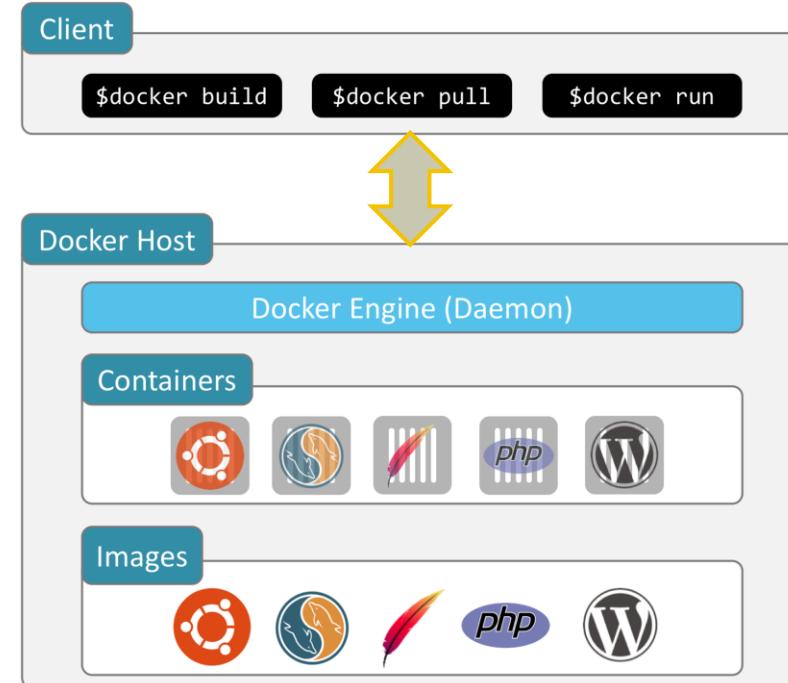
Docker Engine

Docker Registry

Docker Compose

Docker Swarm

- Lives on a Docker host
- Creates and manages containers on the host



Docker Registry

Docker Engine

Docker Registry

Docker Compose

Docker Swarm

Image Storage & Retrieval System



 QUAY by CoreOS

 Nexus

- Docker Engine **Pushes** Images to a Registry
- Version Control
- Docker Engine **Pulls** Images to Run



Docker Registry

Docker Engine

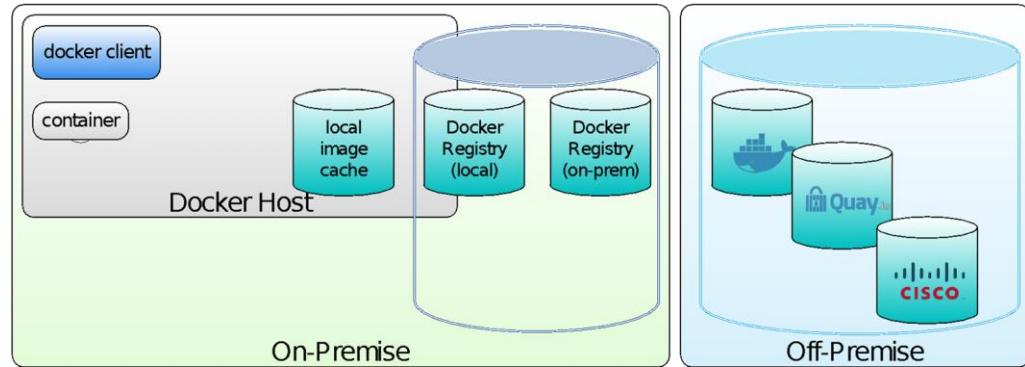
Docker Registry

Docker Compose

Docker Swarm

Types of Docker Registries

- Local Docker Registry (On Docker Host)
- Remote Docker Registry (On-Premise/Off-Premise)
- Docker Hub (Off-Premise)



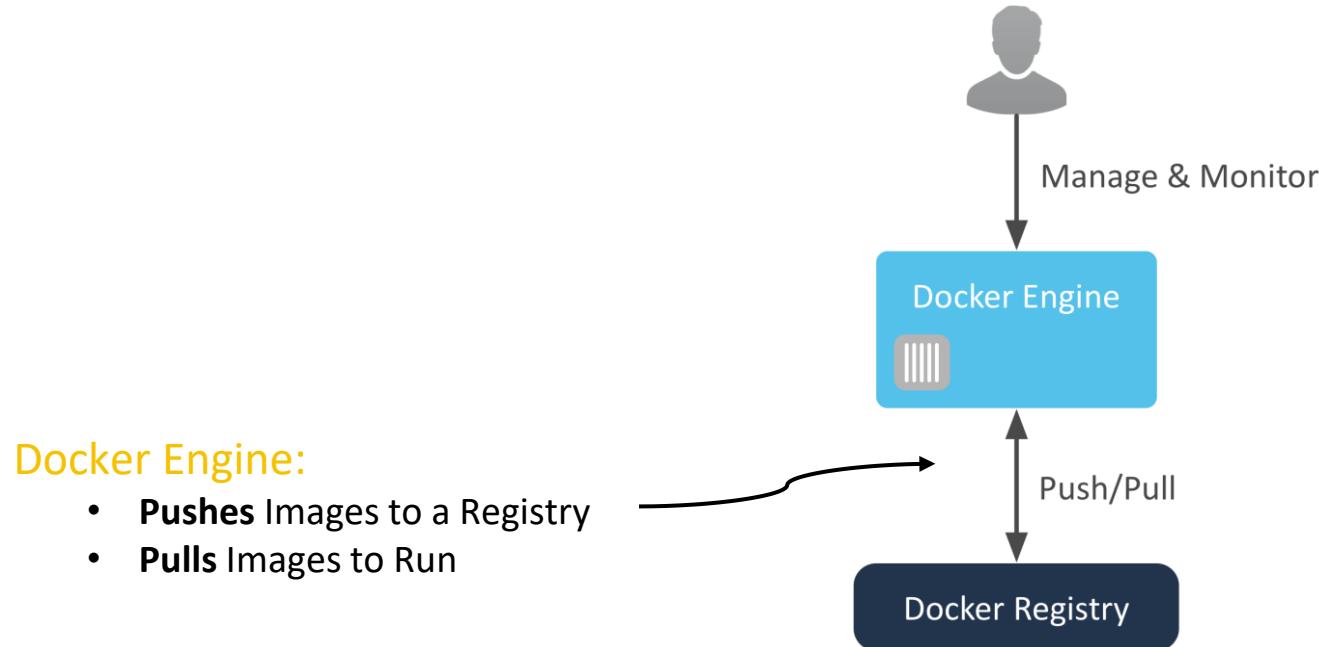
Docker Engine and Registry

Docker Engine

Docker Registry

Docker Compose

Docker Swarm



Docker Registry

Docker Engine

Docker Registry

Docker Compose

Docker Swarm

The registry and engine both present APIs

- All of Docker's functionality will utilize these APIs
- RESTFUL API
- Commands presented with Docker's CLI tools can also be used with curl and other tools

Docker Compose

Docker Engine

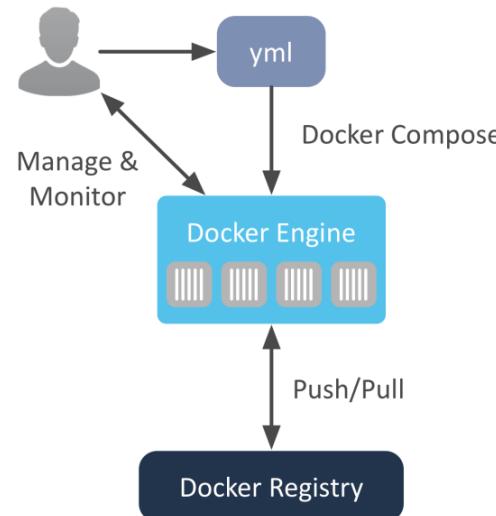
Docker Registry

Docker Compose

Docker Swarm

Tool to create and manage multi-container applications

- Applications defined in a single file:
docker-compose.yml
- Transforms applications into individual containers that are linked together
- Compose will start all containers in a single command



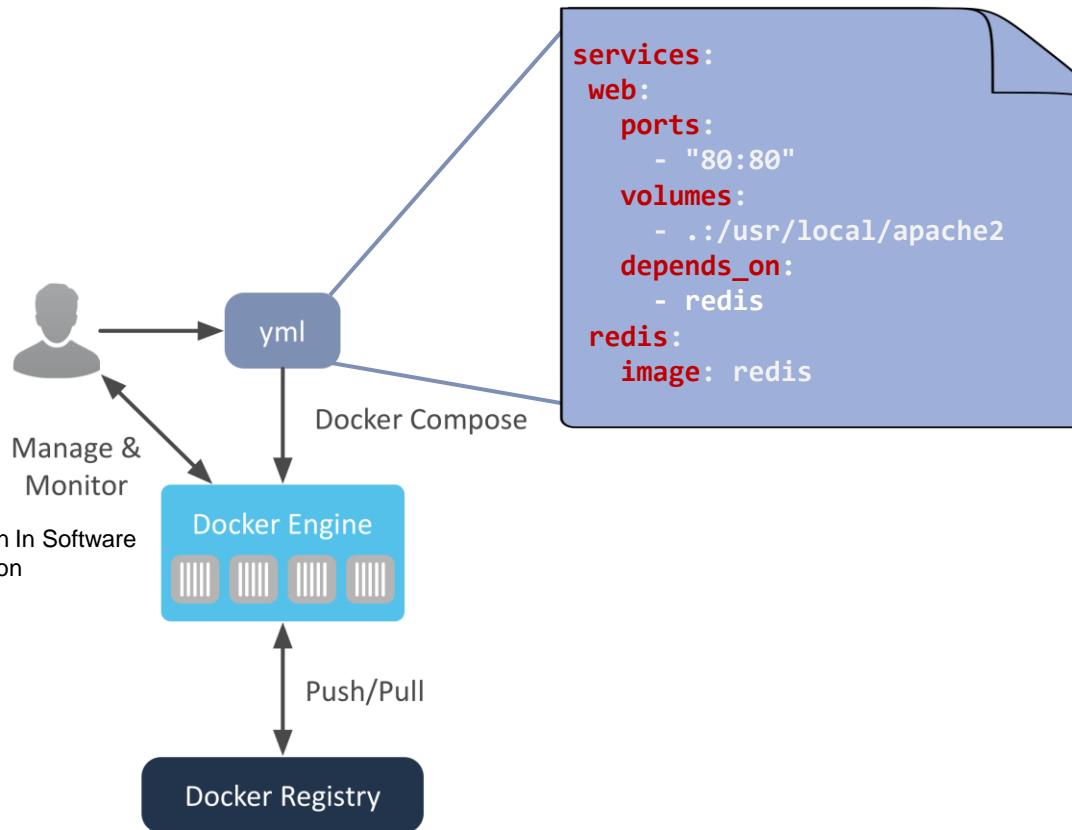
Docker Compose

Docker Engine

Docker Registry

Docker Compose

Docker Swarm



Docker Registry

Start your registry

```
docker run -d -p 5000:5000 --name registry registry:2
```

Pull (or build) some image from the hub

```
docker pull ubuntu
```

Tag the image so that it points to your registry

```
docker image tag ubuntu localhost:5000/myfirstimage
```

Push it

```
docker push localhost:5000/myfirstimage
```

Pull it back

```
docker pull localhost:5000/myfirstimage
```

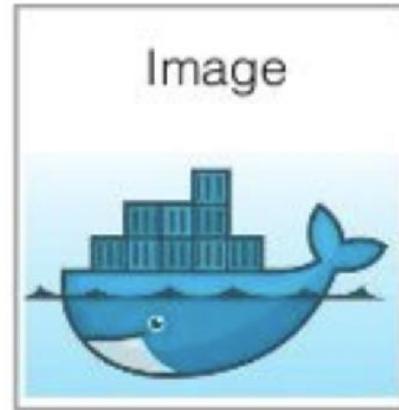
Now stop your registry and remove all data

```
docker container stop registry && docker container rm -v registry
```

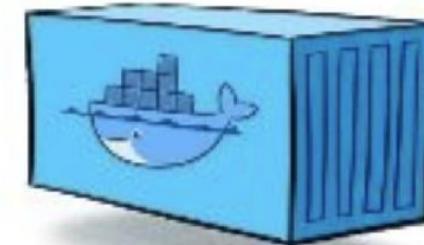
Docker Images

```
# Dockerfile content goes here  
# Dockerfile syntax follows the standard UNIX-style  
# command syntax.  
# The name directory is the local Dockerfile's  
# directory relative to the current working directory.  
  
# FROM uses the official Docker image for Python.  
# Dockerfiles can use any Dockerfile.  
  
# RUN installs Python 3.6.9 and pip 19.0.3.  
# Dockerfiles can use any command.  
  
# EXPOSE defines the ports exposed by the container.  
# Dockerfiles can use any port number.  
# EXPOSED ports are automatically mapped to host ports.  
# Dockerfiles can use any port number.  
  
# CMD specifies the command to run when the container starts.  
# Dockerfiles can use any command.  
# CMD can also be set to an empty string.  
# Dockerfiles can use any command.  
  
# ENTRYPOINT defines the command to run when the container starts.  
# Dockerfiles can use any command.  
# ENTRYPOINT can also be set to an empty string.  
# Dockerfiles can use any command.  
  
# WORKDIR sets the working directory inside the container.  
# Dockerfiles can use any directory path.  
# WORKDIR can also be set to an empty string.  
# Dockerfiles can use any directory path.
```

build



run



Dockerfile

Docker Image

Docker Container

Dockerfile

A Docker image consists of read-only layers each of which represents a Dockerfile instruction. The layers are stacked and each one is a delta of the changes from the previous layer. Consider this [Dockerfile](#):

```
# syntax=docker/dockerfile:1
FROM ubuntu:18.04
COPY . /app
RUN make /app
CMD python /app/app.py
```

Each instruction creates one layer:

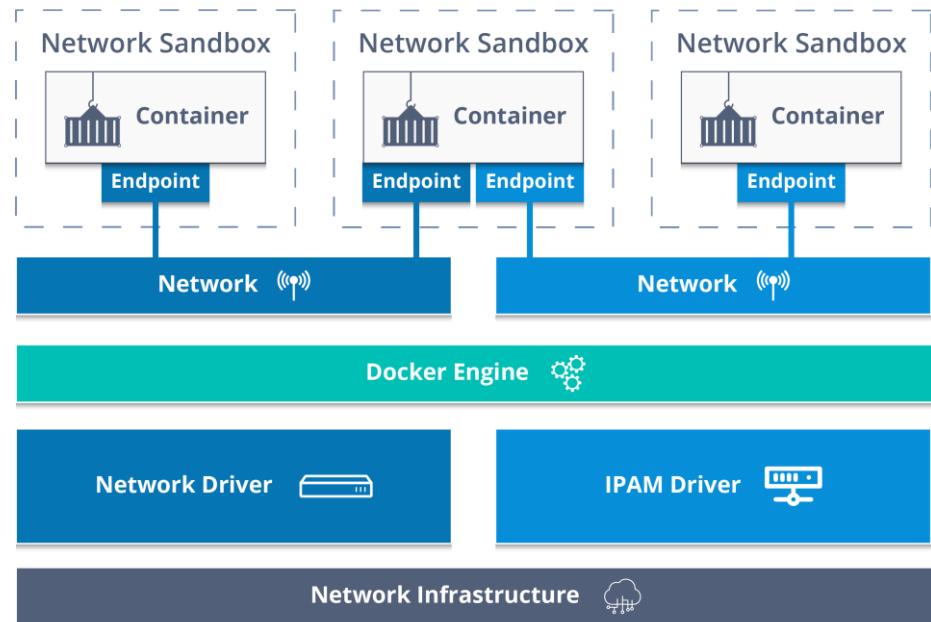
- **FROM** creates a layer from the [ubuntu:18.04](#) Docker image.
- **COPY** adds files from your Docker client's current directory.
- **RUN** builds your application with [make](#).
- **CMD** specifies what command to run within the container.

Dockerfile Best Practices



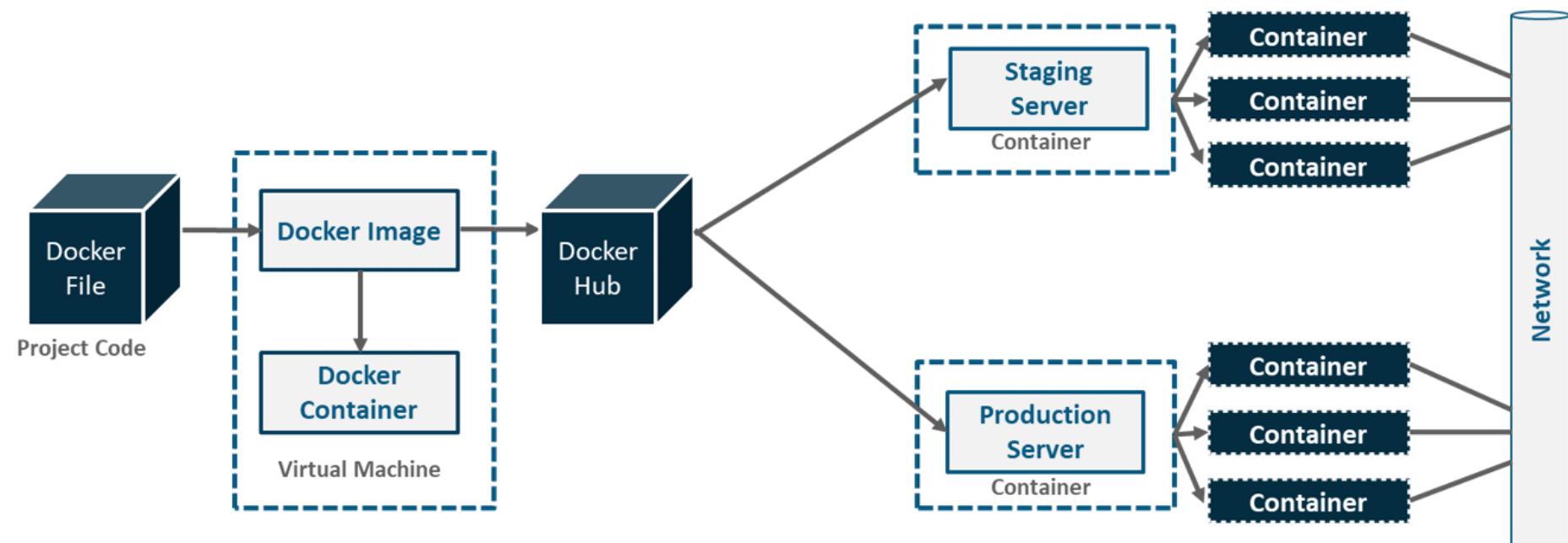
Docker Networking

One of the reasons Docker containers and services are so powerful is that you can connect them together, or connect them to non-Docker workloads. Docker containers and services do not even need to be aware that they are deployed on Docker, or whether their peers are also Docker workloads or not. Whether your Docker hosts run Linux, Windows, or a mix of the two, you can use Docker to manage them in a platform-agnostic way.



Architecture of Container Networking Model

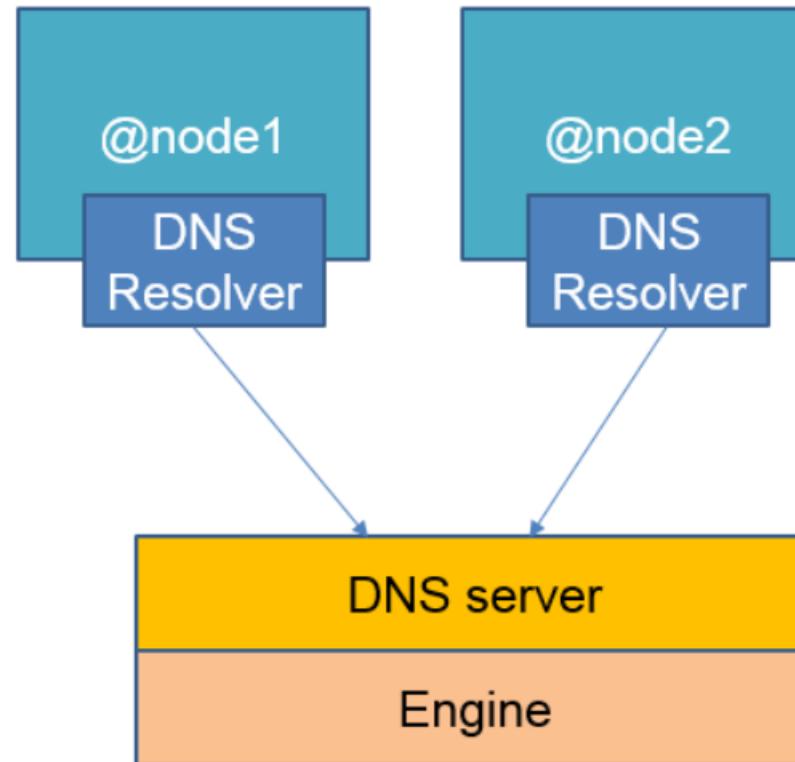
Networking



Docker Service Discovery

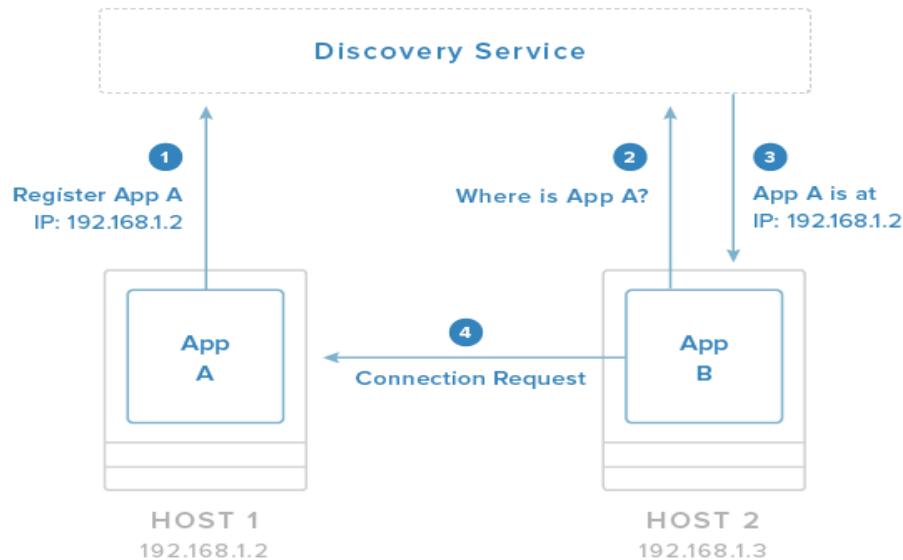
Service Discovery:

- Provided by Embedded DNS
- Highly Available
- Uses Network Control Plane to learn state
- Can be used to discover services and containers



Discovery Flow

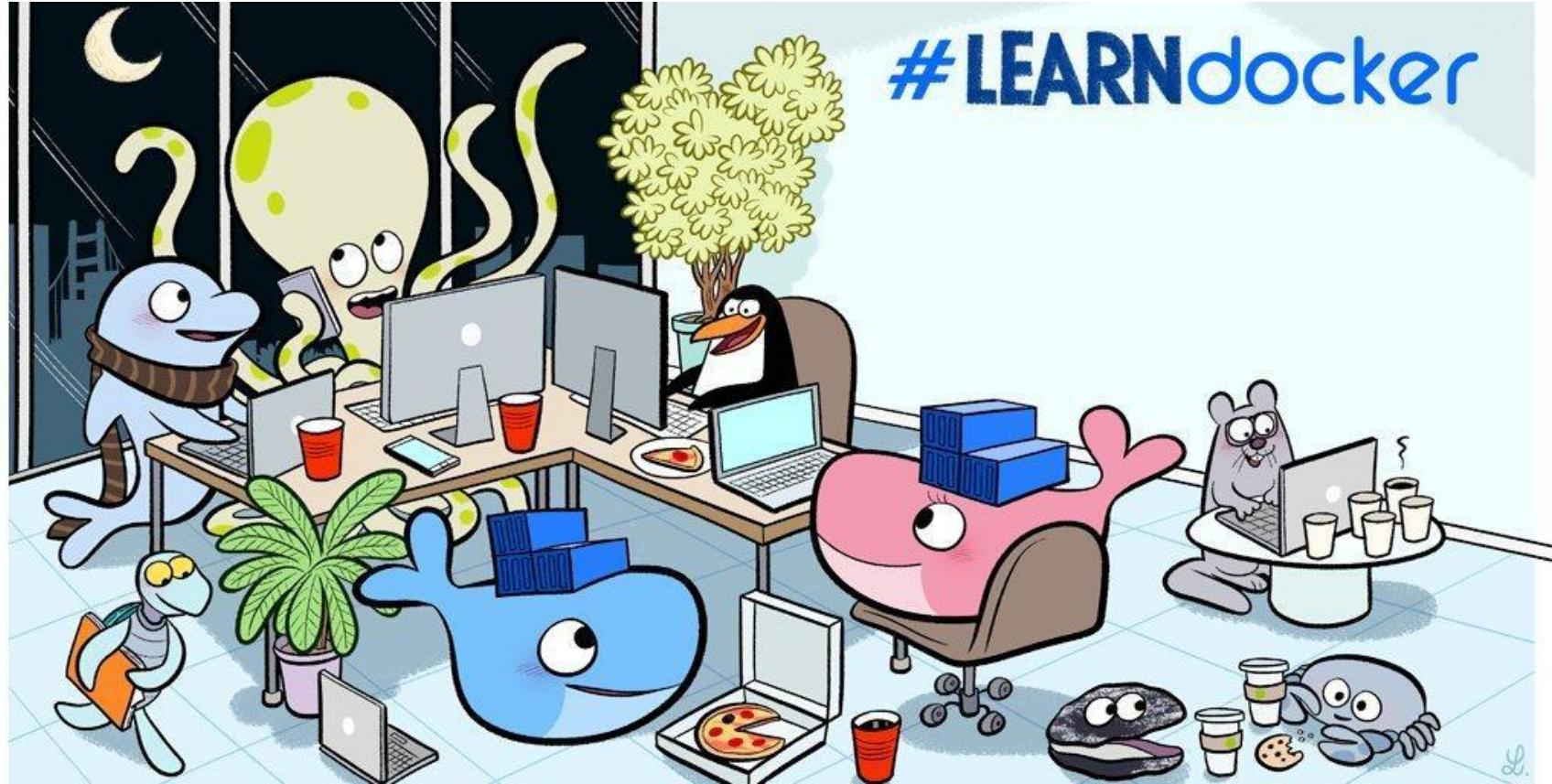
DISCOVERY FLOW



Discovery Tools

- **etcd**: This tool was created by the makers of CoreOS to provide service discovery and globally distributed configuration to both containers and the host systems themselves. It implements an http API and has a command line client available on each host machine.
- **consul**: This service discovery platform has many advanced features that make it stand out including configurable health checks, ACL functionality, HAProxy configuration, etc.
- **zookeeper**: This example is a bit older than the previous two, providing a more mature platform at the expense of some newer features.

STEP 1



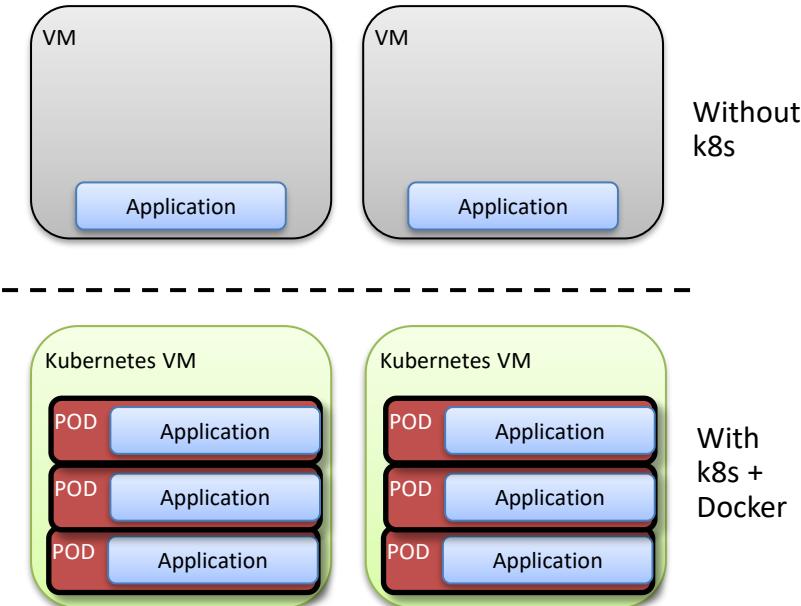
#LEARNdocker

K8S Overview



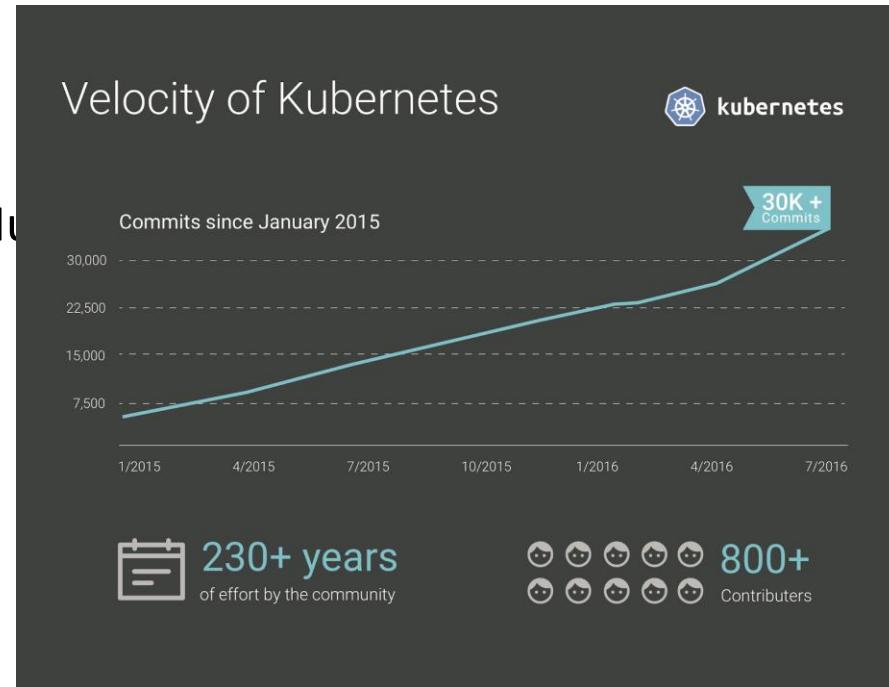
Kubernetes: Google Cloud Platform

- After launch of Google Compute Engine, utilization of VM's by customers was observed to be very low
 - Running apps in whole VM's leading to stranded resources
- GCE itself already running on Borg, which solved utilization through efficient scheduling of containerized applications
- Google engineers pushed to found Kubernetes, as open source project
 - K8s available to GCE customers
 - Not tied to Google or GCP infrastructure

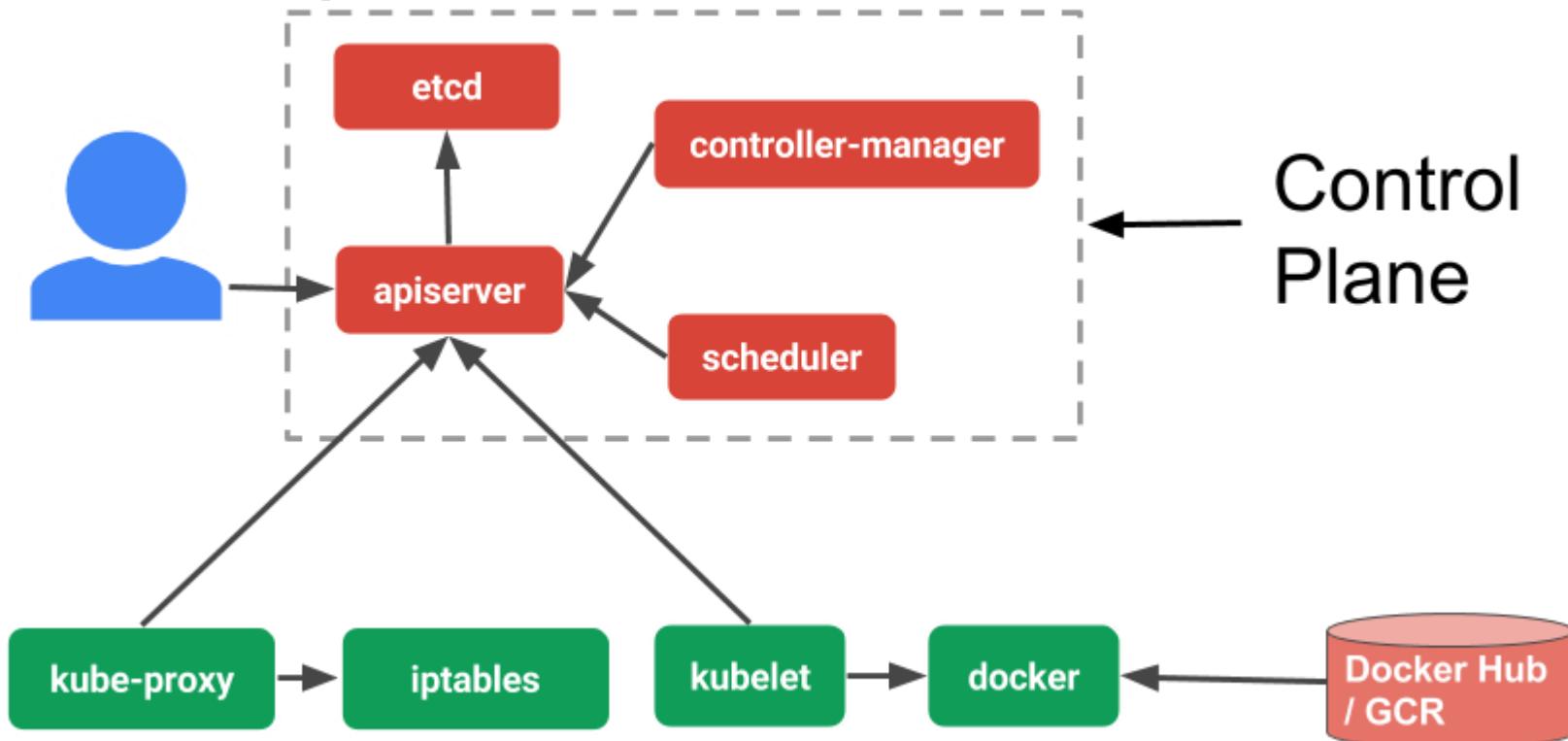


Kubernetes today – CNCF Open Source Project

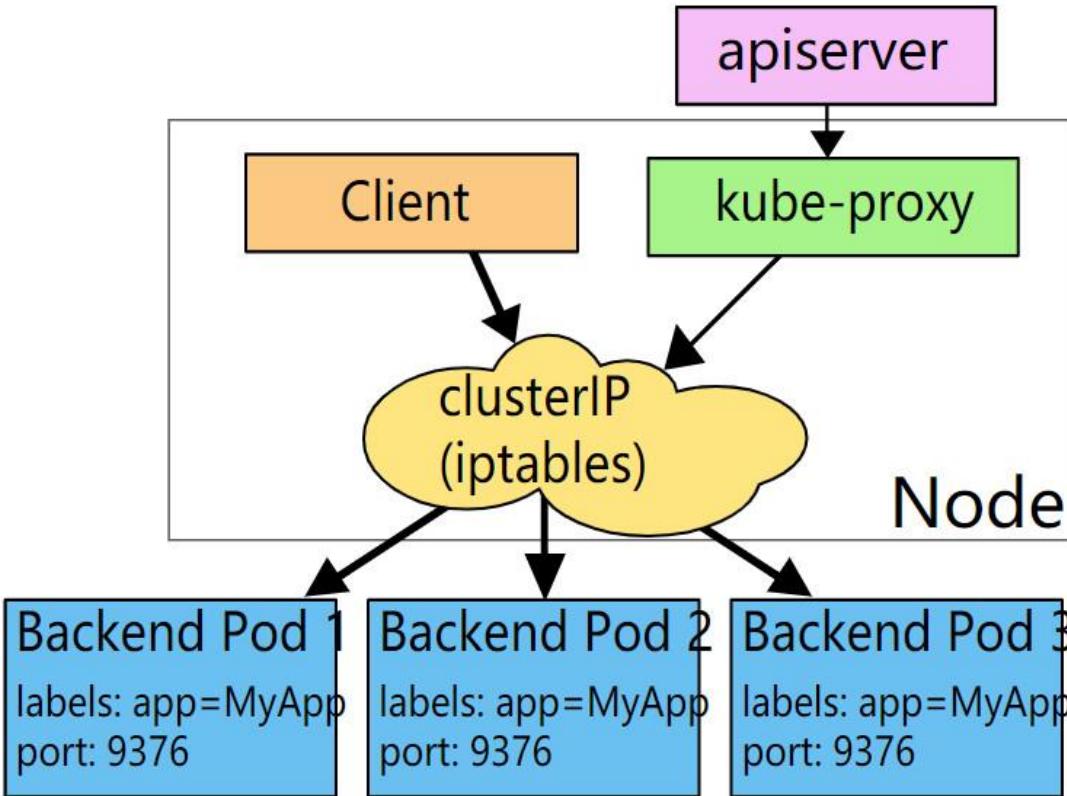
- Cloud Native Computing Foundation (CNCF) founded in 2015 as Linux Foundation Collaborative Project
- Kubernetes contributed by Google to CNCF at same time, still the flagship project
 - One of the most active projects on GitHub
- CNCF membership includes growing range of major industry vendors, including Cisco
- CNCF also governs the major container engine projects, thanks to recent donations
 - containerd (core runtime of Docker Engine)
 - rkt



Kube-apiserver



Kube Proxy

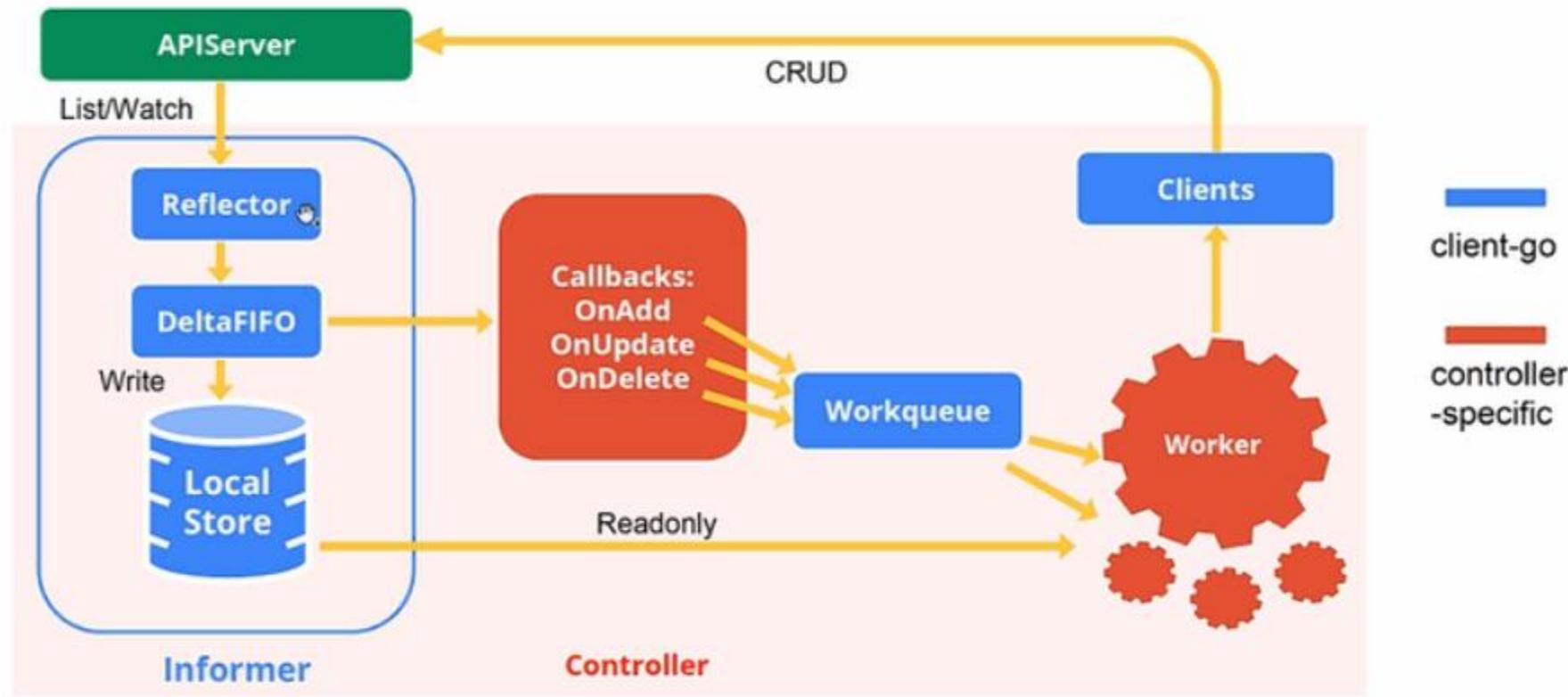


ETCD

ETCD is a distributed, reliable key-value store that is simple, secure and Fast. It is used to hold and manage the critical information that distributed systems need to keep running. It most commonly manages the configuration data, state data, and metadata for Kubernetes, the popular [container orchestration](#) platform.



General pattern of a Kubernetes controller



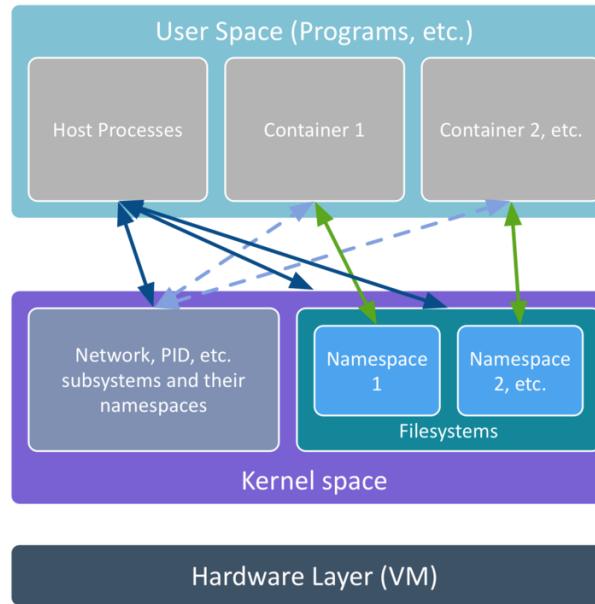
Control Groups

Control Groups

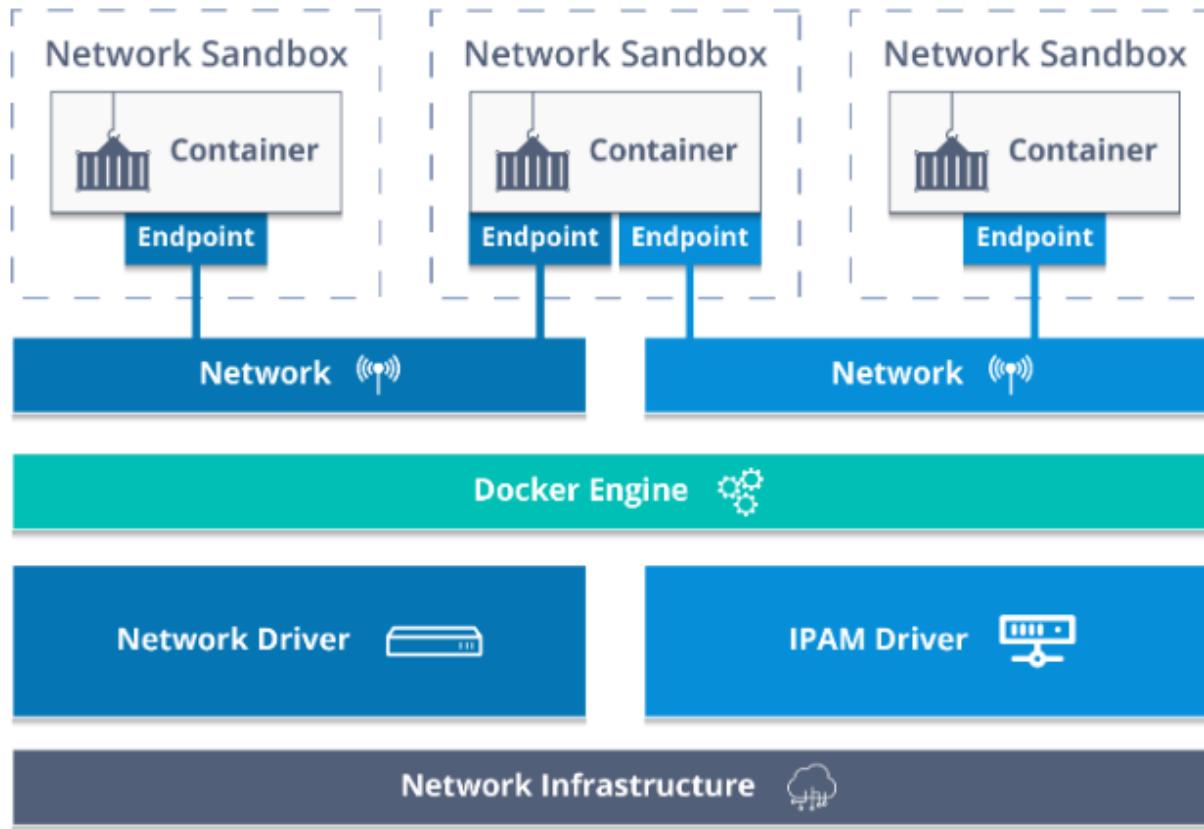
Isolation with Namespaces

Namespaces - Limits what a container can see (and therefore use)

- Namespace wrap a global system resource in an abstraction layer
- Processes running in that namespace think they have their own, isolated resource
- Isolation includes:
 - Network stack
 - Process space
 - Filesystem mount points

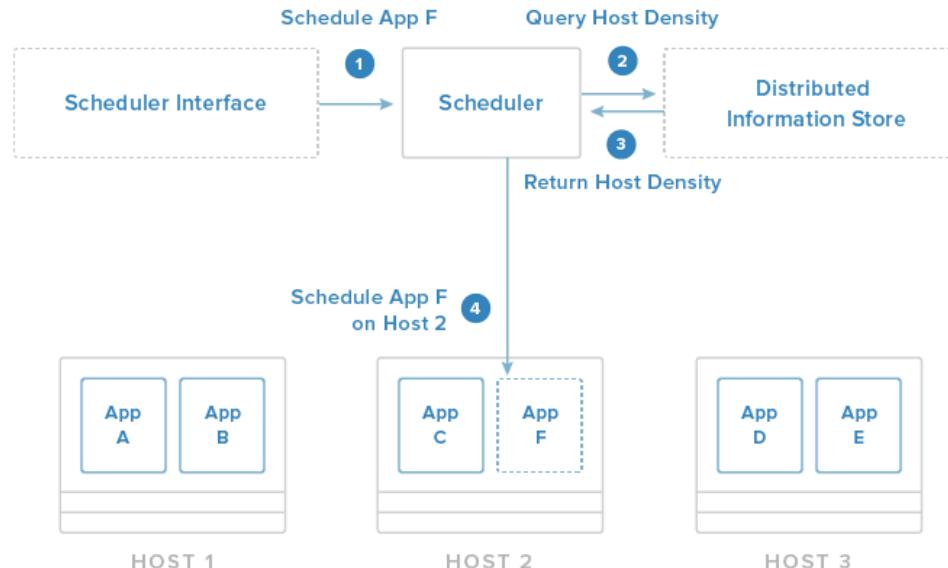


Container Network Model



Scheduling, Cluster Management, and Orchestration

EXAMPLE: SCHEDULE APP F



POP QUIZ:

SAAS – GETTING STARTED



What **is service mesh?**

A: a dedicated infrastructure layer for handling service-to-service communication.

B: A lightweight proxy that is distributed as sidecars.

C: lets you associate multiple resources with a single domain name

Kubernetes





Kubernetes

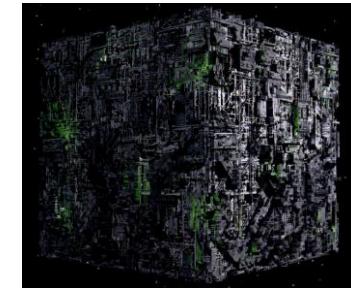
KIND stands for Kubernetes IN Docker, and as the name suggests, it creates a Kubernetes cluster using Docker to host the nodes. This is a novel approach, that takes advantage of Docker's easy, self-contained deployments and cleanup to create the test Kubernetes infrastructure.

kind is a tool for running local Kubernetes clusters using Docker container “nodes”.

kind was primarily designed for testing Kubernetes itself, but may be used for local development or CI

History of Kubernetes

- Google adopted containers as an application deployment standard over a decade ago
 - Contributed cgroups to Linux kernel in 2007
- Google developed generations of container management systems, scaling to thousands of hosts per cluster
 - First was Borg, treated as a trade secret until 2015*
 - Omega built on concepts of Borg, also Google-internal
 - Kubernetes inspired by observed needs of Google Cloud Platform customers, open source
- All major Google services run on Borg
- Other cluster management frameworks, like Apache Mesos, inspired by Borg



Star Trek Borg Cube
A hegemonizing swarm

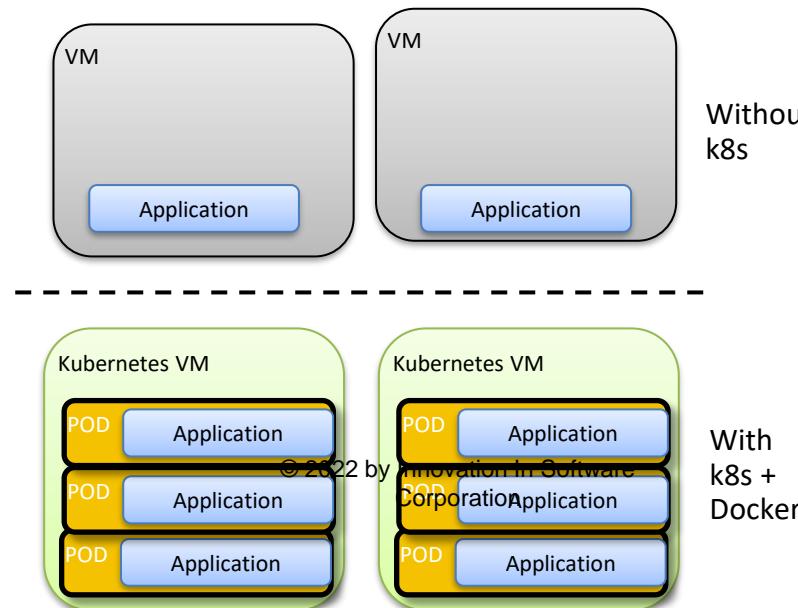


Original project name:
Seven of Nine
(the friendly Borg)

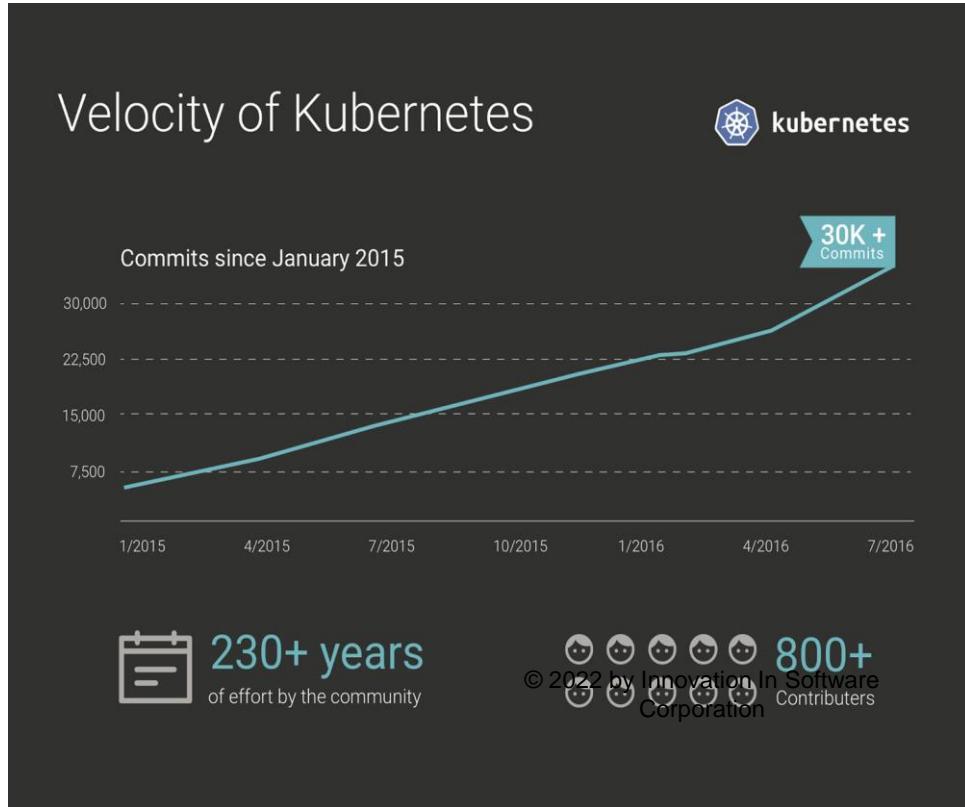
Kubernetes



Kubernetes: Virtualization vs Containerization

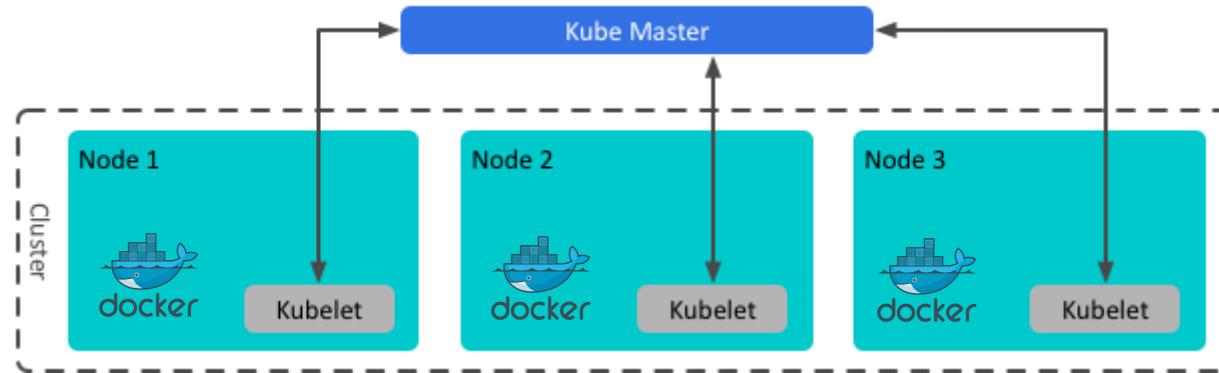


Kubernetes today – CNCF Open Source Project



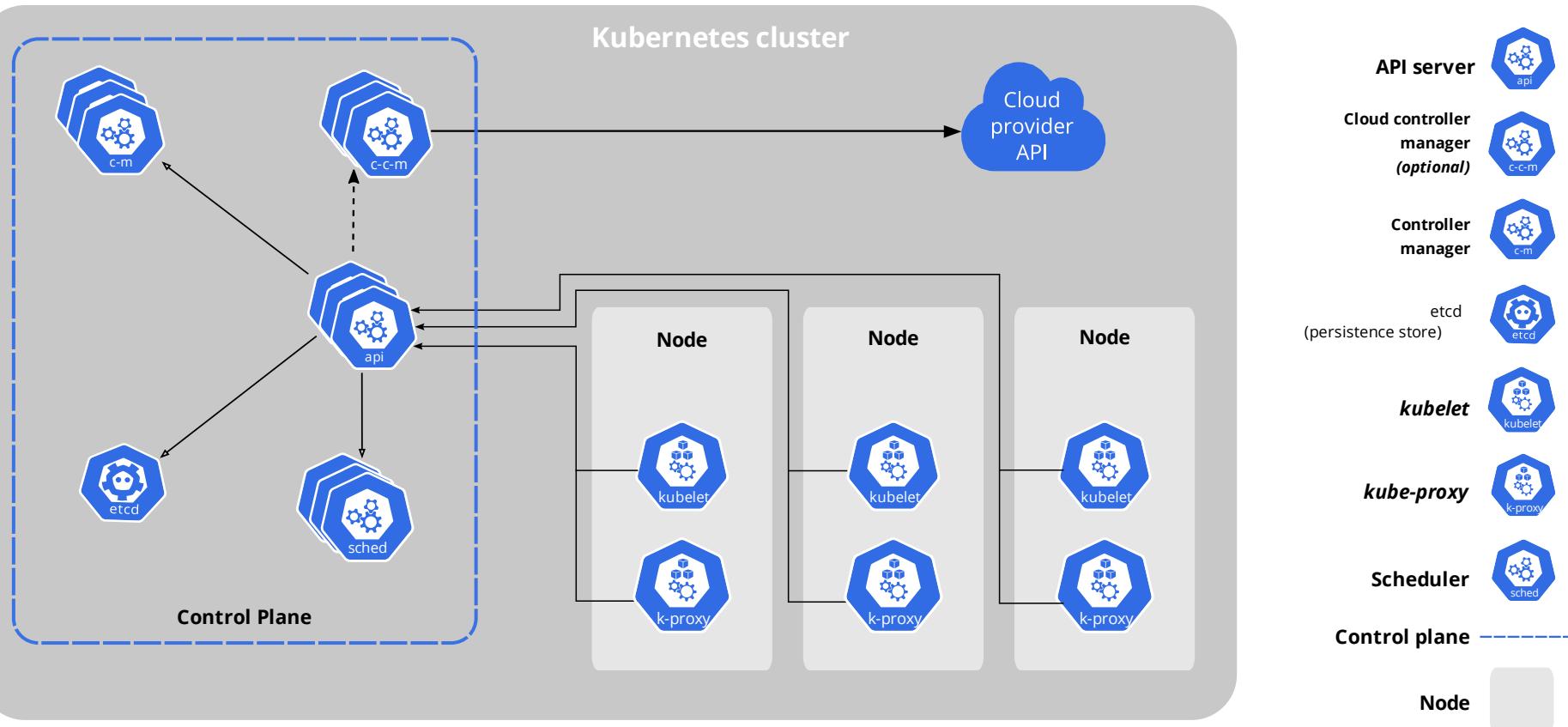
Kubernetes: Containers at Scale

Kubernetes provides the infrastructure for container-centric deployment and operation of applications



- Kubernetes resource objects provide key application management features, including
 - App elasticity and self-healing
 - Naming and discovery
 - Rolling updates
 - Request load balancing
 - Application health checking
 - Log access and resource monitoring

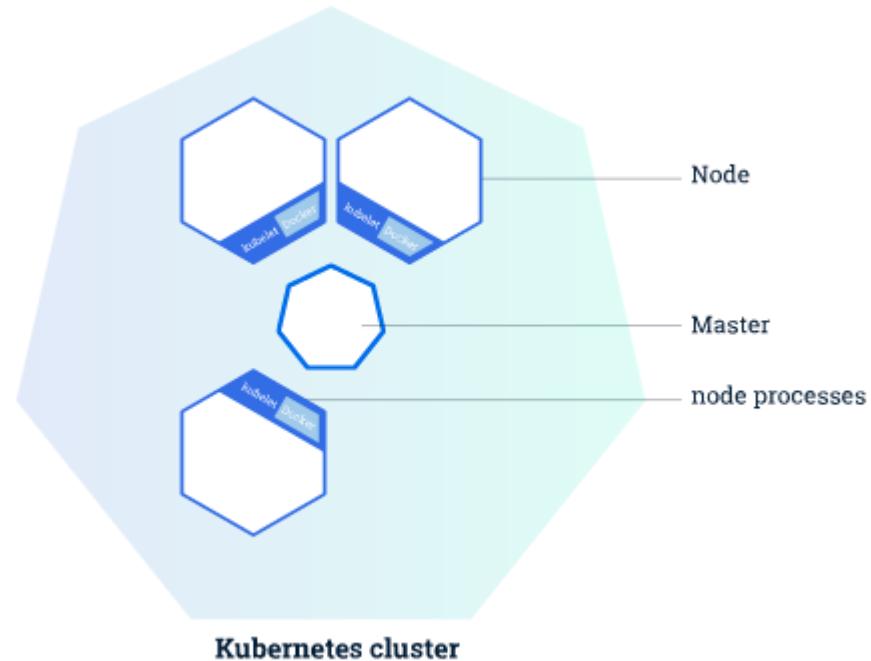
Cluster Components



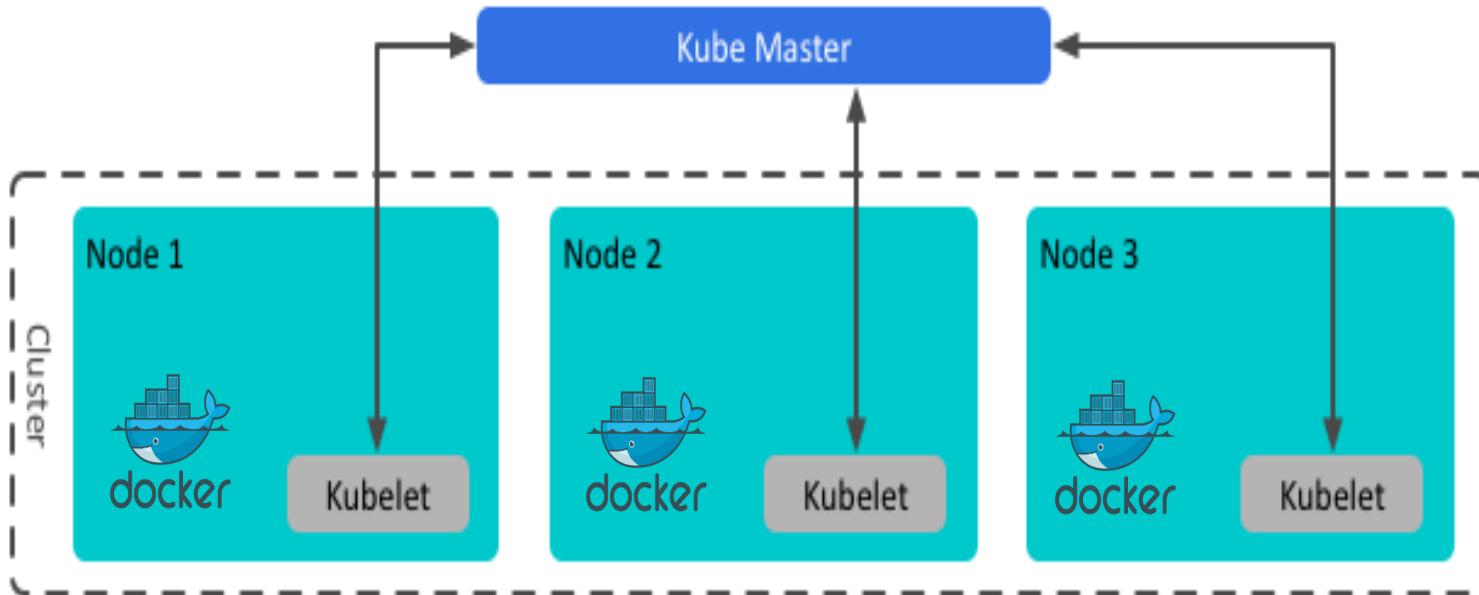
Creating a Cluster

Creating a Kubernetes cluster is as simple as `kind create cluster`.

By default, the cluster will be given the name `kind`. Use the `--name` flag to assign the cluster a different context name.

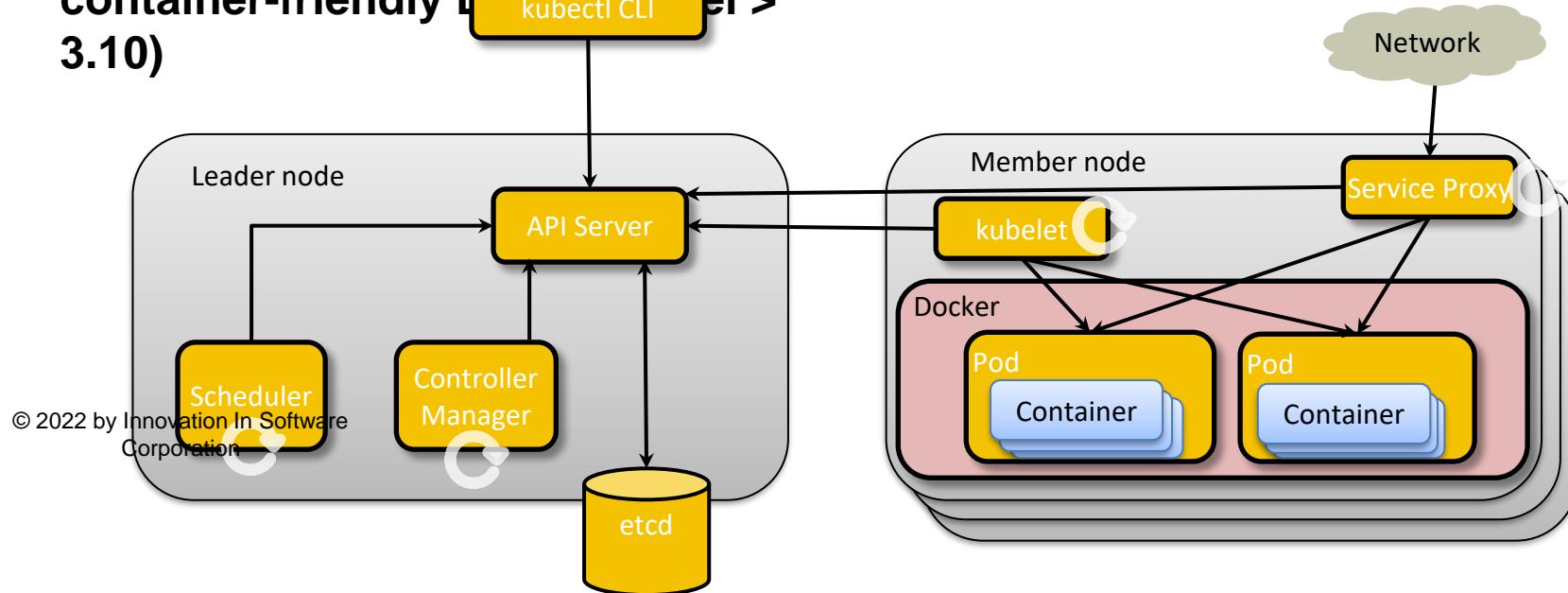


Kubernetes: Cluster



Kubernetes Cluster Architecture

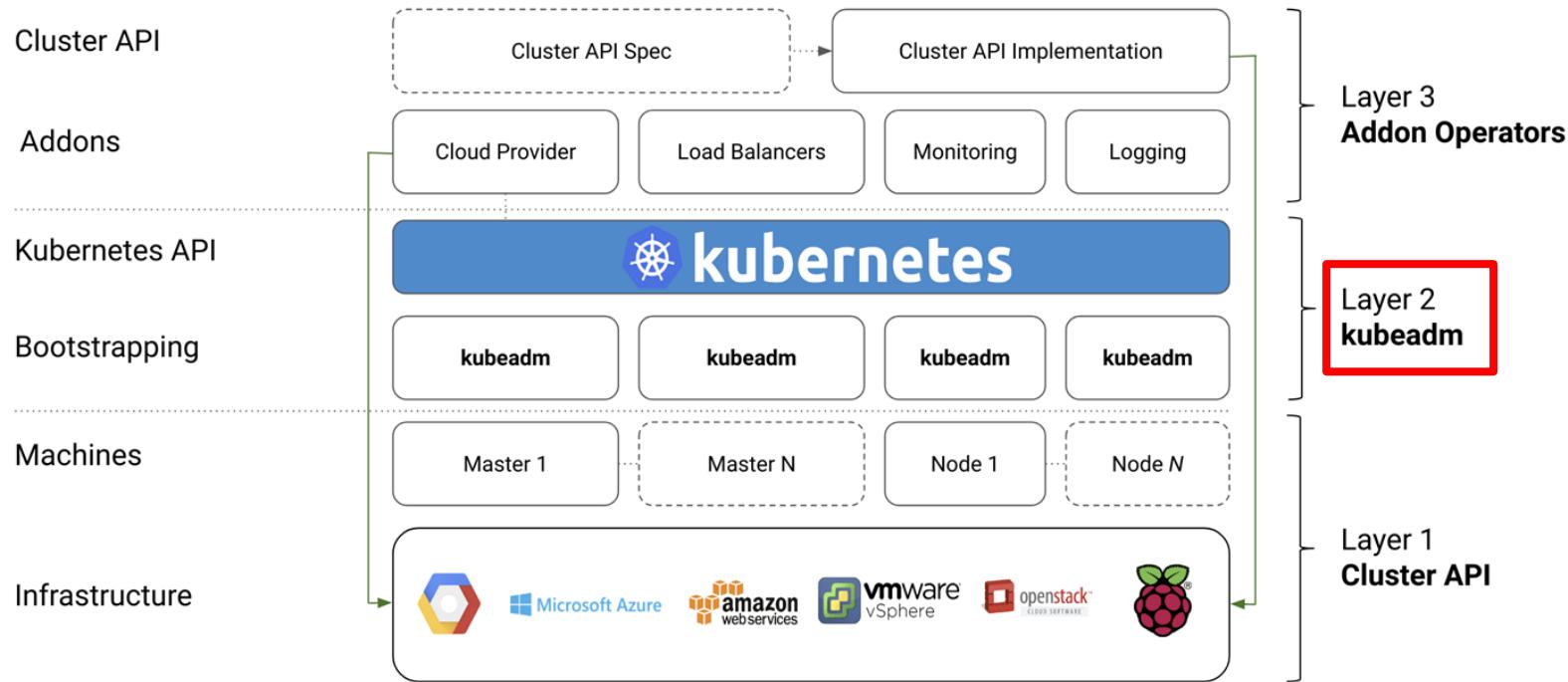
- Kubernetes nodes can be physical hosts or VM's running a container-friendly Linux kernel > 3.10)



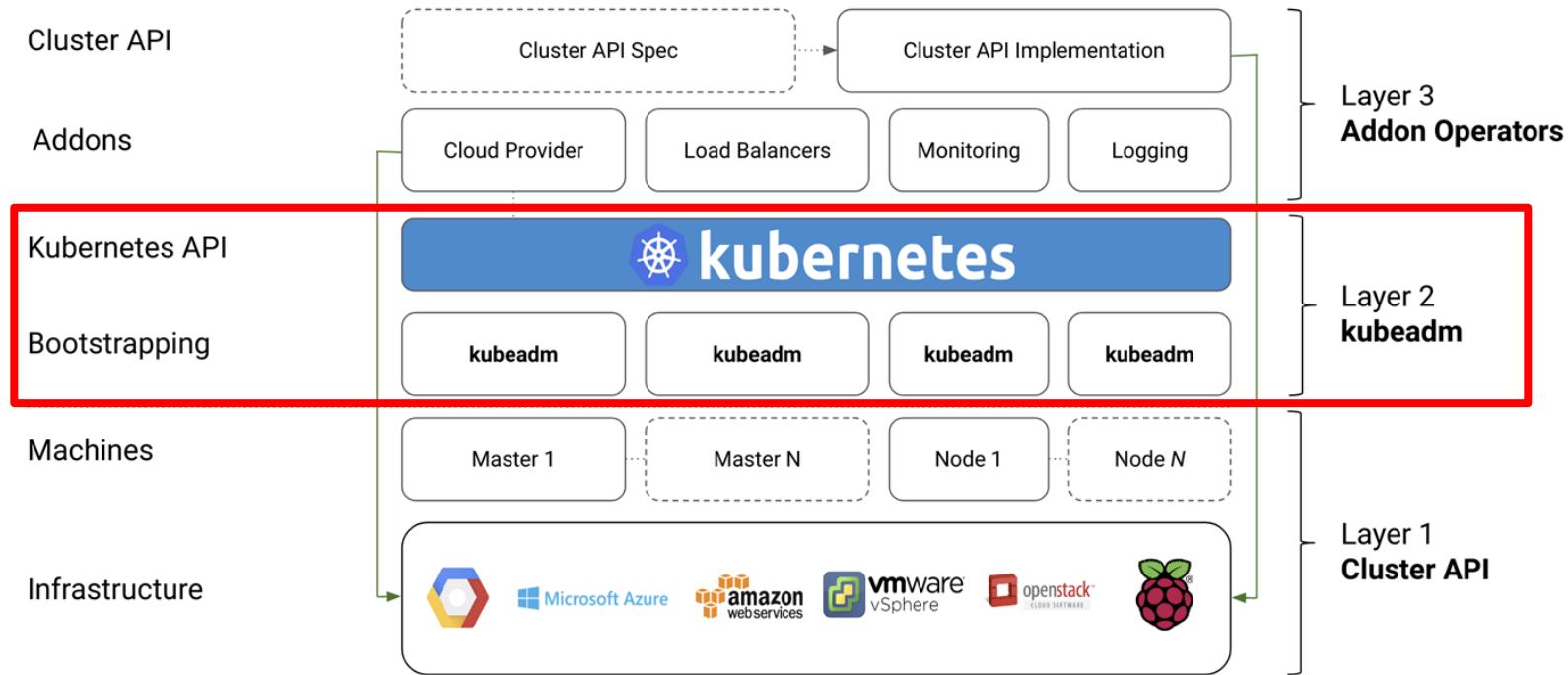
Kubeadm

- Active project since 2016
- kubeadm is focused on building a minimum viable, secure cluster.
- Limited scope, intended to provide building blocks.
 - Only deals with local filesystem and Kubernetes API
 - Agnostic to how the Kubelet is run
- Out of scope
 - Setting up a specific CNI is out of scope
 - non-critical addons (monitoring, logging and visualization)
 - Specific cloud provider integration

Kubeadm



Kubeadm



Kubeadm

- Supports High Availability clusters
 - Beta feature
- kubeadm is focused on building a minimum viable, secure cluster.
- Limited scope, intended to provide building blocks.
 - Only deals with local filesystem and Kubernetes API
 - Agnostic to how the Kubelet is run
- Out of scope
 - Setting up a specific CNI is out of scope
 - non-critical addons (monitoring, logging and visualization)
 - Specific cloud provider integration

Kubeadm

- Supports adding additional nodes through bootstrap tokens
- Manage Kubernetes cluster
 - Upgrades
 - Rollbacks
 - Token management

Kuber Controller Manager

Control plane component that runs [controller](#) processes.

Logically, each [controller](#) is a separate process, but to reduce complexity, they are all compiled into a single binary and run in a single process.

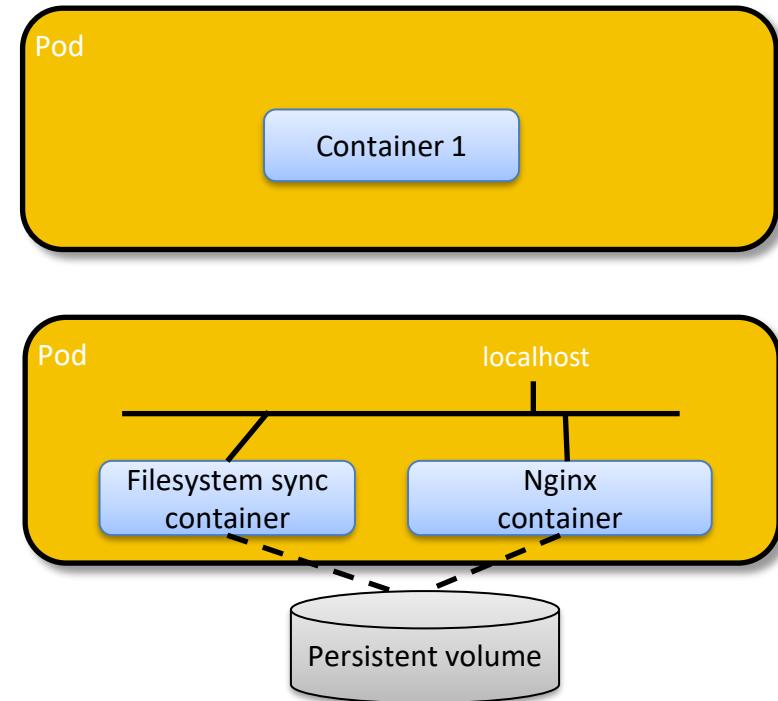
K8S Pods



What is a Kubernetes Pod?

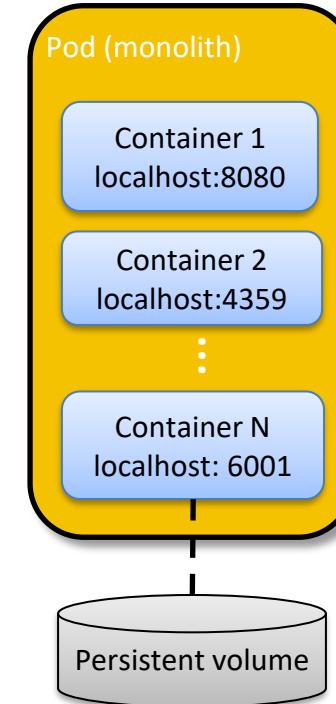
Kubernetes design intention: Pod == application instance

- **Basic unit of deployment is the pod, a set of co-scheduled containers and shared resources**
- **Pods can include more than one container, for tightly-coupled application components, e.g.**
 - Sidecar containers : nginx + filesystem synchronizer to update www from git
 - Content adapter: transform data to common output standard
- **Containers in a pod share network namespaces and mounted volumes**



Pods Enable Deployment Flexibility

- **Possible to use a single pod to run a monolithic application**
 - Each application process can be built as a container
 - All containers can access each other's ports on localhost
- **More advanced features of the K8s system available if application built instead from assemblages of pods, e.g.**
 - Web tier: Apache pods
 - Data tier: Redis master/slave pods
- **Pods provide scale and elasticity via replication – not possible in the monolith**
- **Best practice: assume every pod is mortal**



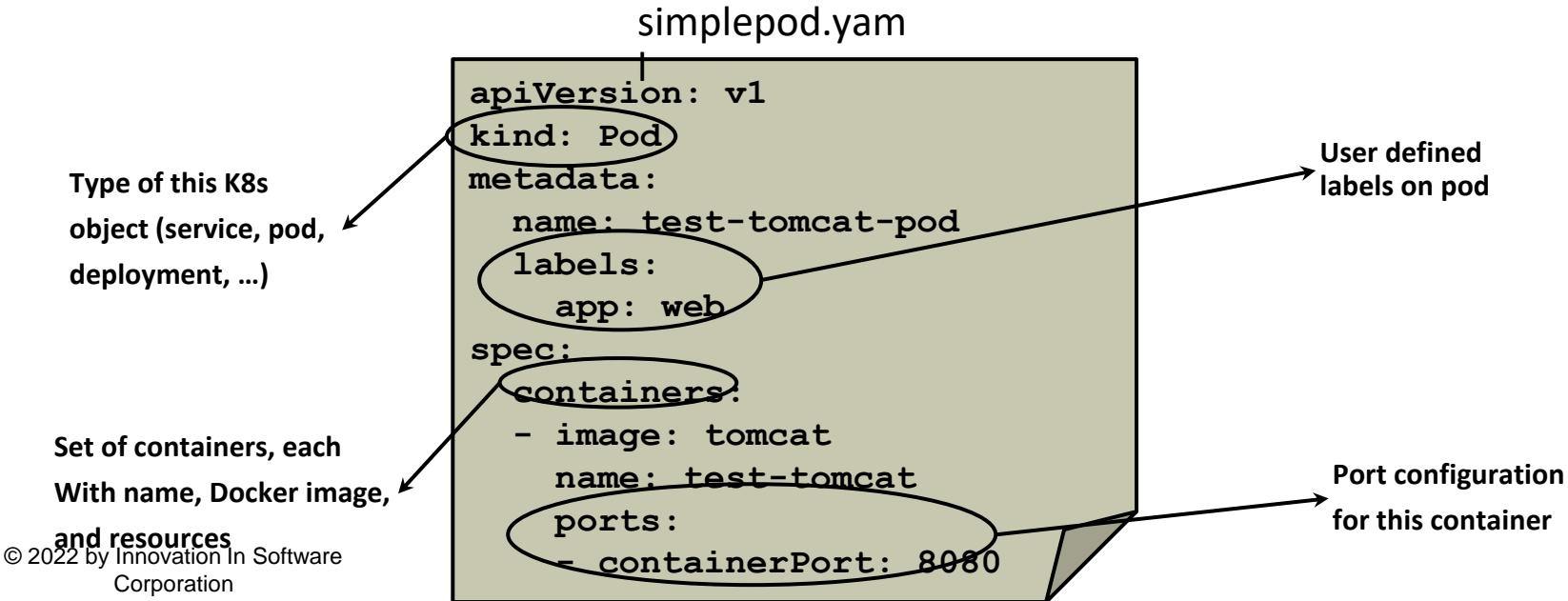
Defining a Pod via a Manifest File

Like other K8s objects, pods can be defined in YAML or JSON files

- **K8s API accepts object definitions in JSON, but manifests often in YAML**
- **YAML format used by a variety of other tools, e.g. Docker Compose, Ansible, etc.**
- **kind field value is ‘Pod’**
- **metadata includes**
 - **name** to assign to pod
 - **label** values
- **spec includes specifics of container images, ports, and other resources**

```
apiVersion: v1
kind: Pod
metadata:
  name: test-tomcat-pod
  labels:
    app: web
spec:
  containers:
  - image: tomcat
    name: test-tomcat
    ports:
    - containerPort: 8080
```

Looking at a Pod Manifest File



© 2022 by Innovation In Software Corporation

- Configuration options similar to creating Docker container directly

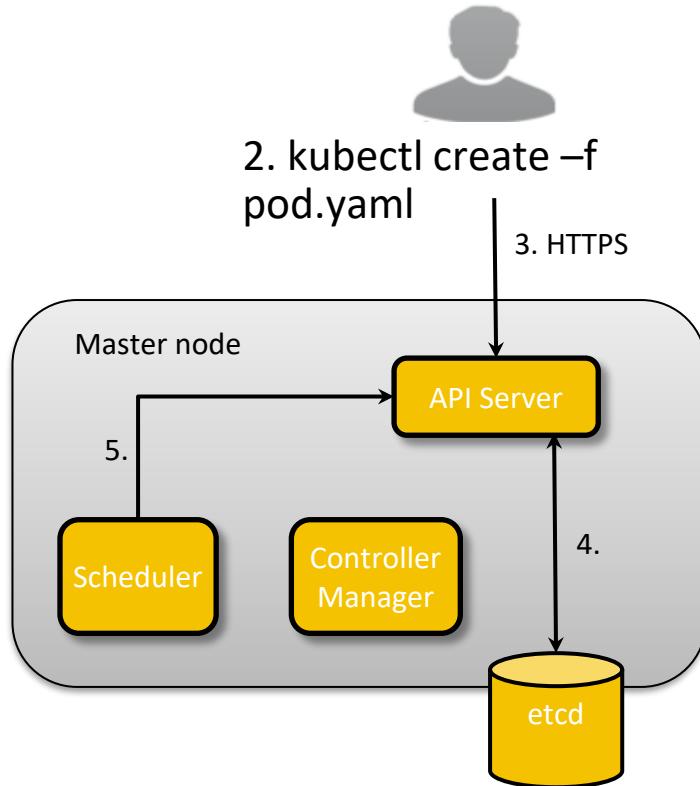
Defining a Pod with Multiple Containers

```
apiVersion: v1
kind: Pod
metadata:
  name: test-tomcat-pod
spec:
  containers:
    - image: tomcat
      name: test-tomcat
      ports:
        - containerPort: 8080
    - image: mysql
      name: test-mysql
      ports:
        - containerPort: 3306
```

→ multipod.yaml
|

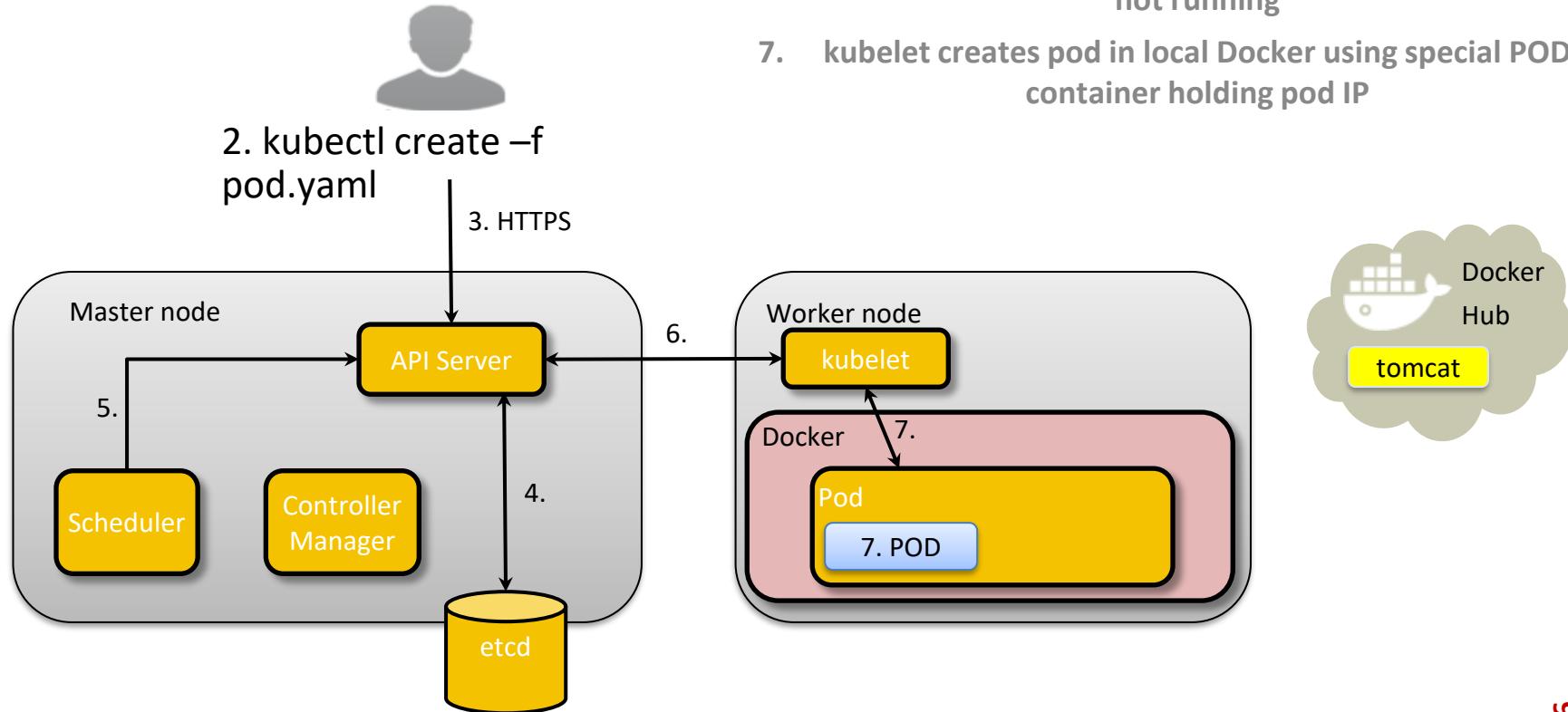
- Pod spec can contain multiple containers from different images
- Containers in pod share local network context and cluster IP for pod

Pod Creation Process

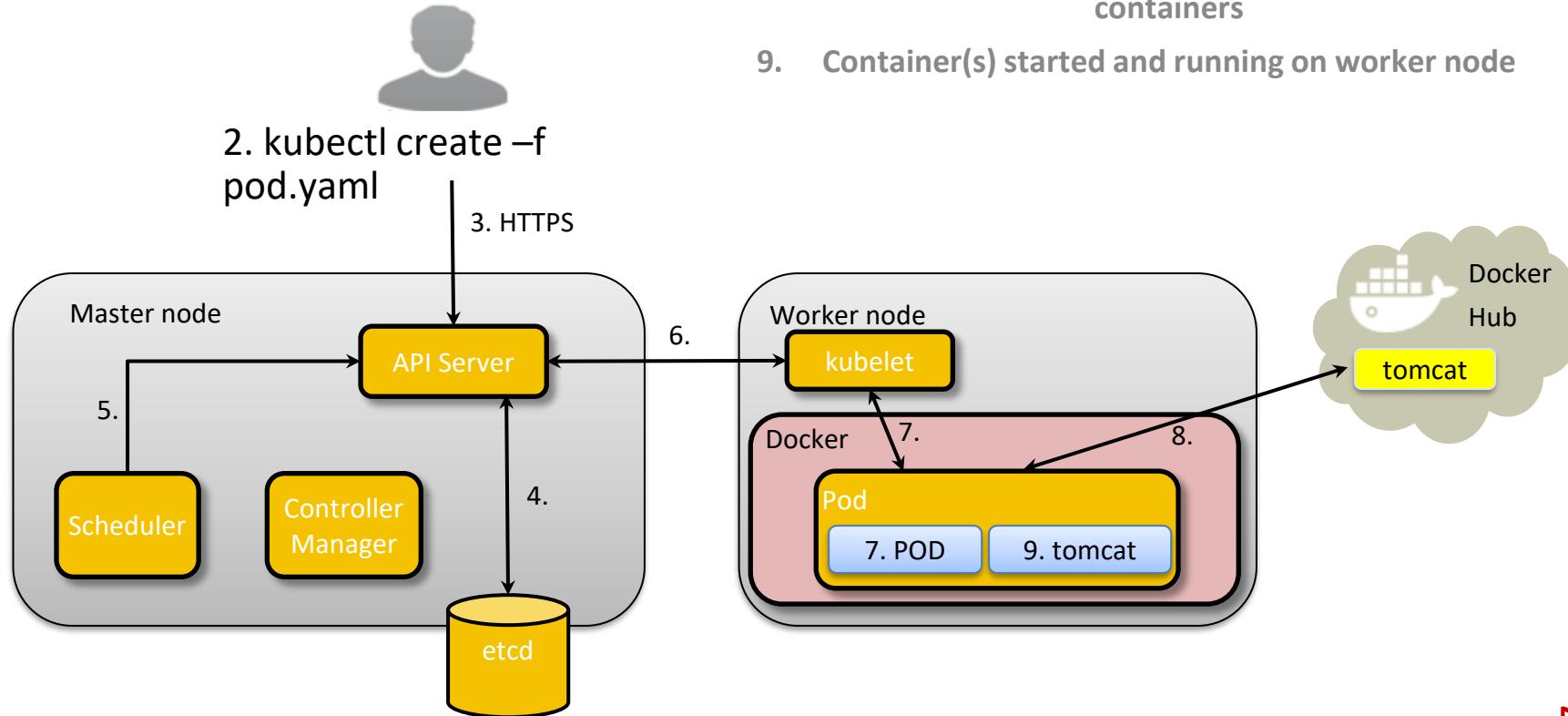


1. User writes a pod manifest file
2. User requests creation of pod from manifest via CLI
3. CLI tool marshals parameters into K8s RESTful API request (HTTP POST)
4. kube-apiserver creates new pod object record in etcd, with no node assignment
5. kube-scheduler notes new pod via API
 - a. Selects node for pod to run on
 - b. Updates pod record via API with node assignment

Pod Creation Process



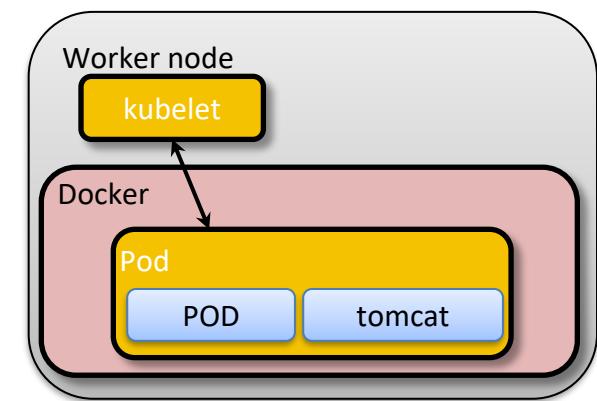
Pod Creation Process



Pod Lifecycles

- By default, K8s Pods have an indefinite lifetime, which is not immortality
 - **restartPolicy** of Always by default
 - **restartPolicy** of Never or OnFailure also available for terminating jobs
- Node's kubelet will create and keep running containers for pods assigned to node, per the pod specs
- If a Pod container fails to start, or unexpectedly exits, kubelet will restart it
 - Can see container lifecycle events via 'kubectl describe pod <PODNAME>'
- If node is lost, its Pods are also lost – K8s will not rebind Pods to another node

```
apiVersion: v1
kind: Pod
metadata:
  name: test-tomcat-pod
  labels:
    app: web
spec:
  containers:
  - image: tomcat
    name: test-tomcat
    ports:
    - containerPort: 8080
```



Modifying a Pod

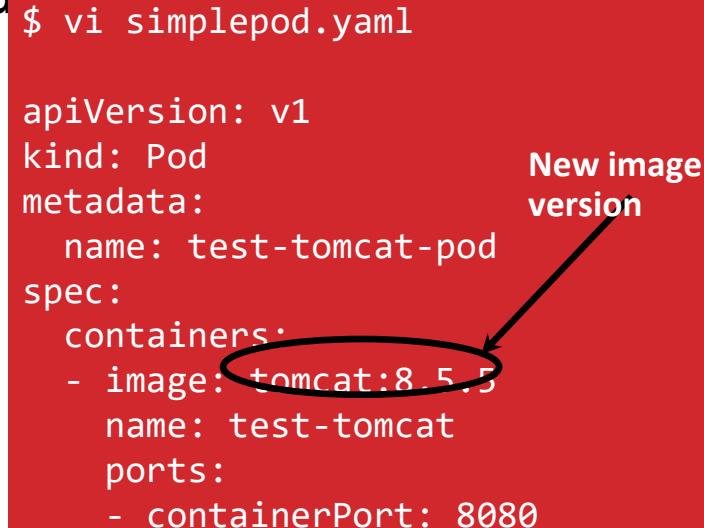
Change the container version

- You can make changes to the desired state of a pod via updating the manifest file
- Changes can then be applied to the pod via the command
 - `kubectl apply -f <manifest.yaml>`
- Changing a container image as shown will result in K8s automatically killing and recreating the pod's workload container

```
$ vi simplepod.yaml
```

```
apiVersion: v1
kind: Pod
metadata:
  name: test-tomcat-pod
spec:
  containers:
    - image: tomcat:8.5.5
      name: test-tomcat
      ports:
        - containerPort: 8080
```

New image version



Modifying a Pod

```
$ kubectl apply -f simplepod.yaml
pod "test-tomcat-pod" configured

$ kubectl describe pod test-tomcat-pod
Name:           test-tomcat-pod
Namespace:      default
...
Labels:         tier=frontend
Status:         Running
...
Containers:
  test-tomcat:
    Image:          tomcat:8.5.5
    Image ID:
    Port:           8080/TCP
    State:          Running
...

```

New version
running



Labeling Pods

User-defined labels help organize K8s resources

- **Labels are key/value pairs that users can assign and update on any K8s resources, including pods**
- **Other K8s objects, like controllers, use labels to select pods to govern**
- **Labels can also be used to filter data queries with *kubectl*, e.g.**
 - `kubectl get pods -l <label=value>`
- **Labels can be used to distinguish pods on any criteria, such as**
 - Application, application tier, version, environment state, etc.
- **K8s system does not require specific labels to be used – all user-defined**

Labeling a Pod

```
$ kubectl label pod test-tomcat-pod tier=frontend  
pod "test-tomcat-pod" labeled
```

```
$ kubectl describe pods test-tomcat-pod  
Name:           test-tomcat-pod  
...  
Labels:         tier=frontend
```

```
$ kubectl get pods -l tier=frontend  
NAME          READY   STATUS    RESTARTS   AGE  
test-tomcat-pod  1/1     Running   0          1d
```

Reviewing Labels on Pods

Changing kubectl output

- You can display pod labels via a flag on the ***kubectl*** command

```
$ kubectl get pods --show-labels
NAME           READY   STATUS    RESTARTS   AGE   LABELS
test-tomcat-pod 1/1     Running   1          1d    tier=frontend

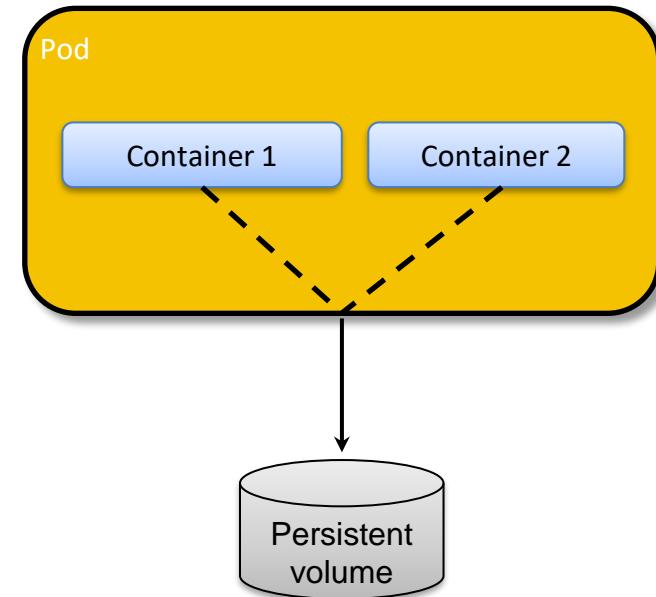
$ kubectl get pods --show-labels --namespace=kube-system
NAME           READY   STATUS    RESTARTS   AGE   LABELS
kube-addon-manager-minikube 1/1     Running   3          8d    component=kube-
addon-manager,addons=addon-manager,version=v6.4-alpha.1
kube-dns-v20-mm0zl          3/3     Running   9          8d    k8s-app=kube-
dns,version=v20
kubernetes-dashboard-kc9rk  1/1     Running   3          8d    app=kubernetes-
dashboard,kubernetes.io/cluster-service=true,version=v1.6.0
```

Deleting Pods

Pod deletion will discard all local pod resources

- When deleting a Pod, its containers will be removed and pod IP relinquished
- If an application needs to persist data, its pods must be configured to use persistent volumes for storage
- If a node dies, its local pods are also gone
- **Best practice: use controller resources instead of managing pods directly**
- **Best practice: use service resources to build reliable abstraction layers for clients**

© 2022 by Innovation In Software
Software Engineering

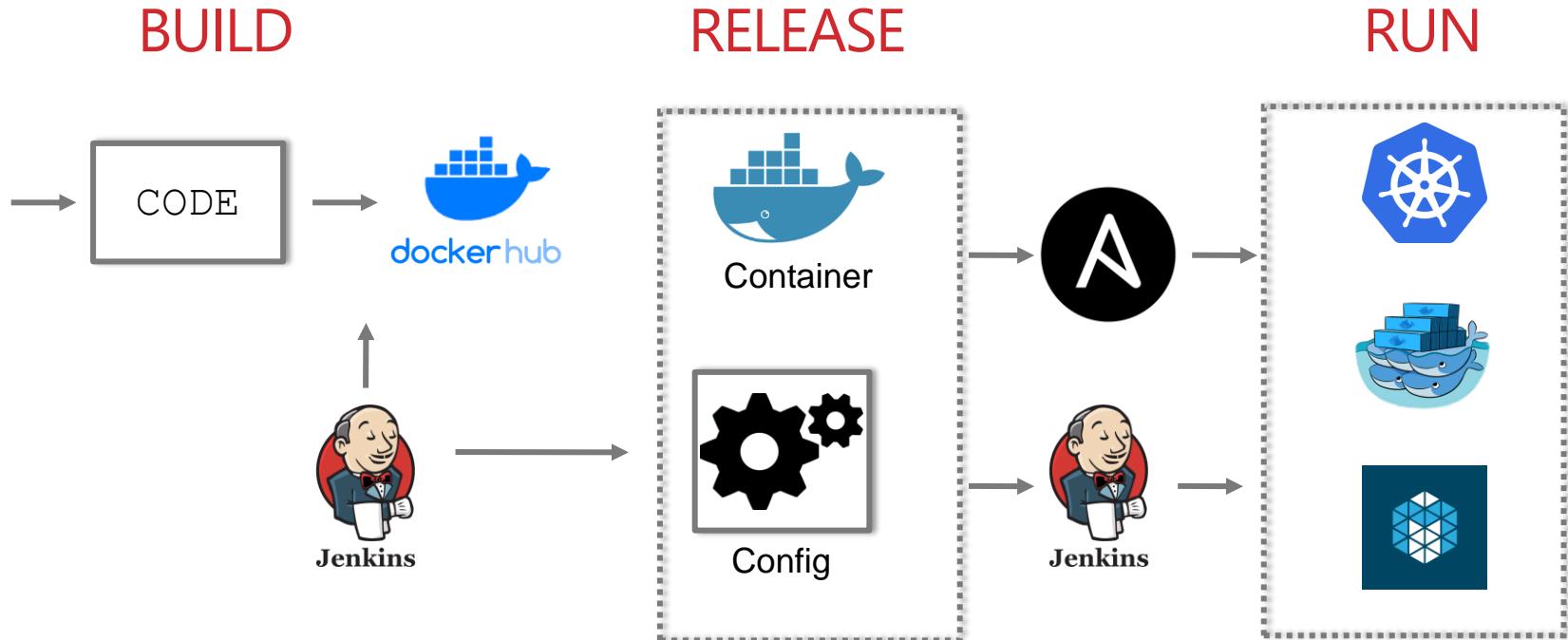


Deployments

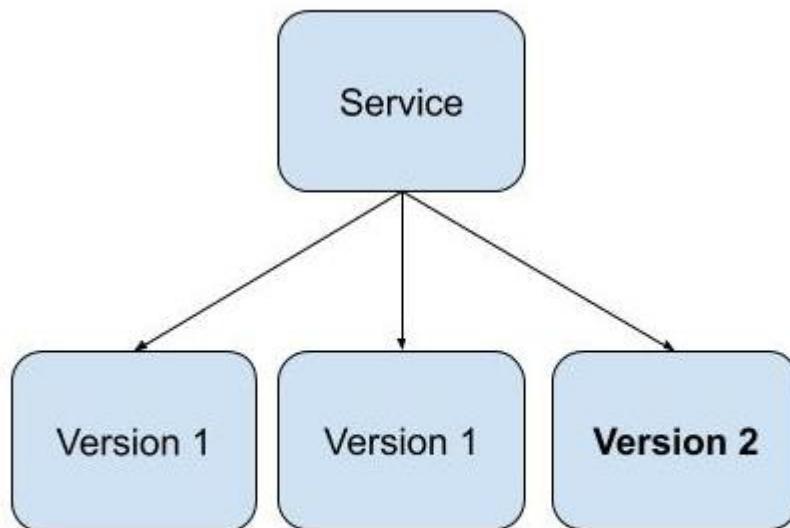


DEVELOPMENT, DEPLOYMENT, OPS

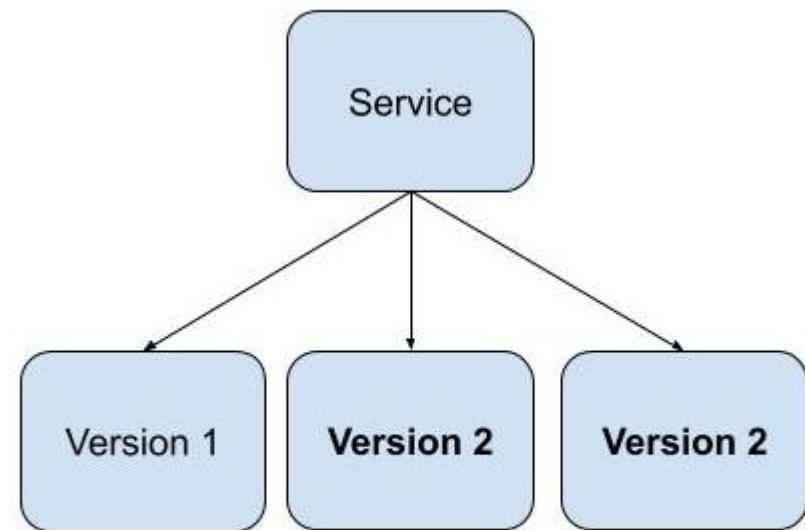
STRICTLY SEPARATE BUILD AND RUN STAGES



Canary



Stage 1: 33% of traffic
handled by canary



Stage 2: 66% of traffic
handled by canary

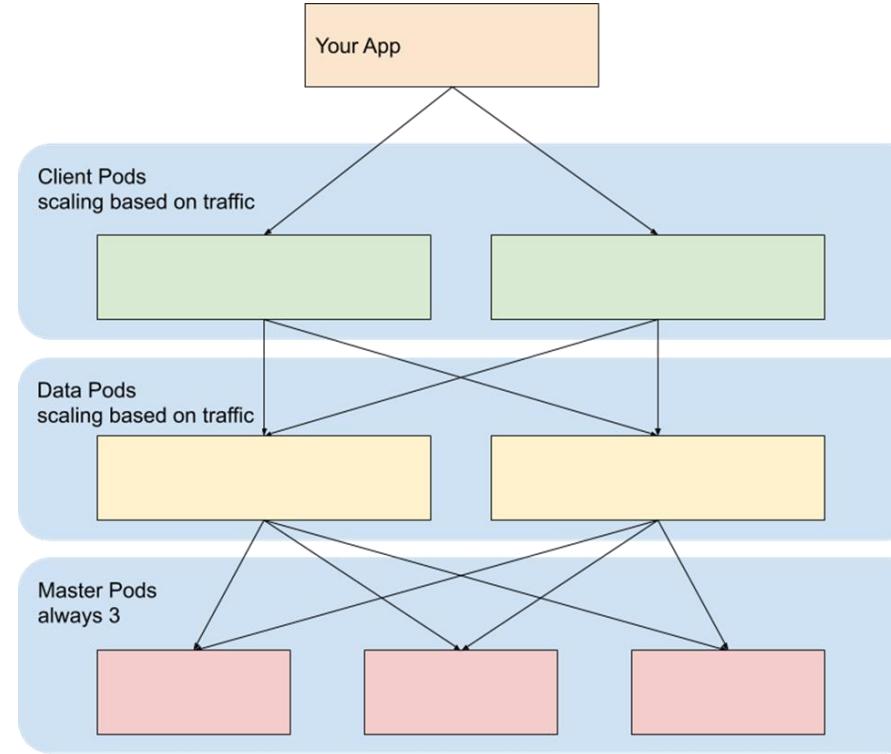
Blue/Green



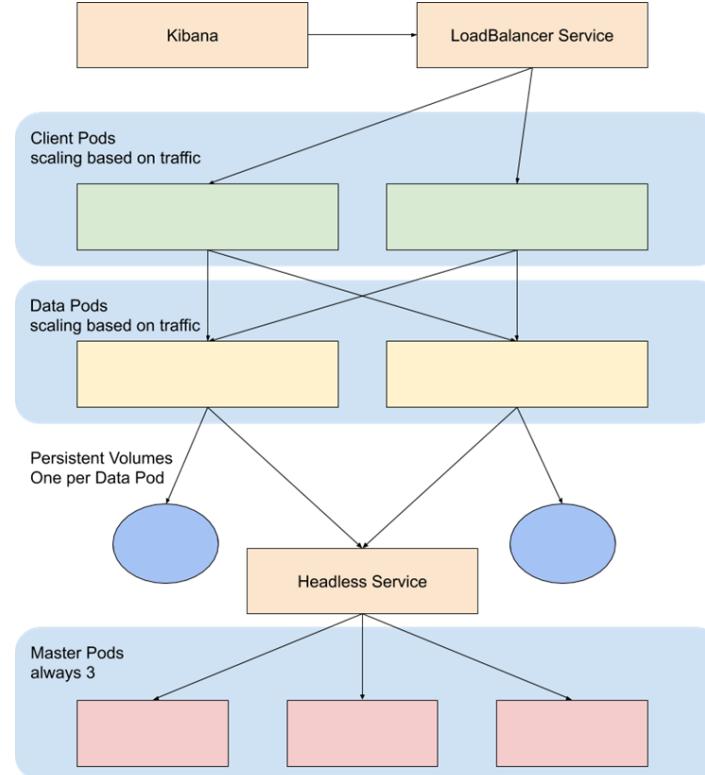
Stateful Applications

- StatefulSets require a persistent volume for storing state.
- Ensure the same PersistentVolumeClaim stays bound to the same Pod throughout its lifetime.
- Deployments:
 - Ensures the group of Pods within the Deployment stay bound to a PersistentVolumeClaim.
- Headless Service:
 - No load balancing, single dedicated IP
- Ordinal scaling
 - Pod with a unique naming convention. e.g. If you create a StatefulSet with name Elastic, it will create a pod with name elastic-0, and for multiple replicas of a StatefulSet, their names will increment like Elastic-0, Elastic-1, Elasic-2, etc

Statefulsets: Elasticsearch

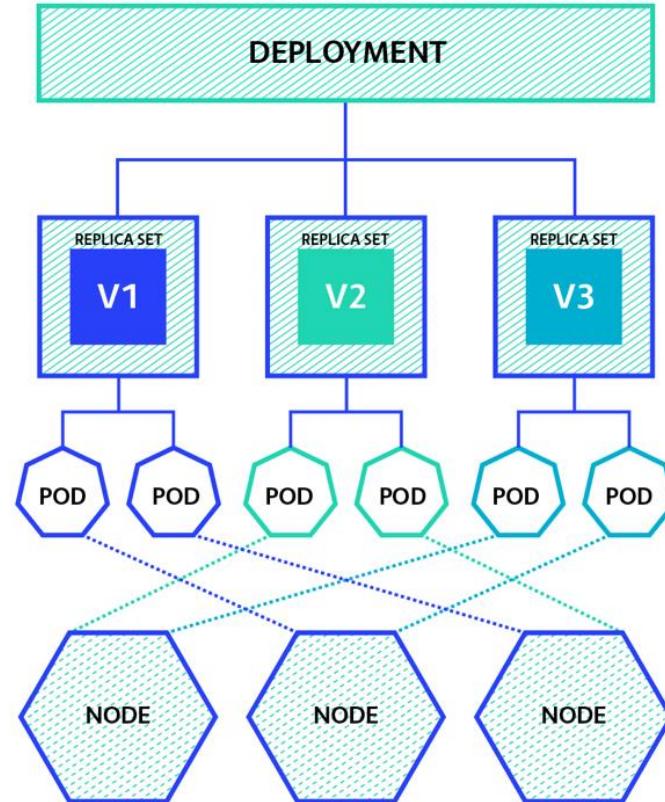


Statefulsets: Elasticsearch



What is a Deployment in K8S?

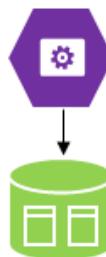
*Kubernetes controller optimal
for stateless applications*



What if my Application isn't Stateless?

Stateless Services

Business microservice A



Stateless service



SQL DB or NoSQL DB

Stateless microservice with separate store

Stateful Services

Business microservice B



Gateway service

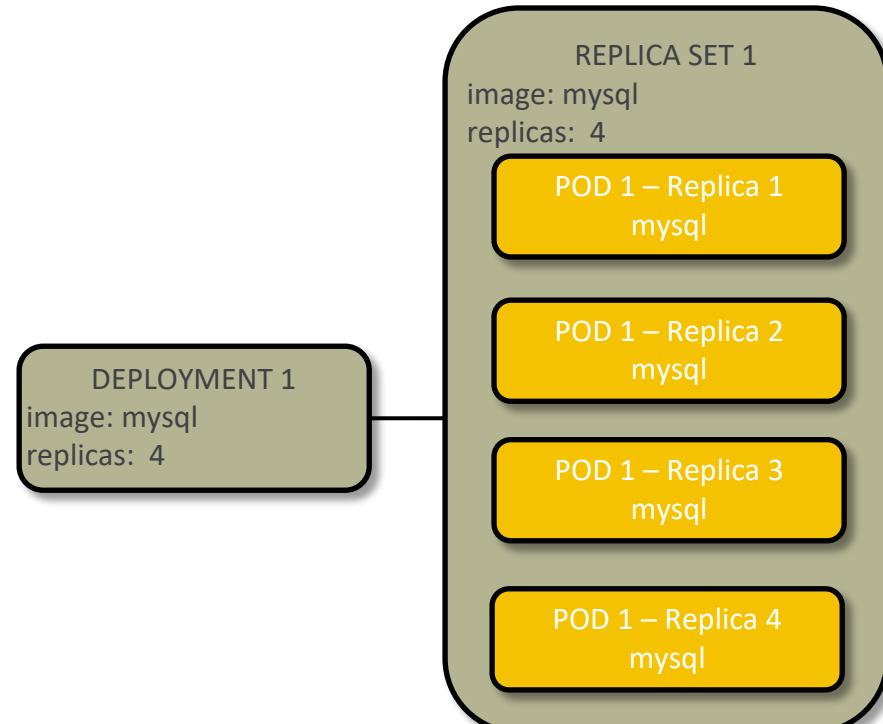
Stateful service partitions

Stateful microservice with in-memory data.
Low latency between business logic and data

Deployments Control ReplicaSet Controllers

Definition of how many replicated Pods should exist

- **Deployment creates and manages a Replica Set that manages a set of pods**
- **Replica count can be adjusted as needed to scale the Replica Set out and back**
- **Replica Set successor to the ReplicationController object**



What is replication for?

Replication Controller is one of the key features of Kubernetes, which is responsible for managing the pod lifecycle. It is responsible for making sure that the specified number of pod replicas are running at any point of time. It is used in time when one wants to make sure that the specified number of pod or at least one pod is running. It has the capability to bring up or down the specified no of pod.

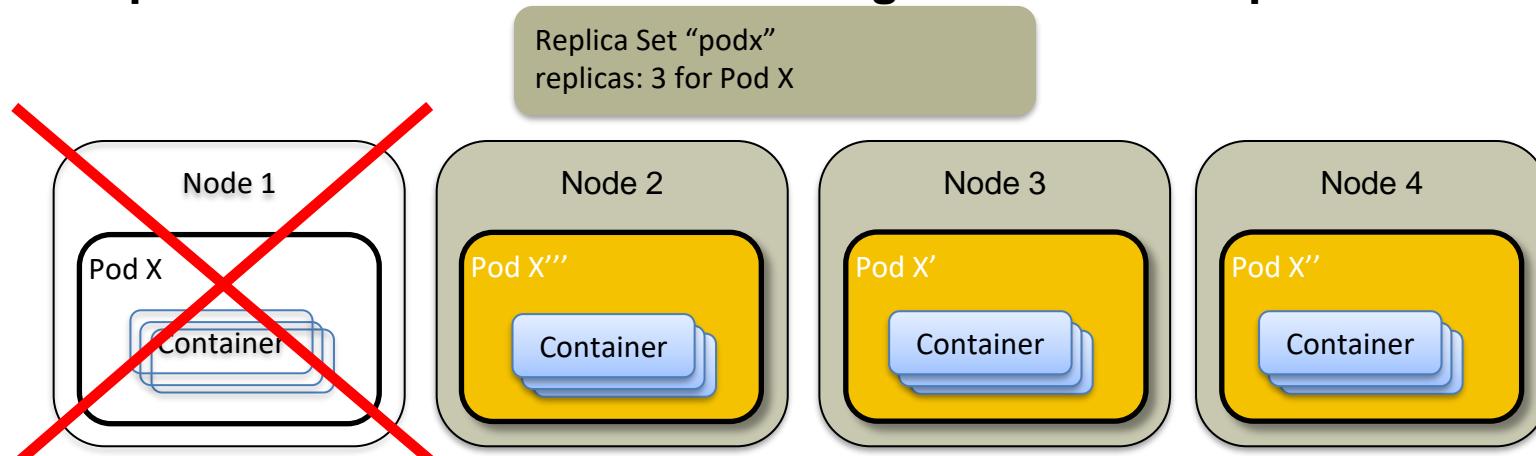
Kubernetes. Replication and self-healing



What is a Replica Set?

Provides scaling and high availability

- **Replica count can be changed to provide scaling on demand as needed**
- **If the node hosting a pod fails, the Kubernetes cluster will recreate the pod elsewhere to achieve the target number of replicas**

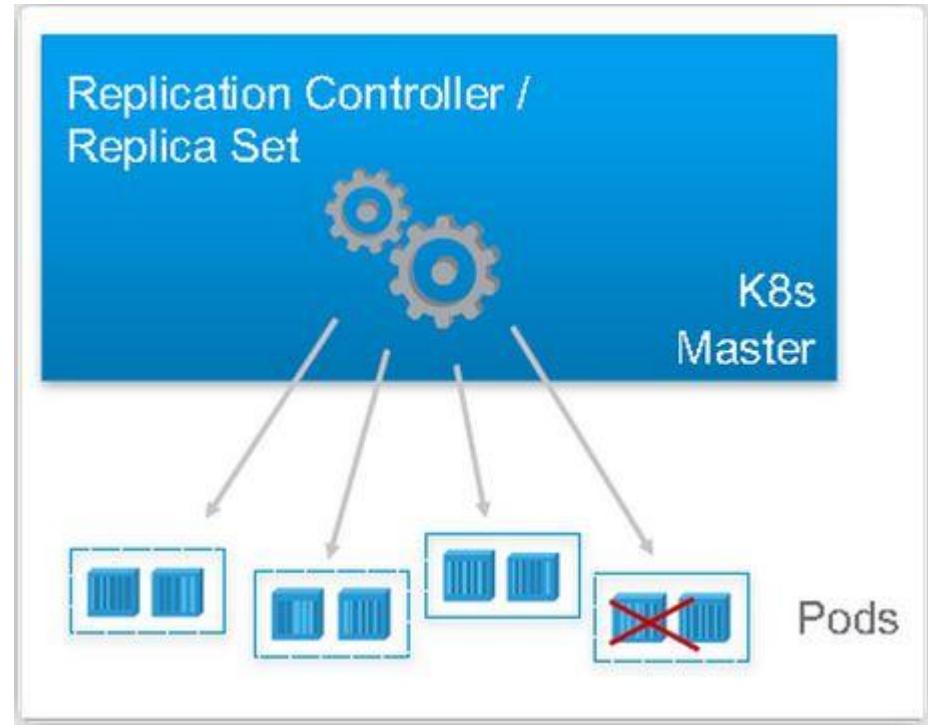


Replication Controller vs Replica Set

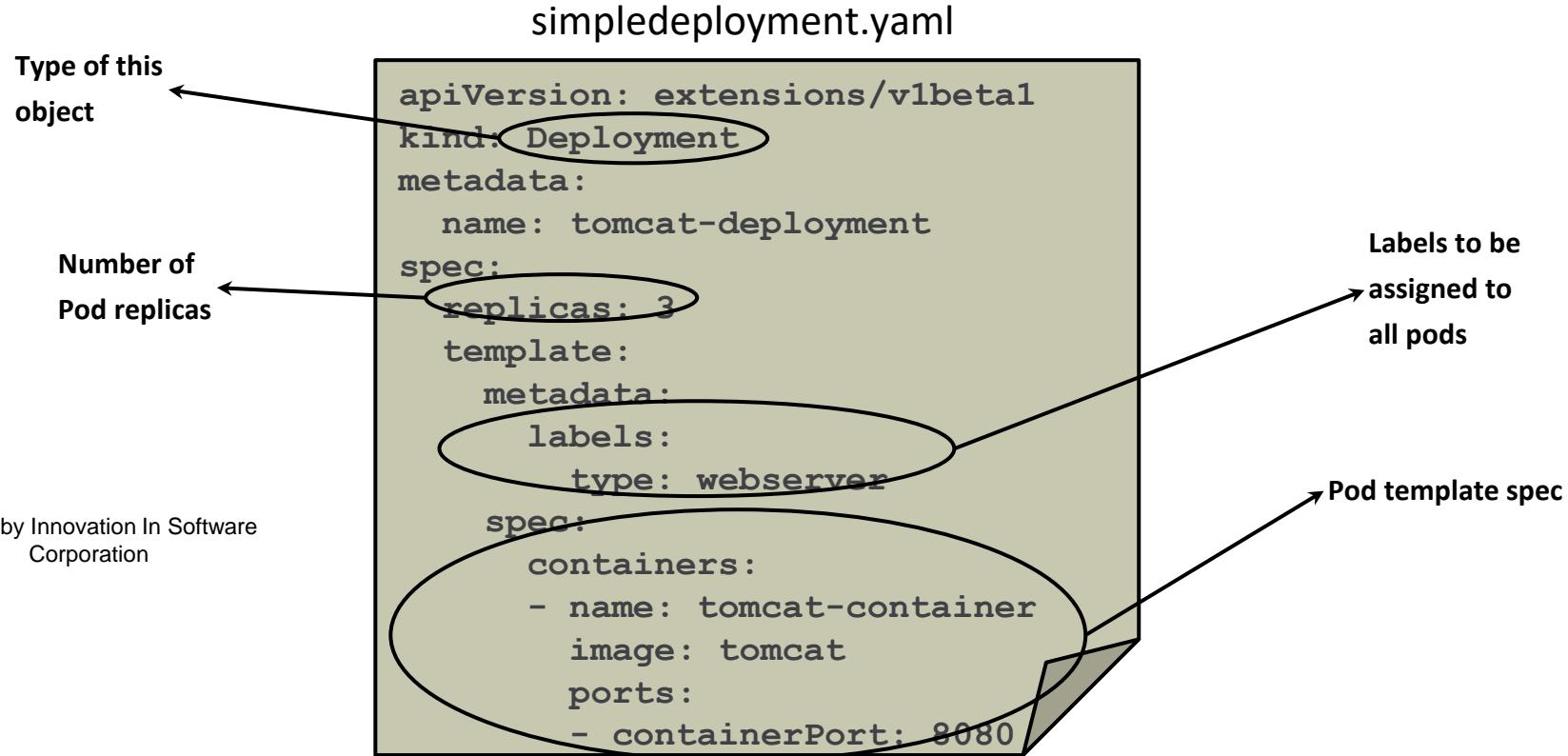
The Replication Controller is the original form of replication in Kubernetes. It's being replaced by Replica Sets, but it's still in wide use, so it's worth understanding what it is and how it works.

A Replication Controller is a structure that enables you to easily create multiple pods, then make sure that that number of pods always exists. If a pod does crash, the Replication Controller replaces it.

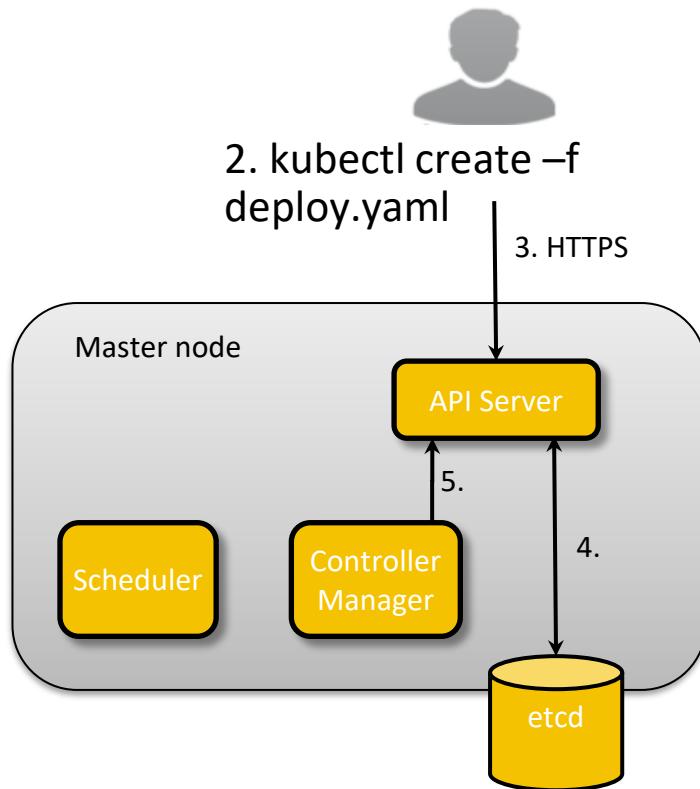
A Kubernetes controller such as the Replication Controller also provides other benefits, such as the ability to scale the number of pods, and to update or delete multiple pods with a single command



Examining a Deployment Manifest File

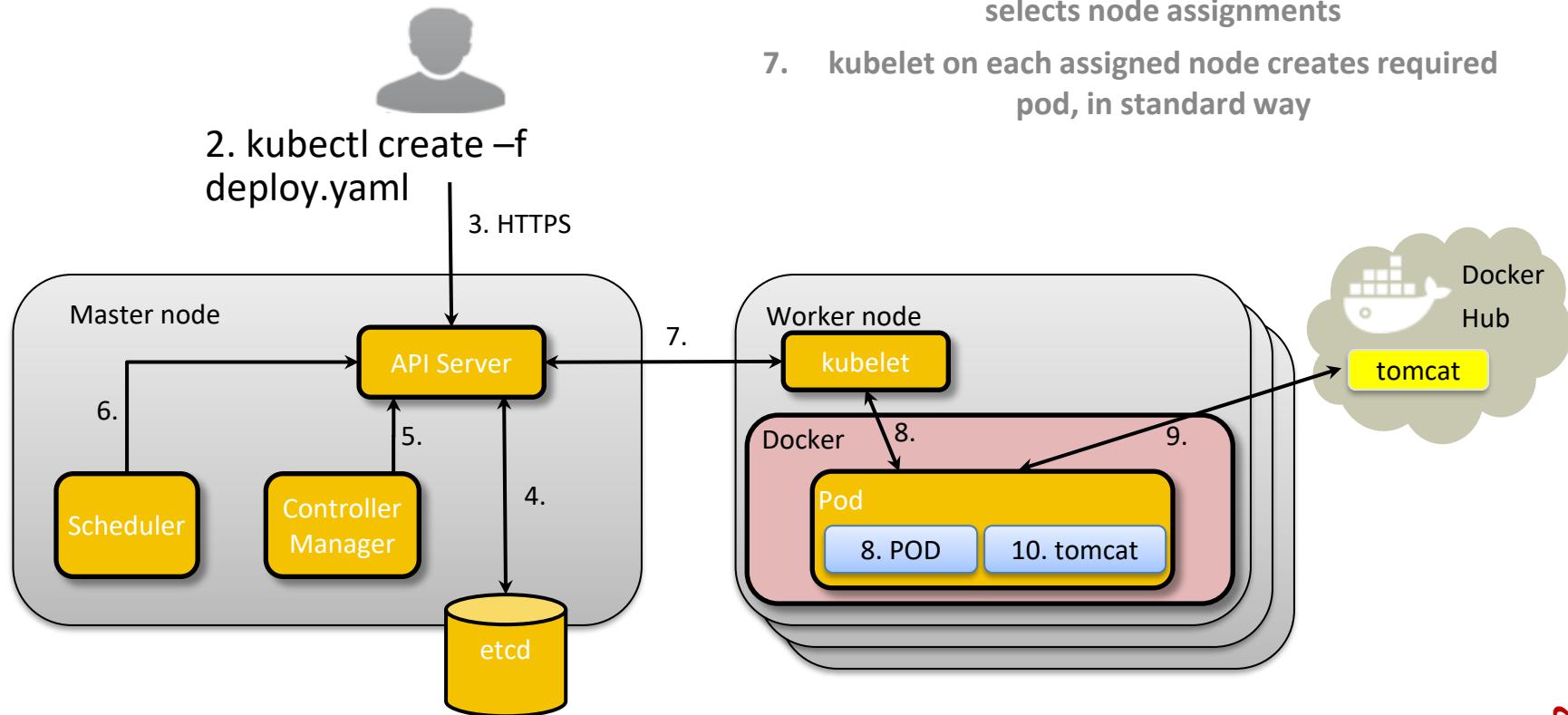


Deployment Creation Process



1. User writes a deployment manifest file
2. User requests creation of deployment from manifest via CLI
3. CLI tool marshals parameters into K8s RESTful API request (HTTP POST)
4. kube-apiserver creates new deployment object record in etcd, and new Replica Set object
5. kube-controller-manager sees new Replica Set and
 - Evaluates state of existing vs. required replicas
 - Submits pod creation requests to API to create required number of replicas

Deployment Creation Process



Deployments Control ReplicaSets

The deployment creates a ReplicaSet that controls pod creation

Deployment name defined in the yaml	Number of replicas defined in the spec	Number of existing pods	Number of Pods that match the Deployment config	Actual number of Pods running
--	---	----------------------------	---	----------------------------------

```
$ kubectl get deployment  
NAME          DESIRED  CURRENT  UP-TO-DATE  AVAILABLE  
tomcat-deployment  3        3        3          3
```

```
$ kubectl get rs  
NAME          DESIRED  CURRENT  UP-TO-DATE  AVAILABLE  AGE  
tomcat-deployment-1699985759  3        3        3          3          2m
```

Replica Set Details

Replica Set name is <Deployment name>-<pod template hash value>

```
$ kubectl describe replicaset tomcat-deployment-1699985759
Name:                  tomcat-deployment-1699985759
Namespace:             default
Image(s):              tomcat
Selector: pod +template-hash=1699985759,type=webserver
Labels:                pod-template-hash=1699985759
                      type=webserver
Replicas: 3 current / 3 desired
Pods Status: 3 Running / 0 Waiting / 0 Succeeded / 0 Failed
...
```



Selector uses labels
from pod template
and template hash

Pod Naming

Pod name is <Replica Set name>-<pod unique random string>

```
$ kubectl get pods
```

NAME	READY	STATUS
tomcat-deployment-1699985759-h11sz	0/1	ContainerCreating
tomcat-deployment-1699985759-ka13j	0/1	ContainerCreating
tomcat-deployment-1699985759-u03b5	1/1	Running

```
$ kubectl get pods
```

NAME	READY	STATUS
tomcat-deployment-1699985759-h11sz	1/1	Running
tomcat-deployment-1699985759-ka13j	1/1	Running
tomcat-deployment-1699985759-u03b5	1/1	Running

Pod name based on
Replica Set name

Random string to
differentiate Pods

Status of Pods

Modifying a Deployment to Trigger a Rollout

Multiple ways to change a Deployment configuration

- **Change the manifest file and apply it via `kubectl apply`**
- **Change specific Deployment attribute via `kubectl set`**
- **Edit the Deployment config in the cluster via `kubectl edit`**
- Any changes to the Pod template will trigger a rollout of the Deployment to create and replicate new Pods with the new template
 - This means any changes – even things like pod labels!

Pausing a Deployment

Pause a Deployment to temporarily halt rollout of updated pods

```
$ kubectl set image deployment/tomcat-deployment tomcat-container=tomcat:8.5.5;
kubectl rollout pause deployment/tomcat-deployment
deployment "tomcat-deployment" image updated
deployment "nginx-deployment" paused

$ kubectl rollout resume deployment/tomcat
deployment "tomcat-deployment" resumed

kubectl rollout status deployment/tomcat-deployment
deployment "tomcat-deployment" successfully rolled out
```

Checking Deployment Rollout History

Kubernetes tracks revisions made to a Deployment

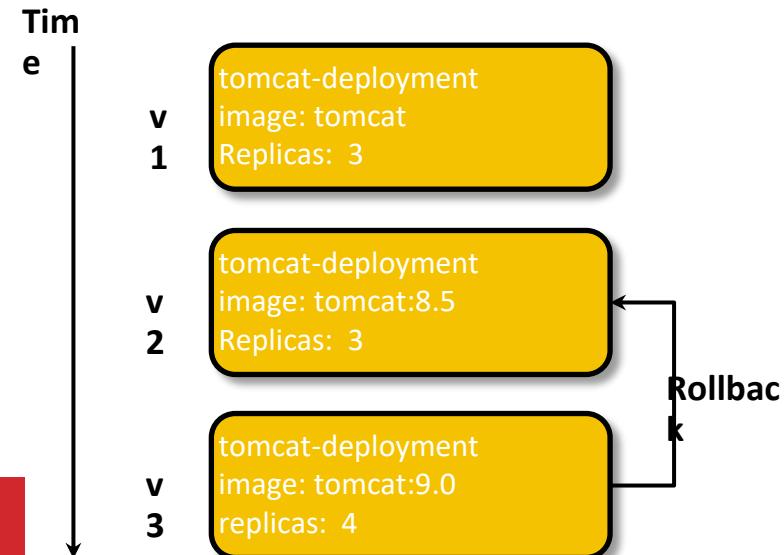
- Users can query history of a deployment and see how many versions existed, and what change was made

- Need to use `--record` flag on kubectl to record details of change commands

© 2022 by Innovation In Software Corporation

▪ History allows you to roll back

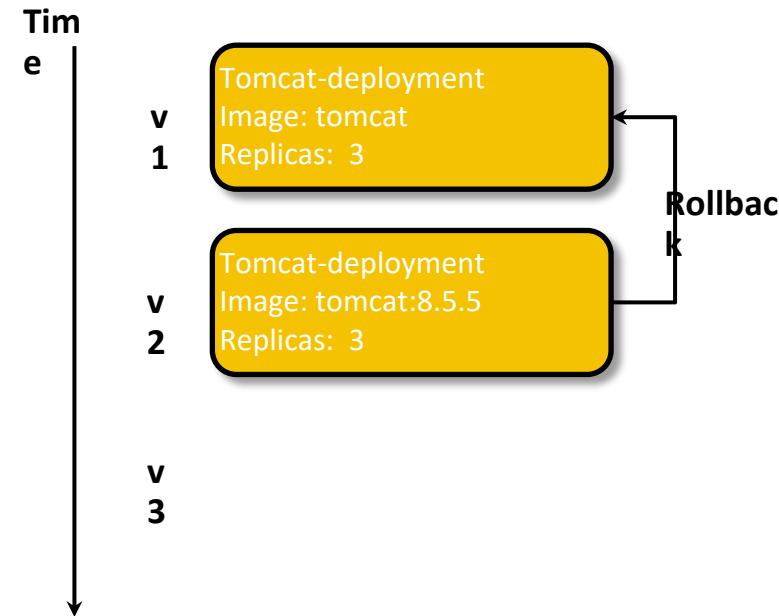
```
$ kubectl rollout history  
deploy/<deployment>  
previous version
```



Rolling back a Deployment

Undoing Deployment changes via rollback operation

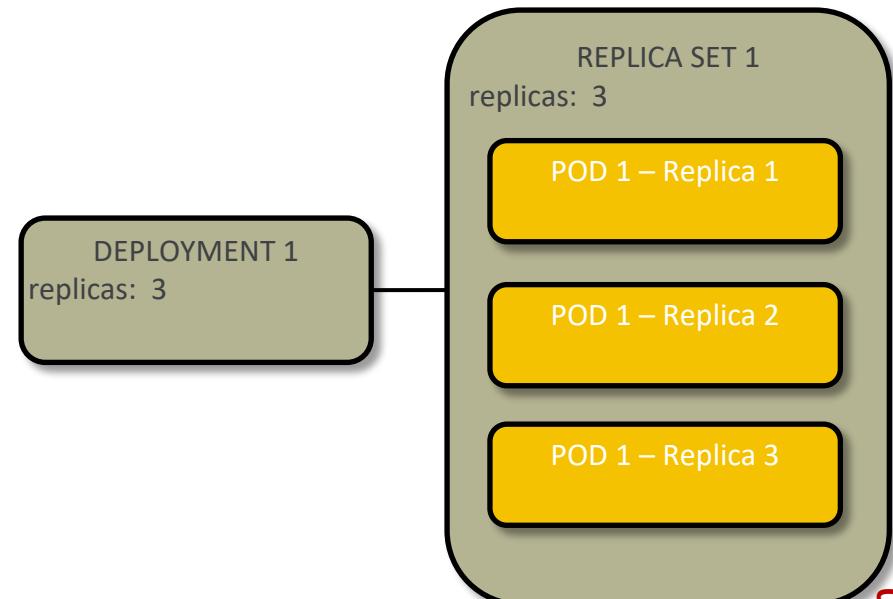
- You can undo the last change or roll back to a specific version
- The revision order will be changed to reflect the change
 - In this example, revision v1 will become the new revision v3
 - Version numbers increase monotonically
- Revision state stored in the corresponding Replica Sets



Deleting a Deployment

Deleting a Deployment will delete its Replica Set and all its Pods

- **By default, when Deployment is deleted, its Replica Sets are deleted**
- **Deletion of Replica Set cascades to deletion of pods managed by the rs**
- **Any Replica Sets reflecting previous versions of the Deployment will also be deleted**



Deployment Strategies Overview



What is a Deployment Strategy?

Approaches to manage risks on updating Deployments

- On each Deployment update/change, all pods in the deployment will be deleted and recreated
- Recreation process can have service impacts, especially for large Deployments
- A Deployment strategy defines how this rebuild process is done, to minimize downtime due to application failures or malfunctions

Types of Deployment Strategies

Kubernetes supports two basic strategies, but users can also leverage multiple Deployments when applying changes

- **Strategies for single Deployments**
 - Recreate
 - RollingUpdate
- **Strategic approaches using two Deployments with a Service**
 - Canary deployments
 - Blue/Green deployments

Each approach has a specific behavior and advantages/disadvantages.

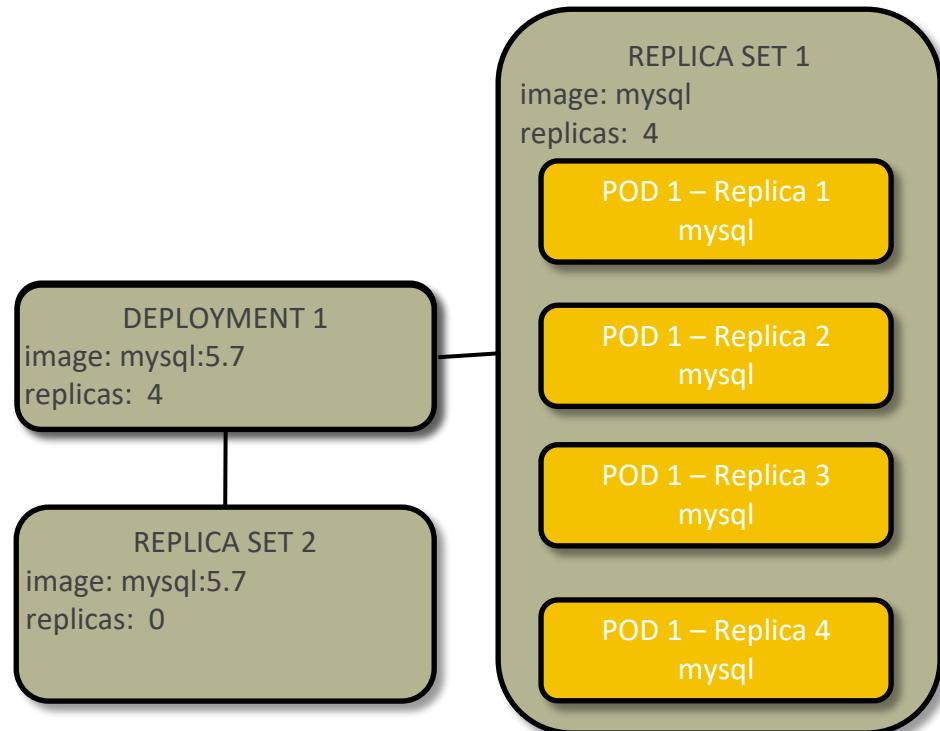
Deployment Strategy: Recreate



Deployment Strategy: Recreate

Simplest strategy for deployments

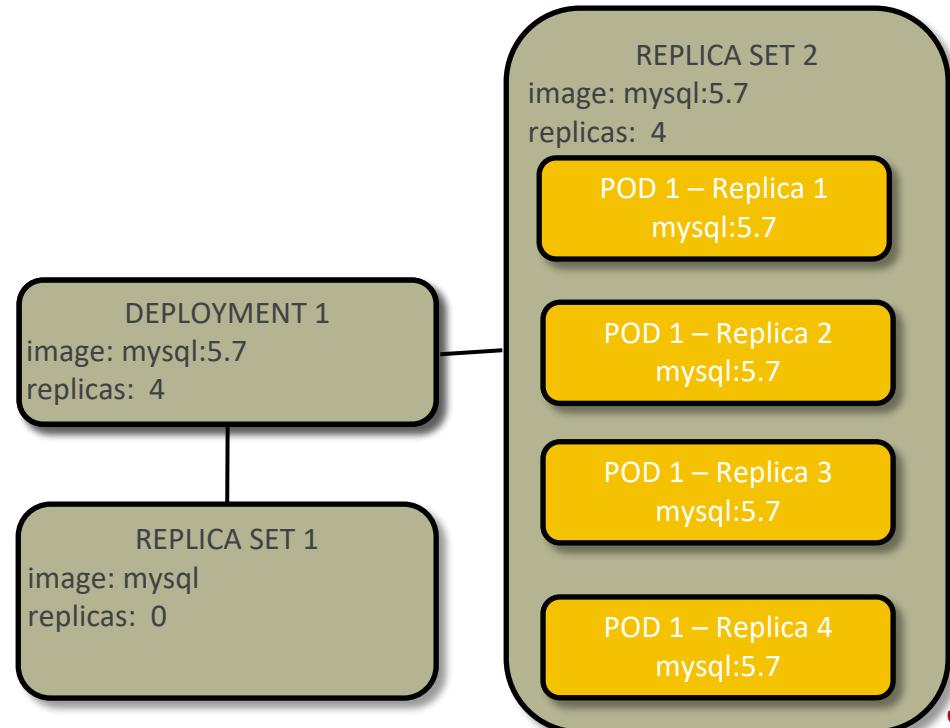
- When a change is made to a Deployment's spec, all Pods are removed and then recreated
 - Old Replica Set pods are killed
 - Then, new Replica Set starts pods
- May lead to downtime during the process while new pods are started



Deployment Strategy: Recreate

Simplest strategy for deployment updates

- When a change is made to a Deployment's template, all Pods are removed and then recreated
 - Old Replica Set pods are killed
 - New Replica Set starts pods
- May cause downtime due to delay between old pods terminating and new pods becoming available



Deployment Strategy: Recreate

Strategies are defined in the spec of a Deployment

- **strategy parameter in Deployment spec sets the strategy to be used for updates**
- **If no parameter value is set, the default is RollingUpdate**

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: tomcat-deployment
spec:
  replicas: 3
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        type: webserver
    spec:
      containers:
        - name: tomcat-container
          image: tomcat
          ports:
            - containerPort: 8080
```

Deployment Strategy: RollingUpdate



Deployment Strategy: RollingUpdate

RollingUpdate is DEFAULT strategy for Deployments

- When a change is made to the Deployment, the old Replica Set pods are scaled down as new pods are created by the new Replica Set
- A minimum number of running Pods is specified, so the Deployment will never be totally out of Pods to respond to service requests
- During the update process, the requested replica count may be temporarily exceeded

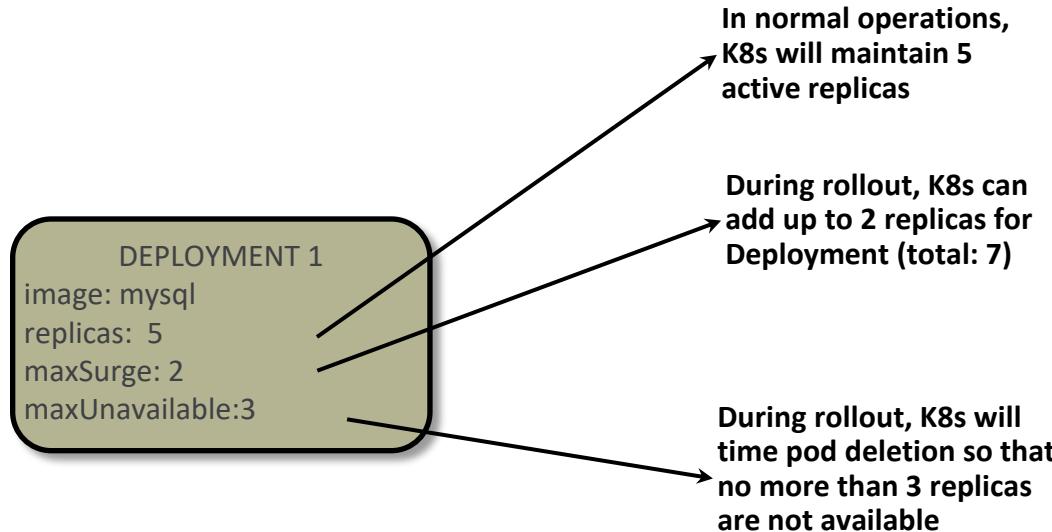
Deployment Strategy: RollingUpdate

Configure parameters to control the update process

```
...  
metadata:  
  name: tomcat-deployment  
spec:  
  replicas: 3  
  strategy:  
    type: RollingUpdate  
    rollingUpdate:  
      maxSurge: 25%  
      maxUnavailable:10%  
template:  
  metadata:  
    labels:  
      type: webserver  
...  
...
```

- **maxSurge:** number or percentage of additional Pods that can be created exceeding the replica count during update
 - Default value of 25%
- **maxUnavailable:** number of Pods that can be unavailable during the update
 - Default value of 25%

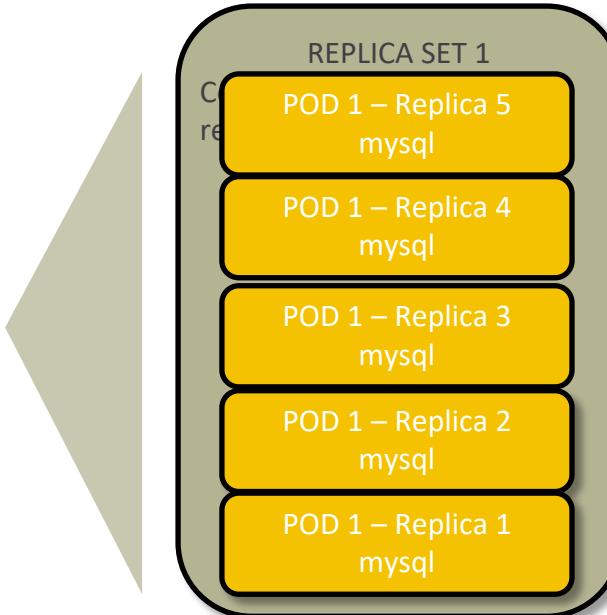
Deployment Strategy: RollingUpdate



Deployment Strategy: RollingUpdate

Initial State

DEPLOYMENT 1
image: mysql
Replicas: 5
maxSurge: 2
maxUnavailable:3



© 2022 by Innovation In Software

Corporation

```
$ vi simpledeployment.yaml
```

```
...
```

```
    image: mysql:5.7
```

```
...
```

```
$ kubectl apply -f simpledeployment.yaml
```

Deployment Strategy: RollingUpdate

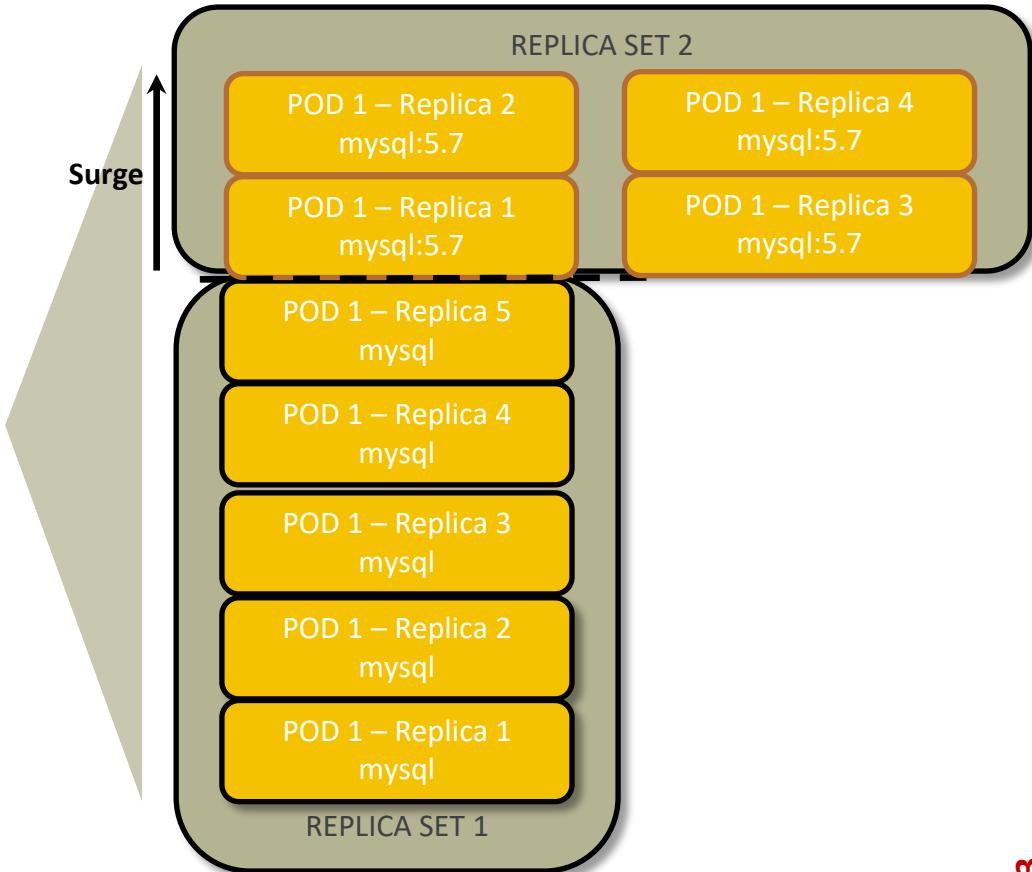
Rollout in progress

DEPLOYMENT 1
image: mysql:5.7
Replicas: 5
maxSurge: 2
maxUnavailable:3

© 2022 by Inno**data** Solutions Corporation

Initial surge of new pods on new Replica Set

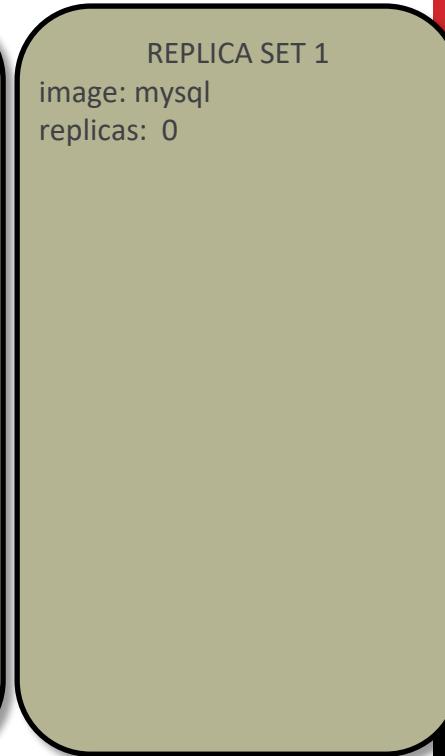
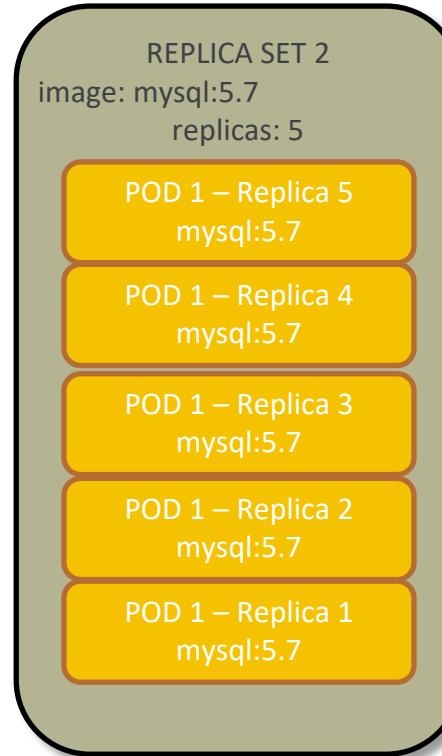
- **Original Replica Set scaled back as new RS scaled out**



Deployment Strategy: RollingUpdate

Rollout complete

DEPLOYMENT 1
image: mysql:5.7
replicas: 5
maxSurge: 2
maxUnavailable:3



© 2022 by Innovation In Software

By default, old, inactive
Replica Set saved –
previous version of the
Deployment

Updating Using Multiple Deployments



RollingUpdate using Multiple Deployments

Controlled testing of new versions in production

- Assume an application running as a Deployment, exposed as a Service
- To apply a new application version in production, a second Deployment can be used using labels in common with the first Deployment
 - Canary deployment allows for limited testing of new version in production

Strategic Approach: Canary Deployment

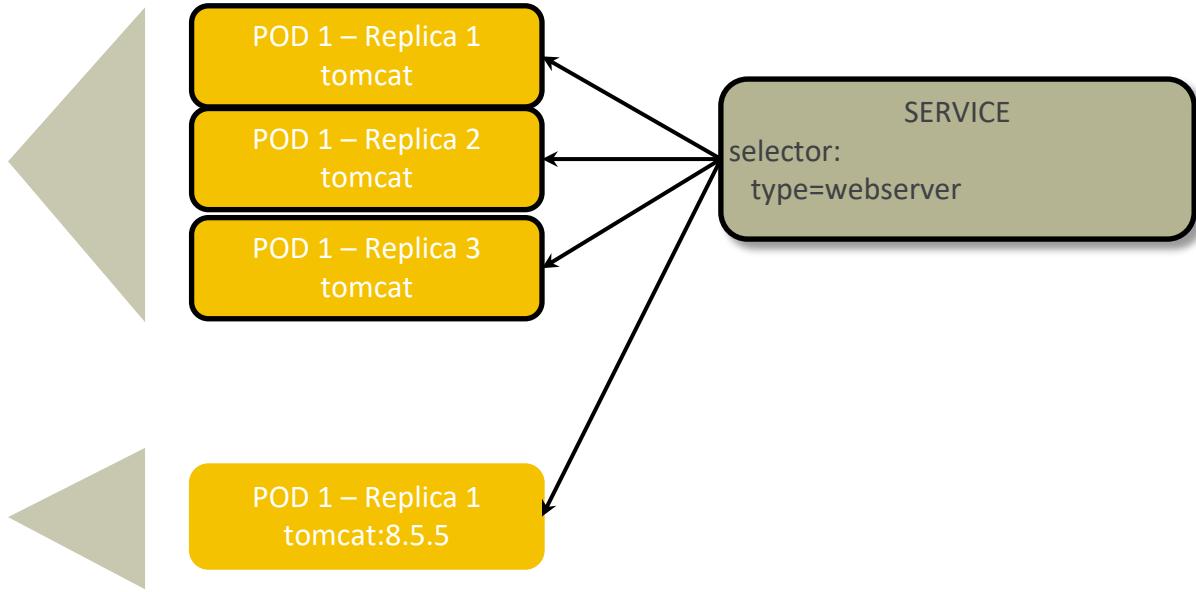
Controlled testing of the update on production

- Consider a Service selecting pods from a Deployment of application pods
- In a canary deployment, a second Deployment (Canary Deployment) is created with pods for the new version, with labels matching the Service's selector
- Service directs some requests to pods on the Canary, allowing testing of changes in production
- If a malfunction is detected, it will only impact a small portion of the Pods and can be undone.

Strategic Approach: Canary Deployment

DEPLOYMENT 1
image: tomcat
replicas: 3
type=webserver
channel=**production**

DEPLOYMENT 2
image: tomcat:8.5.5
Corporation
replicas: 1
type=webserver
channel=**canary**



Strategic Approach: Canary Deployment

Decisions after running the canary Deployment in production

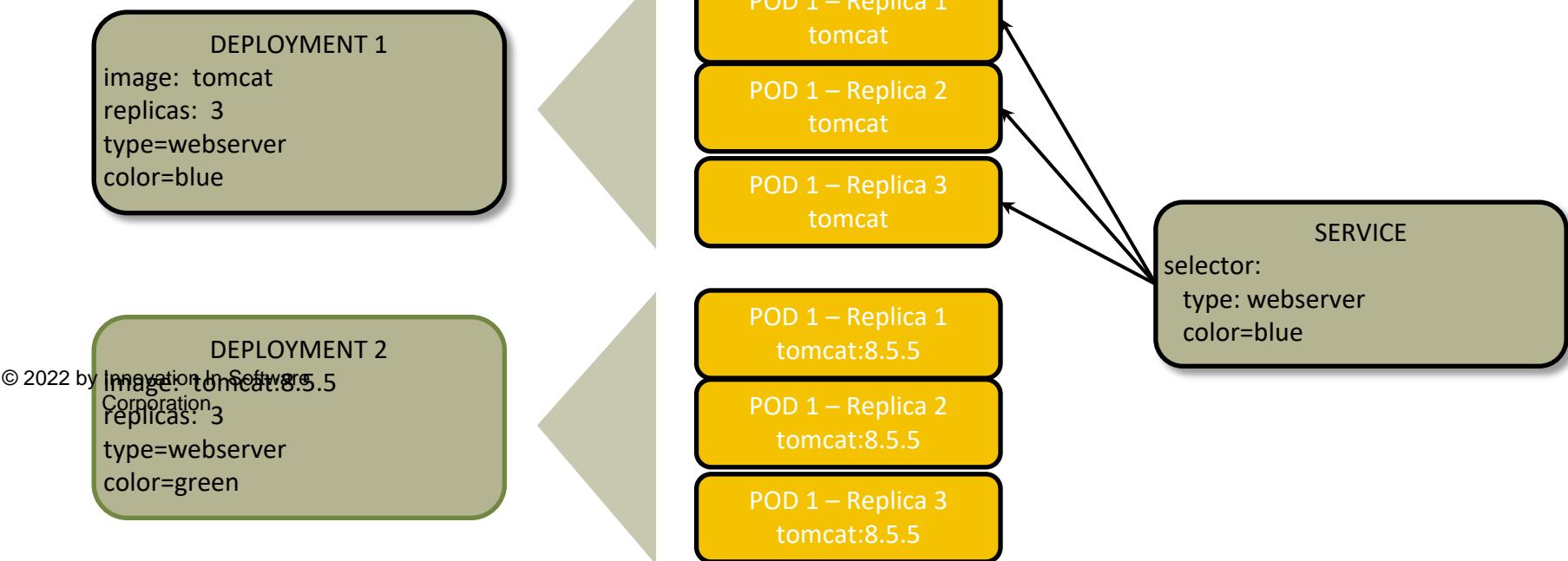
- **If the application error rate is not increasing and Canary Deployment is stable:**
 - The main Deployment can be updated to the newer version (using Rolling Update for example) and then the Canary can be discarded; OR
 - The Canary can be scaled up and reconfigured and the old Deployment can be discarded.
- **If the test results in failure, the Canary deployment can be deleted**

Strategic Approach: Blue/Green Deployment

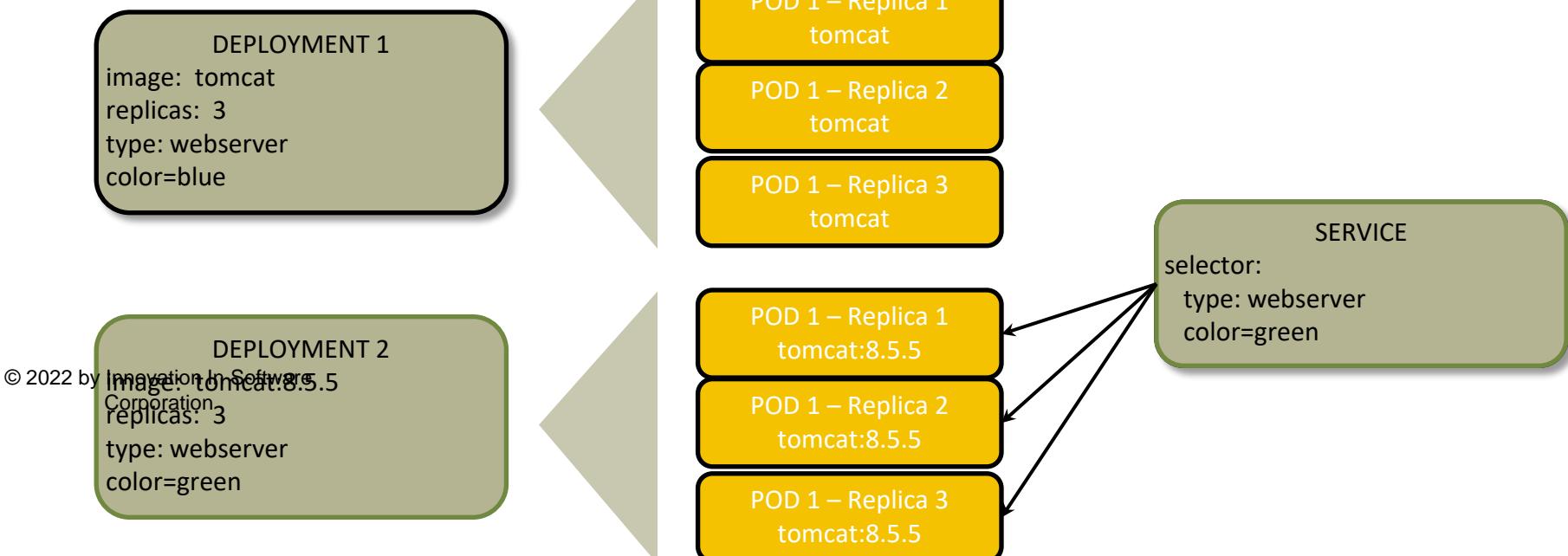
Complete environment switch from one version to another

- **With a Blue/Green deployment, you create a new full-scale Deployment in addition to the current production Deployment**
- **Reconfiguring the pod label selector on the application's Service allows choice of directing requests to old Deployment or new Deployment**
- **Similar to effect of Replace strategy without application downtime**

Strategic Approach: Blue/Green Deployment



Strategic Approach: Blue/Green Deployment



ConfigMaps & Secrets



ConfigMap

- Many applications require configuration via:
 - Config Files
 - Command-Line Arguments
 - Environment Variables
- These need to be decoupled from images to keep portable
- ConfigMap API provides mechanisms to inject containers with configuration data
- Store individual properties or entire config files/JSON blobs
- Key-Value Pairs

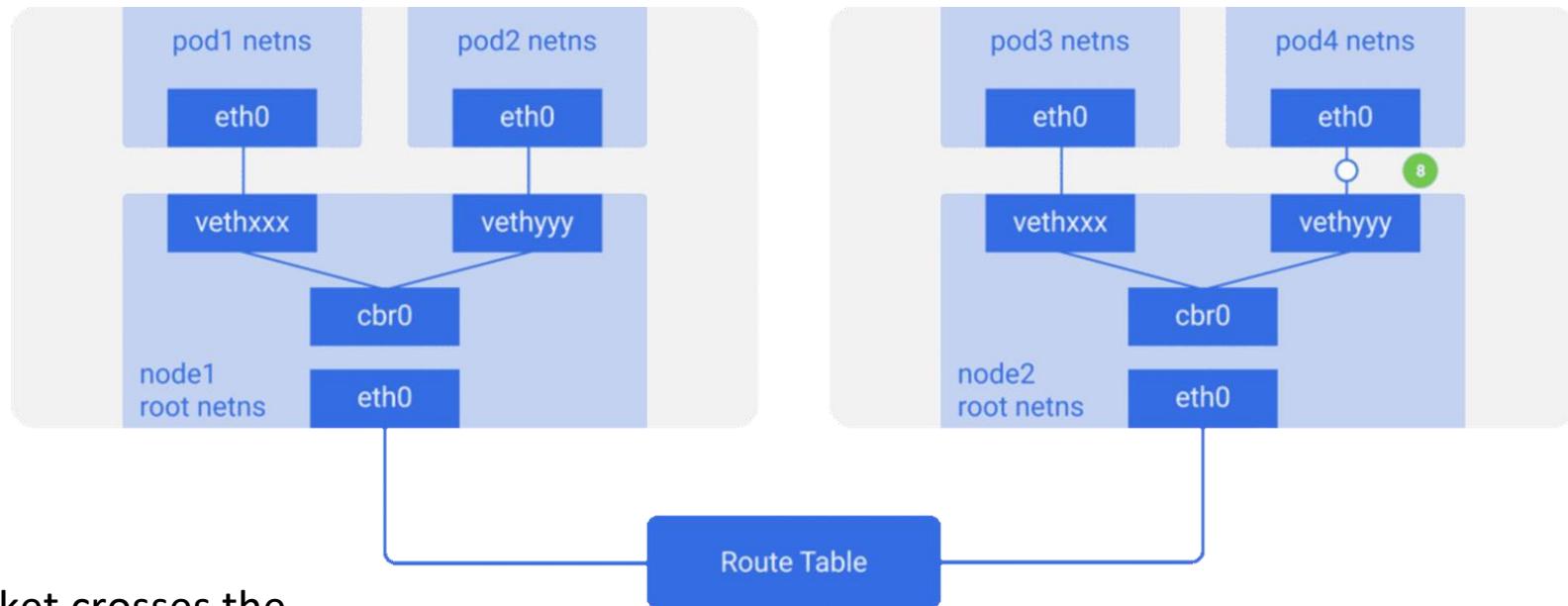
ConfigMap

- Not meant for sensitive information
- PODs or controllers can use ConfigMaps

1. Populate the value of environment variables
2. Set command-line arguments in a container
3. Populate config files in a volume

```
kind: ConfigMap
apiVersion: v1
metadata:
  creationTimestamp: 2016-02-18T19:14:38Z
  name: example-config
  namespace: default
data:
  example.property.1: hello
  example.property.2: world
  example.property.file: |-  
    property.1=value-1  
    property.2=value-2  
    property.3=value-3
```

Inter-node Communication



8. The packet crosses the pipe-pair and reaches POD4

Secrets



Application Secrets

What secrets do applications have?

- Database credentials
- API credentials & endpoints (Twitter, Facebook etc.)
- Infrastructure API credentials (Google, Azure, AWS)
- Private keys (TLS, SSH)
- Many more!

Application Secrets

It is a bad idea to include these secrets in your code.

- Accidentally push up to GitHub with your code
- Push into your file storage and forget about
- Etc.

Application Secrets

There are bots crawling GitHub searching for secrets

Real life example:

Dev put keys out on GitHub, woke up next morning with a ton of emails and missed calls from Amazon

- 140 instances running under Dev's account.
- \$2,375 worth of Bitcoin mining

Create Secret

- Designed to hold all kinds of sensitive information
- Can be used by Pods (filesystem & environment variables) and the underlying kubelet when pulling images

```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
Type: Opaque
data:
  password: mmyWfoidfluL==
  username: NyhdOKwB
```

Pod Secret

```
kind: Pod
metadata:
  name: secret-env-pod
spec:
  containers:
    - name: mycontainer
      image: redis
      env:
        - name: SECRET_USERNAME
          valueFrom:
            secretKeyRef:
              name: mysecret
              key: username
```

Volume Secret

```
spec:  
  containers  
    - name: mycontainer  
      image: redis  
      volumeMounts:  
        - name: "secrets"  
          mountPath: "/etc/my-secrets"  
          readOnly: true  
      volumes:  
        - name: "secrets"  
          secret:  
            secretName: "mysecret"
```

Resiliency

High Availability/Fault Tolerance/Disaster Recover



Basic Microservices Principles

- **Application is composed of in(ter)dependent, independently deployable components**
 - E.g. containerized components packaged as Docker images
- **Design of application is service-oriented, with information exchange between components via standard interfaces, e.g.**
 - RESTful APIs
 - AMQP messaging
- **Maximize use of stateless components that can be scaled horizontally via replication**
 - Horizontal scaling provides elastic capacity for handling demand

Application Scaling Strategies

Several ways to achieve scalability

- **Pre-defined application scale:** application deployed with a predefined, static number of resources => not the Kubernetes way
- **Manual scaling:** deployment scale reconfiguration driven by operator
- **Auto-scaling:** automatic scaling of resources based on a defined trigger (number of hits, CPU usage, etc)

Automatic Scaling of a Deployment

Creating a HorizontalPodAutoscaler resource

- **The hpa is created similarly to any other Kubernetes resource**
 - Use a manifest file for source tracking
 - Can create directly with *kubectl*
- **30 seconds is the default for considering the threshold**
- **Scaling is metrics-driven, either resource metrics from metrics-server, custom metrics API, or external metrics API**

```
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: tomcat-autoscaler
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: tomcat-deployment
  minReplicas: 2
  maxReplicas: 10
  targetCPUUtilizationPercentage: 50
```

simplescaler.yaml

Application Scaling Strategies

Manual scaling

- **K8s supports manual scaling of Deployment to adjust replica count**
- **Operator just needs to adjust declaration of how many replicas are desired, and K8s system will manage to that number**
 - Pod creation and placement completely automatic
 - Pods automatically re-created if nodes fail

```
$ kubectl scale --replicas=2 deployment/tomcat
```

```
$ kubectl scale --replicas=8 deployment/tomcat
```

```
$ kubectl scale --replicas=4 deployment/tomcat
```

```
$ kubectl edit deployment/tomcat
```

```
$ kubectl apply -f simpledeployment.yaml
```

Application Scaling Strategies

Auto-scaling in Kubernetes

- **Kubernetes has a controller object for automatic scaling of Deployments (or Replica Sets or ReplicationControllers)**
- **HorizontalPodAutoscaler is control loop to adjust scale of pod set depending on one or more metrics, evaluated at configurable interval (default 30s)**
 - Requires metrics-server to be deployed on the cluster to provide metric data
 - Metrics include CPU and RAM utilization for pods
- **Typical scenario: increase Deployment replica count when average Pod CPU utilization is above threshold value for specified period of time**
- **Note: application must support horizontal scaling!**

Application Scaling Strategies

Auto-scaling in Kubernetes

- HPA will either reduce or increase the number of pods in the set to a level that will enable each pod in the set to match the desired baseline usage as closely as possible. The baseline is calculated based on the CPU limit in the pod specifications, for example:

```
resources:  
requests:  
  cpu: 25m  
limits:  
  cpu: 100m
```

Application Scaling Strategies

Auto-scaling in Kubernetes

- HPA will either reduce or increase the number of pods in the set to a level that will enable each pod in the set to match the desired baseline usage as closely as possible. The baseline is calculated based on the CPU limit in the pod specifications, for example:

```
resources:  
requests:  
  cpu: 25m  
limits:  
  cpu: 100m
```

Automatic Scaling of a Deployment

Creating a HorizontalPodAutoscaler resource

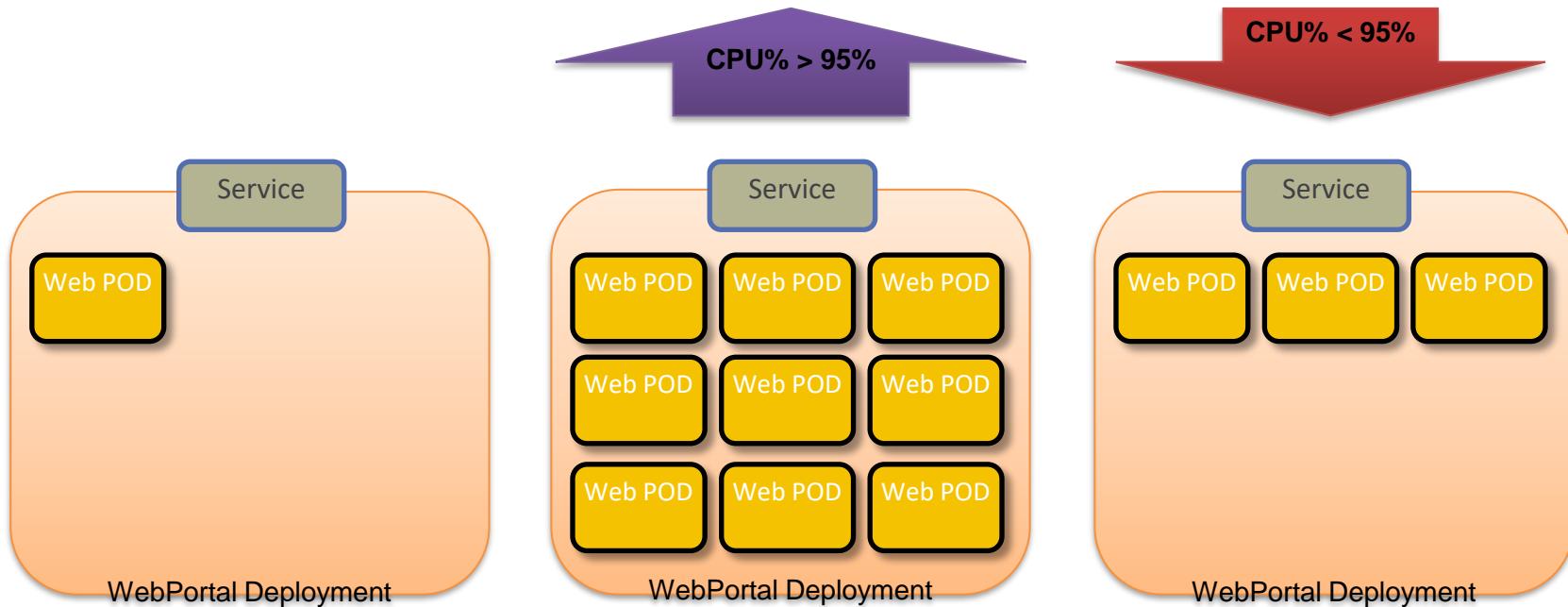
- **The hpa is created similarly to any other Kubernetes resource**
 - Use a manifest file for source tracking
 - Can create directly with *kubectl*
- **30 seconds is the default for considering the threshold**
- **Scaling is metrics-driven, either resource metrics from metrics-server, custom metrics API, or external metrics API**

```
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: tomcat-autoscaler
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: tomcat-deployment
  minReplicas: 2
  maxReplicas: 10
  targetCPUUtilizationPercentage: 50
```

simplescaler.yaml

Auto-Scaling of a Deployment

HorizontalPodAutoscaler with a 95% CPU usage threshold



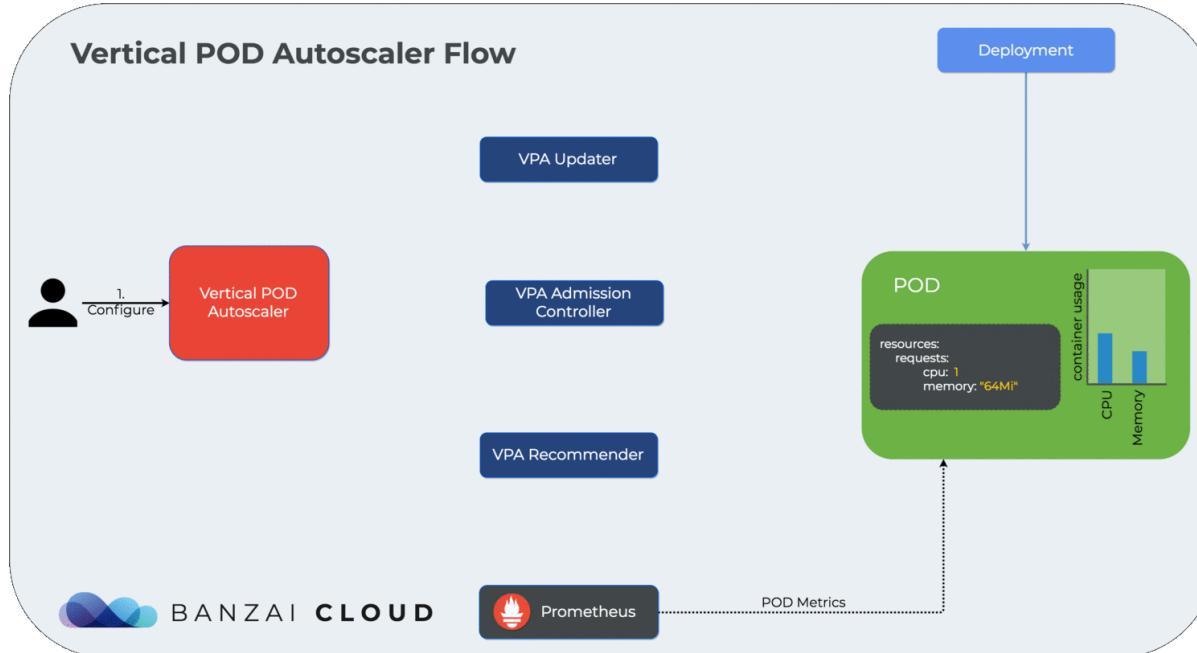
Application Scaling Strategies

Vertical auto-scaling in Kubernetes

- **VerticalPodAutoscaler** is control loop to adjust scale of pod set depending on one or more metrics, evaluated at configurable interval (default 30s)
 - Requires metric-server to be deployed on the cluster to provide metric data
 - Metrics include CPU and RAM utilization for pods
- **Typical scenario: Allocates CPU/Memory to pods based on historical utilization**
- **Note: Currently pods are rescheduled**
- **Note 2: Can not combine HPA and VPA for same deployments**

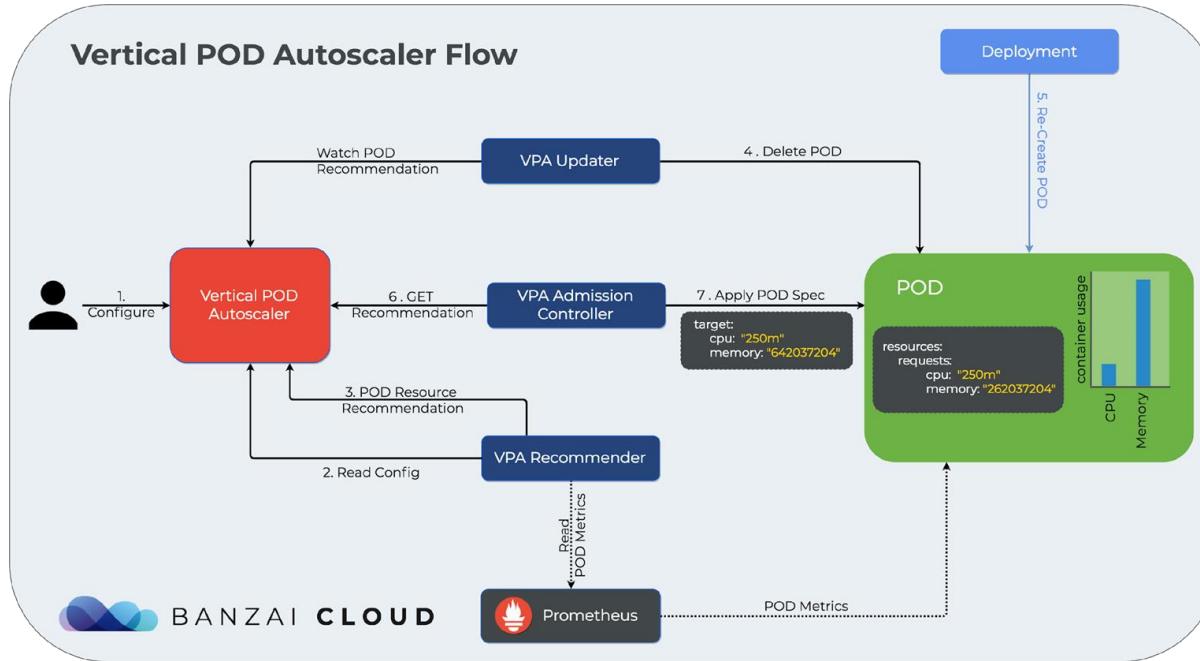
Vertical Scaling of Pods

VerticalPodAutoscaler Architecture



Vertical Scaling of Pods

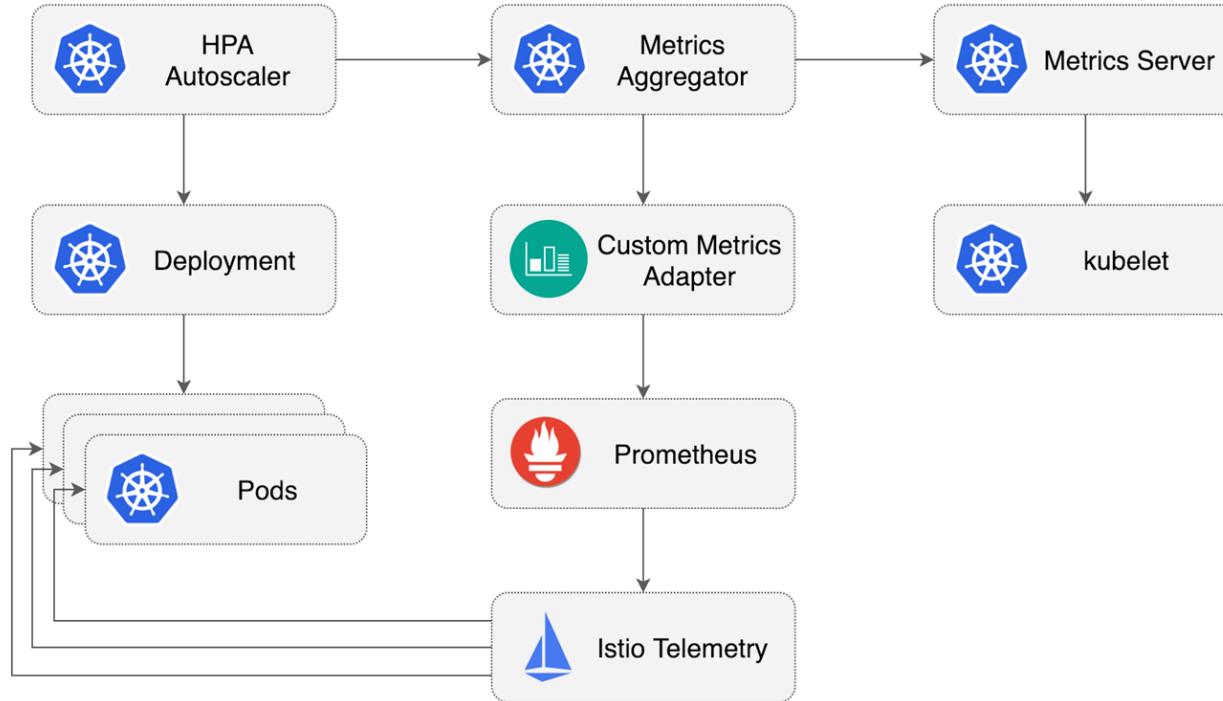
VerticalPodAutoscaler Architecture



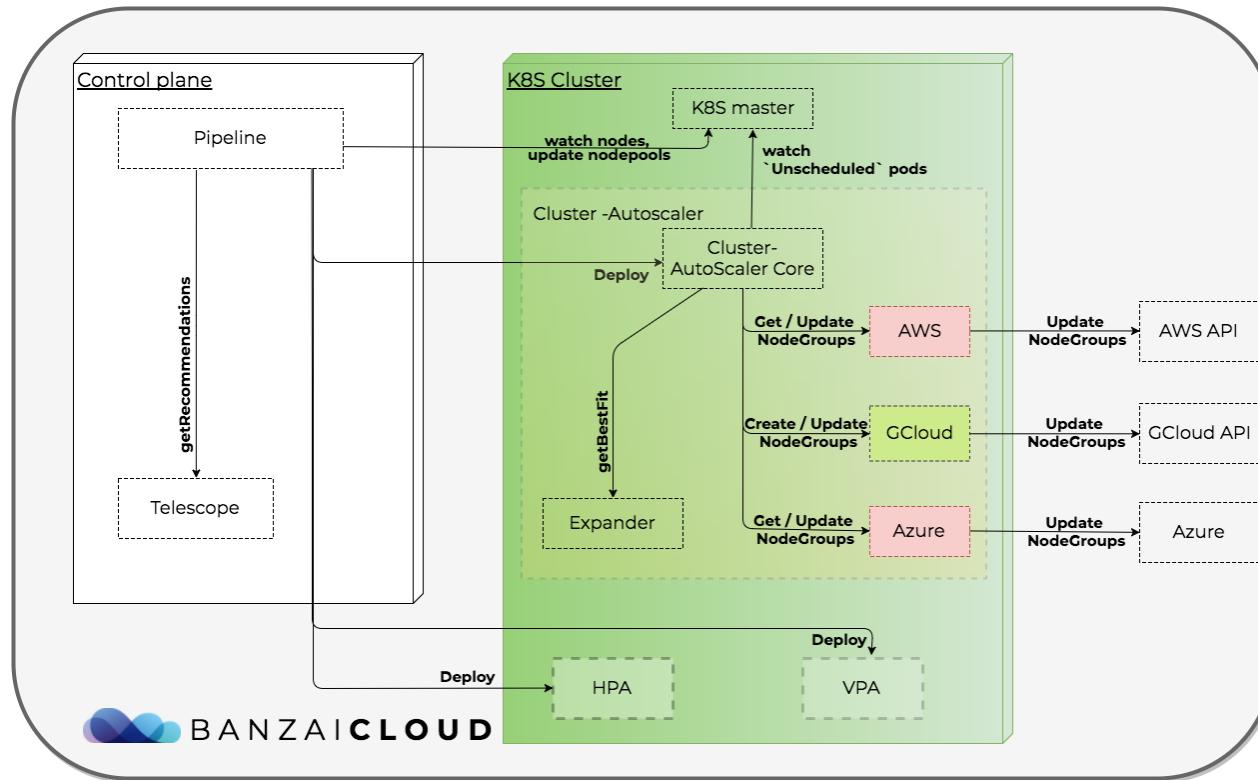
Custom & External Metric APIs

- **Scaling on CPU is not effective for most real-world applications**
- **Custom metrics are available at `custom.metrics.k8s.io`**
- **Scale based on metrics Kubernetes objects like pods or nodes, or metrics that are written by your application using a `/metrics` endpoint**
 - Requires a metrics adapter
- **External metrics are available at `external.metrics.k8s.io`**
- **Scale based on metrics from external aggregator (Prometheus, Stackdriver etc.)**

Custom & External Metric APIs



Custom & External Metric APIs



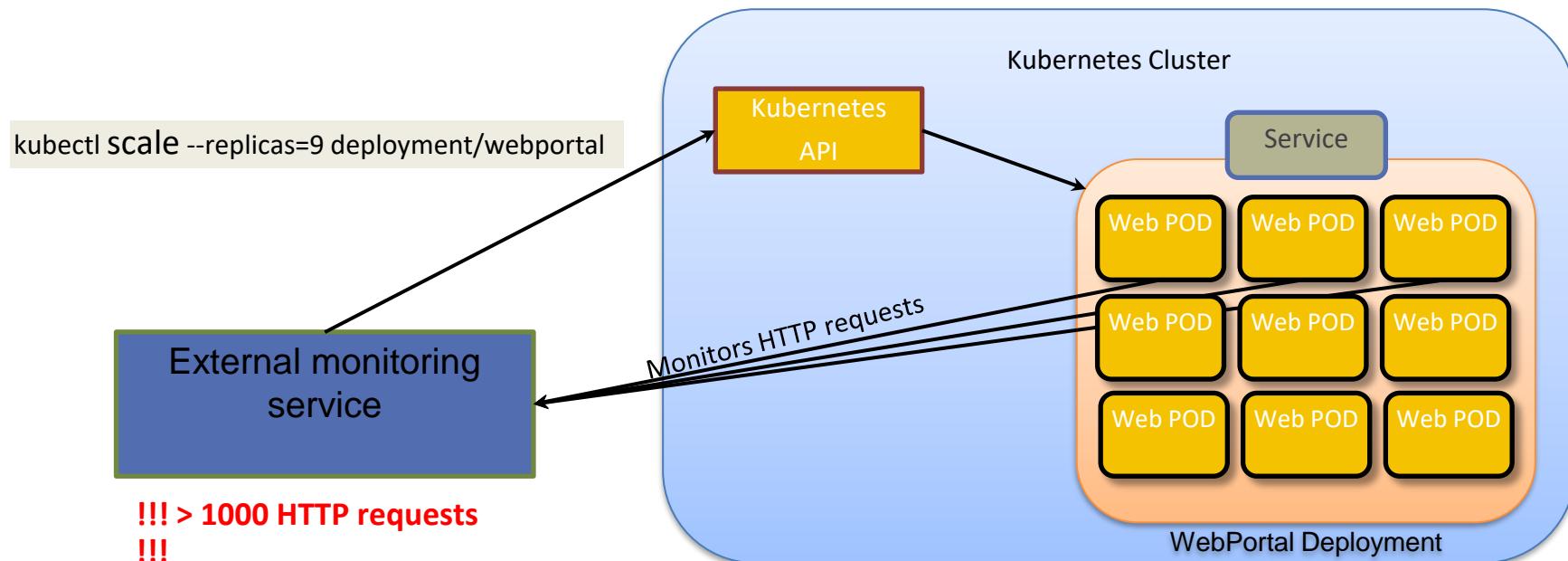
Application Scaling Strategies

Externally-driven auto-scaling

- **Kubernetes API can be used to change Deployment replica counts remotely**
- **Any monitoring application that can send REST calls (e.g. Nagios) could be used to drive automatic scaling externally**
- **Very flexible approach, but more complex to implement**
- **Needs robust testing before going into production**

Application Scaling Strategies

Auto-scaling driven by external system



Cluster Autoscaler



Cluster Scaling Strategies

Cluster node auto-scaling

The Cluster Autoscaler is a tool that automatically right sizes your Kubernetes cluster. It runs periodically and reacts to the following events:

- There are pods that failed to run in the cluster due to insufficient resources, usually these pods are in a 'Pending' state
- Some nodes in the cluster are underutilized for a configurable extended period of time, so they can be deleted, and their pods easily placed on other existing nodes.

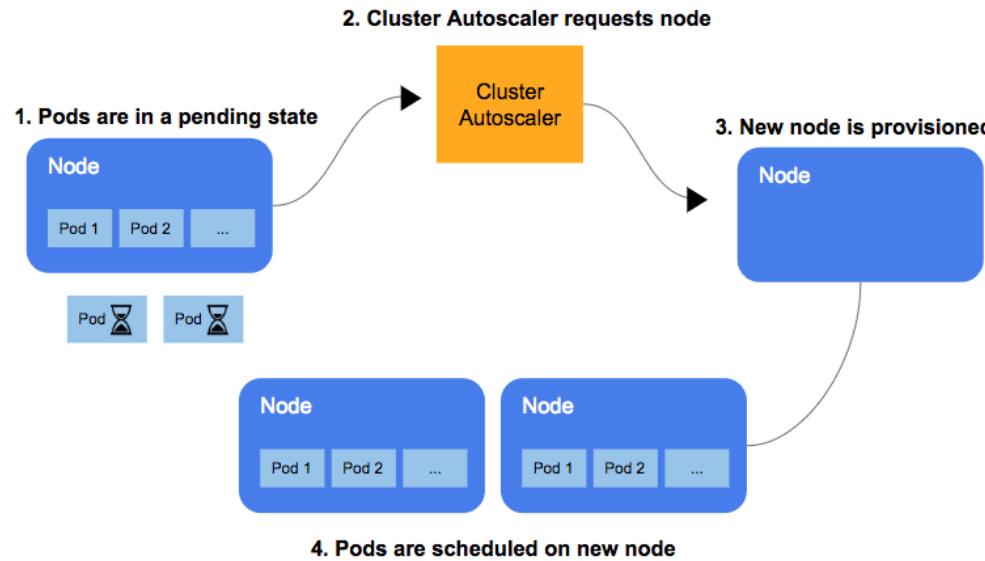
Cluster Scaling Strategies

Cluster node auto-scaling

Cluster Autoscaler is cloud agnostic. It can be used to scale Kubernetes cluster on different infrastructure.

When the Cluster Autoscaler identifies that it needs to scale up a cluster due to unschedulable pods, it increases the number of nodes in one of the available node groups. When there is more than one node group, it must decide which one to expand. Expanders provide different strategies for selecting the node group to which new nodes will be added.

Cluster Scaling: Diagram



Cluster Scaling: Add Nodes

- 1. When enabled, the cluster autoscaler algorithm checks for pending pods.**
- 2. Autoscaler requests a newly provisioned if:**
 - 1. There are pending pods due to not having enough resources.**
 - 2. Cluster or node-pool has not reached maximum node count.**
- 3. Kubernetes detects new node and adds it to cluster.**
- 4. Pending pods are allocated to new node**
- 5. If there are still pods in Pending status go back to Step 1**

Cluster Scaling: Remove Nodes

Node removal is a more complicated process. There are checks to confirm it is safe to remove nodes without causing downtime.

- If the pod is part of a daemonset, it is safe to turn down, since daemonsets run statelessly on all nodes. Removing the node should not reschedule a pod in a daemonset.
- Removing the node does not bring the running pods below the minimum replica count defined in the deployment.
- pod doesn't use any local storage on the node; since the node is going away, that local storage will be lost.

Cluster Scaling: Expander

Expander strategies:

- **random** - randomly selects a node group.
- **most-pods** - selects the node group that would be able to schedule the most pods when scaling up. This could be useful when you are using `nodeSelector` to make sure certain pods land on certain nodes.
- **least-waste** - selects the node group with the least idle CPU (or, if groups are tied, unused memory) after scale-up. This is useful when you have different classes of nodes - for example, high CPU or high memory nodes - that you only want to expand when there are pending pods that need a lot of those specific resources.

Cluster Scaling: Expander

You can also configure the cluster autoscaler to scale nodes based on cost.

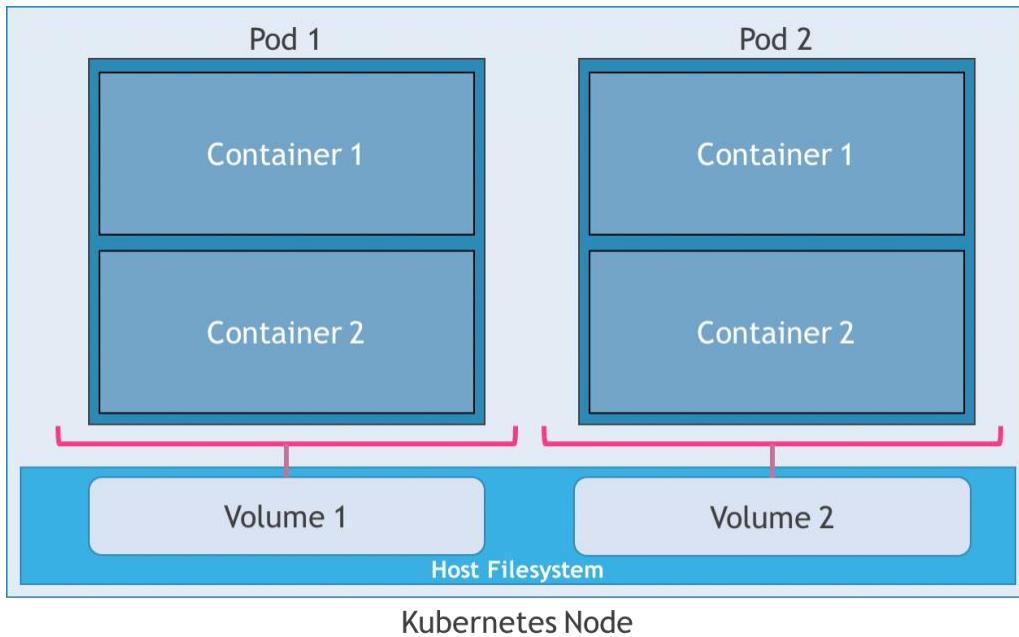
- **price** - selects the node group that costs the least and, at the same time, whose machines match the cluster size.

Cluster Scaler: Limitations

- **The cluster autoscaler doesn't consider actual CPU/GPU/Memory usage, just resource requests and limits. Most teams overprovision at the pod level, so in practice we see aggressive upscaling and conservative downscaling.**
- **Scaling up isn't immediate. Autoscaler sends request in 30-60 seconds**
 - Adjust your scaling threshold to account for provisioning time.
- **Scaling down isn't guaranteed with the autoscaler.**
 - Nodes can have non removable pods (persistent local storage, unschedulable nodes leading to minimum replica count).

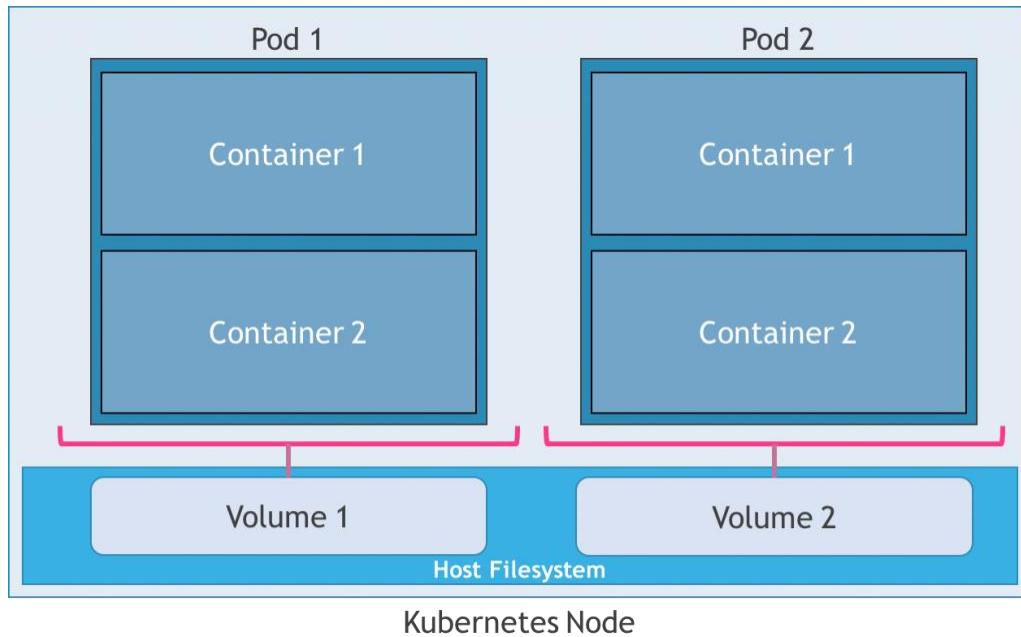
Host Based Storage

- Containers in a Pod share the same volume
- Host based storage
 - Local filesystem
- Removed when Pod is deleted
- Types of local storage
 - emptyDir
 - hostPath



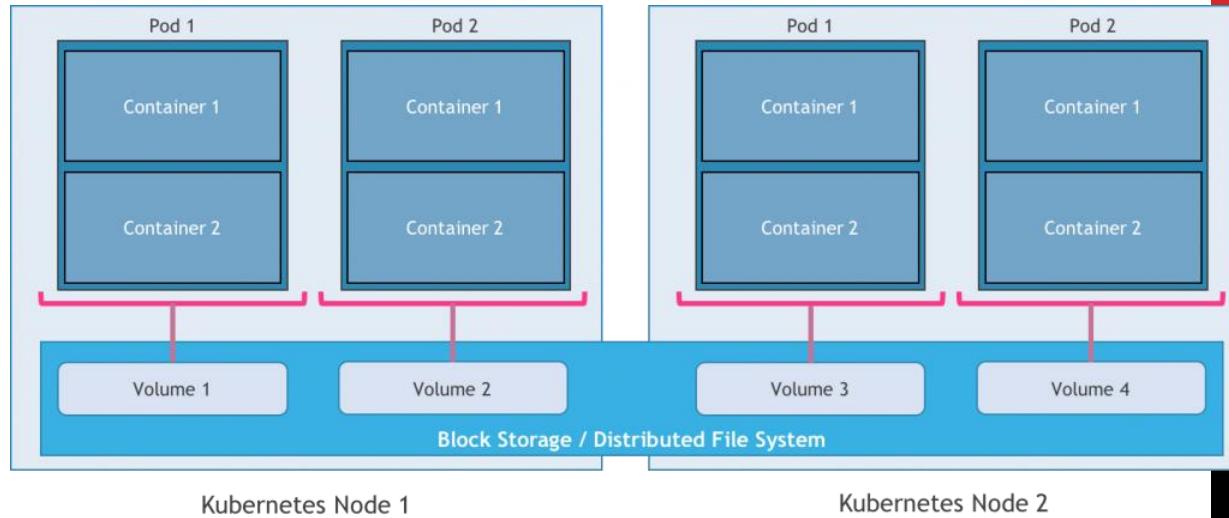
Host Based Storage

- Common use cases
 - Scratch disk
 - Store temp config data
- hostPath
 - Directories on host
 - Owned by root



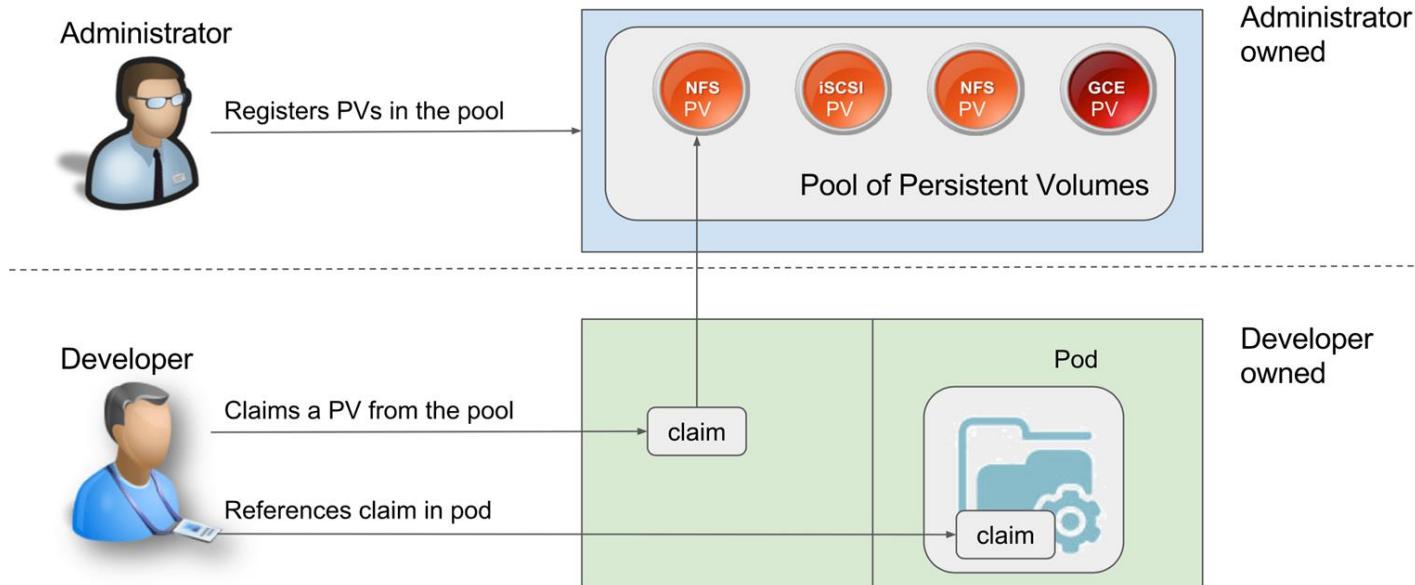
Non-Host Based Storage

- Common use cases
 - Persistent data
 - DB, Stateful apps
- EBS
- GCE Persistent Disks
- NFS, NetApp, GlusterFS



Persistent Volumes & Claims

- Operations creates PVs
 - Real storage details
 - Storage “LUN”



Key CI/CD Concepts

Jenkins



What is Continuous Delivery?

We'll use the definition from Martin Fowler, a DevOps master craftsperson:



- Continuous Delivery is a software development discipline where you build software in such a way that the software can be released to production at any time.
- Core idea of CD is creation of a repeatable, reliable and agile SDLC process from ideation to client
- Goals are sustainability, agility, and repeatability

Continuous Delivery comes from XP Agile

- Agile Development begins with Extreme Programming (XP)
- Invented/Promoted by Kent Beck and associates
- Beck describes XP as:
 - A philosophy of software development based on the values of communication, feedback, simplicity, courage, and respect
 - A body of practices proven useful in improving software development.
 - A set of complementary principles, intellectual techniques for translating the values into practice, useful when there isn't a practice handy for your particular problem.
 - A community that shares these values and many of the same practices.

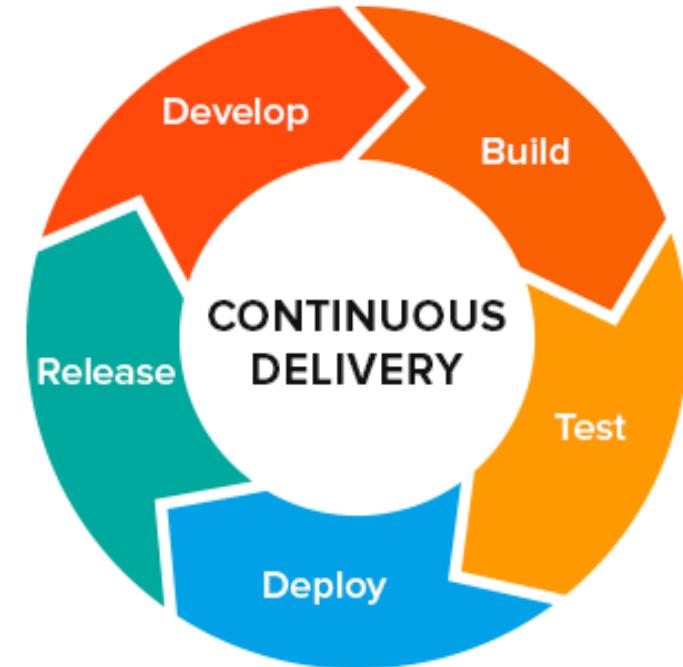
Delivery Pipelines

- Vehicle for continuous delivery
- Tasks separated by stages
- Implement various testing techniques and processes
- Support Shift-Left
- Task orchestration AKA Application Release Automation
 - Netflix Conductor
 - Datical DB
 - XebiaLabs CodePipeline, XL Deploy, & XL Release
 - CA Release Automation

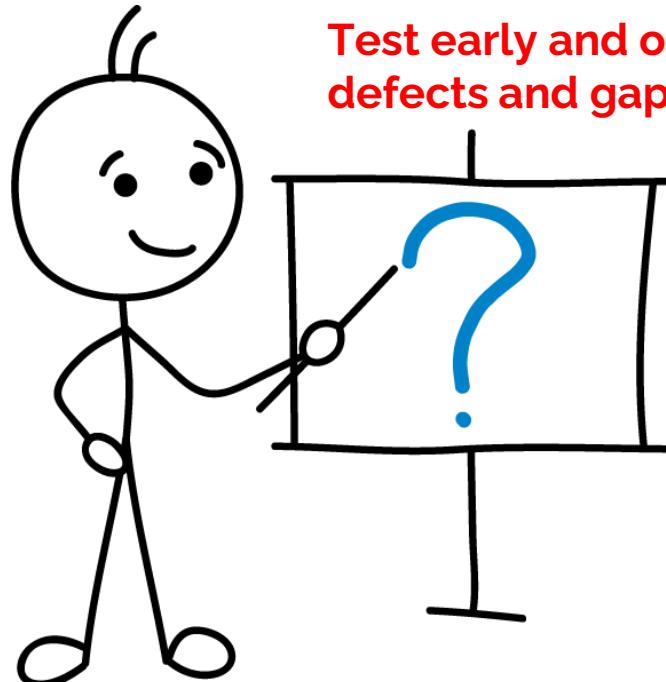


Your Organization is Doing CD if ...

- Software is deployable throughout its life-cycle
- Team prioritizes keeping the app deployable over working on new features
- Automated feedback is available to anybody on the production readiness of any system at any time when anyone makes a change
- Deployments can be kicked off by a push of a button for any version of the software to any environment on request



What is Continuous Integration?



Test early and often to reduce the late stage identification of defects and gaps in intent:

- CI is a development practice where members of a team integrate their work frequently
- Each person integrates daily – ensuring that the component is still build and deployment ready
- Each integration is verified by automation
- Shift Left

CI & CD Methodologies

- Enable automation of the build early
- Front-load creation of a Dev server environment
- Include unit testing on the Dev server
- Quickly stand up production-like environment, e.g. clustered, integration points, roles, and permissions
- Enterprise Agile

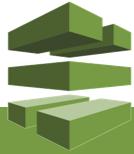


DevOps CI/CD Maturity

Dev & Ops Collaboration

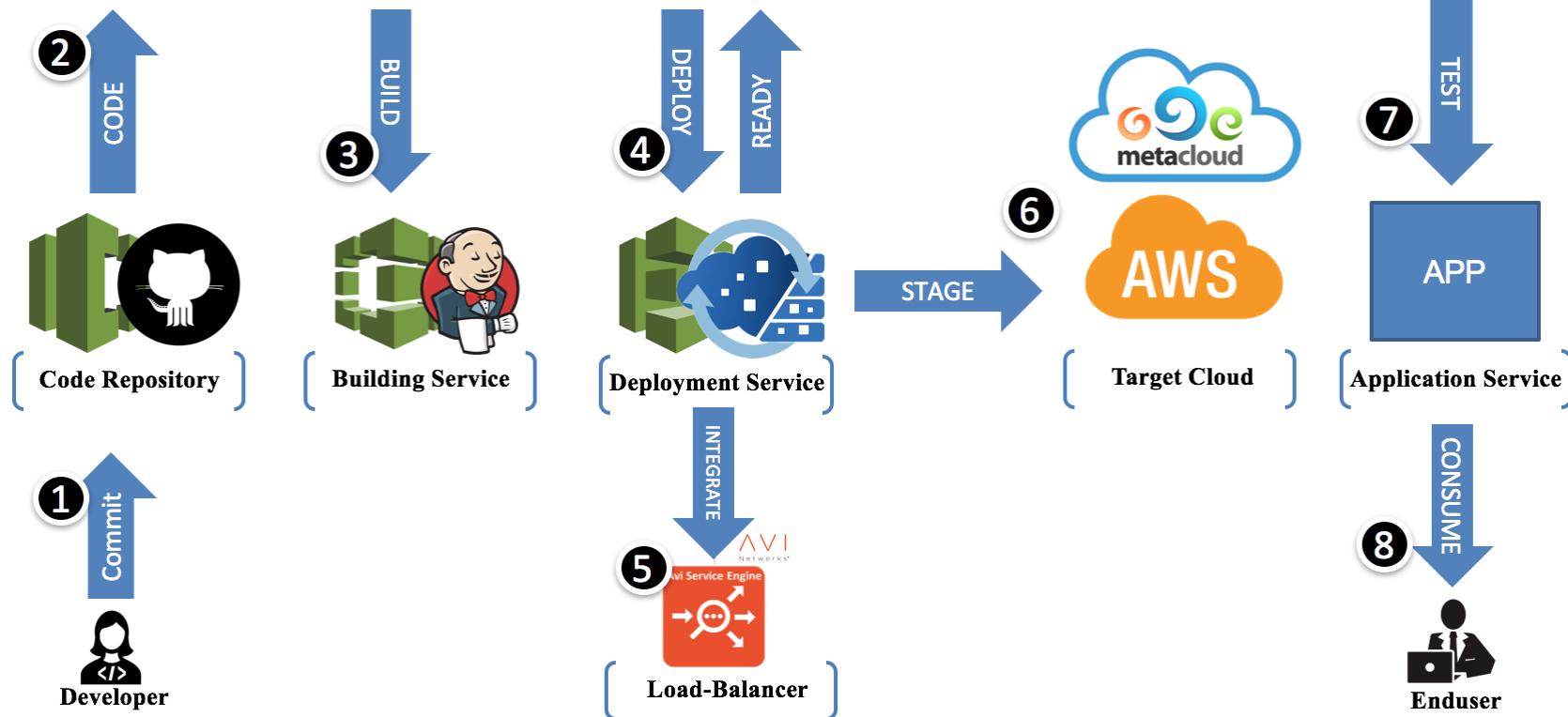


- Enable automation of the build early
- Front-load creation of a Dev server environment
- Include unit testing on the Dev server
- Quickly stand up production-like environment, e.g. clustered, integration points, roles, and permissions
- Agile methodology
- Ensure development, architecture, governance, security, operations, test/QA, and business or product team members are engaged



CI/CD BluePrint Steps

AWS Code Pipeline #CICD Service



Key CI/CD Tool Categories

- Monitoring
- Automation
- Configuration Management
- Build Management
- Version Control
- Notification
- Release automation
- Testing
- Knowledge Management
- Orchestration
- Design/Prototyping
- Workflow

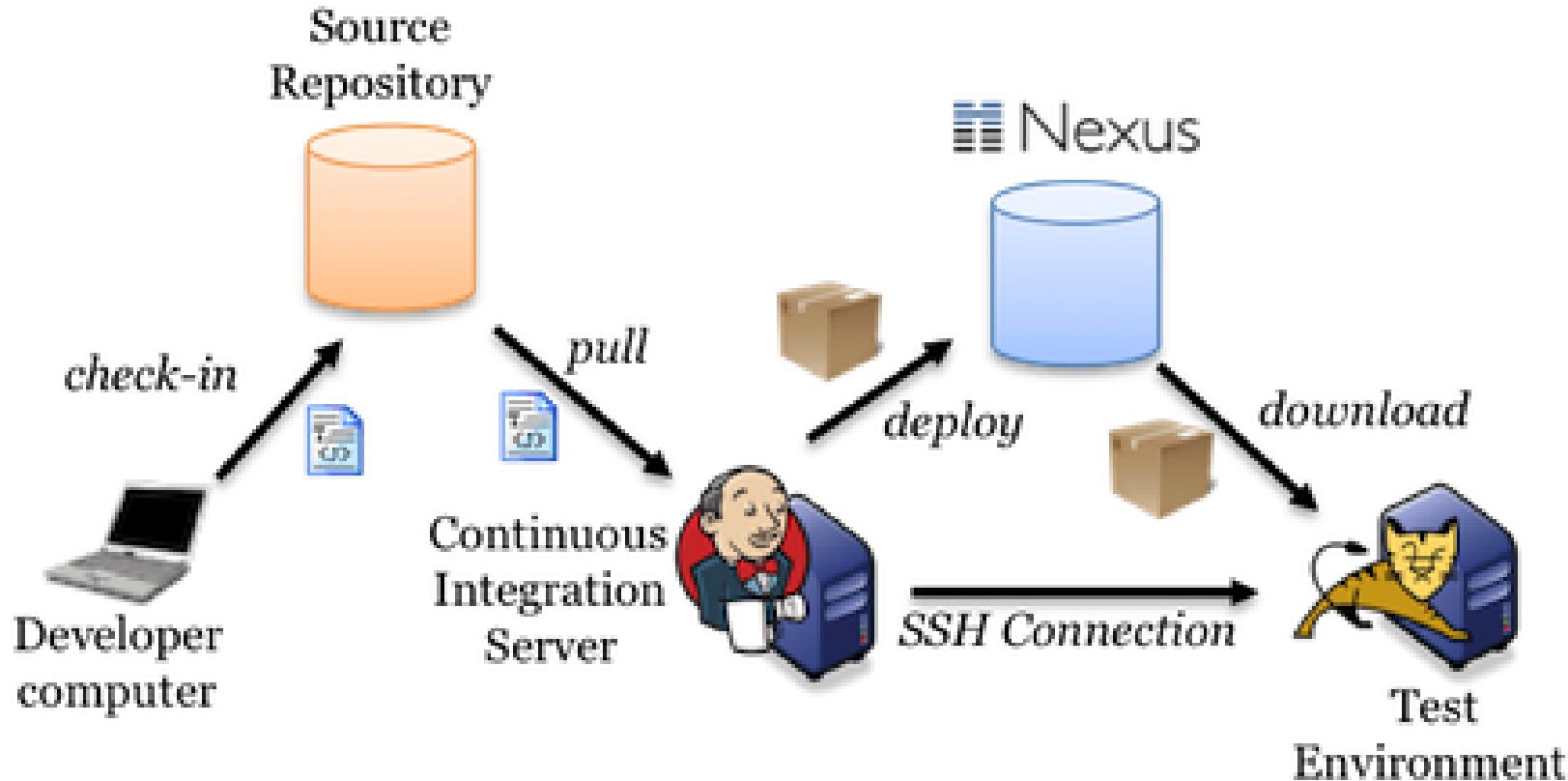
***Don't leave out your Database, Architecture, Performance, Governance, Security, and QA folks in these tool considerations**

Business Continuity

- Business continuity is a very broad term, and CI/CD are a key area for most organizations
- Reliable releases, scalability, sustainability are all key maturity metrics
- Releases focused on time-to-market is the opposite end of the continuity spectrum from HA/DR
- Stress is a key business continuity metrics, tied to release, development, and maintenance
- Time-to-value measures business continuity across value streams for SDLC

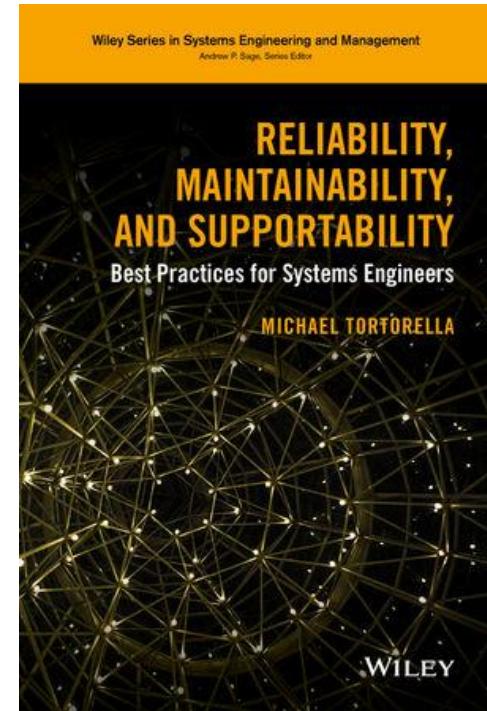


Typical CI/CD Setup



Sustainability & Supportability

- Sustainability means continuing to adapt.
- Supportability is tied to your organizational ability to utilize what you have for technology and process.



Why Continuous Integration & Delivery?

- Purposes
 - You find out quickly about integration problems
 - Immediately evident if a code change “breaks the build”
 - Prevents a tedious integration step at the end of code changes
 - Should be complete enough that eventual first deployment of the system is “no big deal”
 - CI system gets the code directly from version control system of record
 - Build is independent of any local artifacts that are on the developer’s machine
 - Organizational repository may be distributed or centralized
 - Controlled links to corporate repository and Maven Central
 - Goal is to ensure that the application can be built from the corporate repository

CI/CD Reduces Gaps in Intent



How do we get started?

The start of CI or CD in organizations usually begins with:



- Installing a CI/CD tool like Jenkins
- Using Jenkins tasks to wrap existing scripts running on servers
- Automation of sequenced tasks
- Connecting the CI/CD tool to the existing technology, e.g. XL Release, Packer, Nexus, GitHub

CI/CD Vocabulary

- Task – An individual job that is automated
- Process – decomposed into a sequence of tasks and then prioritized for automation
- Pipeline – Orchestrated tasks are automated for a process
- CI – Continuous Integration
- CD – Continuous Delivery
- Jenkins – A popular CI/CD console
- (D)VCS – (Distributed) Version Control System
- Trigger – manual, scheduled, or event-driven methods for kicking off a CI/CD task or pipeline
- Dashboard – CI/CD console view of processes that have run across our application lifecycle

CI/CD: Challenges Solved

- Rapidly deliver reliable, scalable services that are critical for the business.
 - “Integrate and test changes after no more than a couple of hours.”
 - Beck & Andres “Extreme Programming Explained”.
 - Turning snowflake builds into the phoenix.
- CI is focused on automation of all development testing at the earliest stage where issues can be identified across an application, service or code product.
- CD brings the same application packaging across the infrastructure for consistency, sustainability, and supportability.

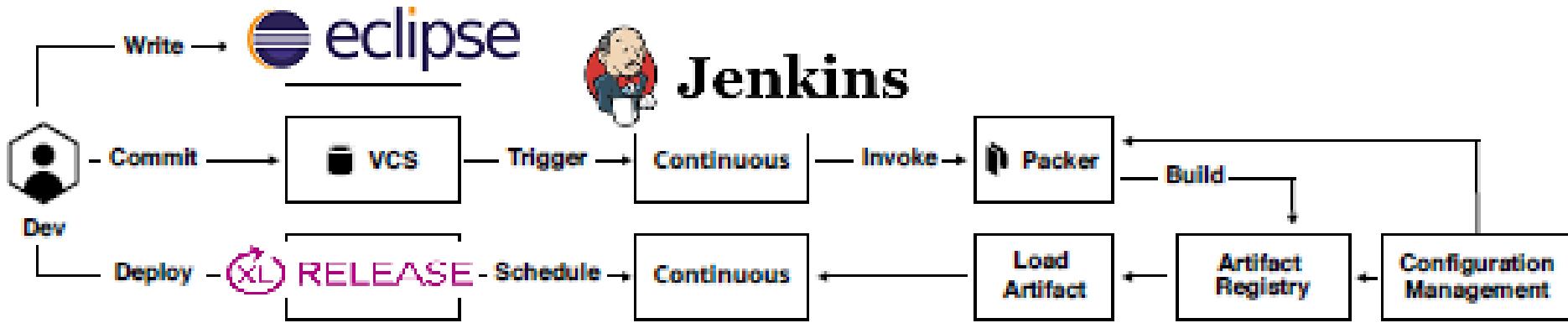


Benefits of CI/CD

- Install to Dev, QA, UAT, BVT, Preprod
- Always have an installable artifact
 - It's a great time to generate development metrics, e.g. Code Coverage, Standards Compliance, Static Analysis
- Generate development reports
 - SonarQube
 - Findbugs
 - PMD
- Dashboard visibility to success & opportunities

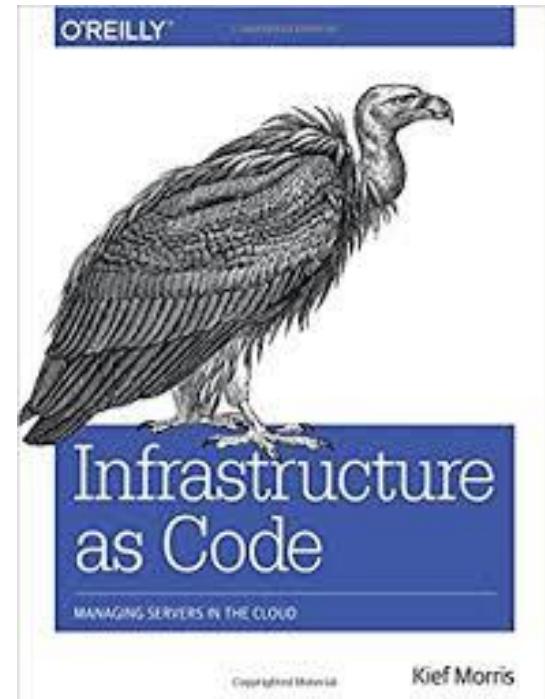


DevOps Release Pipelines



Infrastructure as Code (IaC)

- IaC or programmable infrastructure, means writing code to automate provisioning of infrastructure into Private or Public Cloud.
- Frequently used in Continuous Integration and Delivery



Jenkins

pipeline butler



Introduction to Jenkins

- Jenkins is a robust CI platform with hundreds of plugins and thousands of open-source developers globally who have worked on the platform
- Jenkins forked in November 2010 from Hudson after Oracle purchased Sun
- Jenkins is the point of entry CI tool for many organizations
- <https://jenkins.io/index.html>



Jenkins is pretty Groovy

- Jenkins supports Groovy:
- Pipeline plugin
- Groovy Script Console for troubleshooting
- Groovy post build plugin to run a groovy script when your build is done.
- Groovy plugin to run a groovy script on selected steps of the build

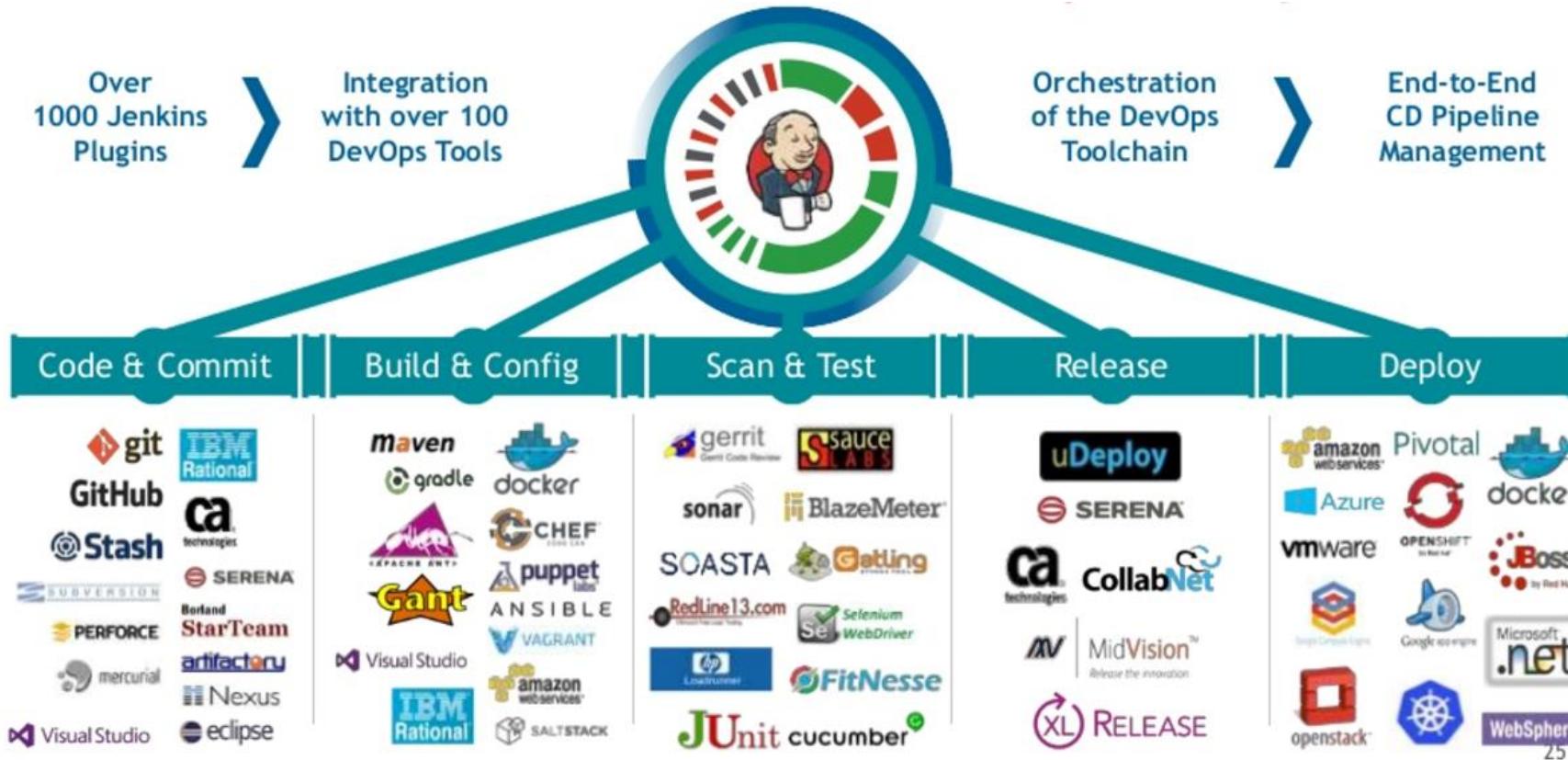


Introduction to Groovy

- Built on top of Java
- Functional language
- Dynamic language
- Runs in JVM

```
def sayHello(name) {  
    println("Hello $name!")  
}  
  
def names = ["World", "Class"]  
names.sort()  
for (name in names) {  
    sayHello(name)  
}  
}
```

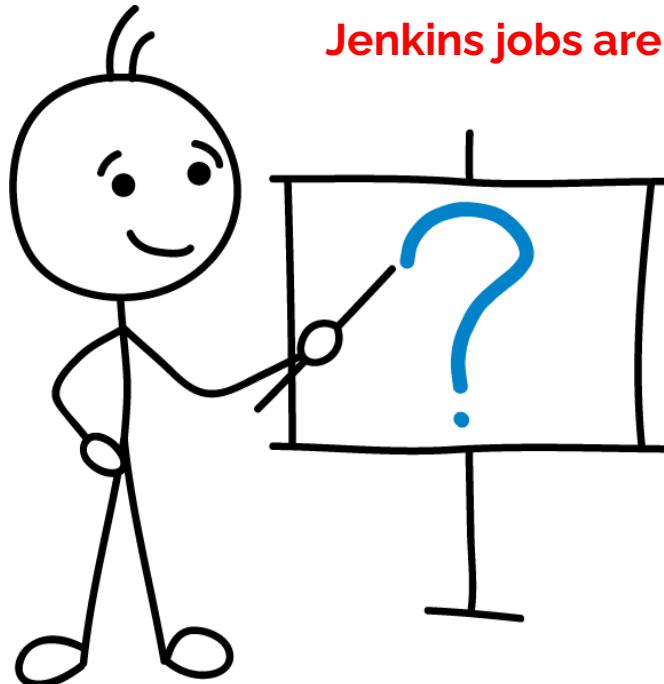
What is Jenkins?



Jenkins Features

- Executes jobs based on a number of triggers
 - Change in a version control system
 - Time
 - Manual Trigger
- A Job consists of some instructions
 - Run a script
 - Execute a Maven project or Ant File
 - Run an operating system command
- User Interface can gather reports
 - Each job has a dashboard showing recent executions
- Jenkins 2 adds the "Pipeline"
 - Orchestrate Continuous Deployment processes more easily

What is a Jenkins Job?



Jenkins jobs are used to create CI/CD automations:

- Traditional Continuous Integration tasks are called "Jobs"
- More complicated tasks can be put together in a sequence
- Create a new job by clicking 'New Item' from the main menu
- Jobs automate tasks, pipelines automate processes

Jenkins Job Types available

Enter an item name

» Required field

 **Freestyle project**
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

 **Maven project**
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

 **Pipeline**
Orchestrates long-running activities that can span multiple build slaves. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

 **External Job**
This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system.

Jenkins Orchestration Job Types

- **Multi-configuration job:**
 - Run same build job in many different configurations.
 - Powerful feature, useful for testing an application in many different environments.
- **Folder:**
 - aka the CloudBees Folders Plugin
 - Lets you group jobs in folders
 - Creates separate namespaces for jobs
- **Pipeline**
 - Runs an orchestration script that can have multiple steps and span restarts
- **Multibranch Pipeline**
 - Creates a pipeline for more than one branch in an SCM repository

Connecting to Code (SCM)

- Jenkins monitors version control system, and checks out the latest changes as they occur, e.g. Subversion, Endeavor, Git, GitHub, BitBucket
- Compiles, creates artifacts, and tests the most recent version of the code
- Simply check out and build the latest version of the source code on a regular basis.
- SCM configuration options in Jenkins are identical across all sorts of build jobs.
- Jenkins supports Git and Subversion out of the box (default plugins)
- Integrates with a large number of other version control systems via installable plugins.



Working with Git

- **Git support is in a plugin**, and is installed by default if you select "Install suggested plugins" at initial setup.
- Any type of remote repository can be used:
 - Could be https, ssh, or local
- Jenkins will check that the URL is valid as soon as you enter it.
- You can store ssh credentials or http credentials
- There are a wide variety of other "**Additional Checkout Behaviors**" that are possible, which we note in the next slide



Configuring SCM in Jenkins

The screenshot shows the Jenkins configuration interface for a Git repository. On the left, there's a sidebar with 'Git' selected under 'SCM'. The main area has two sections: 'Repositories' and 'Branches to build'. The 'Repositories' section contains a list of advanced behaviors, and the 'Branches to build' section contains a list of build triggers.

Section	Configuration Item
Repositories	Advanced checkout behaviours
	Advanced clone behaviours
	Advanced sub-modules behaviours
	Calculate changelog against a specific branch
	Check out to a sub-directory
	Check out to specific local branch
	Clean after checkout
	Clean before checkout
	Create a tag for every build
	Custom SCM name
Branches to build	Custom user name/e-mail address
	Don't trigger a build on commit notifications
	Force polling using workspace
	Merge before build
Polling ignores commits from certain users	

Storing Credentials

- Usually, SCM systems require login
- The values we use for login are called "**Credentials**" and can be managed centrally
- As a convenience, you can add them directly from the job configuration screen
- Click '**Add**' and then select '**Jenkins Credential Provider**'

Jenkins Credentials Provider: Jenkins

Add Credentials

Domain: Global credentials (unrestricted)

Kind: SSH Username with private key

Username: []

Private Key:

- Enter directly
- From a file on Jenkins master
- From the Jenkins master `~/.ssh`

Passphrase: []

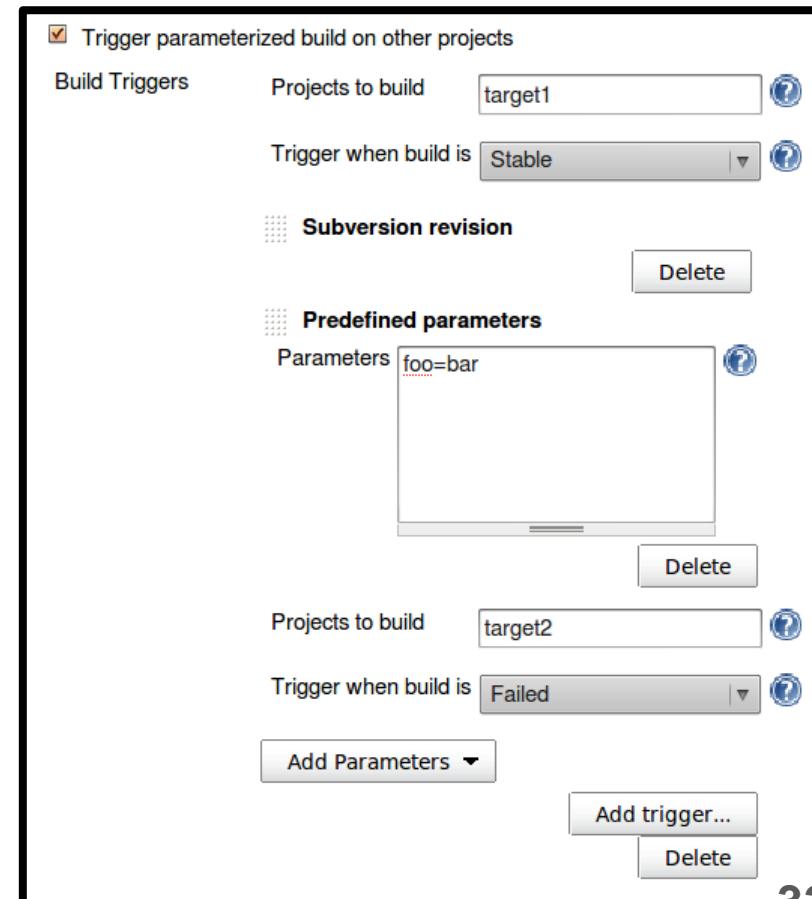
ID: []

Description: []

Add Cancel

Freestyle Build Triggers

- In a **Freestyle build**, there are three basic ways a build job can be triggered
 - Start a build job once another build job has completed
 - Kick off builds at periodic intervals
 - Poll the SCM for changes
- In addition, for more complex chaining of builds you can use the **Parameterized or Parameterized Remote Trigger Plugins**



Build Trigger Options

Build Triggers

- Build whenever a SNAPSHOT dependency is built
 - Schedule build when some upstream has no successful builds
- Trigger builds remotely (e.g., from scripts)
- Build after other projects are built
- Build periodically
- Build when a change is pushed to GitHub
- Poll SCM

Scheduling Jenkins Jobs

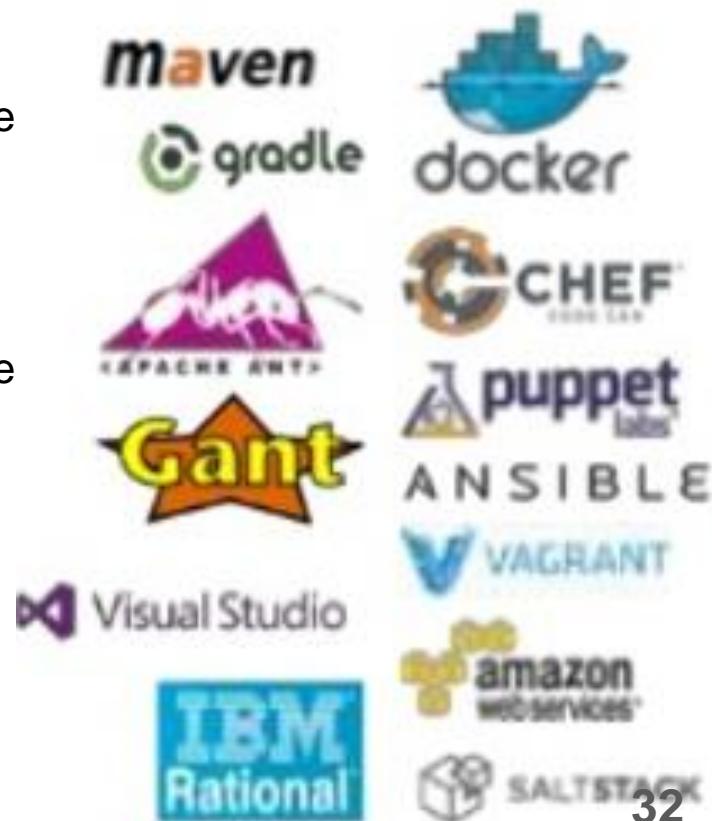
- Build job at regular intervals.
- For all scheduling tasks, Jenkins uses a cron-style syntax, consisting of five fields separated by white space in the following format:
 - MINUTE : Minutes within the hour (0–59)
 - HOUR : The hour of the day (0–23) DOM
 - DOM : The day of the month (1–31)
 - MONTH : The month (1–12)
 - DOW : The day of the week (0–7) where 0 and 7 are Sunday.

Polling the Version Control System

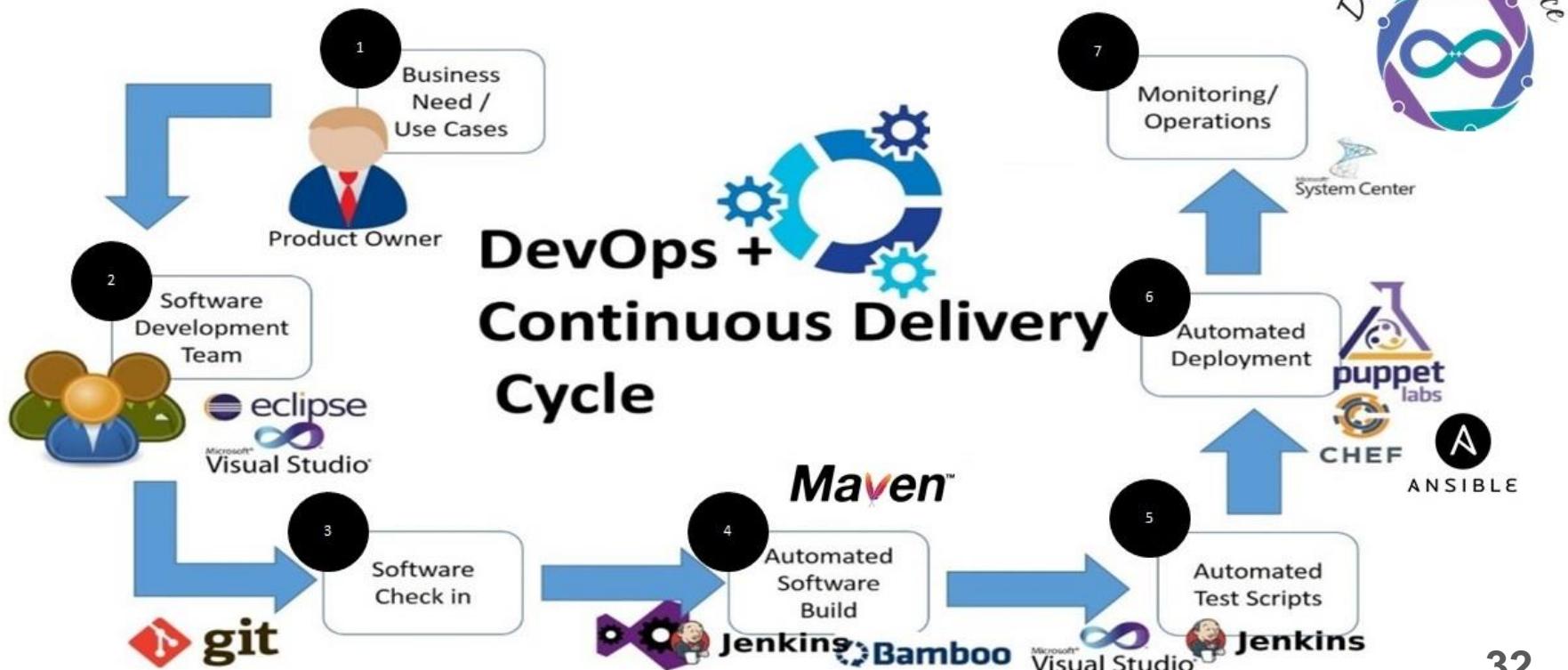
- Poll your DVCS/VCS/SCM system for changes
- Poll SVN or Git server at regular intervals if any changes have been committed.
- The more frequent the polling is, the faster the build jobs will start, and the more accurate.
 - But, there's more load on the SCM server
 - In Jenkins, SCM polling is easy to configure, and uses the same cron syntax we discussed previously.

Jenkins Build Tool Support

- As an example, Jenkins has excellent Maven support, and Maven build steps are easy to configure and very flexible.
- Select “Invoke top-level Maven targets” from the build step lists.
- Select a version of Maven to run (if you have multiple versions installed)
- Enter the Maven goals you want to run. Jenkins freestyle build jobs work fine with both Maven 2 and Maven 3.
- The optional POM field lets you override the default location of the Maven pom.xml file.



Jenkins in the Toolchain Life Cycle



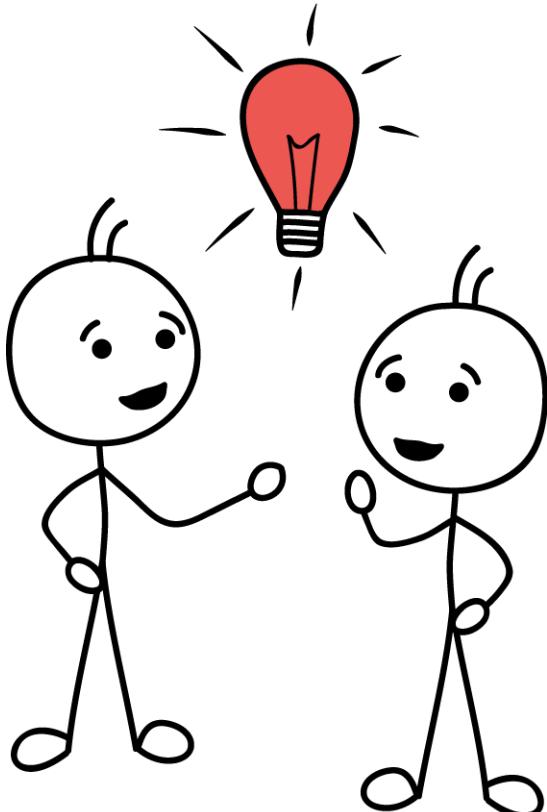
How do we get started?

The start of Jenkins Job automation in organizations usually begins with:



- Prioritizing processes and tasks for automation
- Identifying the toolchains that will be integrated with or tied to Jenkins
- Identifying what version control systems will be used with Jenkins
- Automating the process of creating Jenkins worker nodes with Docker or utilizing CloudBees Jenkins

Discussion



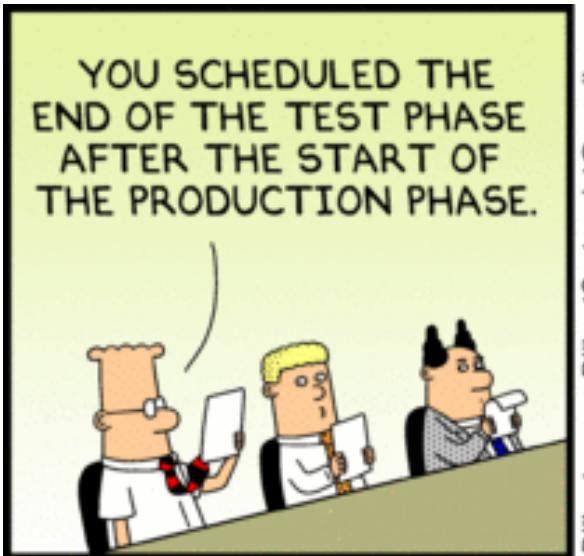
- What are current usages of Continuous Integration and Delivery in my role, group, or line of business?
- How does my role change in a world CI/CD?
- What toolchains are part of your organization's Continuous Integration stack?
- What environments are part of my application Continuous Delivery pipeline?
- What can we automate, e.g. build, deploy, risk assessment, to improve flow?

Continuous Testing

QA as a part of workflow



Testing your Big Data Pipeline



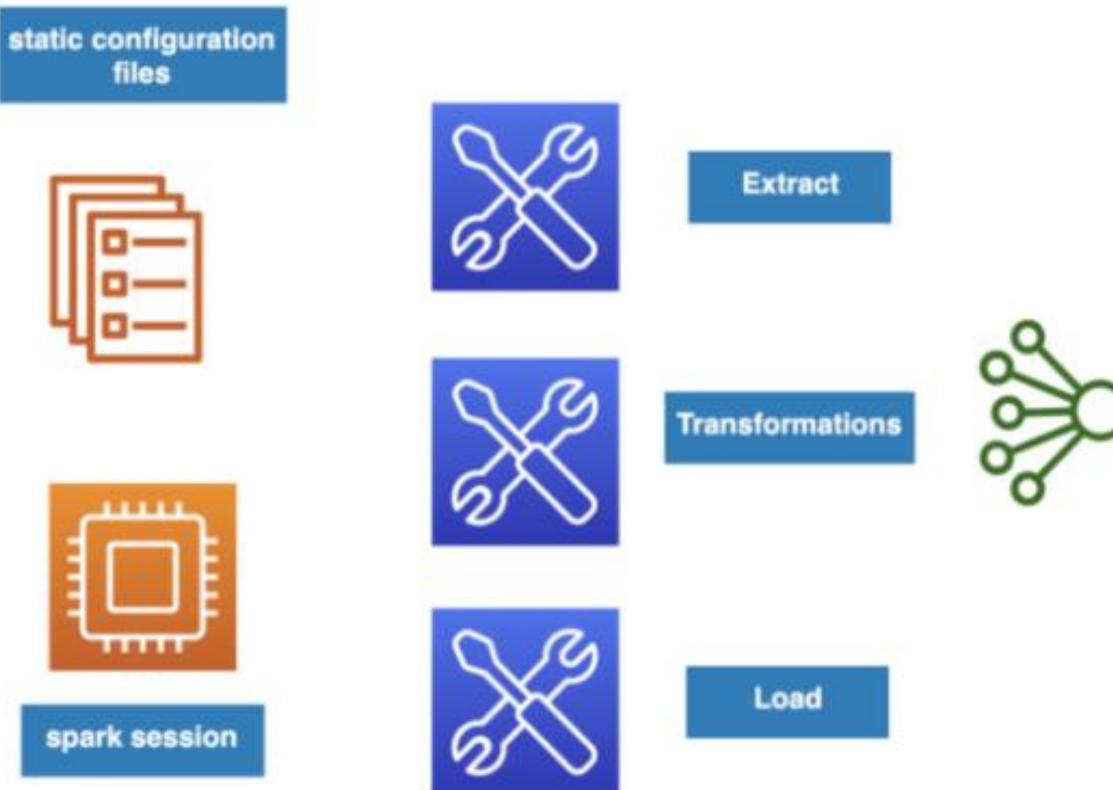
Continuous Code Quality

- “Improve quality and you improve productivity” – Deming
- DevOps involves development, operations, and quality assurance
- Plays a very important role in DevOps
- Application / software should be tested
- Test types: unit tests, integration tests, functional tests, smoke tests, ...
- Continuous quality assurance also involves checking the code quality
- Code quality can be checked by using software, such as, SonarQube, Clover, and FxCop

Driven Design

- Behavior Driven Development – Acceptance TDD
- Test Driven Development
- Feature Driven Development– deliver per sprint
- Story TDD
- Unit Tests through XUnit framework
- Understanding of what the system should do before writing code
- PyUnit (Python), Junit (Groovy or Java), Flex Unit, Qunit, and other frameworks
- Code or script only what is required

Testing your Big Data Pipeline

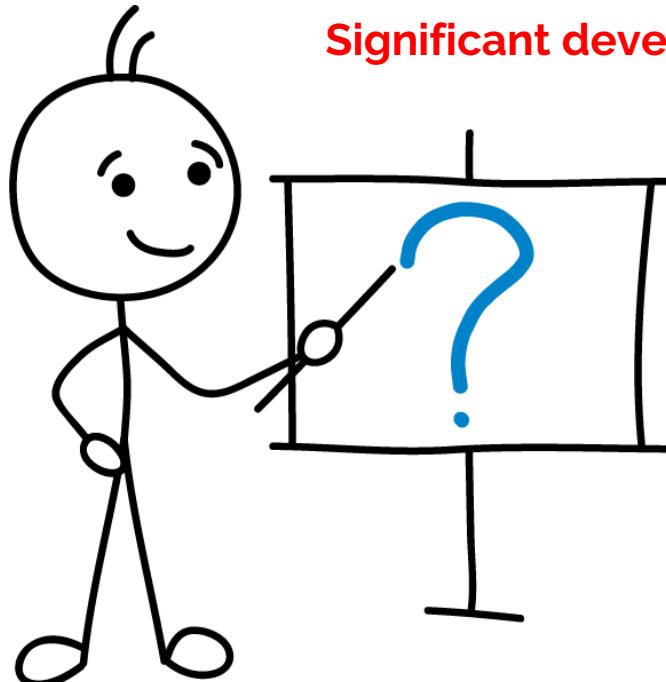


Jenkins Plugins

Artifact management



What are Jenkins Plugins?



Significant development by organizations and open source:

- Plugins provide the functionality in Jenkins
- Without Plugins Jenkins does nothing
- Common plugins are included with installation (a checkbox is provided to stop that behavior)
- Plugins for almost every build, configuration, source system, and many others
- Plugins can be created for special purpose needs

plugins.jenkins.io



Plugins Index

Discover the 1000+ community contributed Jenkins plugins building, deploying and automating any project.

Browse

Find plugins...



Pull Request Builder Plugin

- Provides assist in support of automating code review with github/gitlab.
- For any new pull request this plugin does runs the build on the code as well as gathering configured static analysis.
- Helps reviewers get an idea of the health of the code which is going to be merged.
- You can define automatic merge if build passes.

The screenshot shows the Jenkins configuration interface for the "GitHub Web Hook" and "Github pull requests builder" sections. In the "GitHub Web Hook" section, the "Manually manage hook URLs" option is selected. In the "Github pull requests builder" section, the "Github server api URL" is set to <https://api.github.com>, the "Username" is "example-builds", and the "Password" is masked. A checkbox for "Use comments to report results when updating commit status fails" is checked. An "Admin list" input field contains "admin1 admin2 admin3".

Job Generator Plugin

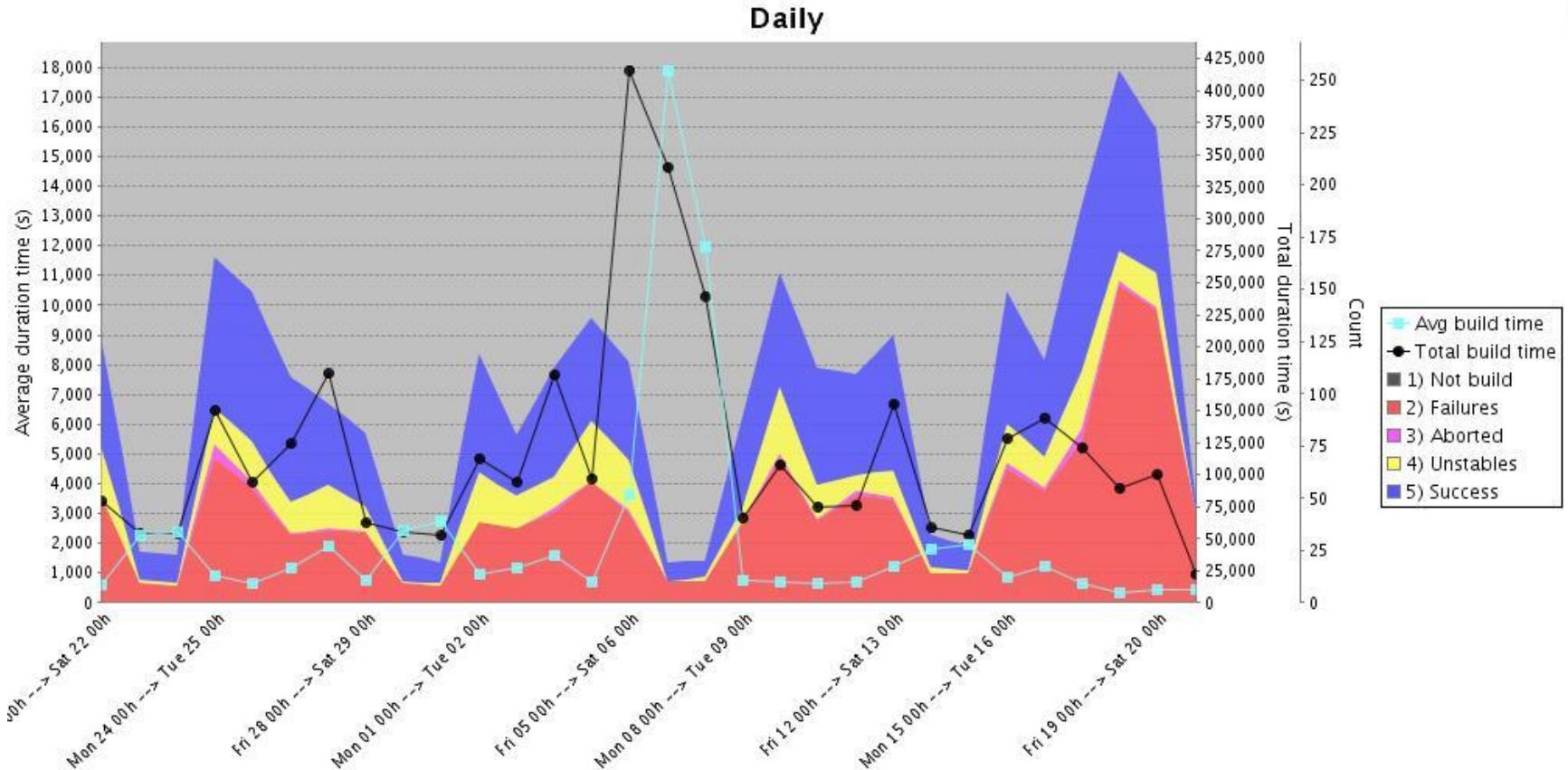
- A job generator copies itself when it is executed replacing all generator parameters by the given values. The conditional entries (Conditional BuildStep Plugin and Flexible Publish Plugin) are resolved during the generation so generated job configurations are clean.
- By using the Parameterized Trigger Plugin you can generate multiple jobs in one pass by calling a downstream job generator multiple times with different parameter values; this way you can fork Jenkins jobs easily.

The screenshot shows the configuration page for a new Jenkins job named "my_job_generator". The "Job name" field is filled with "my_job_generator". Below it, there are four project selection options:

- Build a free-style software project**
This is the central feature of Jenkins. Jenkins will...
- Build a maven2/3 project**
Build a maven2 project. Jenkins takes advantage...
- Build multi-configuration project**
Suitable for projects that need a large number of...
- Job Generator**
A job which generates a new job when executed. I...

The "Job Generator" option is selected and highlighted with a red border.

Global Build Stats Plugin



Post-Build Task Plugin

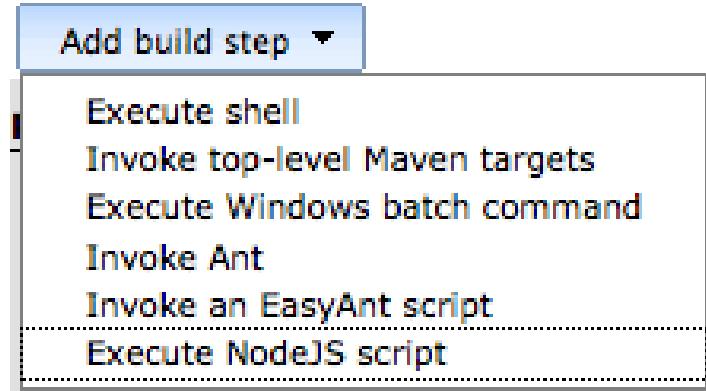
- Used when performing some actions on the basis of the results of a build
- For example, if build passed
 - upload artifact(ex debian) to some repo (apt)
 - perform some packaging part or similar.
- In case of failure you may want to
 - roll back something (like release) .
- This plugin helps you to define the pass/fail criteria and let's you decide what to do after that.

Post build task

Tasks	Log text	Operation	Value	Action
	Total time: (\d+)	-- OR --		Delete Log Text
	build	-- AND --		Delete Log Text
		Add		
Script	script.sh			

Node.js Plugin

- Running Node projects from a shell script will be fine for many projects
- There is also a Node.js plugin
 - It only runs on Linux
 - Provides auto-installers
 - Lets you create multiple Node environments
 - Different versions
 - Different sets of global modules



SonarQube Plugin

Build

SonarQube Scanner for MSBuild - Begin Analysis



Project key

FabrikamFiber

Project name

Fabrikam Fiber

Project version

1.0

Additional arguments

Delete

Additional command line arguments

Execute Windows batch command



Command

"C:\Program Files (x86)\MSBuild\14.0\Bin\MSBuild.exe" /t:rebuild

See [the list of available environment variables](#)

Delete

SonarQube Scanner for MSBuild - End Analysis



Delete

Save

Apply

Plugin Dependencies

- Many plugins have dependencies, and those may have subordinate dependencies.
- Installing a plugin can install the required, optional and implied additional plugins
- For example, the current Ansible build plugin has the dependencies noted

Dependencies

[Credentials v.1.16.1](#) (required)

[Plain Credentials v.1.4](#) (required)

[SSH Credentials v.1.10](#) (required)

[Pipeline: Step API v.1.10](#) (optional)

[Job DSL v.1.36](#) (optional)

[bouncycastle API v.2.16.0](#) (implied)

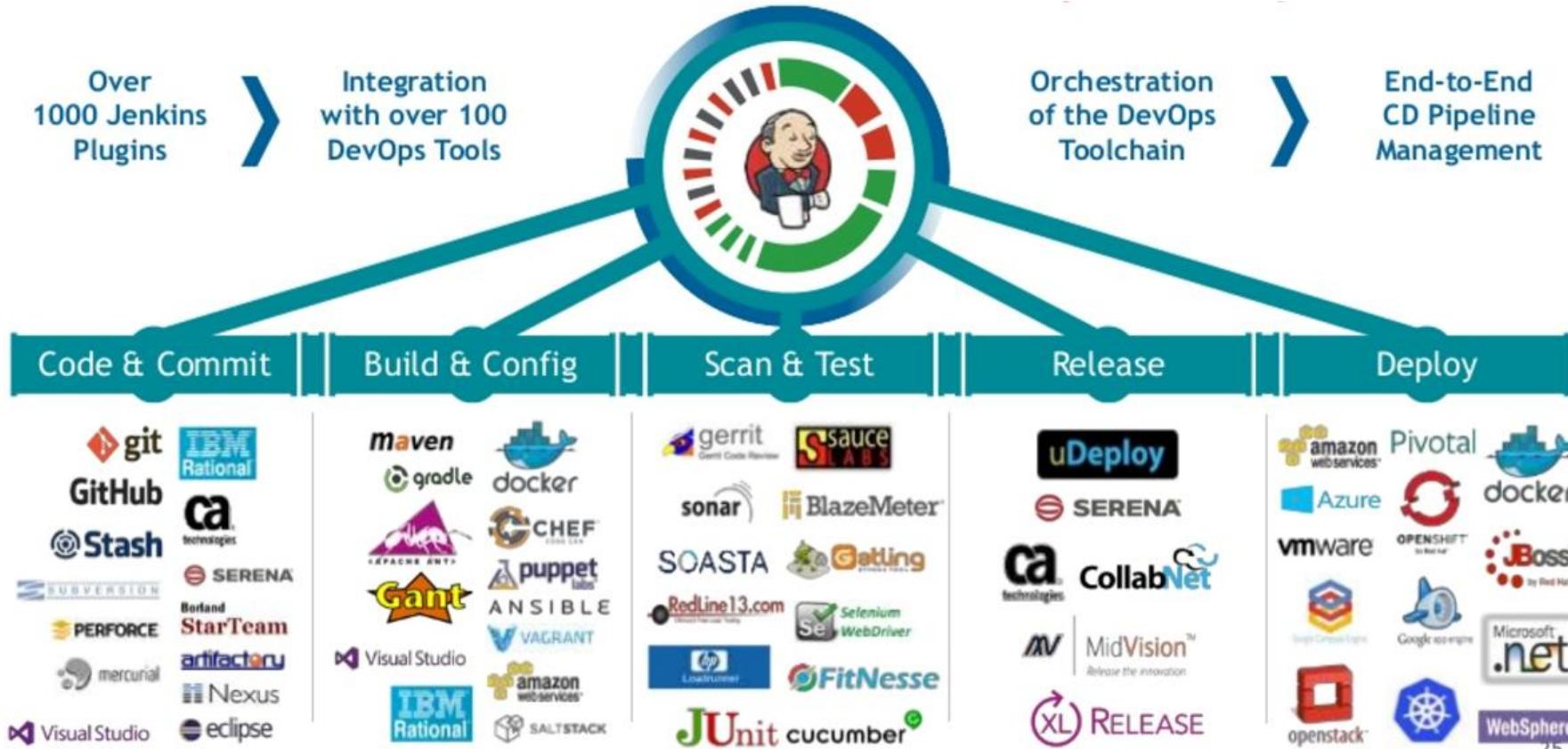
(what's this?)

[Command Agent Launcher v.1.0](#)

(implied) (what's this?)

[JDK Tool v.1.0](#) (implied) (what's this?)

Jenkins Plugin Ecosystem

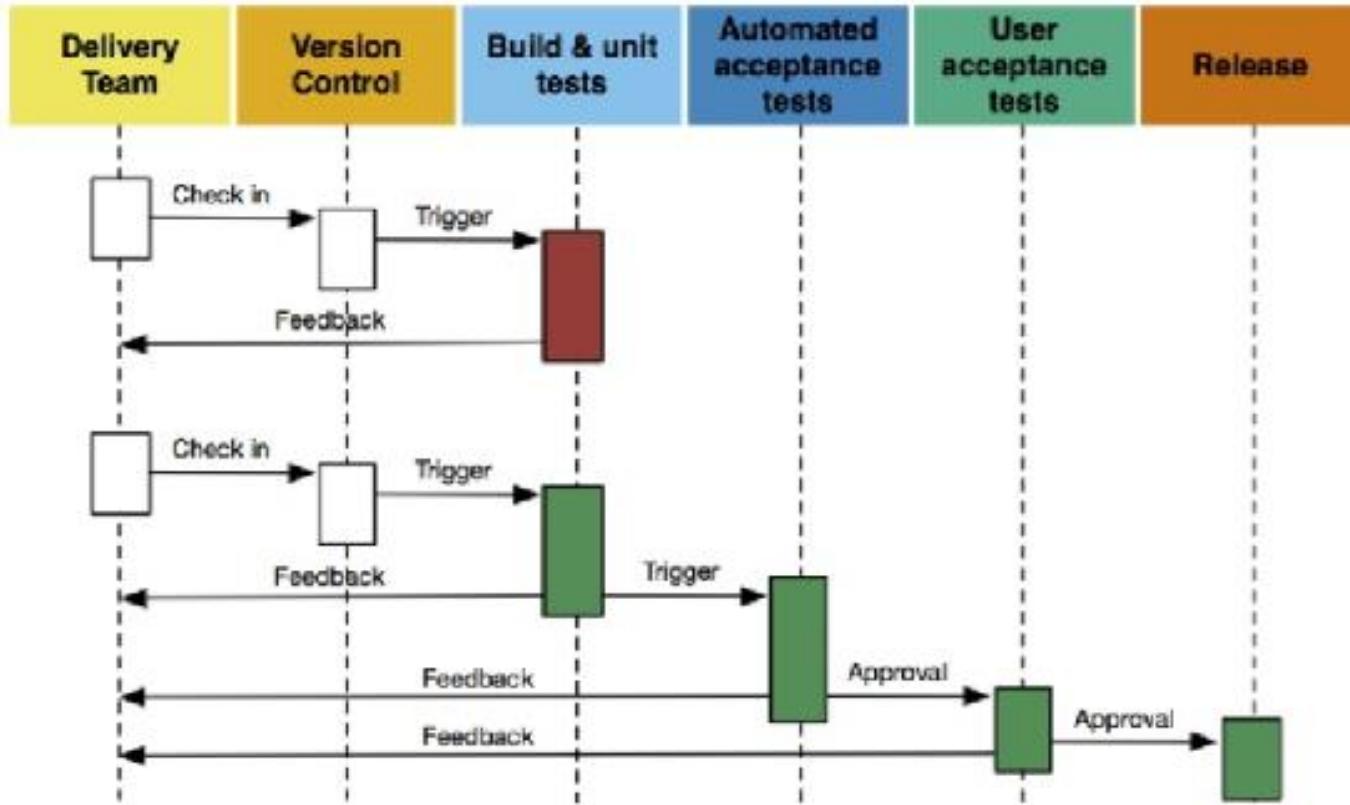


Building CD Pipelines

Continuous Delivery



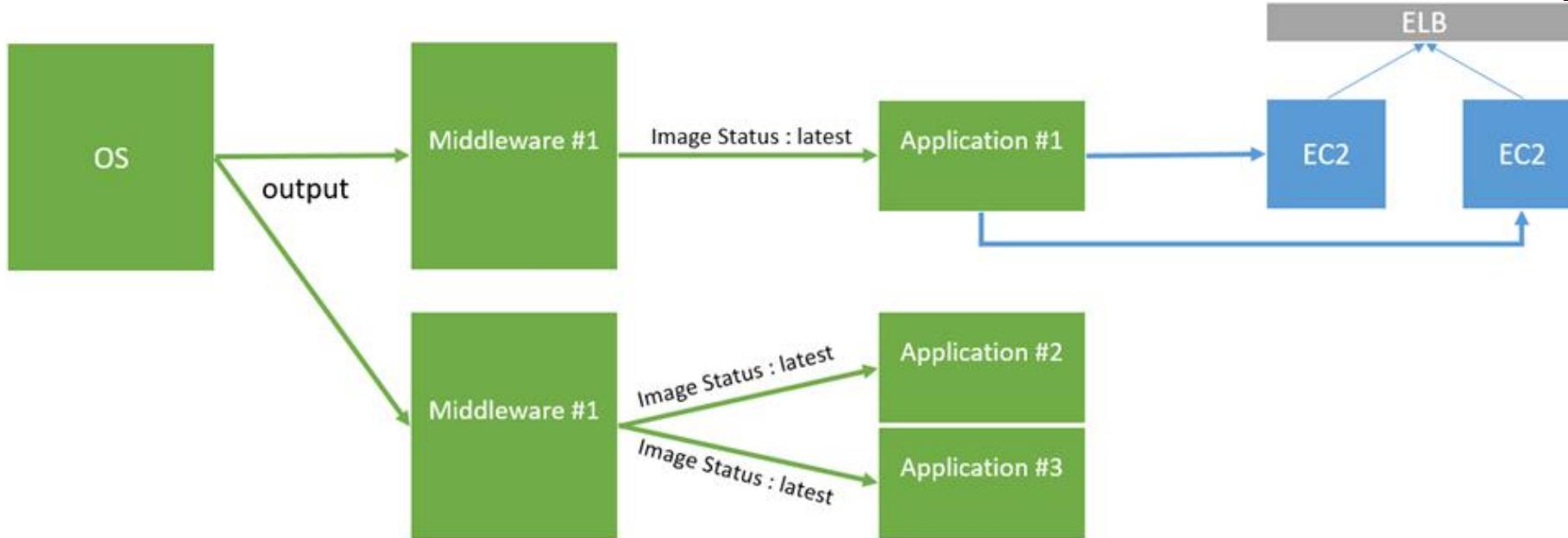
Why Pipelines?



Jenkins Pipeline – Challenges Solved

- Jenkins has always been useful for executing tasks in sequences across servers
 - A job completed successfully can trigger another job
 - Jobs can be run on different machines
 - Parameterized jobs can gather information from users
 - Plugins can manage deployment
- Workflow processing used to be more difficult in Jenkins
 - Builds in-process don't persist across restart
 - No logic in linking jobs
 - No unified user interface
 - Job definitions are not in Version Control
- **Solution: The Pipeline Plugin**

Jenkins Continuous Delivery Use Cases



Jenkins Pipeline in a Nutshell

- Jenkins support the creation of pipelines. They are built with simple text scripts that use a Pipeline DSL (domain-specific language) based on the Groovy programming language.
- The script, typically called **Jenkinsfile**, defines multiple steps to execute both simple and complex tasks according to the parameters that you establish.
- Once created, pipelines can build code and orchestrate the work required to drive applications from commit to delivery.

- A pipeline is executed on Jenkins nodes
- A pipeline often consists of multiple stages.
- A stage consists of multiple steps.

Jenkins Pipeline Plugin

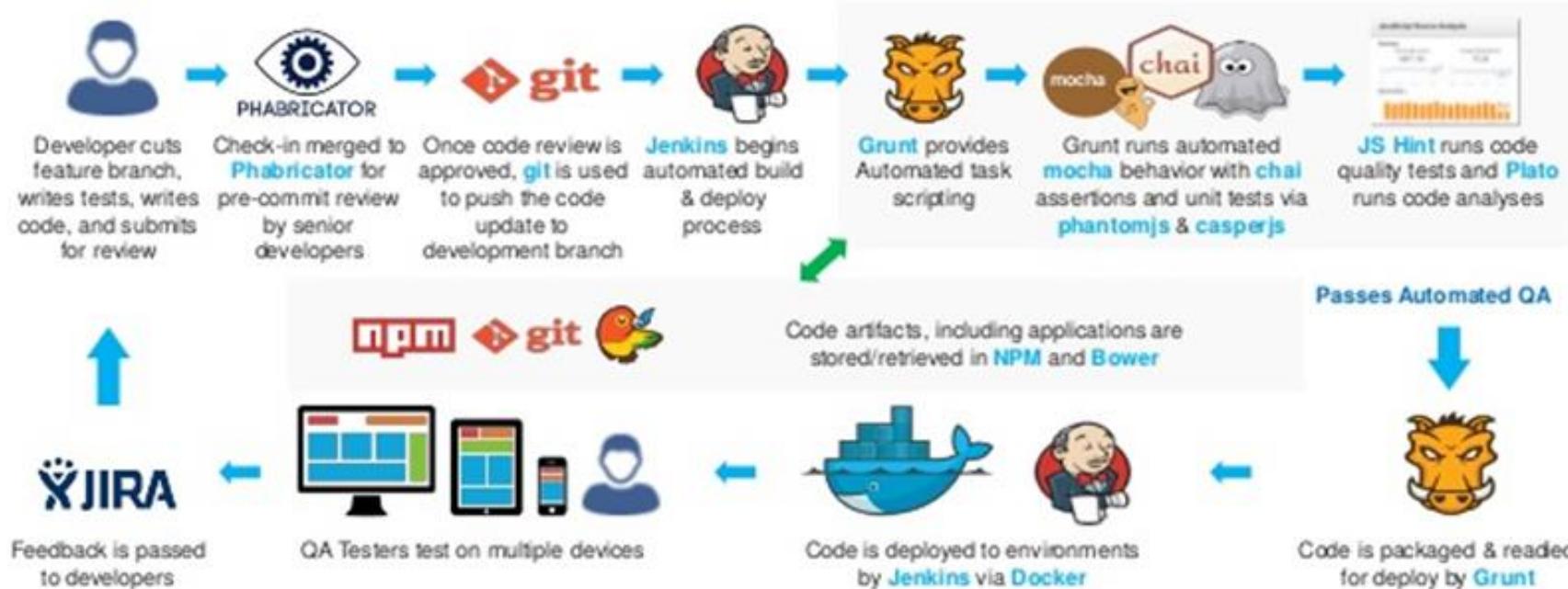
- Represents a workflow
- More like a business process in SOA whereas a "Job" is more like a "Service Operation"
- Define a process and then execute an "instance"
- Completion of the process can span more than one machine
- Process has "State", "where are we in the process"
 - State is stored persistently and distributed across worker nodes
 - Steps can be executed conditionally and in parallel across nodes
- Definition of Pipeline is stored in Version Control in the principle of Infrastructure as Code (IaC)
- Jenkins Pipeline User Interface lets us see multiple instances, reviews the I/O and see the span of an orchestrated workflow

Pipeline Views Across Orchestration

	Sync raw data and census files	Process raw logs	Generate census data	Generate stats	Publish census	Publish stats
Average stage times: (Average full run time: ~1h 19min)	3s	4s	31min 3s	48min 1s	593ms	2s
#29 Jun 29 05:25 No Changes	3s	4s	33min 57s	47min 48s	579ms	1s
#28 Jun 28 19:25 7 commits	2s	4s	32min 31s	47min 33s	565ms	3s
#27 Jun 28 05:25 No Changes	2s	5s	39min 16s	48min 37s	565ms	3s
#26 Jun 27 05:25 No Changes	3s	4s	29min 5s	49min 37s	564ms	1s

Organization Specific Toolchains

Modern Web Architecture requires a journey toward the next generation of agile development methods, DevOps capabilities, and quality-first engineering principles



Helm



Helm sample application

The screenshot shows a web browser window with the title "Guestbook" and the URL "dev.frontend.minikube.local/". The page displays a guestbook application titled "Guestbook : "MyPopRock Festival 2.0"" from "GLOBOMANTICS". The interface includes a "Filter" input field, a table of messages, and a form for leaving feedback.

Messages Table:

Name	Message
John	Very nice show !!

Items per page: 5 | 1 - 1 of 1

Feedback Form:

Leave your feedback !

Your name *

Jane

Your questbook message

Congrats to Globomantics DevOps !

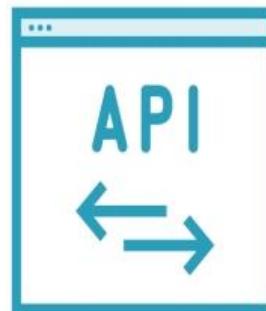
Leave message

Sample application components



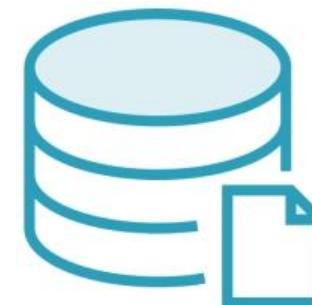
Frontend

- ✓ ConfigMap
- ✓ Pod
- ✓ Service
- ✓ Ingress



Backend API

- ✓ Secret
- ✓ Pod
- ✓ Service



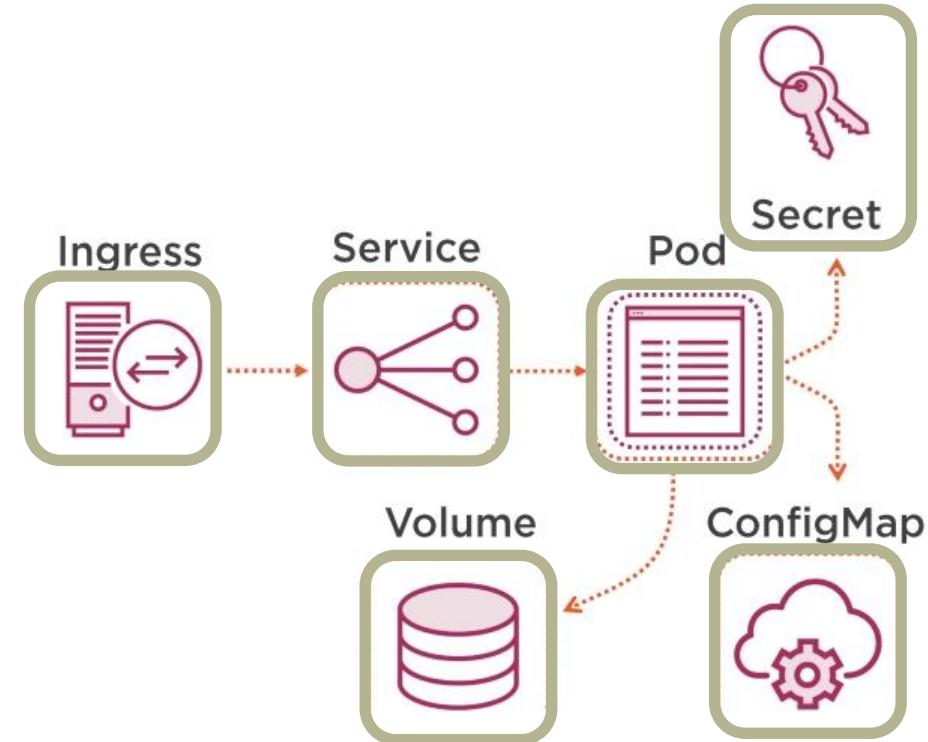
Database

- ✓ Secret
- ✓ PV
- ✓ PVC
- ✓ Pod
- ✓ Service



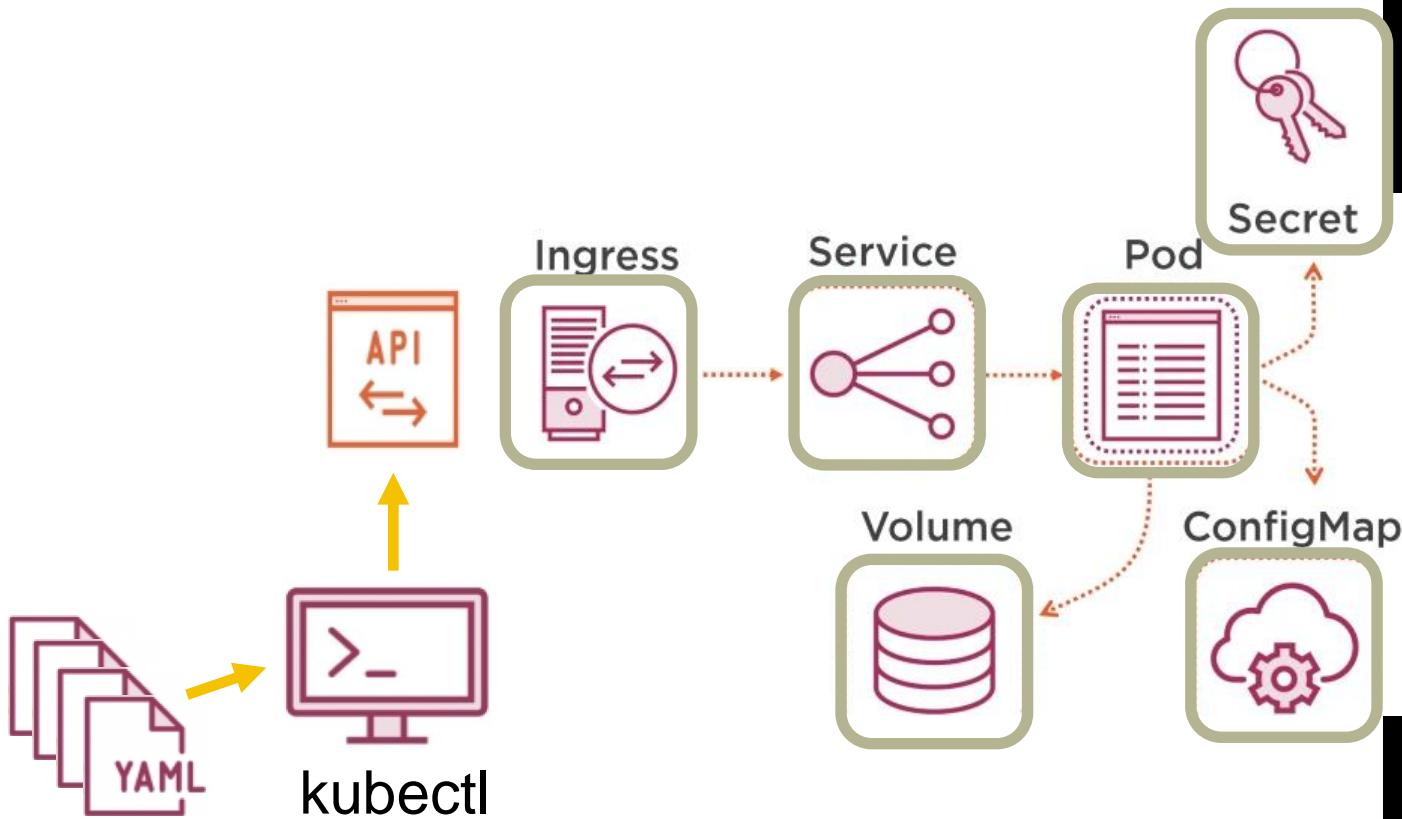
Native Kubernetes way

- Application
- Container
- Pod
- Service
- Ingress
- ConfigMap
- Secrets
- Volumes: PV, PVC, Storage

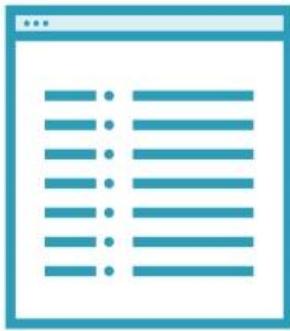


Native Kubernetes way

- Limitations
 - Packaging
 - Versioning



Guestbook: version 1



- ConfigMap
- Pod
- Service
- Ingress



Fronten
d

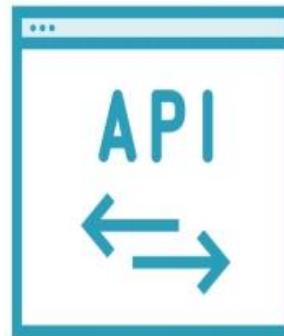
Guestbook: version 2



- ConfigMap
- Pod
- Service
- Ingress



Fronten
d



Backen
d



- Secret
- Pod
- Service



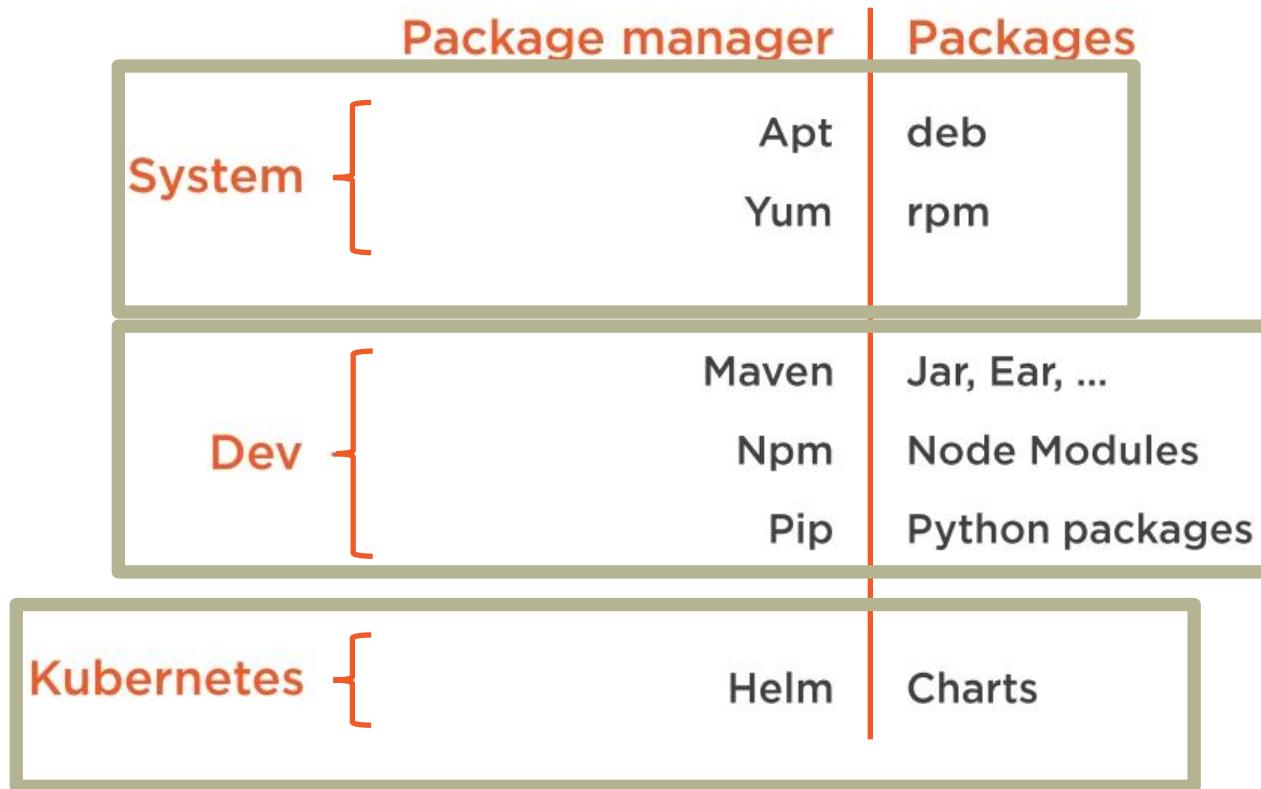
Databas
e

- Secret
- PV
- PVC
- Pod
- Service

Painful to manage

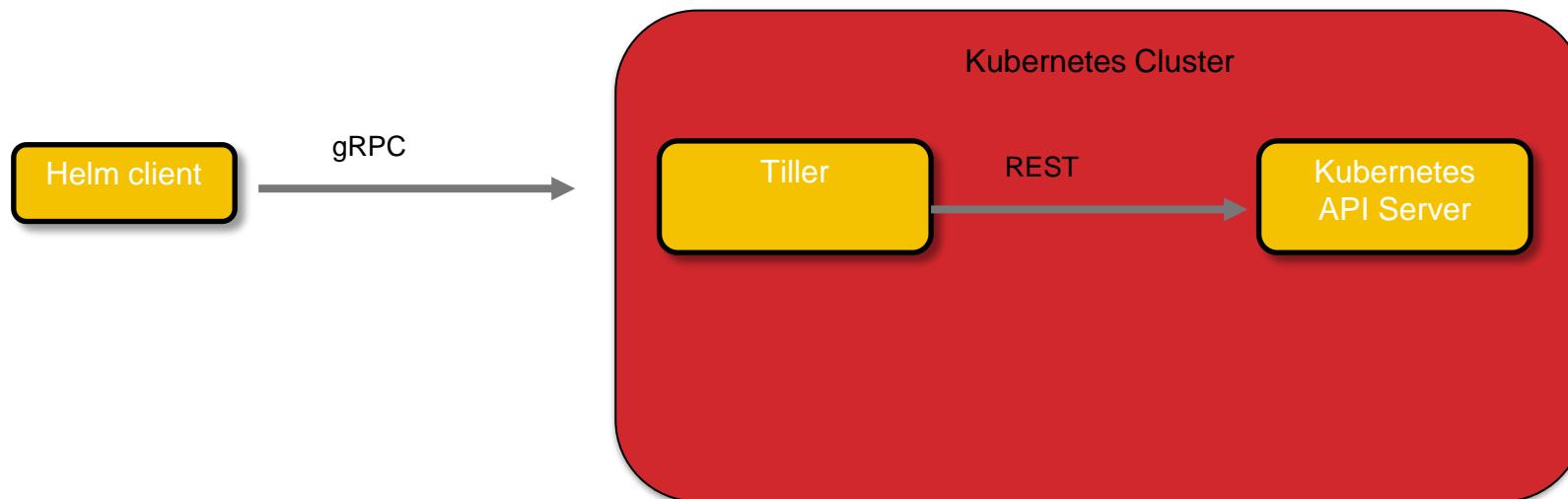


Helm features



Helm architecture

- Helm client
 - command-line
 - Interacts with Tiller
 - Local chart development
- Tiller
 - In-cluster
 - Listens to Helm client
 - Interacts with Kube API
 - Manages lifecycle



Helm features



Charts



Templates



Dependencies



Repositories

Helm Chart Structure



Helm Chart structure



- nginx-demo
 - Chart.yaml
- Chart properties
 - name
 - version
 - more..

Helm Chart structure



- nginx-demo
 - Chart.yaml
 - README.md
- Document chart
 - Overrides
 - Maintainer
 - Instructions

Helm Chart structure



- nginx-demo
 - Chart.yaml
 - README.md
- templates
 - deployment.yaml
 - ingress.yaml
 - service.yaml
- Kubernetes object definitions
 - customizable YAML templates

Helm Chart structure



- nginx-demo
 - Chart.yaml
 - README.md
- templates
 - deployment.yaml
 - ingress.yaml
 - service.yaml
- values.yaml
- Kubernetes object definitions
 - customizable YAML templates
 - Provide default values.

Helm Chart structure



- nginx-demo
 - Chart.yaml
 - README.md
- templates
 - deployment.yaml
 - ingress.yaml
 - service.yaml
 - NOTES.txt
- values.yaml
- Provide helpful output after installation
 - How to access application
 - Create user/pass

Helm Chart structure



- nginx-demo
 - Chart.yaml
 - README.md
 - templates
 - deployment.yaml
 - ingress.yaml
 - service.yaml
 - NOTES.txt
 - tests
 - test-connection.yaml
 - values.yaml
- You can create tests to confirm Chart works as expected.

Helm Chart structure

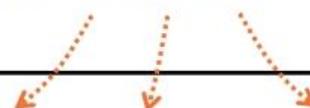


- nginx-demo
 - Chart.yaml
 - README.md
 - requirements.yaml
 - templates
 - deployment.yaml
 - ingress.yaml
 - service.yaml
 - NOTES.txt
 - tests
 - test-connection.yaml
 - values.yaml
- Define sub-charts and dependencies

Helm Chart.yaml

Chart.yaml

```
apiVersion: v1
appVersion: "1.0"
description: A Helm chart for Kubernetes
name: nginx-demo
version: 0.1.0
```



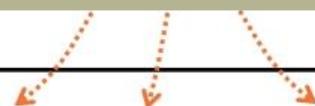
Major,Minor,Patch (SemVer 2.0)

- App you are installing

Helm Chart.yaml

Chart.yaml

```
apiVersion: v1
appVersion: "1.0"
description: A Helm chart for Kubernetes
name: nginx-demo
version: 0.1.0
```



Major, Minor, Patch (SemVer 2.0)

- Version of Helm chart

Common Helm commands

Action	Command
Install a Release	<code>helm install [chart]</code>
Upgrade a Release revision	<code>helm upgrade [release] [chart]</code>
Rollback to a Release revision	<code>helm rollback [release] [revision]</code>
Print Release history	<code>helm history [release]</code>
Display Release status	<code>helm status [release]</code>
Show details of a release	<code>helm get [release]</code>
Uninstall a Release	<code>helm delete [release]</code>
List Releases	<code>helm list</code>

K8S Pod Deep Dive



Pod Scheduling Overview

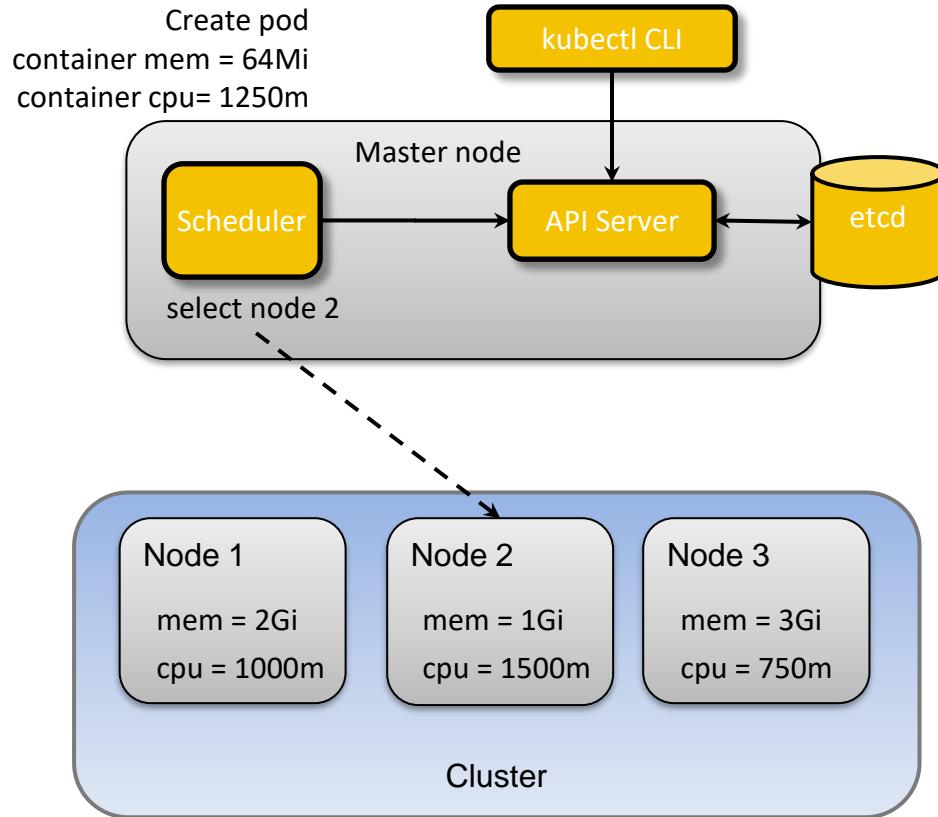
- **kube-scheduler on the master node assigns new pod requests to appropriate worker nodes**
- **Default scheduler takes account of**
 - Available node CPU/RAM
 - Resource requests from new pod – sum of resource requests of pod containers

Default scheduler will automatically

Schedule pod on node with sufficient free resources

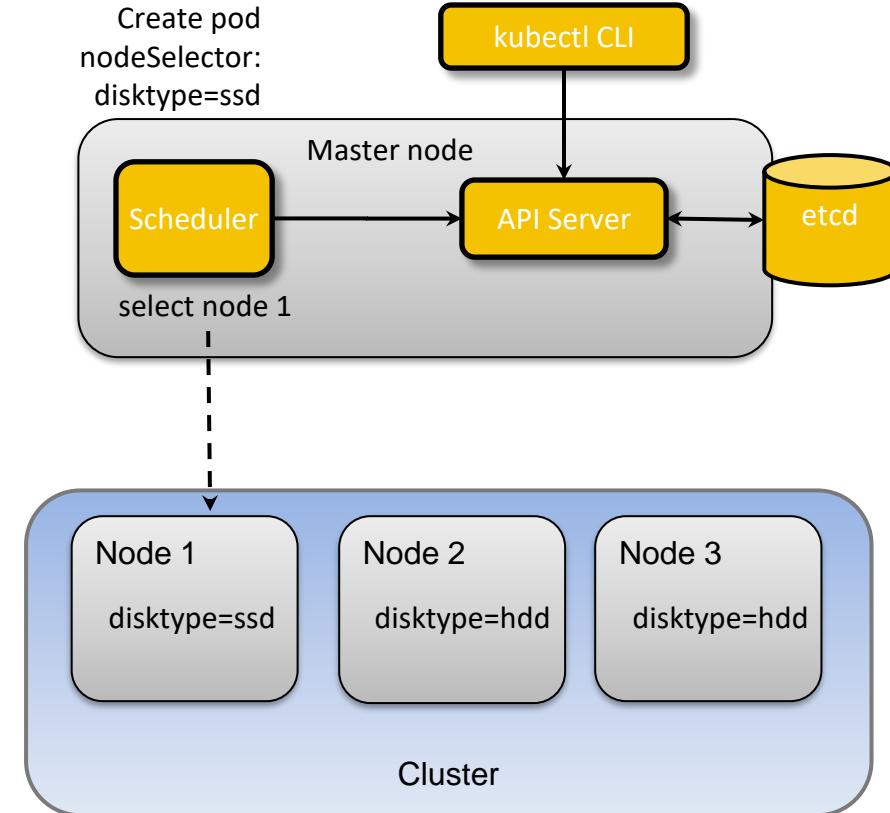
Spread pods across nodes in the cluster

Can specify custom schedulers for pods



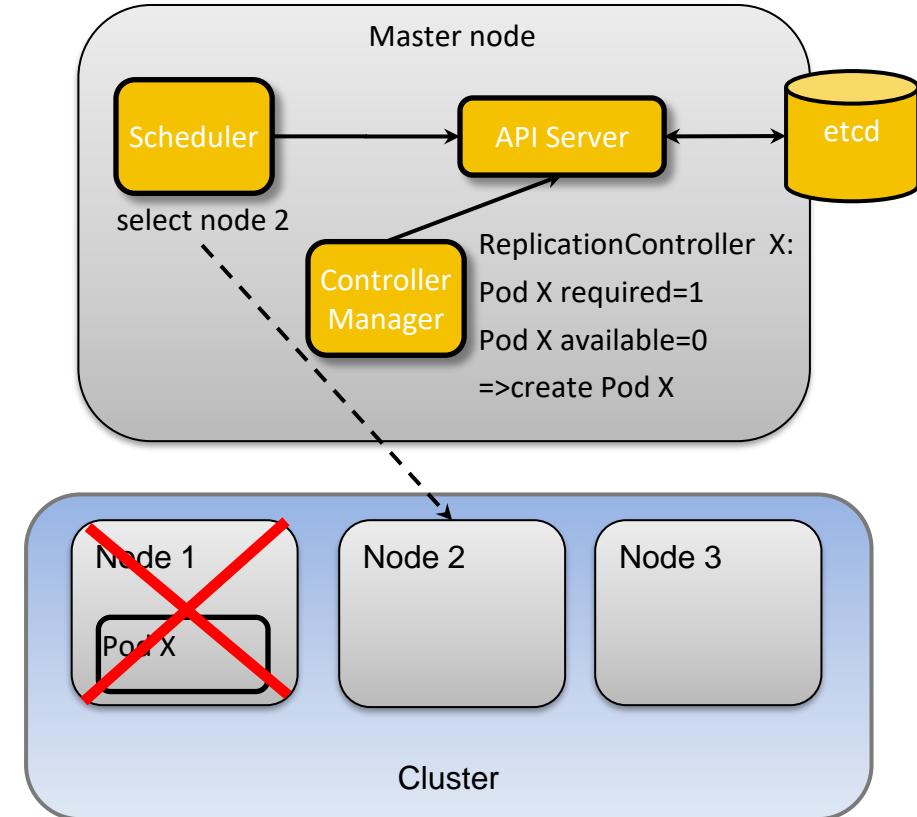
Controlling Pod Scheduling with Default Scheduler

- Nodes carry labels indicating topology and other resource notes
- Users can require pods to be scheduled on nodes with specific label(s) via a nodeSelector in container spec



Scheduling Pod Re-creation Driven by Control Loops

- kube-scheduler performs same node selection operation when new pod created due to e.g. node loss
- kube-controller-manager runs controllers like ReplicationController managing number of pod instances available
- kube-controller-manager will initiate request for new pods as needed, which will be scheduled by kube-scheduler per pod/container spec



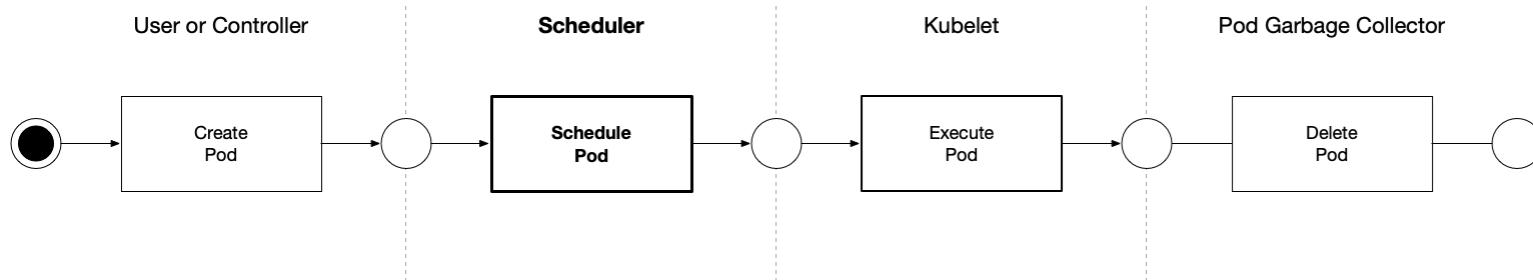
Kubernetes Scheduling

Node Selector

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    env: test
spec:
  container:
  - name: nginx
    image: nginx
  nodeSelector:
    disktype: ssd
```

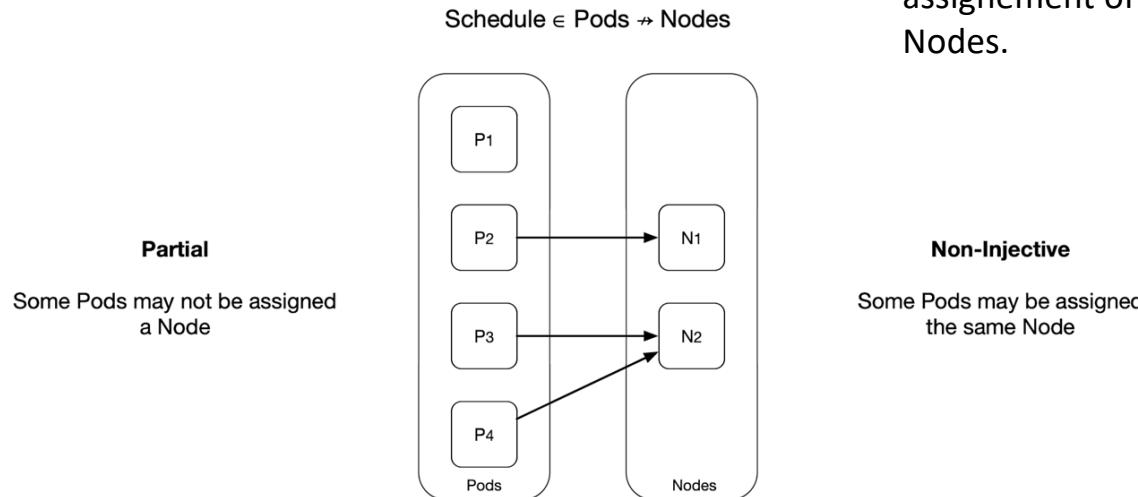
Scheduling flow

- **Scheduler (`kube-scheduler`)**: selects nodes for newly created pods to run on, this is called a "placement"
 - Placement: a partial, non-injective assignment of a set of Pods to a set of Nodes.



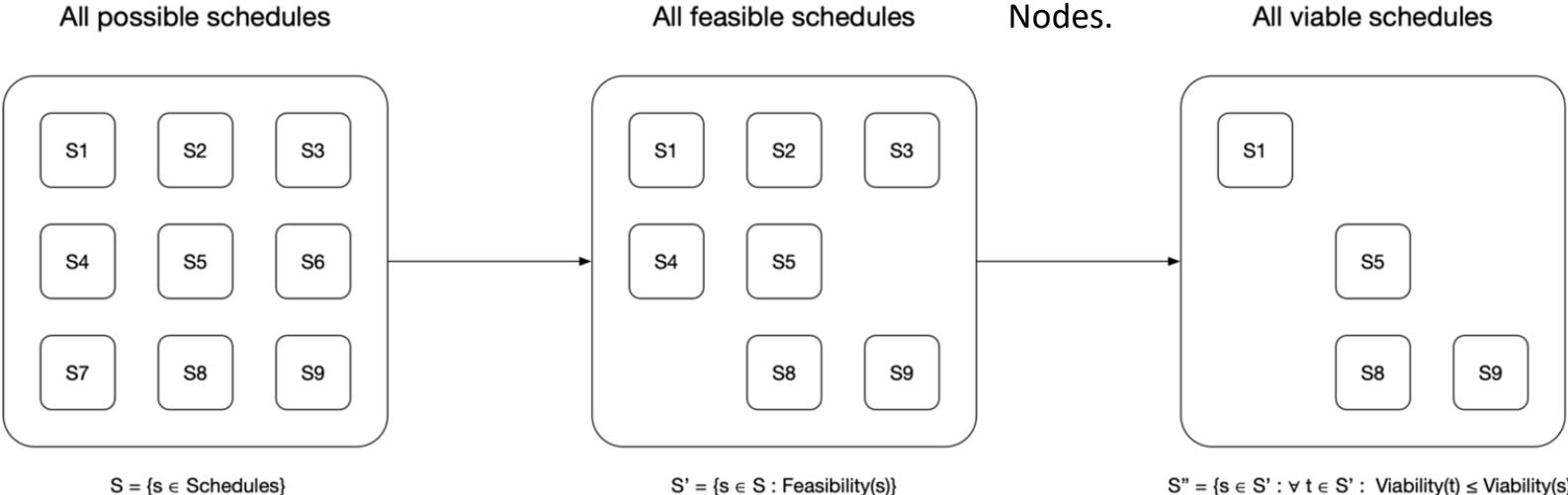
Scheduler details

- **Scheduler (`kube-scheduler`)**: selects nodes for newly created pods to run on, this is called a "placement"
 - Placement: a partial, non-injective assignment of a set of Pods to a set of Nodes.



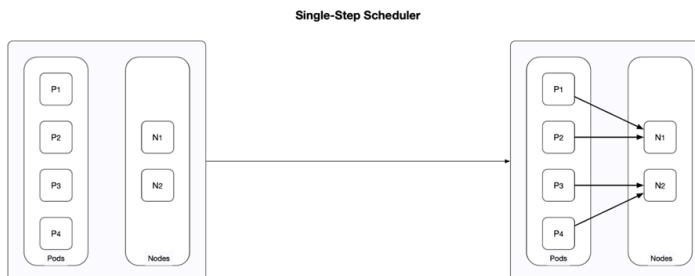
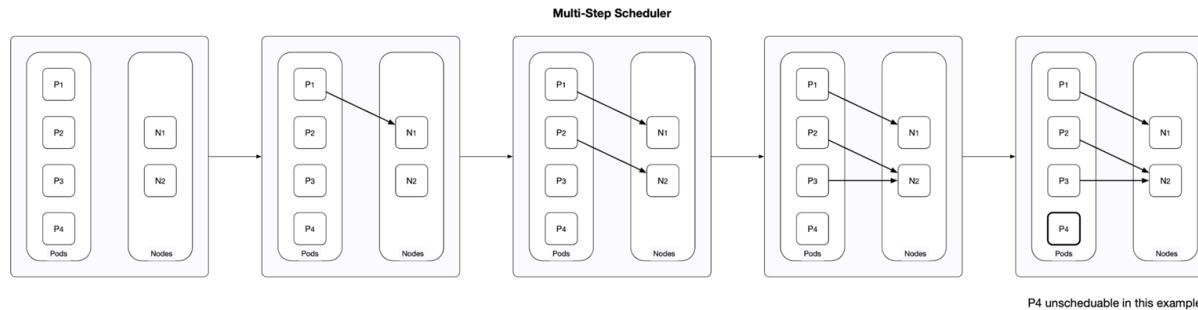
Scheduler decisions

- **Scheduler (kube-scheduler)**: selects nodes for newly created pods to run on, this is called a "placement"
 - Placement: a partial, non-injective assignment of a set of Pods to a set of Nodes.



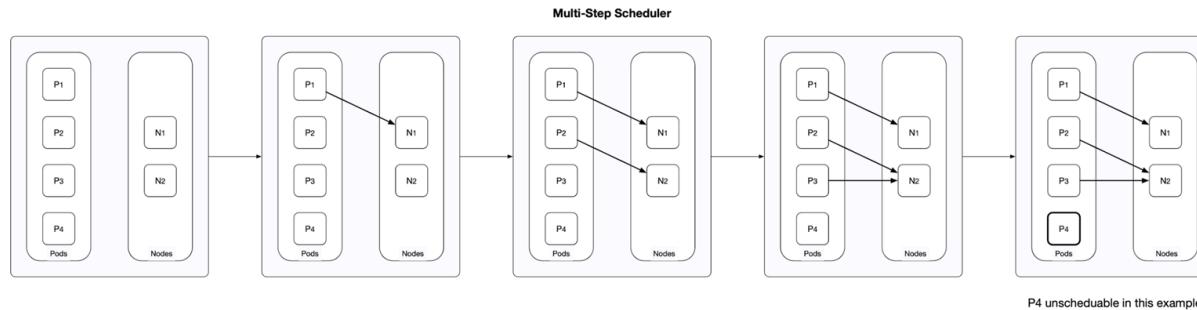
Scheduler decisions

- **Multi-step scheduler**
 - local optimum instead of global optimum

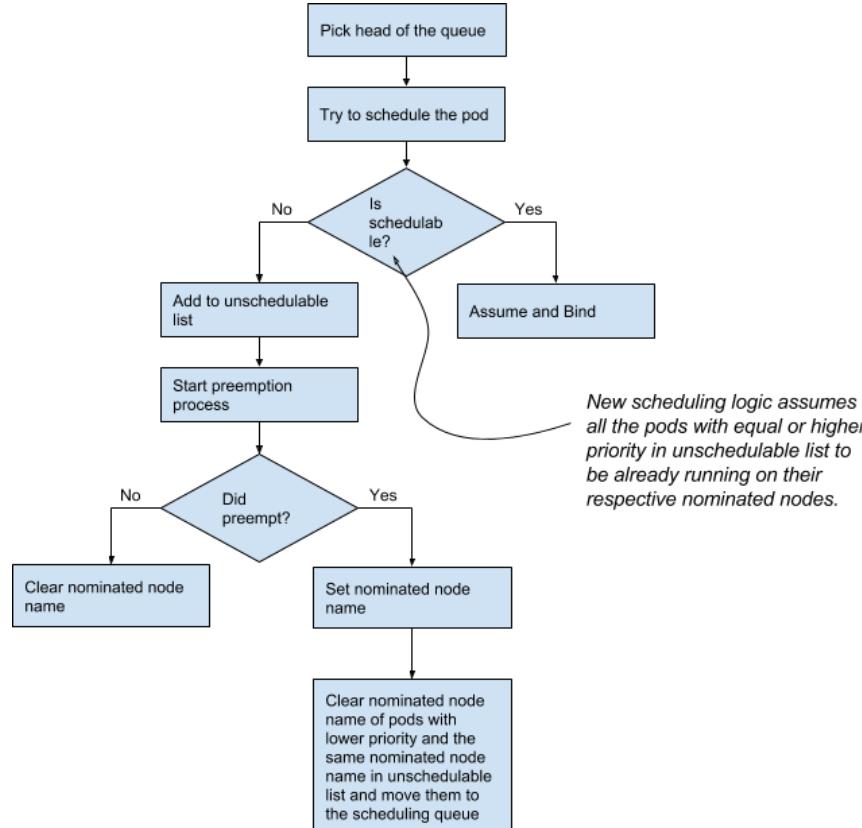


Scheduler decisions

- **Multi-step scheduler**
 - local optimum instead of global optimum



Scheduler preemption

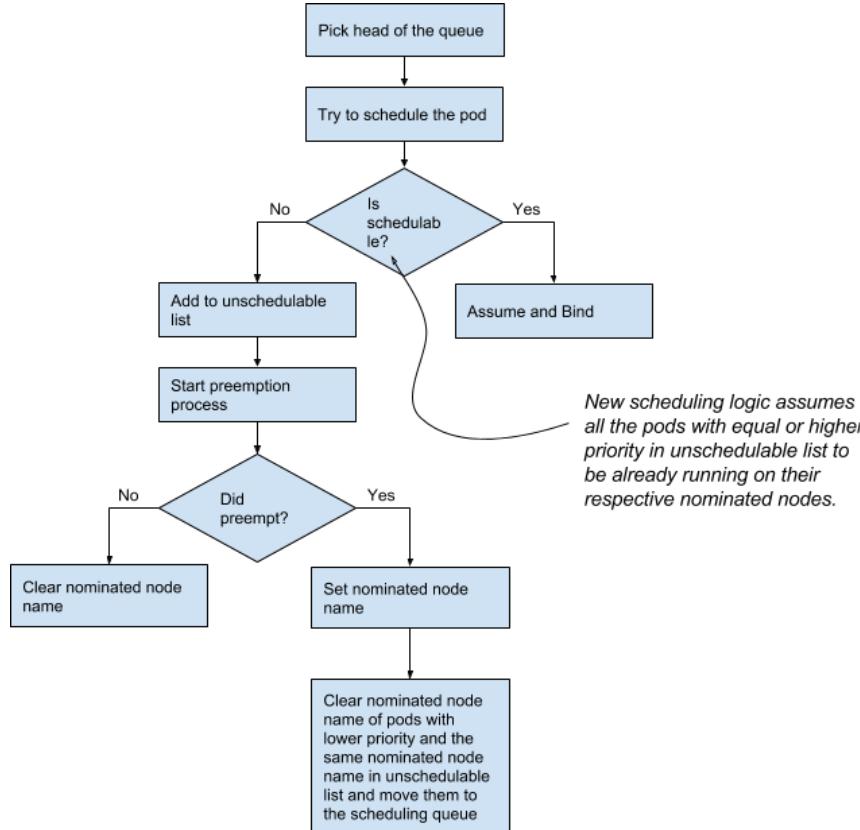


- **Preemption**

- Lower priority Pods are destroyed to free up resources for higher priority Pods.

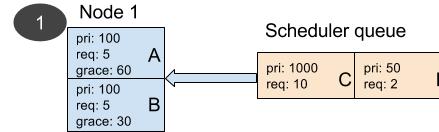
New scheduling logic assumes all the pods with equal or higher priority in unschedulable list to be already running on their respective nominated nodes.

Scheduler preemption



- **Example**

- 1 node in the cluster with capacity 10 units
- 2 pods (A,B) running on the node with priority 100 and each using 5 units
- Scheduler has 2 pods (C,D) in queue. Pod C needs 10 units and priority is 1000. Pod D needs 2 units and it's priority is 50
- Scheduler determines that Pod C has highest priority and destroys (A,B) so it can schedule.



Built-in Node Labels

- kubernetes.io/hostname
- failure-domain.beta.kubernetes.io/zone
- failure-domain.beta.kubernetes.io/region
- beta.kubernetes.io/instance-type
- beta.kubernetes.io/os
- beta.kubernetes.io/arch

Operators

Valid operators

- In
- NotIn
- Exists
- DoesNotExist
- Gt (Greater Than)
- Lt (Less Than)

Node Taints

Allows you to mark (“taint”) a node

- No pods can be scheduled onto tainted nodes
- Useful when all or most PODs should not be scheduled on a node
- Mark your master node as schedulable only by Kubernetes system components
- Keep regular pods away from nodes that have special hardware so as to leave room for pods that need the special hardware.

Node Tolerations

To allow a POD to be scheduled onto a ‘tainted’ node it must have:

```
tolerations:  
- key: "key"  
  operator: "Equal"  
  value: "value"  
  effect: "NoSchedule"
```

Pod Affinity

- Define how pods should be placed relative to one another
- Spread or pack pods within a service or relative to pods in other services

```
affinity:  
  podAffinity:  
    requiredDuringSchedulingIgnoredDuringExecution:  
      - labelSelector:  
          matchExpressions:  
            - key: service  
              operator: In  
              values: ["S1"]  
        topologyKey: failure-domain.beta.kubernetes.io/zone
```

Pod Affinity

- Define how pods should be placed relative to one another
- Spread or pack pods within a service or relative to pods in other services

```
affinity:  
  podAffinity:  
    preferredDuringSchedulingIgnoredDuringExecution:  
      - labelSelector:  
          matchExpressions:  
            - key: service  
              operator: In  
              values: ["S1"]  
        topologyKey: failure-domain.beta.kubernetes.io/zone
```

Pod Affinity

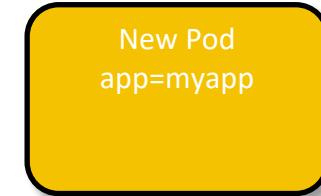
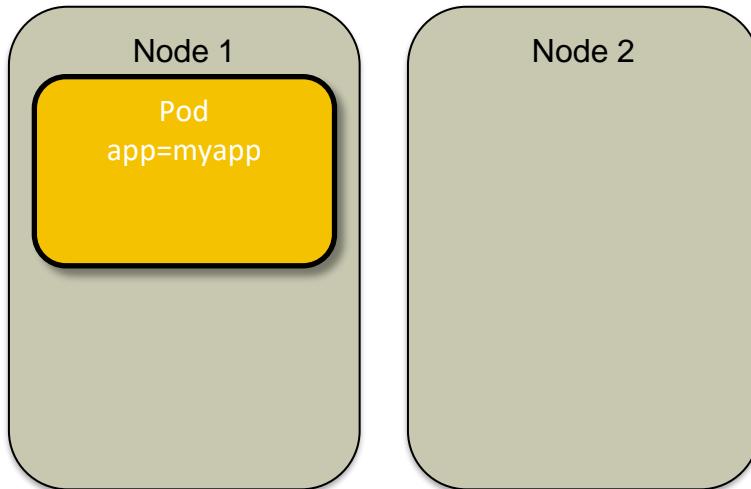
- Define how pods should be placed relative to one another
- Spread or pack pods within a service or relative to pods in other services

```
affinity:  
  podAffinity:  
    preferredDuringSchedulingIgnoredDuringExecution:  
      - labelSelector:  
          matchExpressions:  
            - key: service  
              operator: In  
              values: ["S1"]  
        topologyKey: failure-domain.beta.kubernetes.io/zone
```

Pod Affinity

- Schedule Pods onto nodes that have same labels

- app=myapp
- operator: In
- kubernetes.io/hostname

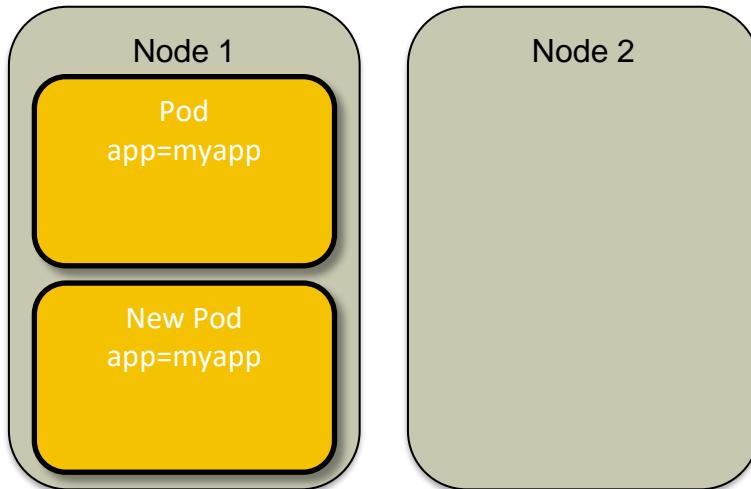


```
affinity:  
podAffinity:  
requiredDuringSchedulingIgnoredDuringExecution:  
- labelSelector:  
  matchExpressions:  
  - key: "app"  
    operator: In  
    values:  
    - myapp  
topologyKey: "kubernetes.io/hostname"
```

Pod Affinity

- Schedule Pods onto nodes that have same labels

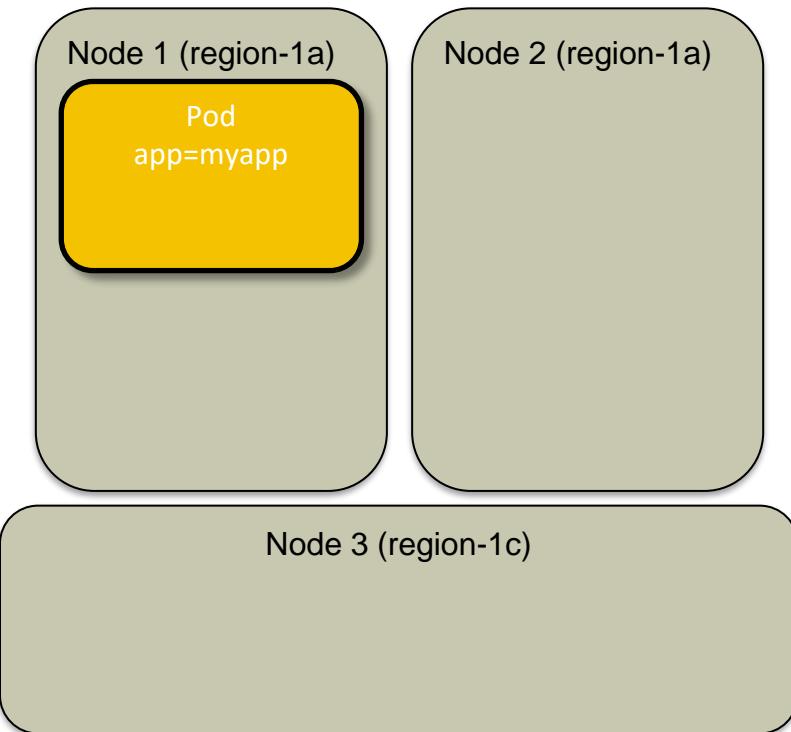
- app=myapp
- operator: In
- kubernetes.io/hostname



```
affinity:  
podAffinity:  
requiredDuringSchedulingIgnoredDuringExecution:  
- labelSelector:  
  matchExpressions:  
  - key: "app"  
    operator: In  
    values:  
    - myapp  
topologyKey: "kubernetes.io/hostname"
```

Pod Affinity

- **Schedule Pods onto nodes in same region**
 - app=db
 - failure-domain.beta.kubernetes.io/zone



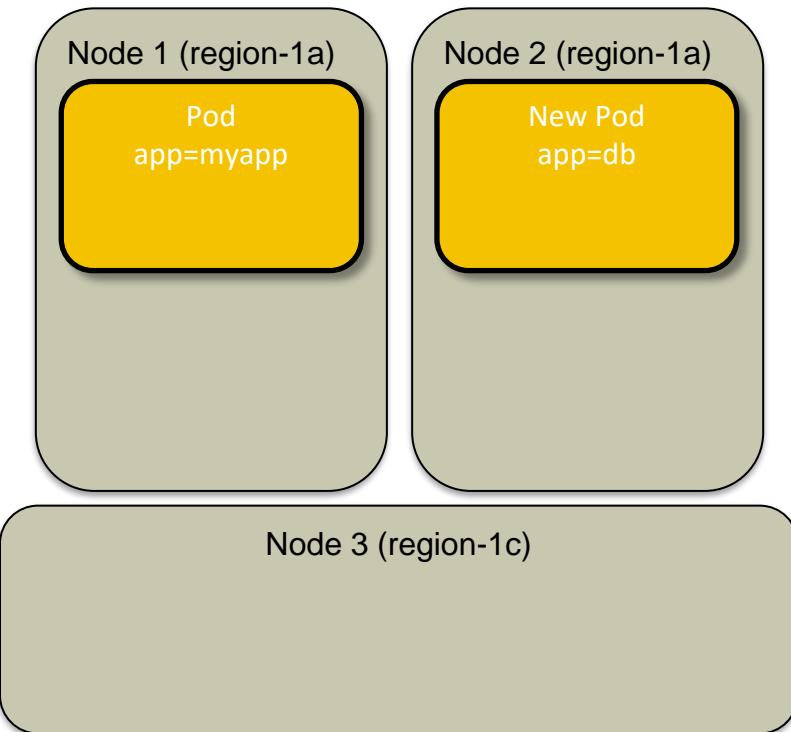
New Pod
app=db

```
affinity:  
podAffinity:  
  requiredDuringSchedulingIgnoredDuringExecution:  
    - labelSelector:  
        matchExpressions:  
          - key: "app"  
            operator: In  
            values:  
              - myapp  
    topologyKey: "failure-domain.beta.kubernetes.io/zone"
```

Pod Affinity

- **Schedule Pods onto nodes in same region**

- app=db
- failure-domain.beta.kubernetes.io/zone

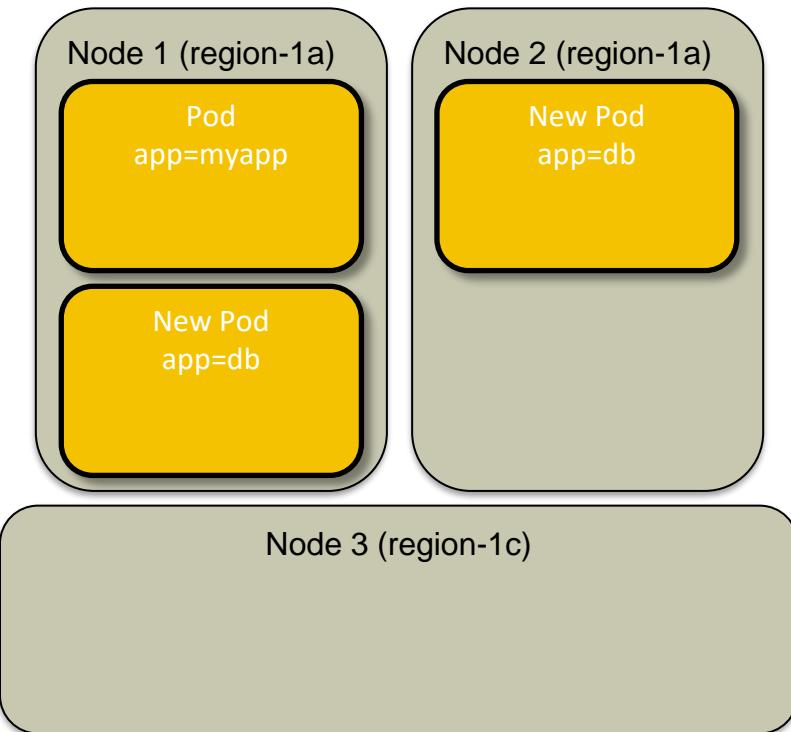


```
affinity:  
podAffinity:  
  requiredDuringSchedulingIgnoredDuringExecution:  
    - labelSelector:  
        matchExpressions:  
          - key: "app"  
            operator: In  
            values:  
              - myapp  
  topologyKey: "failure-domain.beta.kubernetes.io/zone"
```

Pod Affinity

- **Schedule Pods onto nodes in same region**

- app=db
- failure-domain.beta.kubernetes.io/zone



```
affinity:  
podAffinity:  
  requiredDuringSchedulingIgnoredDuringExecution:  
    - labelSelector:  
        matchExpressions:  
          - key: "app"  
            operator: In  
            values:  
              - myapp  
  topologyKey: "failure-domain.beta.kubernetes.io/zone"
```

Pod Anti-Affinity

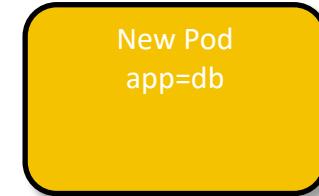
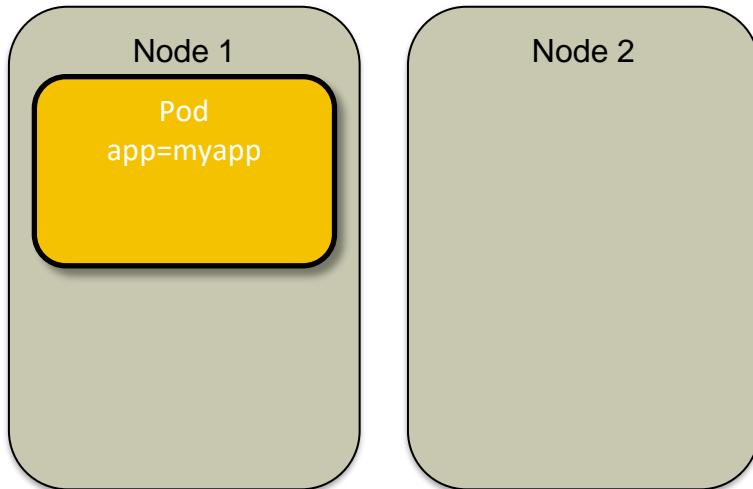
- Define how pods should be placed relative to one another
- Pods of different services run on different nodes.

```
affinity:  
  podAntiAffinity:  
    preferredDuringSchedulingIgnoredDuringExecution:  
      - labelSelector:  
          matchExpressions:  
            - key: service  
              operator: In  
              values: ["S1"]  
        topologyKey: kubernetes.io/hostname
```

Pod Anti-Affinity

- Do not schedule Pods onto nodes with matching labels.

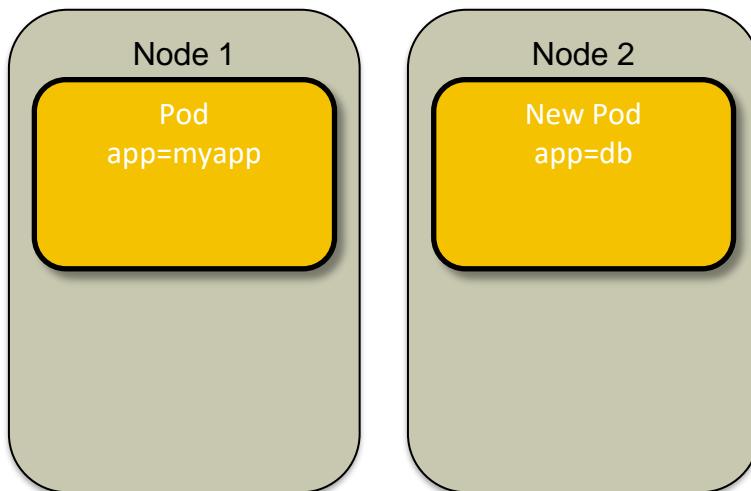
- app=db
- operator: In
- kubernetes.io/hostname



```
affinity:  
podAntiAffinity:  
requiredDuringSchedulingIgnoredDuringExecution:  
- labelSelector:  
  matchExpressions:  
  - key: "app"  
    operator: In  
    values:  
    - myapp  
topologyKey: "kubernetes.io/hostname"
```

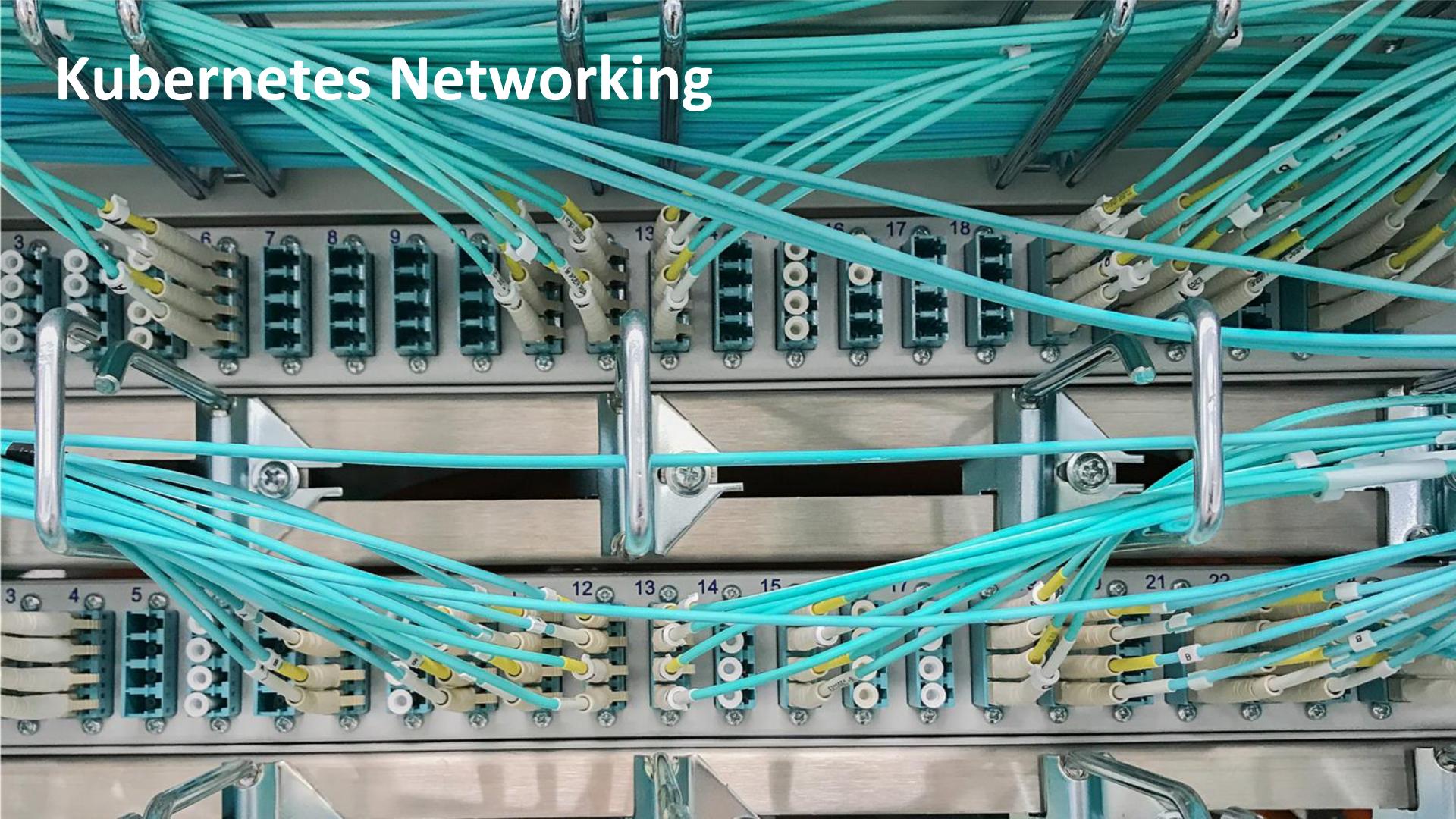
Pod Anti-Affinity

- Do not schedule Pods onto nodes with matching labels.
 - app=db
 - operator: In
 - kubernetes.io/hostname



```
affinity:  
podAntiAffinity:  
requiredDuringSchedulingIgnoredDuringExecution:  
- labelSelector:  
  matchExpressions:  
  - key: "app"  
    operator: In  
    values:  
    - myapp  
topologyKey: "kubernetes.io/hostname"
```

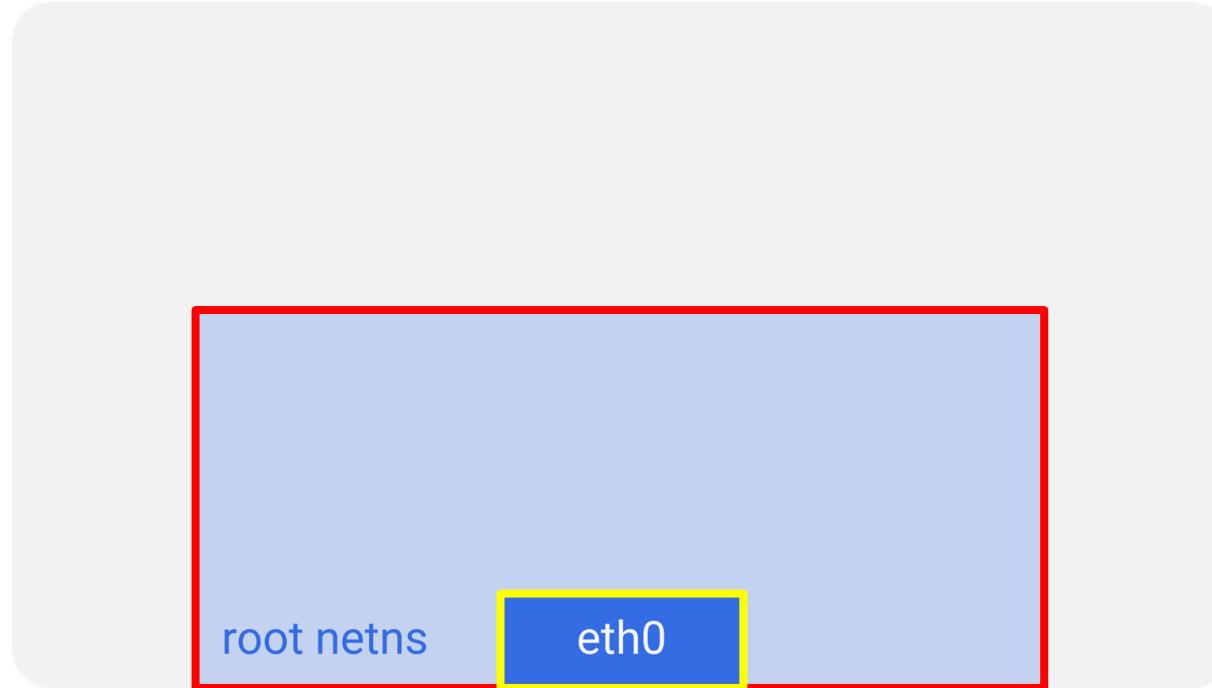
Kubernetes Networking



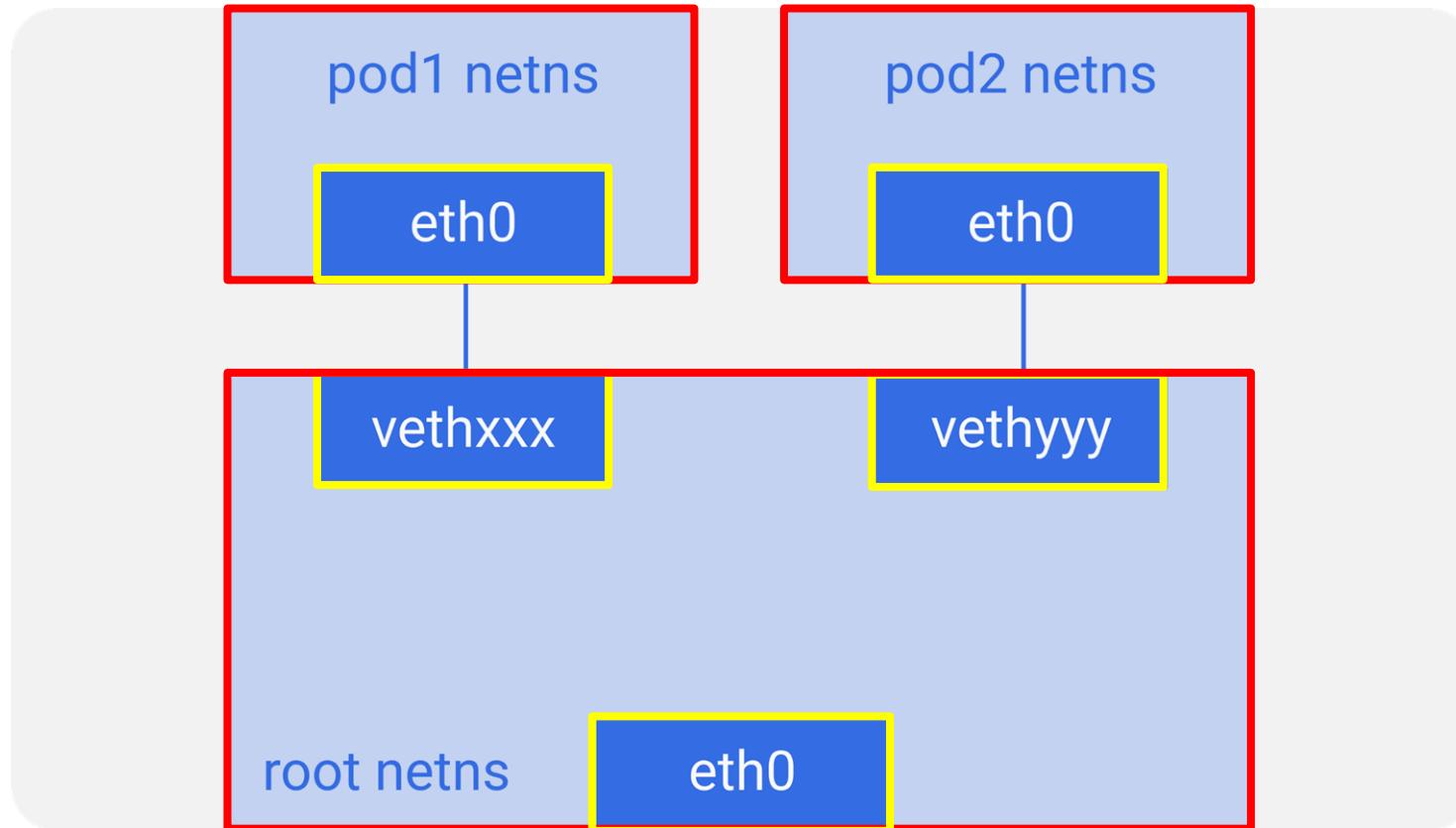
Pod Networking Approaches

- **Using an overlay network**
 - An overlay network obscures the underlying network architecture from the pod network through traffic encapsulation (for example vxlan).
 - Encapsulation reduces performance, though exactly how much depends on your solution.
- **Without an overlay network**
 - Configure the underlying network fabric (switches, routers, etc.) to be aware of pod IP addresses.
 - This does not require the encapsulation provided by an overlay, and so can achieve better performance.

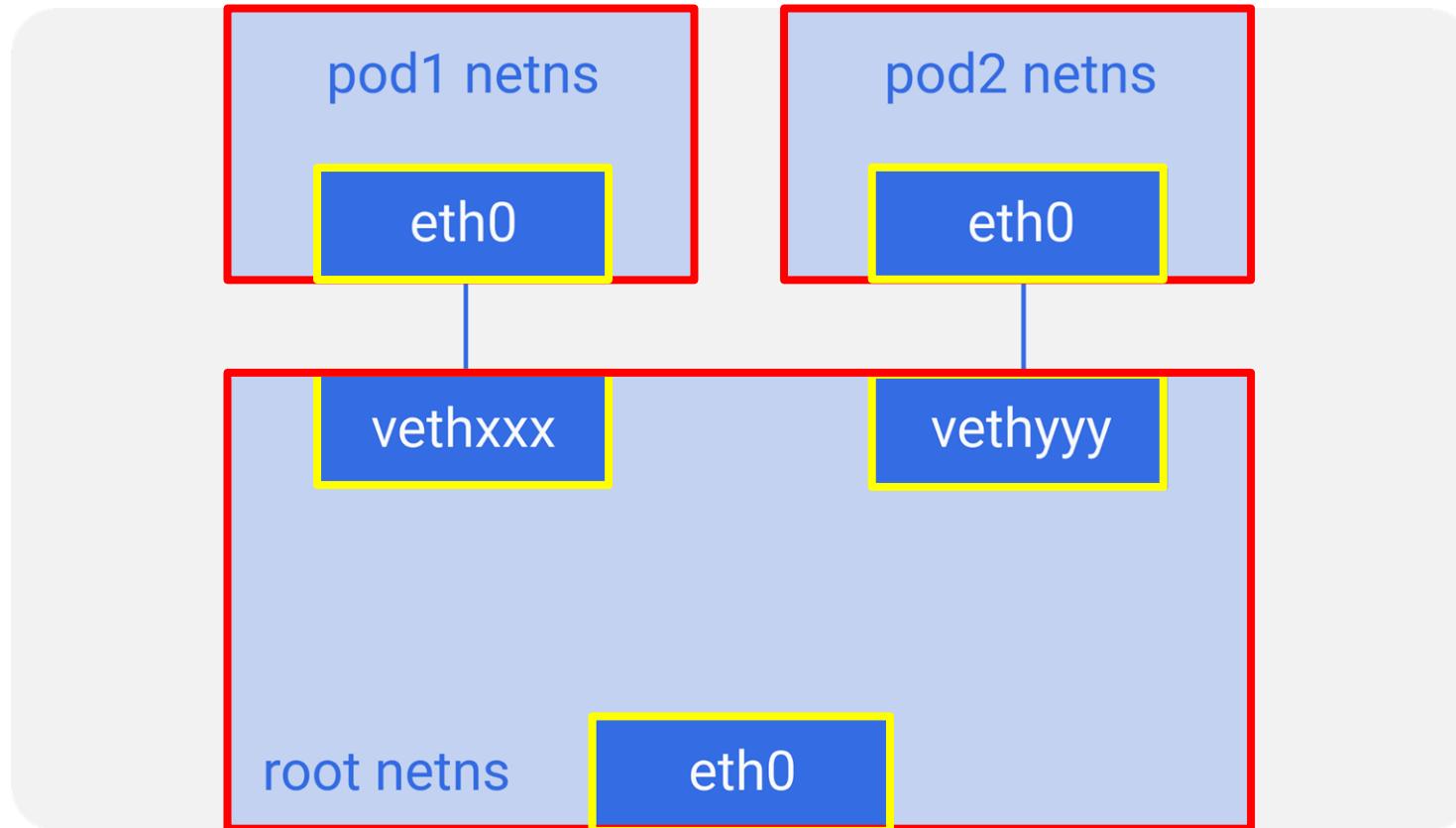
Node Namespace



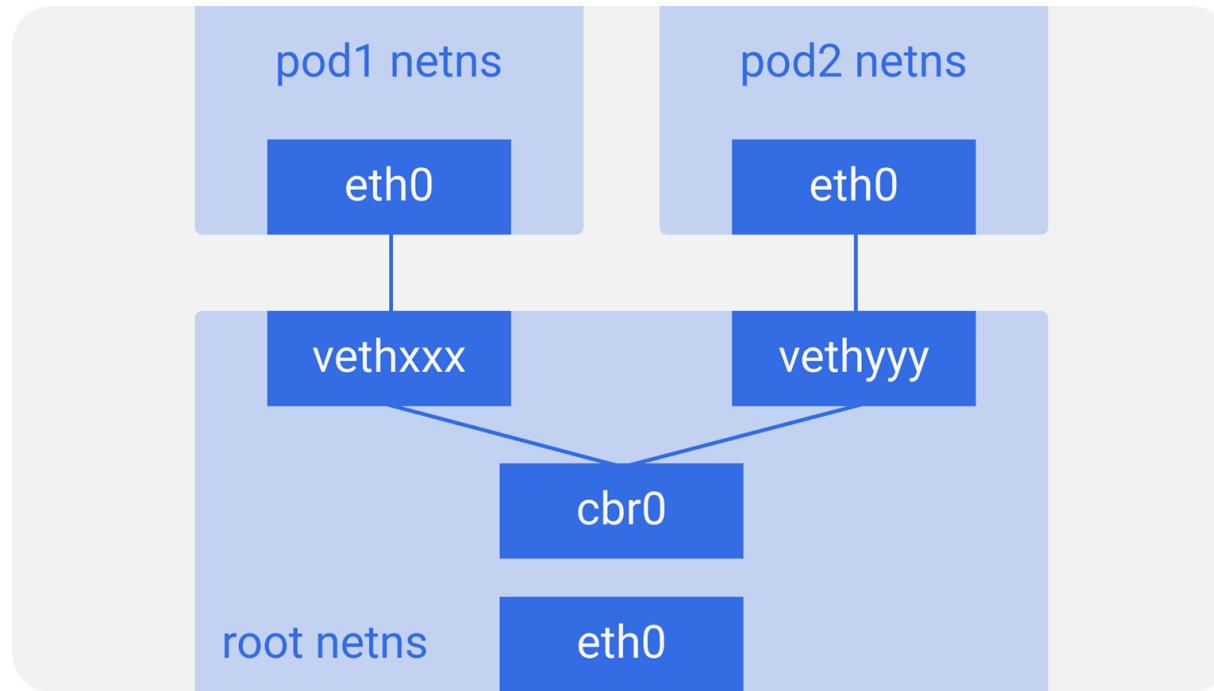
Node Namespaces



Node Namespaces

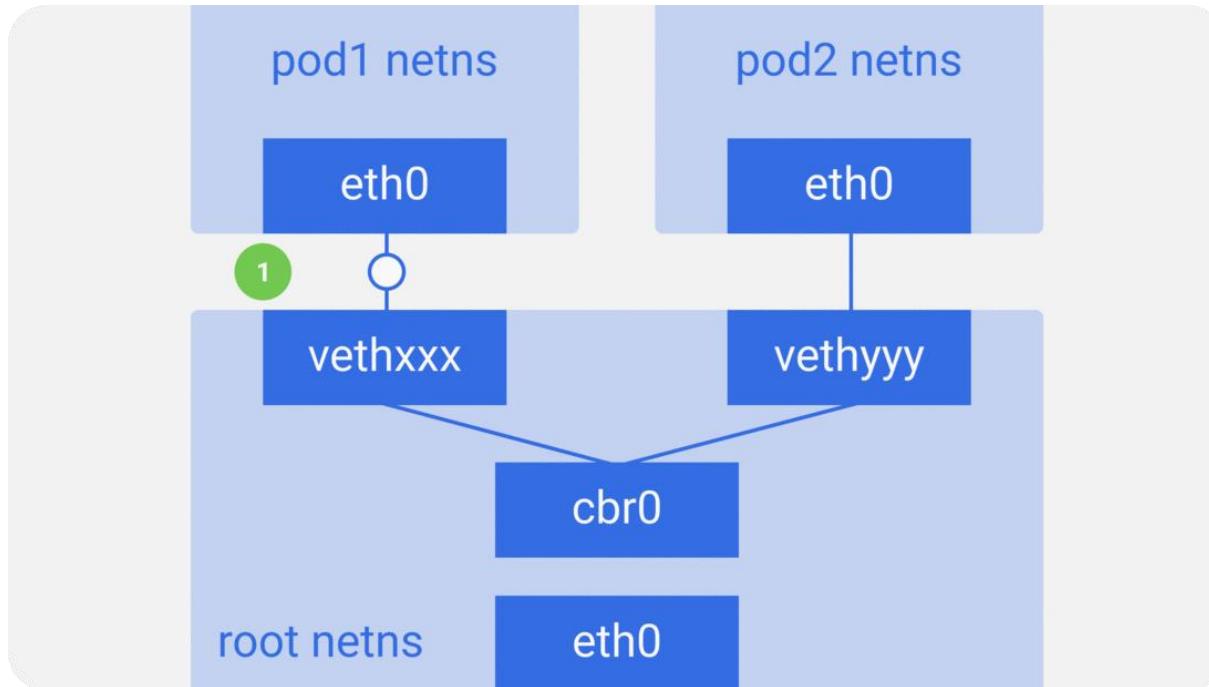


Pod to Host



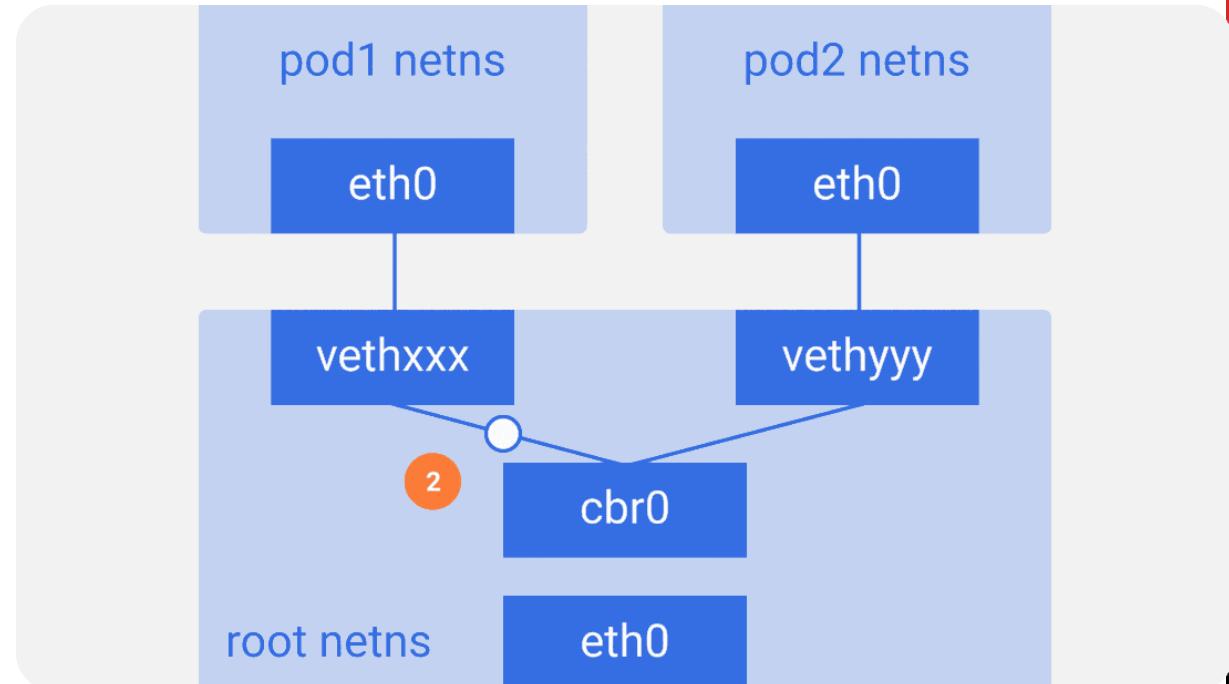
Single Host POD to POD Network

1. Packet leaves pod1's netns at eth0 and enters the root netns at vethxxx.



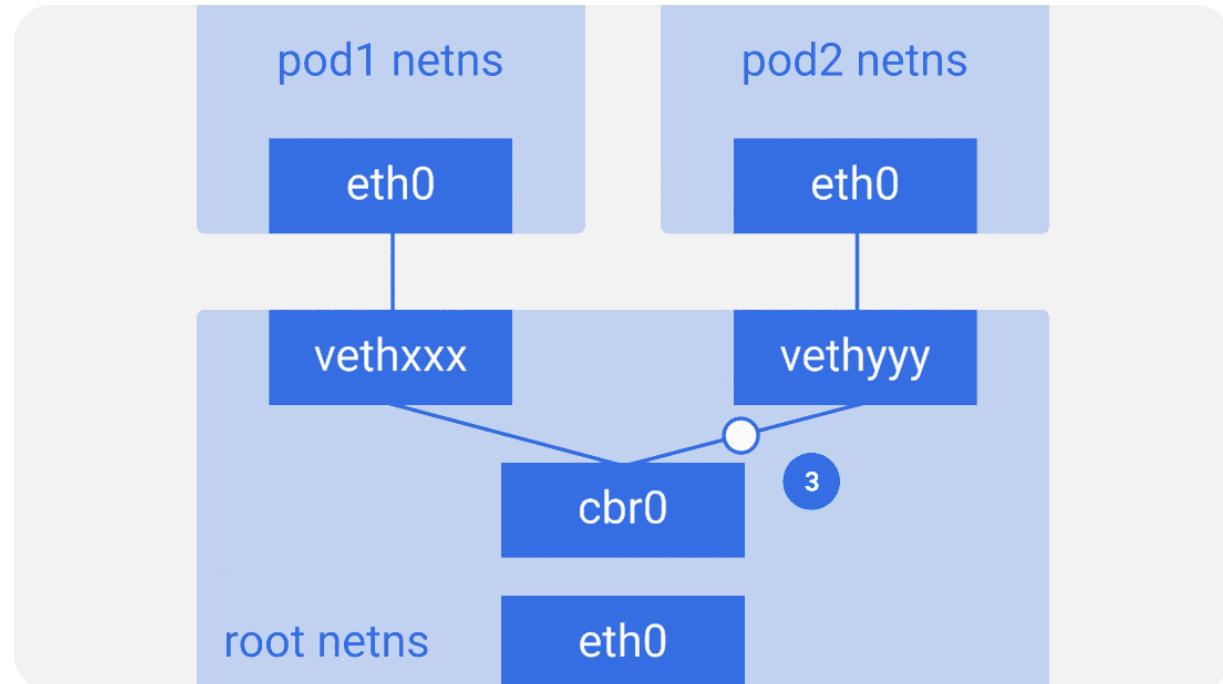
Single Host POD to POD Networking

2. It's passed on to cbr0, which discovers the destination using an ARP request, saying "who has this IP?"



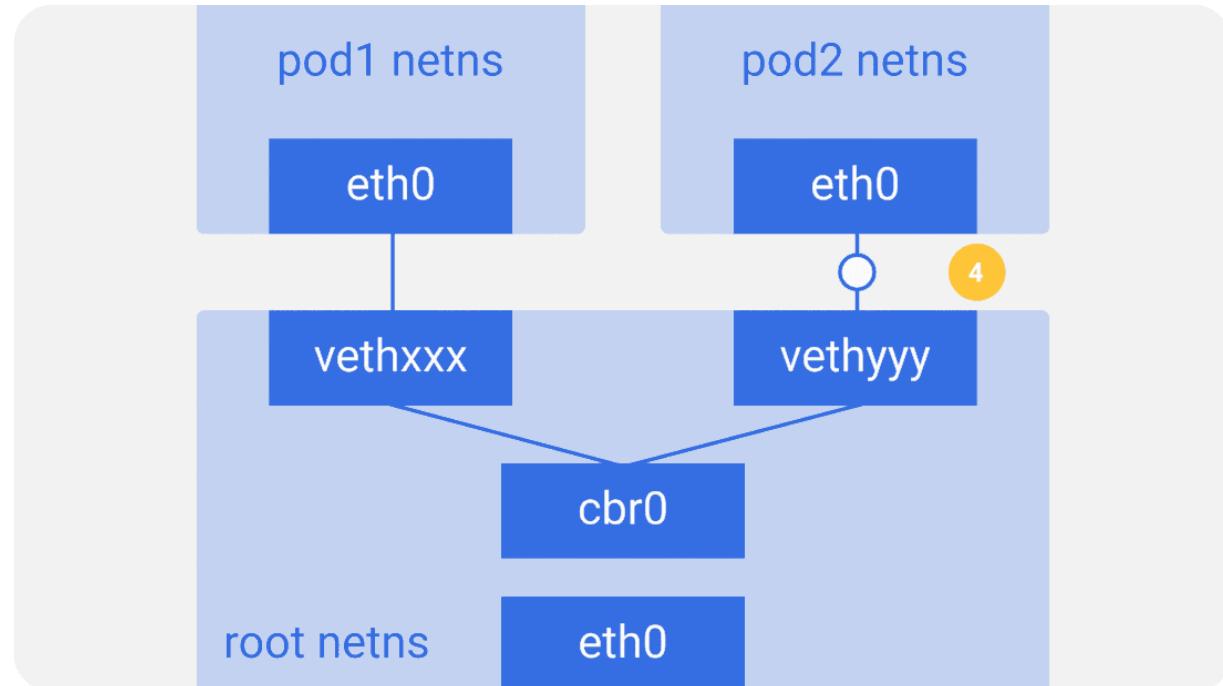
Single Host POD to POD Networking

3. vethyyy says it has that IP, so the bridge knows where to forward the packet.



Single Host POD to POD Networking

4. The packet reaches vethyyy, crosses the pipe-pair and reaches pod2's netns.



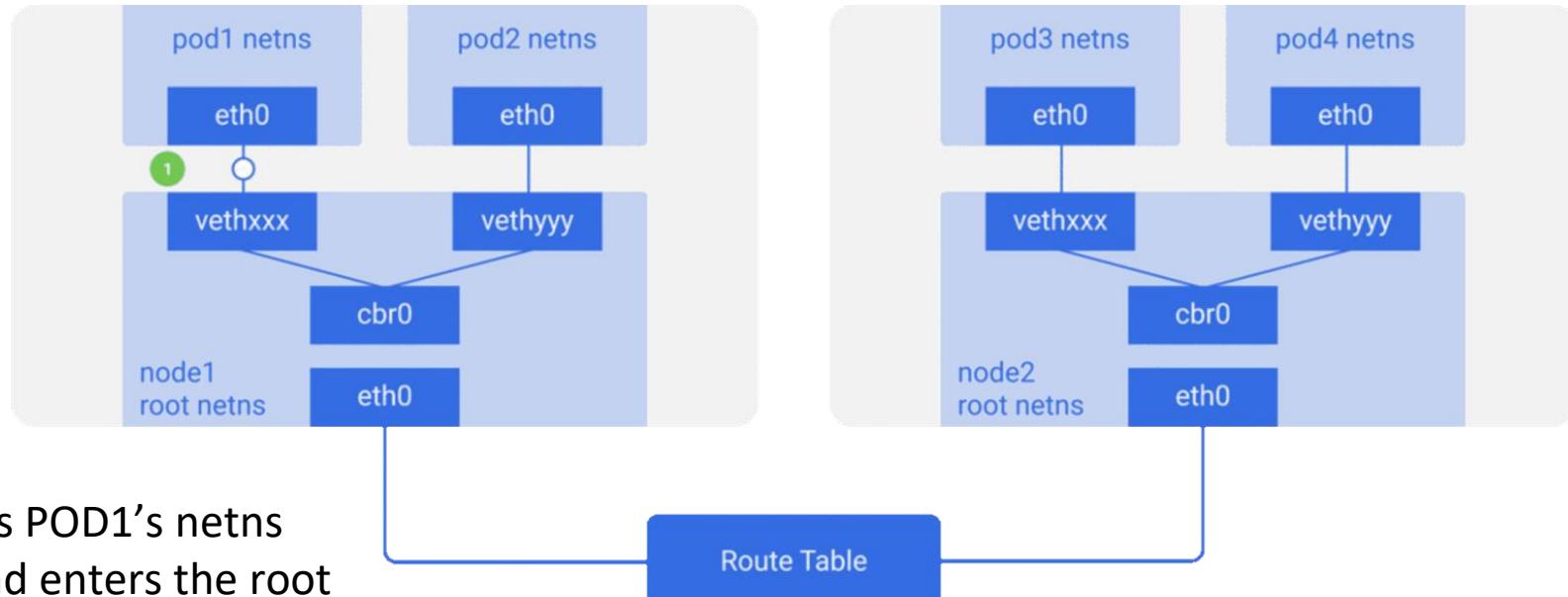
K8S Node Deep Dive



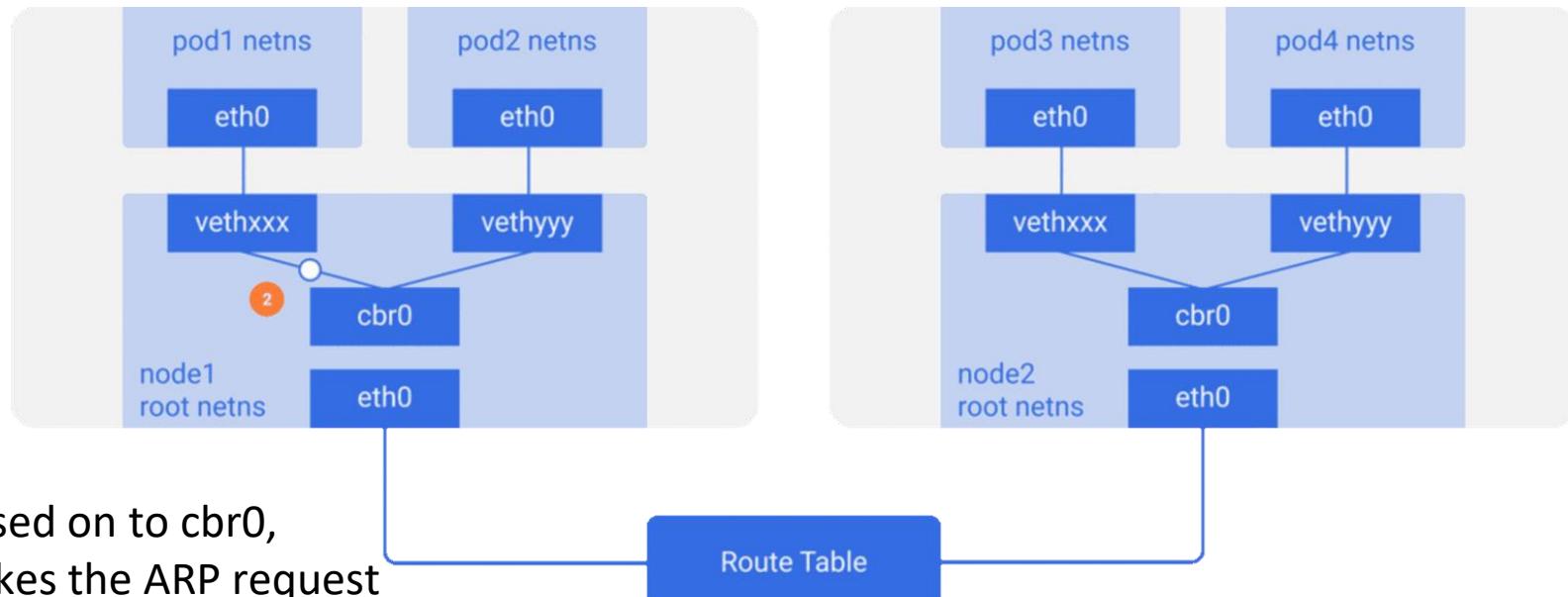
Inter-node Communication

- Pods must be reachable across nodes
- Kubernetes doesn't care how:
 - L2 (ARP across nodes)
 - L3 (IP routing across nodes)
 - Overlay networks

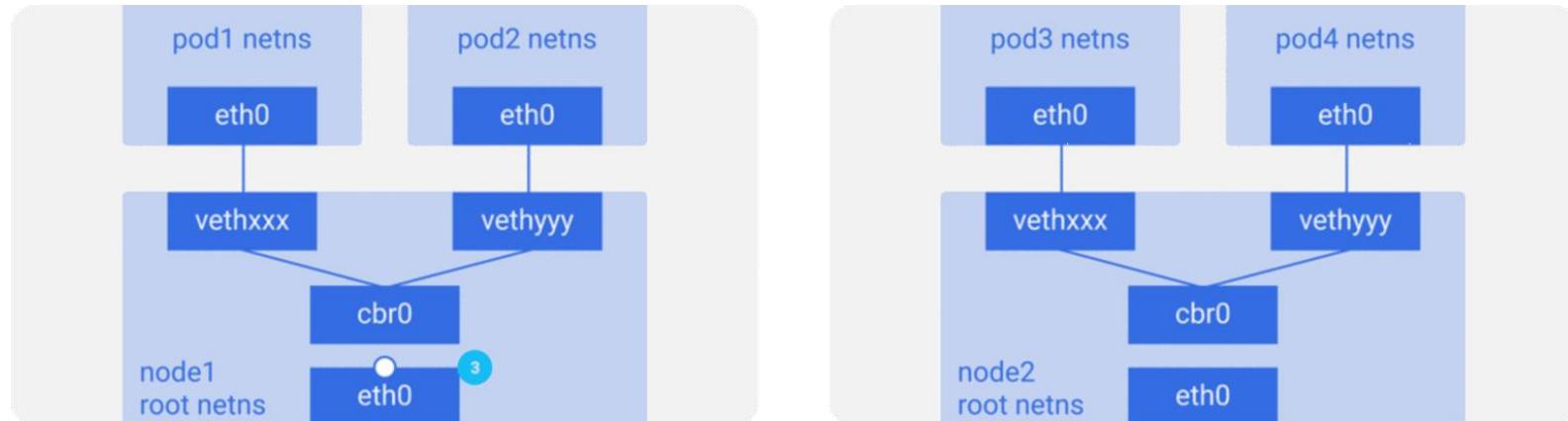
Inter-node Communication



Inter-node Communication

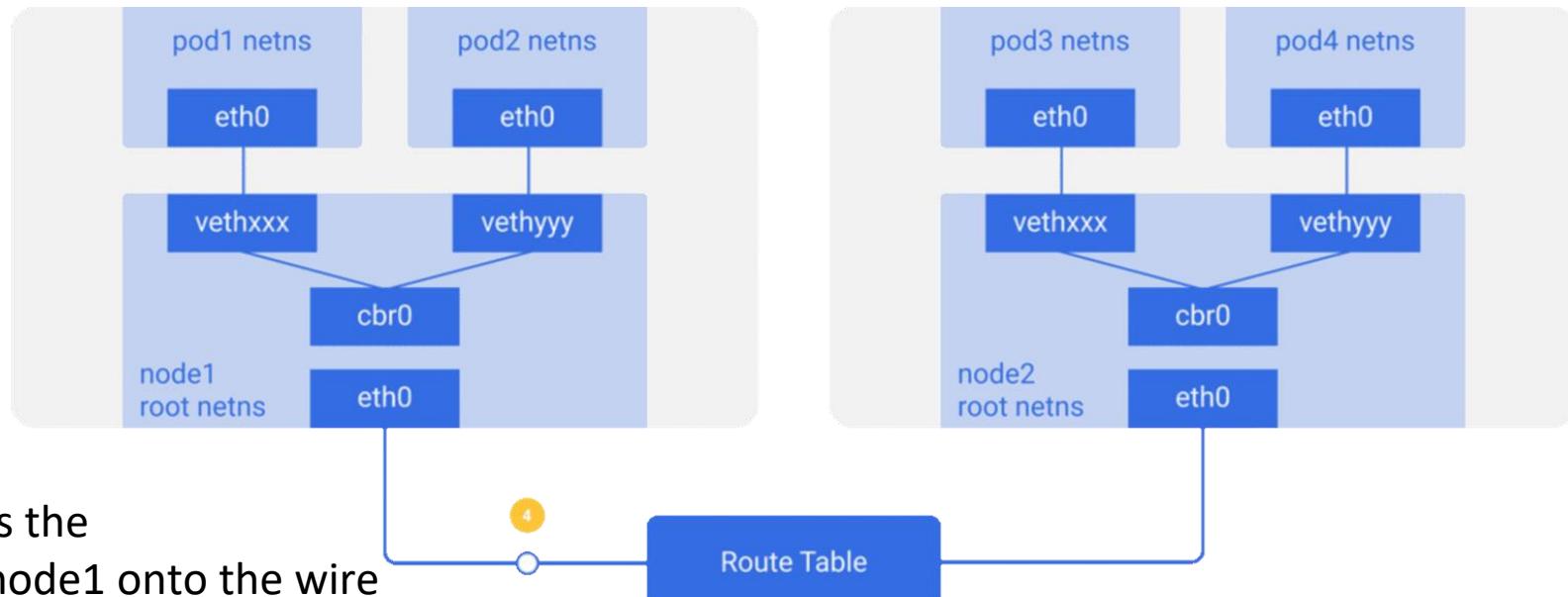


Inter-node Communication

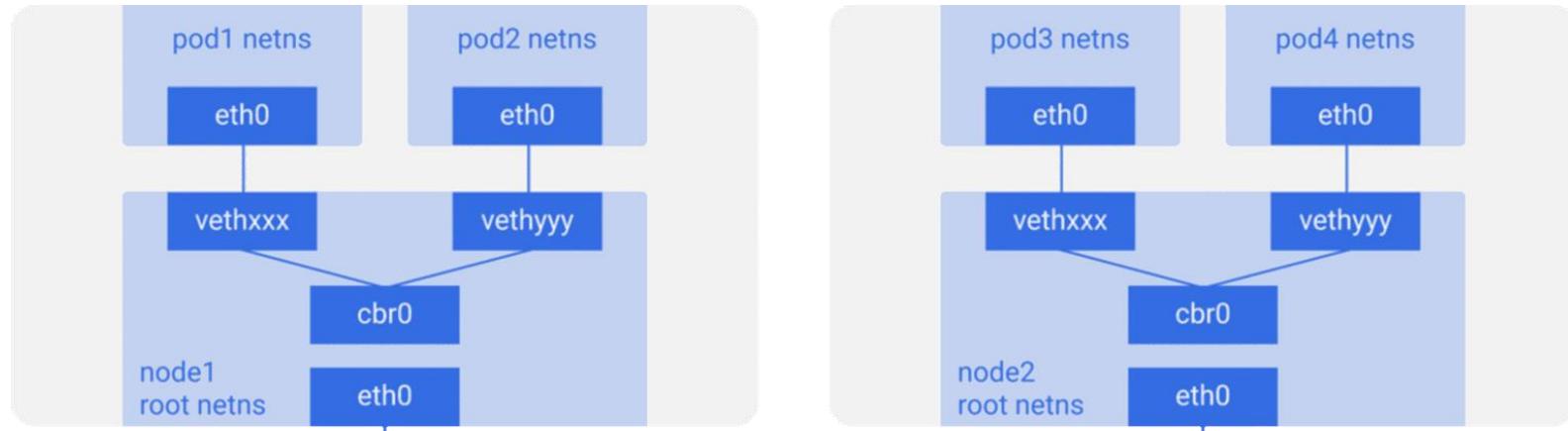


3. It comes out of cbr0 to the main network interface eth0 since nobody on this node has the IP address for POD4.

Inter-node Communication

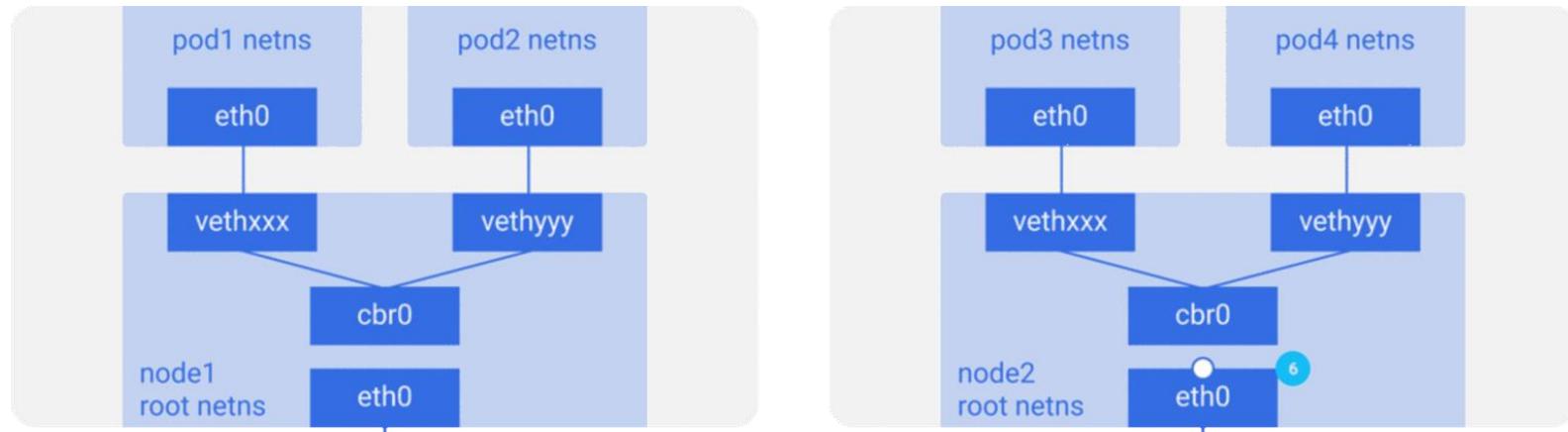


Inter-node Communication



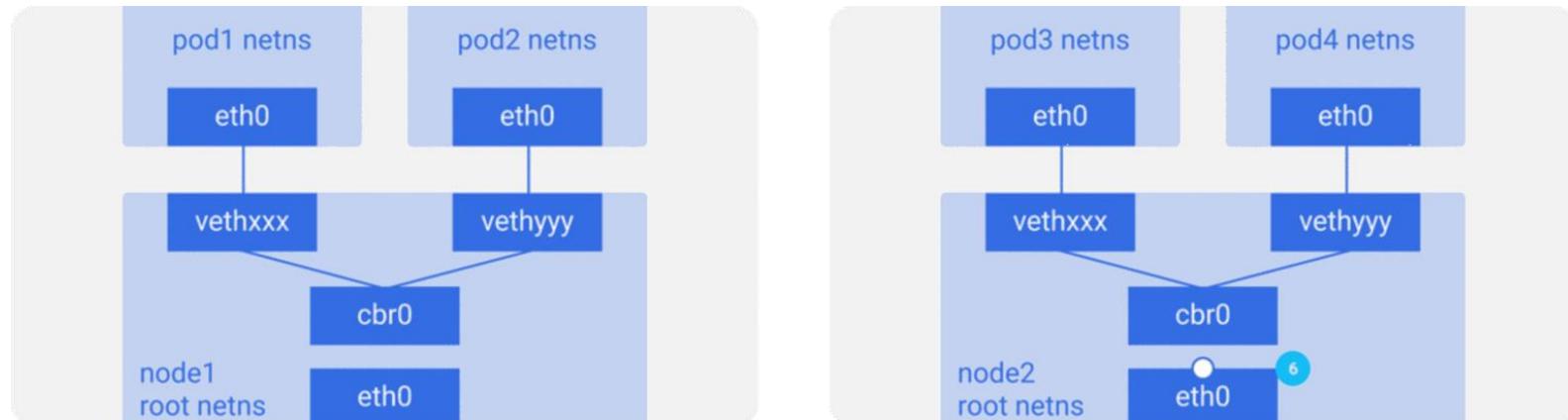
5. The route table has routes setup for each of the node CIDR blocks, and it routes the packet to the node whose CIDR block contains the POD4 IP.

Inter-node Communication



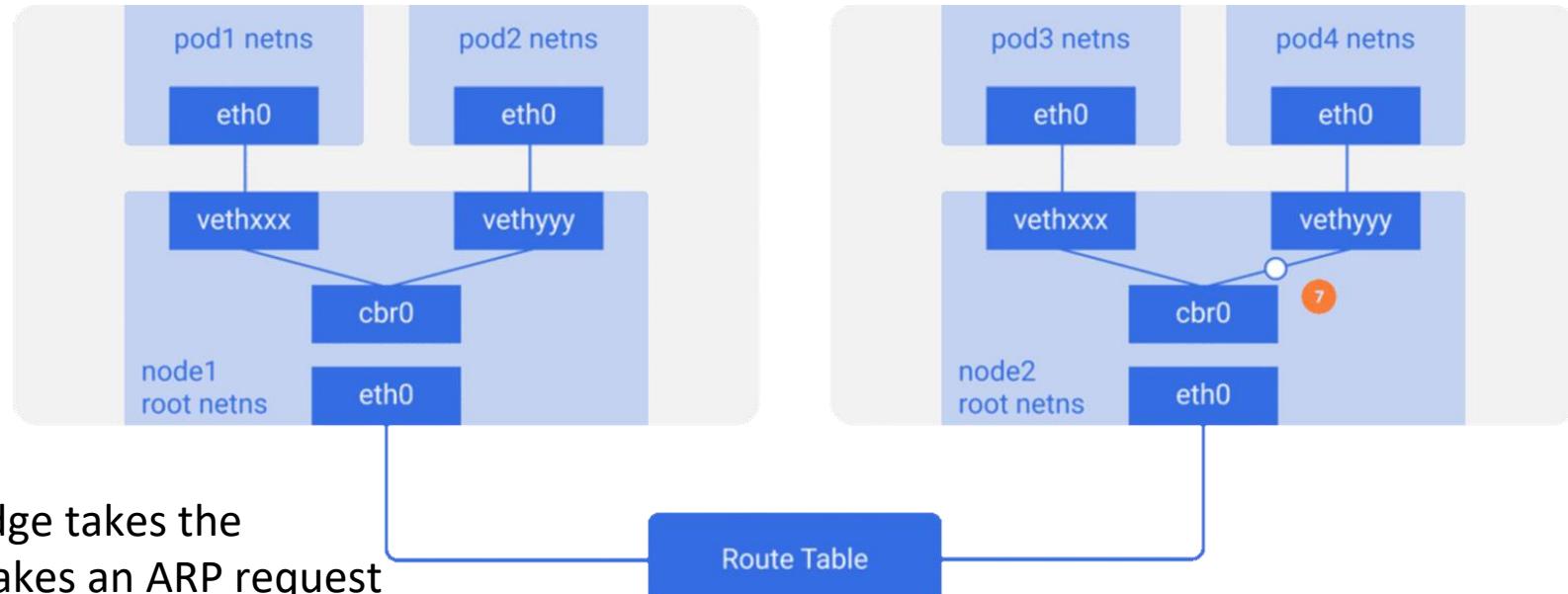
6. Packet arrives at node2 at `eth0`. POD4 isn't the IP of `eth0`, the packet is forwarded to `cbr0`, nodes are configured with IP forwarding enabled.

Inter-node Communication



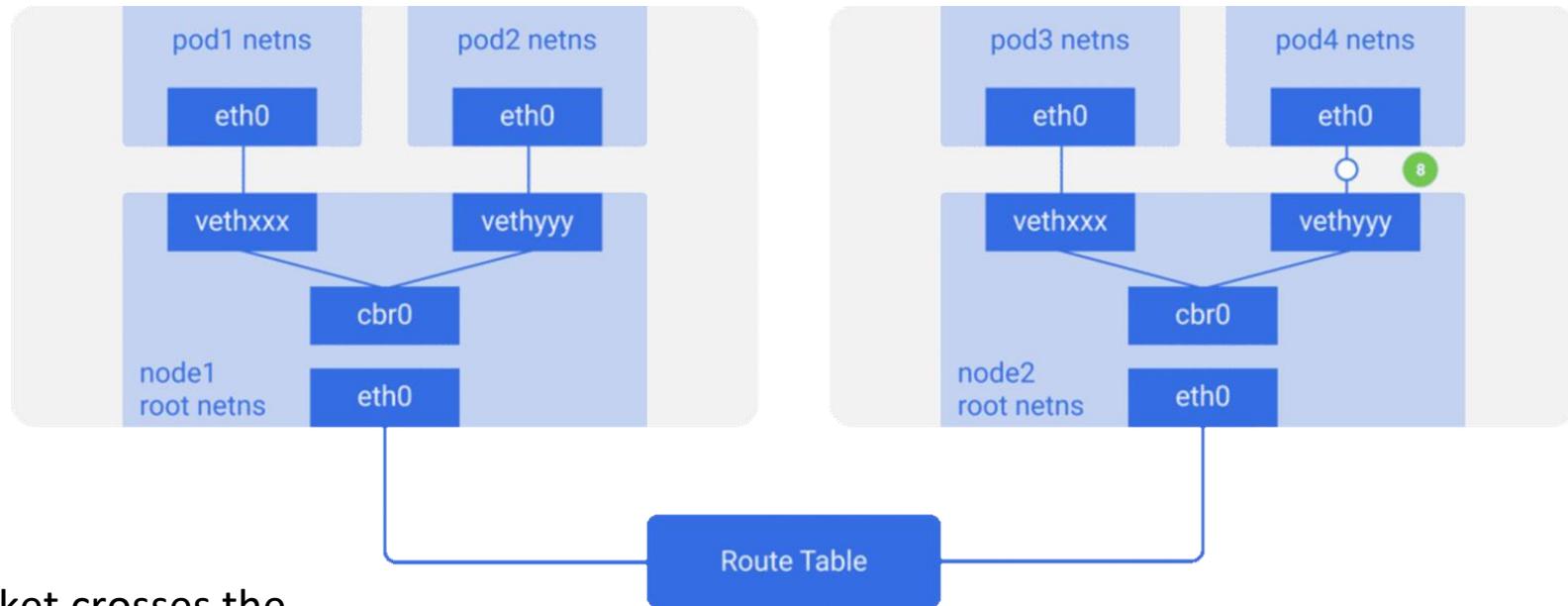
6. The node's routing table is looked up for any routes matching the POD4 IP. It finds cbr0 as the destination for this node's CIDR block.

Inter-node Communication



7. The bridge takes the packet, makes an ARP request and finds out that the IP belongs to vethyyy.

Inter-node Communication



8. The packet crosses the pipe-pair and reaches POD4

ReplicaSets



Reference

Develop a passion for
learning.

© 2022 Innovation in Software (2022)

SITE: MEDIUM.COM

<https://medium.com/search?q=azure>

Medium.com is a website with comprehensive technical content for developers. The content is detailed and with plenty of examples. Great online resource for Azure data engineering content.

BOOK: ANTIFRAGILE SOFTWARE

<https://leanpub.com/antifragilesoftware>

© 2022 by Innovation In Software Corporation

Written by Russ Miles with
Sylvain Hellegoarch, and
Grant Tarrant-Fisher.

These are the folks who have
driven a boatload of the
progress in Chaos
Engineering in the last few
years.