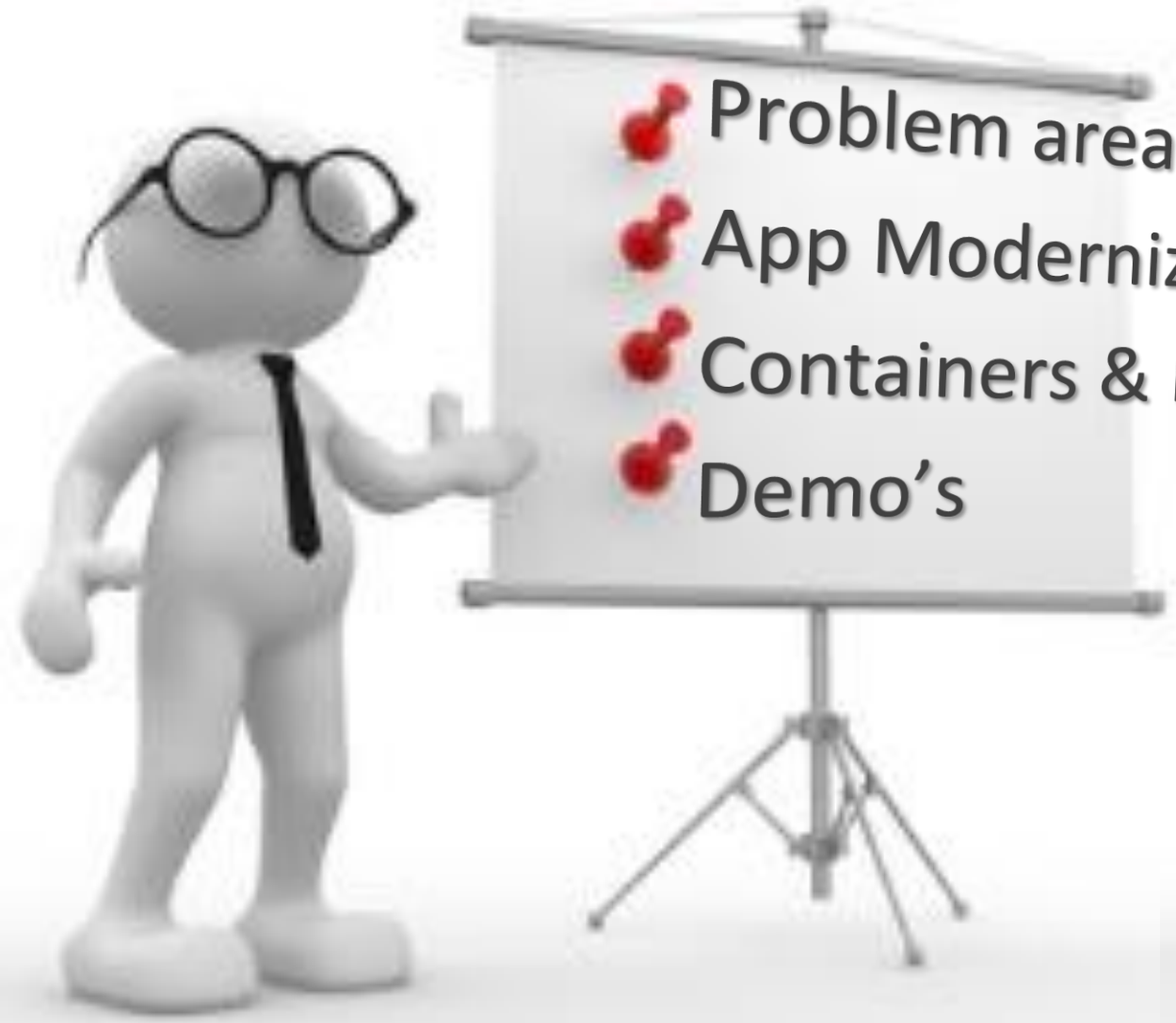
Managed Kubernetes in Azure

Even voorstellen…

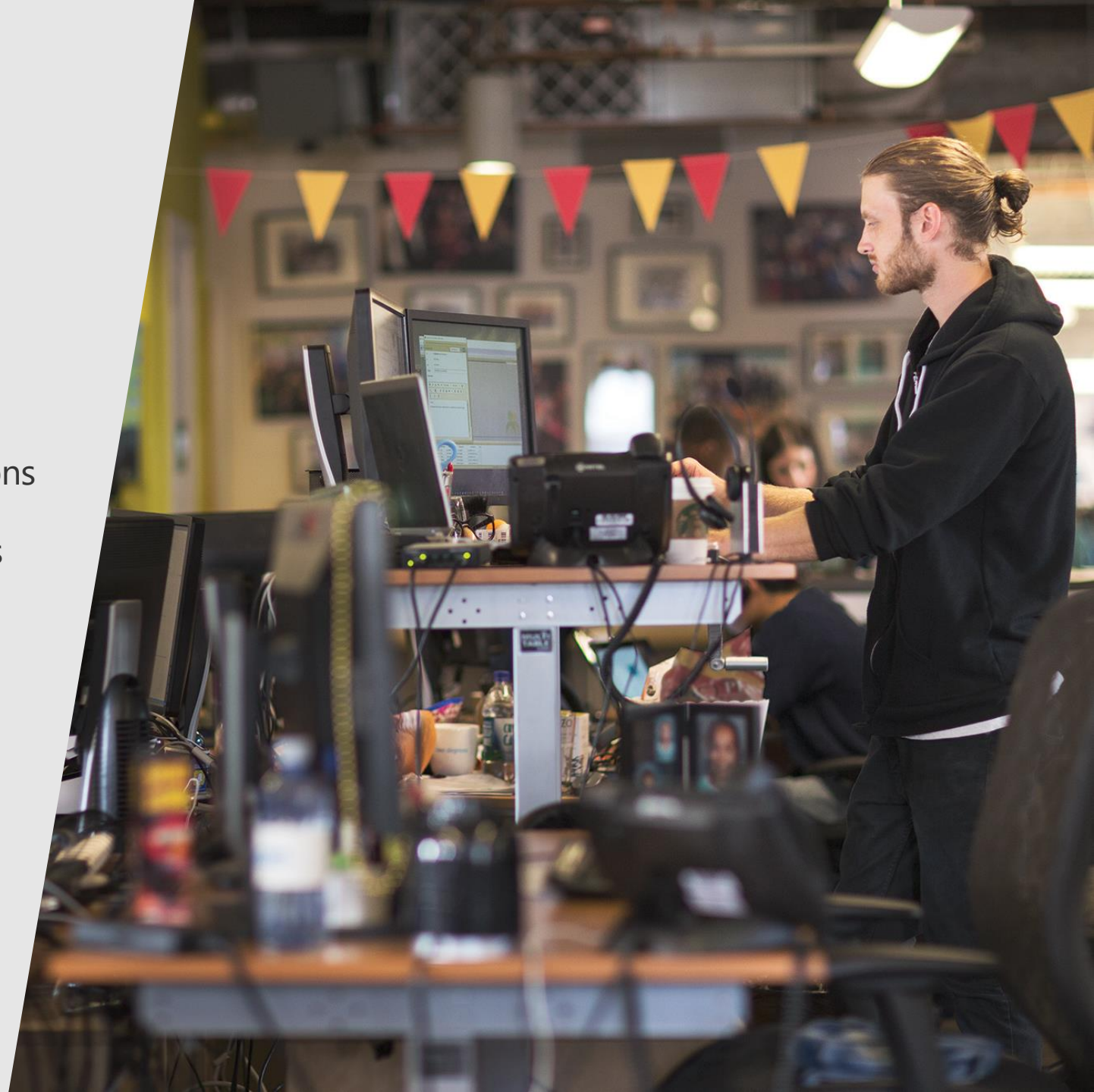
- Arjen Steinhauer asteinhauer@ilionx.com
- Principle Consultant / Architect bij ilionx (Utrecht)
- Werkzaam in IT sinds 1995
- Microsoft platform (.NET)
- Microsoft Azure
- Integration
- Continuous Delivery / DevOps

ilionx
Your partner in digital business


Problem area
App Modernization Trends
Containers & Kubernetes
Demo’s

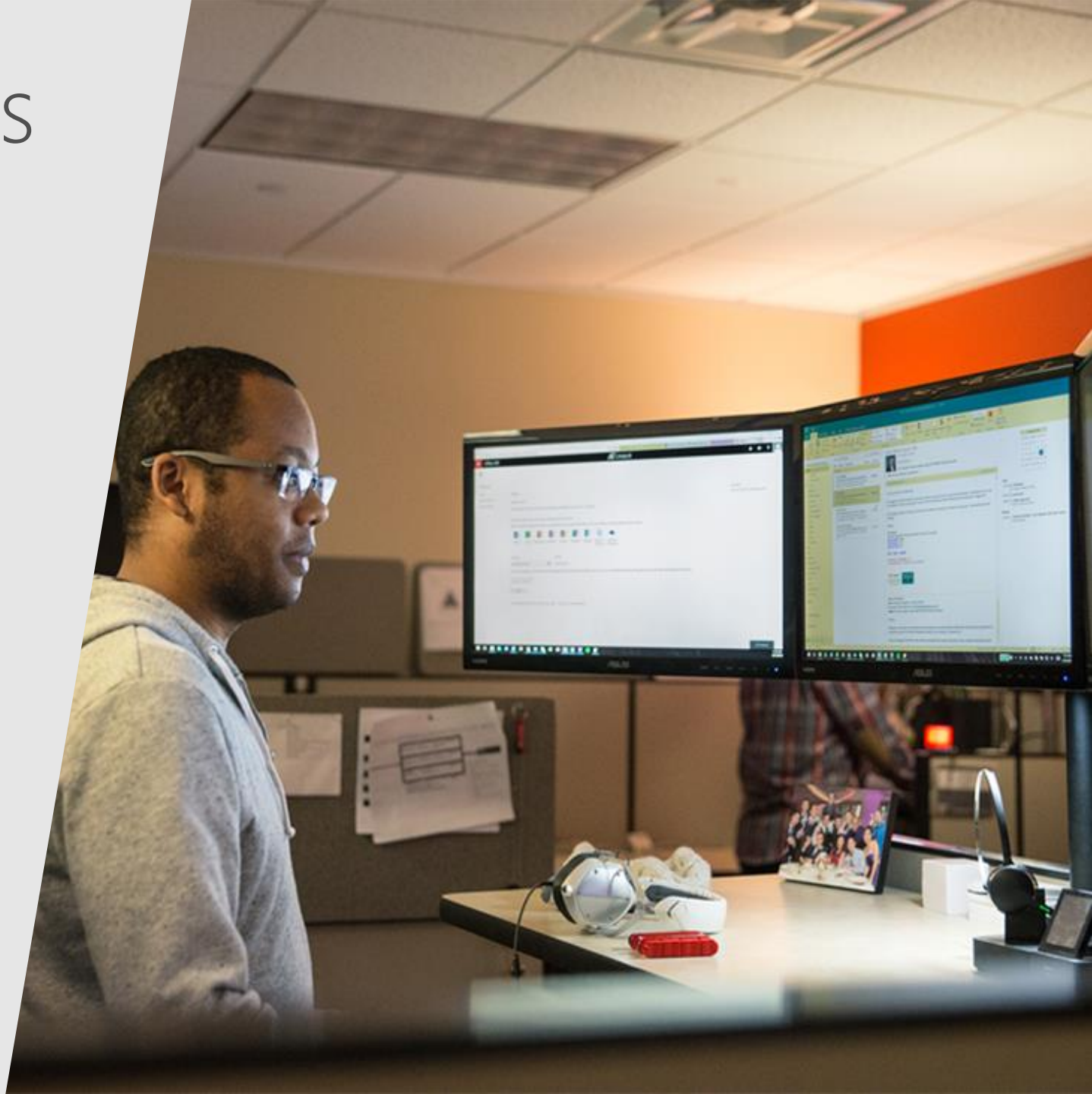
How do app admins spend their time?

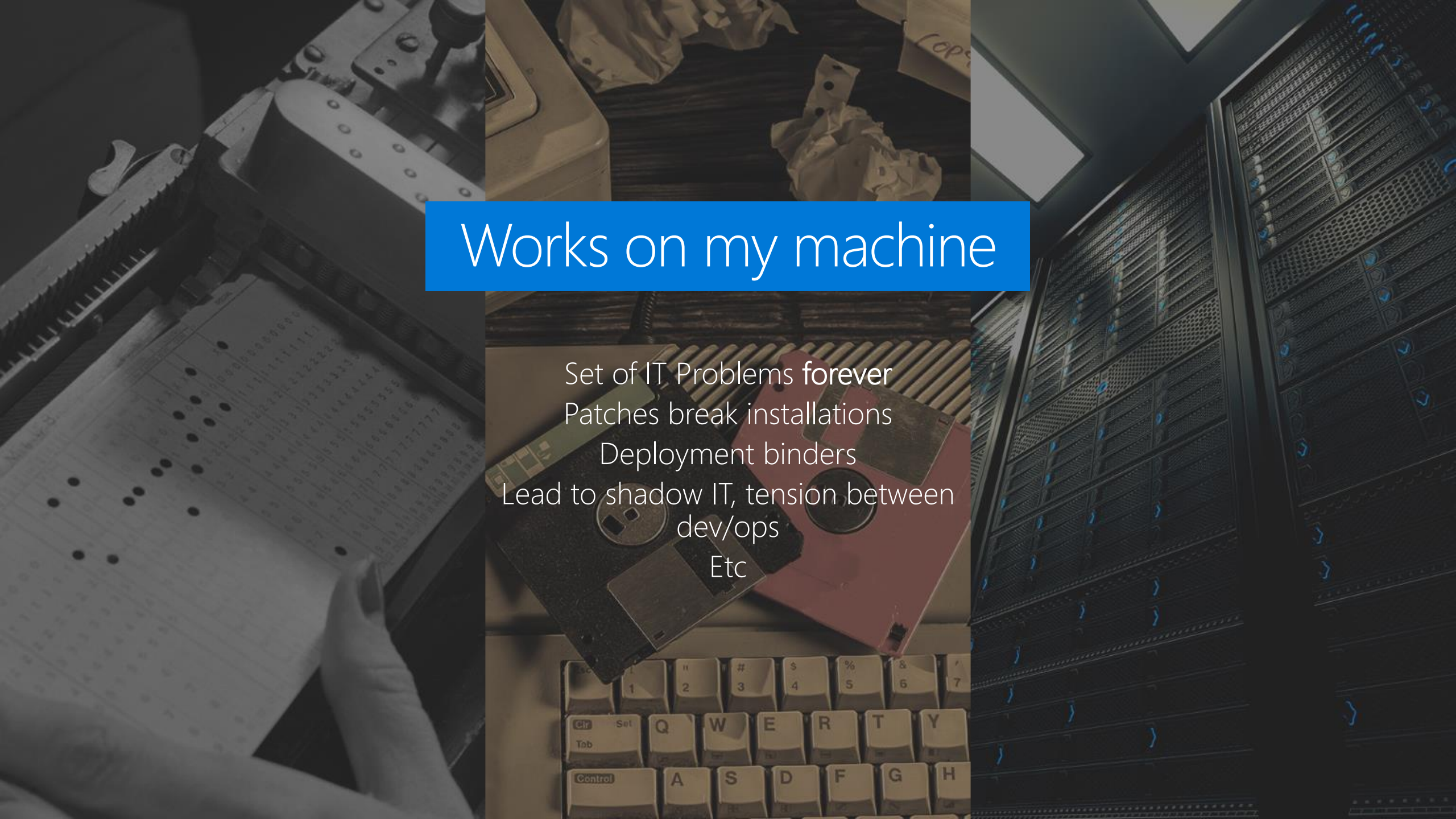
1. Maintaining existing applications
2. Re-deploying new versions of existing applications
3. Troubleshooting issues from feature additions to existing applications
4. Deploying a new application



Where do developers spend their time?

1. Maintaining existing applications
2. Fixing* existing applications
3. Adding features to existing applications
4. Fixing the features someone else added to existing applications
5. Building a new application (maybe … if there's time!)





Works on my machine

Set of IT Problems **forever**

Patches break installations

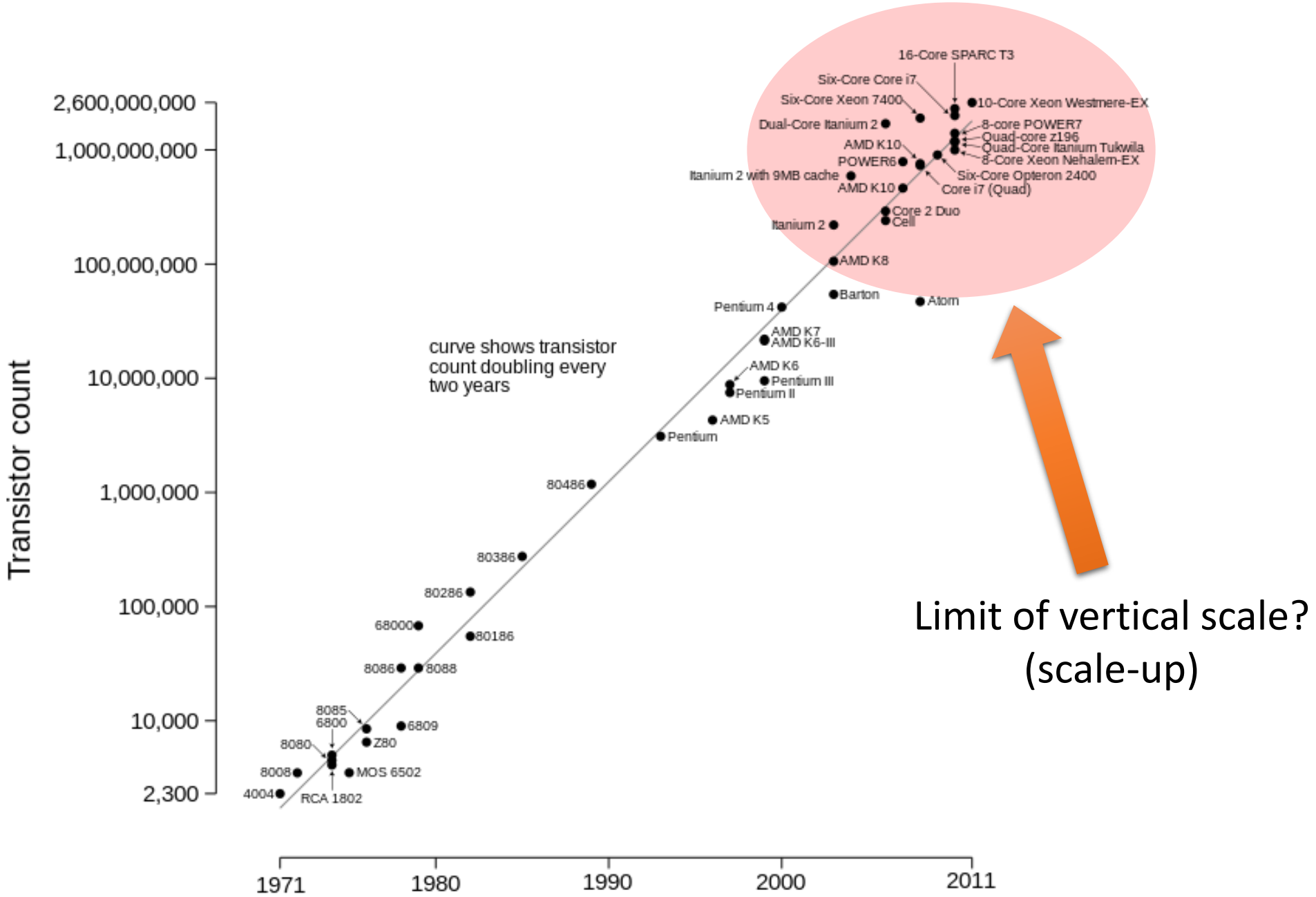
Deployment binders

Lead to shadow IT, tension between dev/ops

Etc

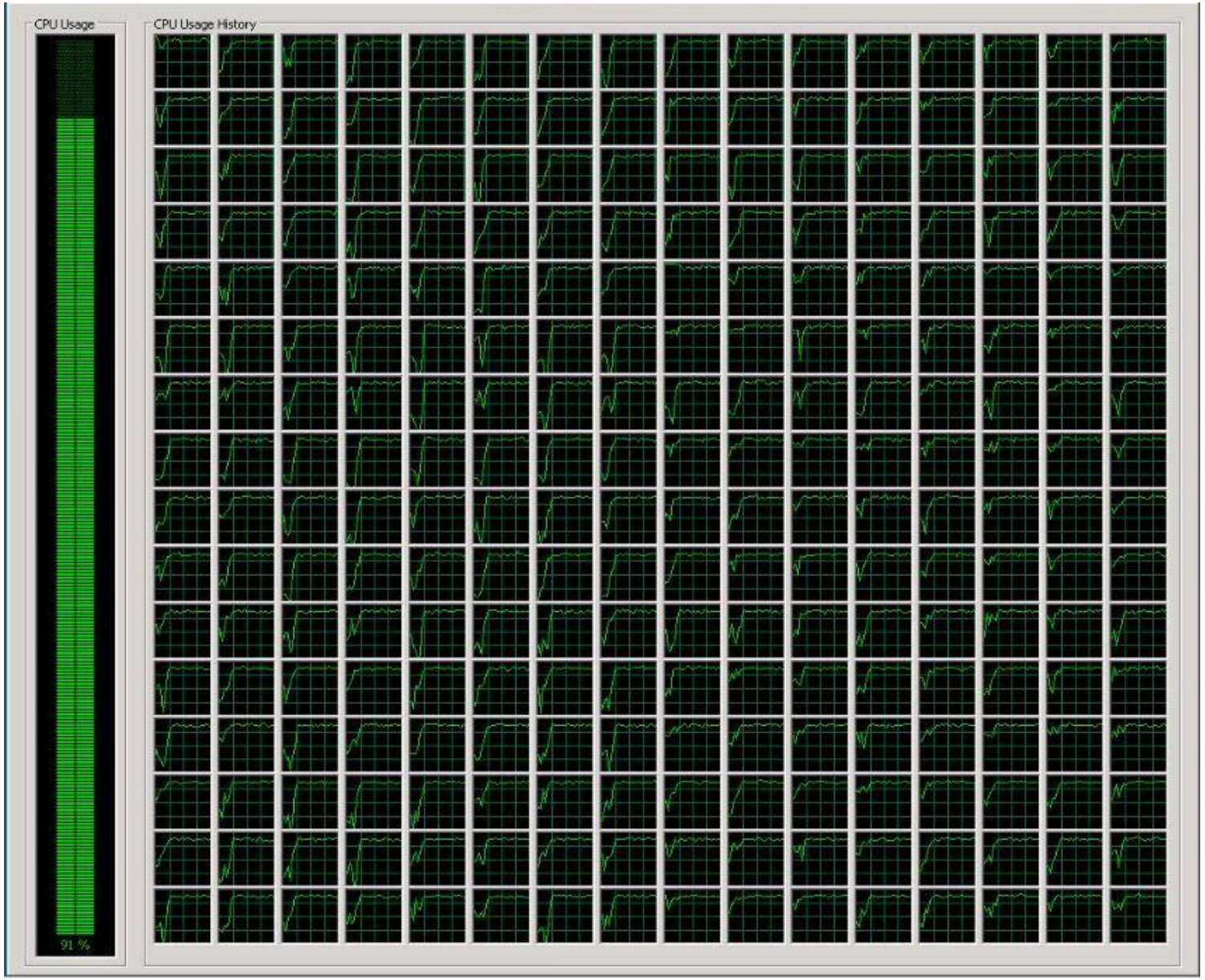
App modernization trends

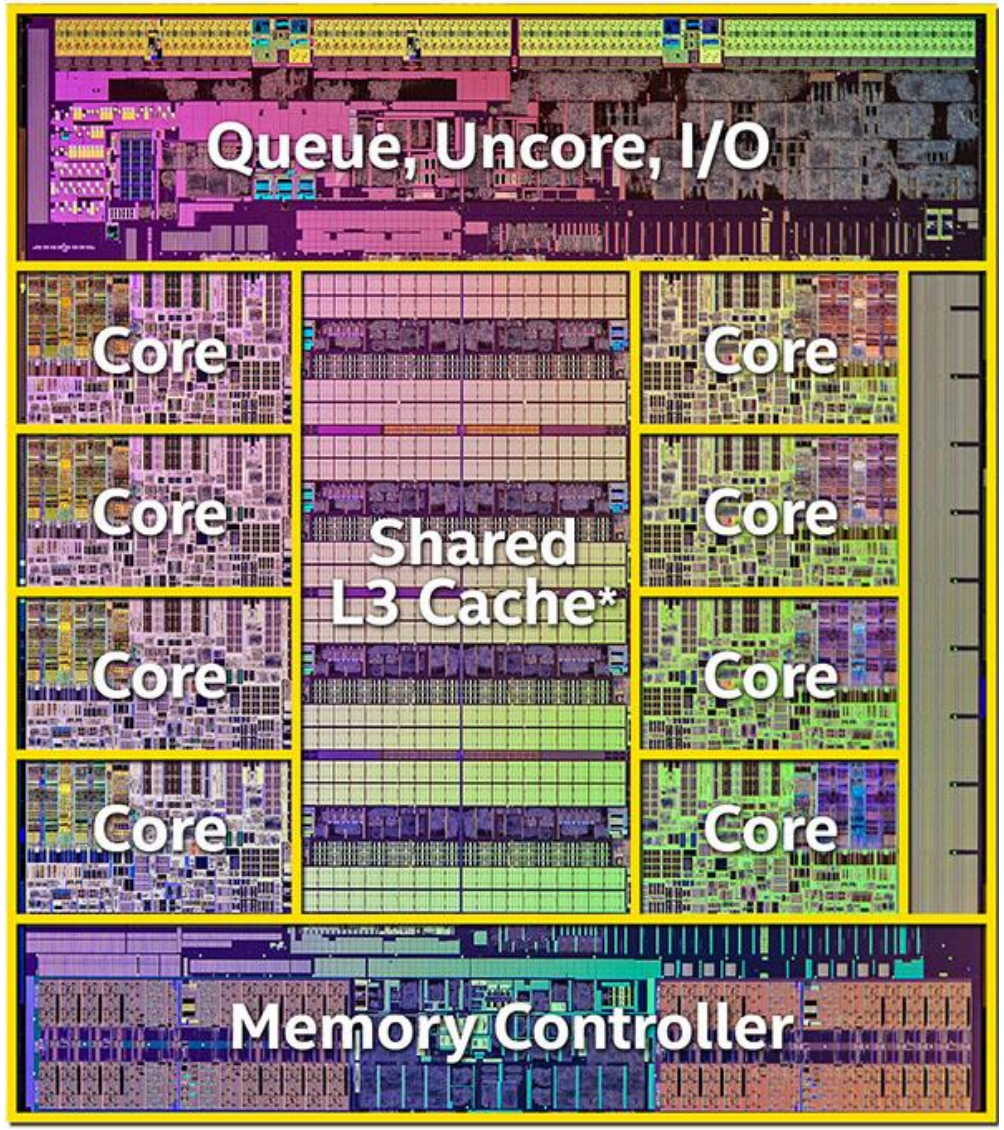
Moore's Law



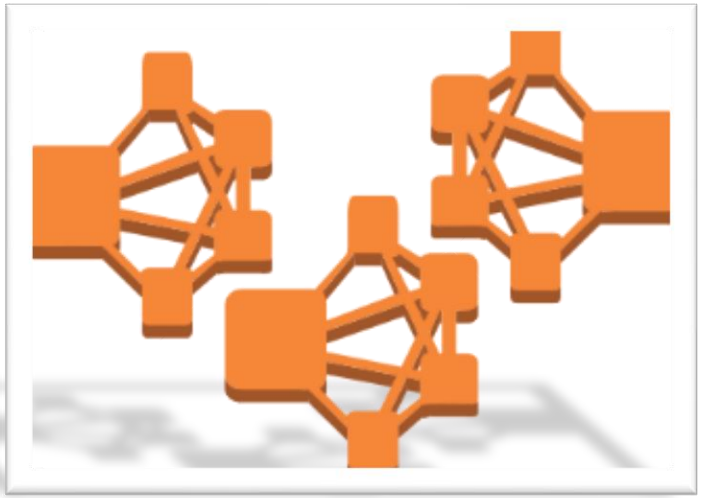
Limit of vertical scale?
(scale-up)

256 core processor



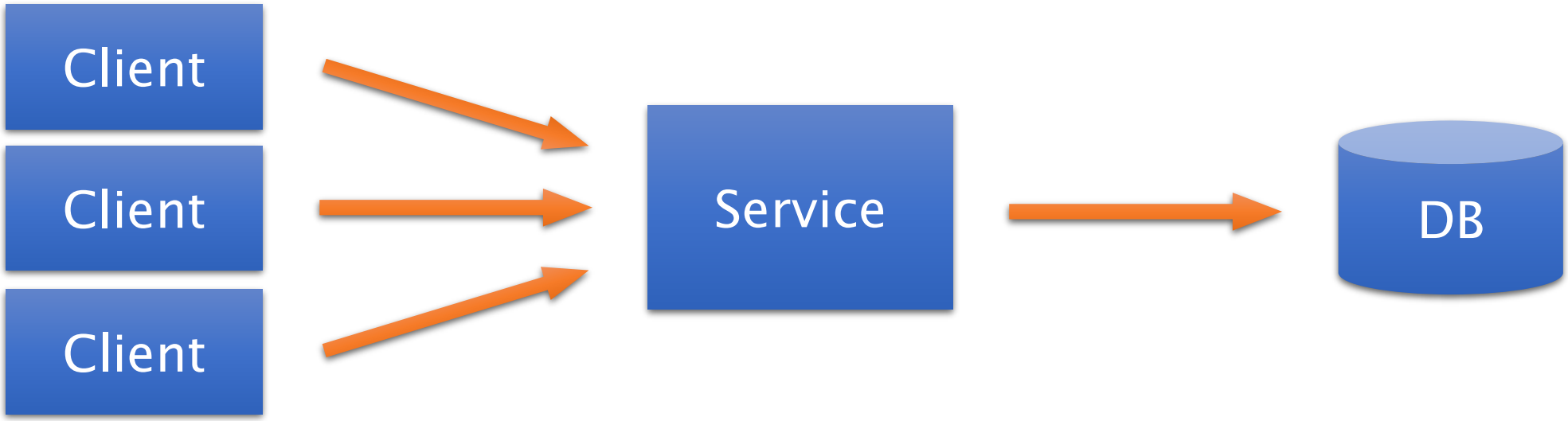


Distributed Systems

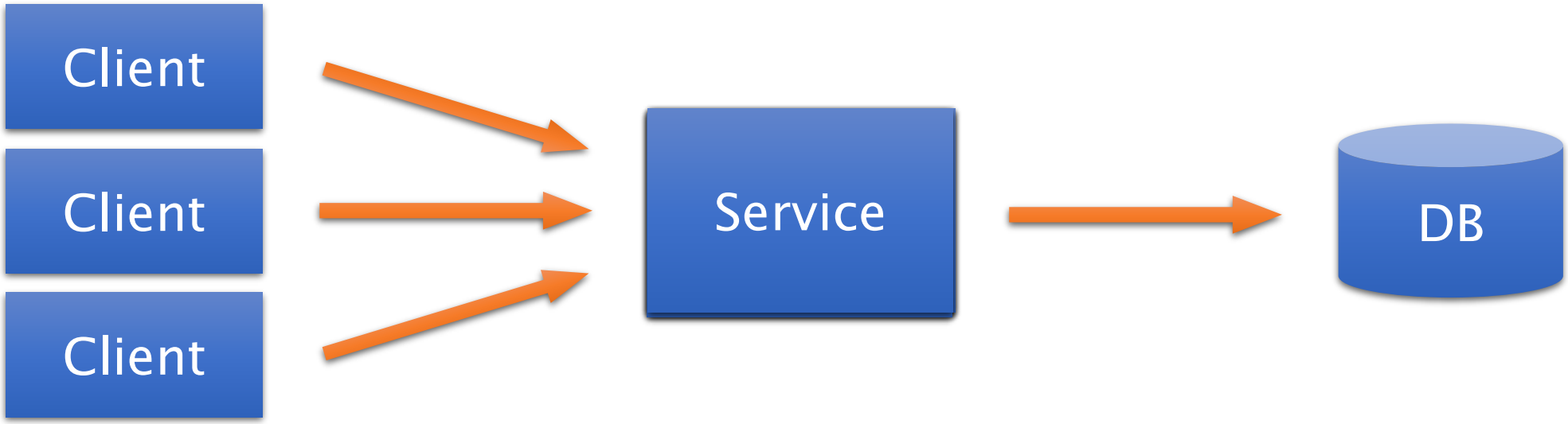


Common approach

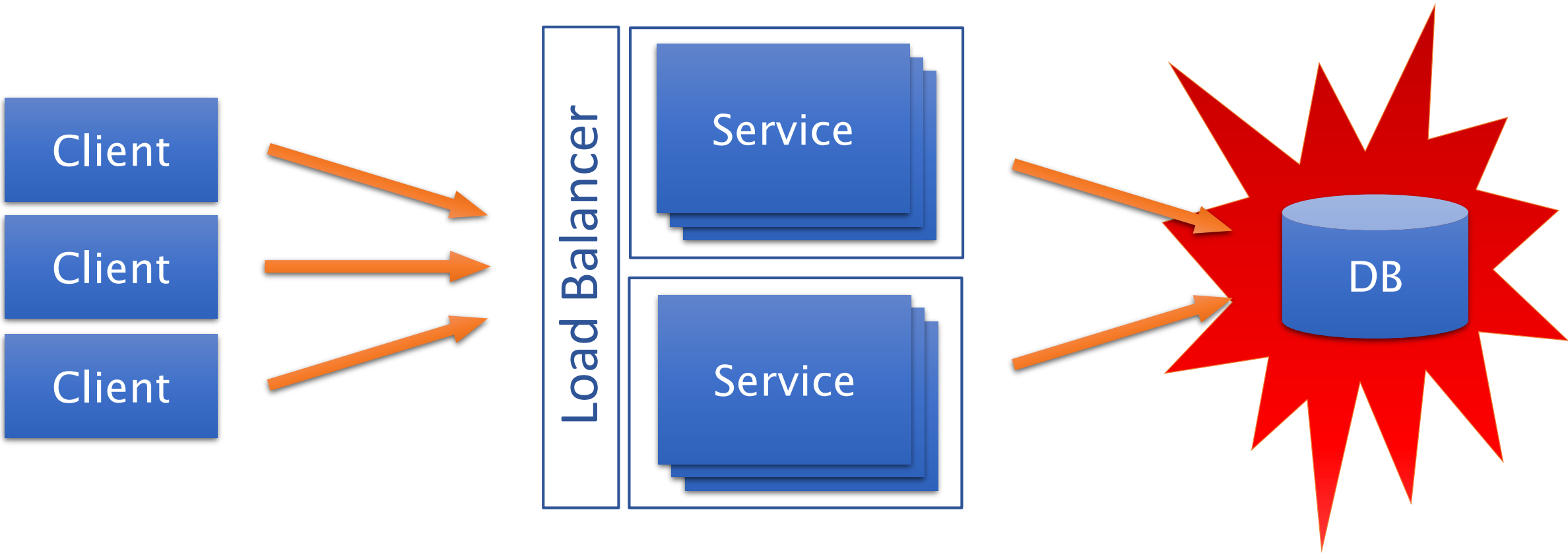
- Client Server model
- 3-tier architecture
- Object Oriented (C#, java, C++, …)
- RPC, DCOM, SOAP, WCF, REST, …



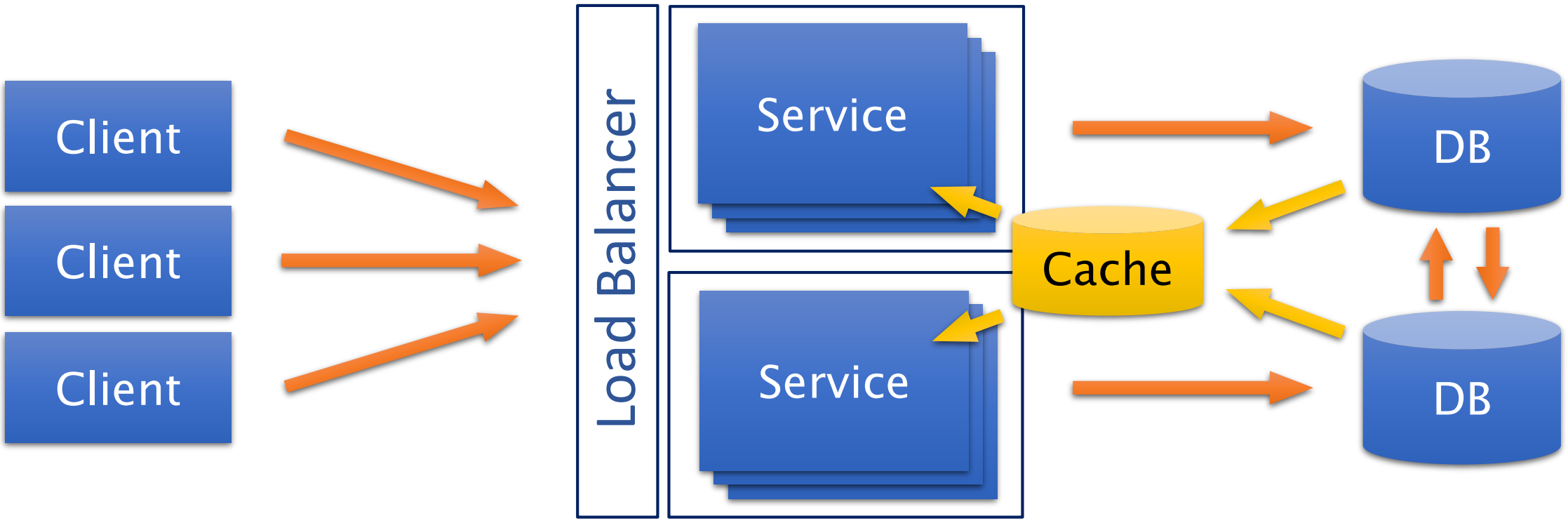
Middleware scale-out



Middleware scale-out

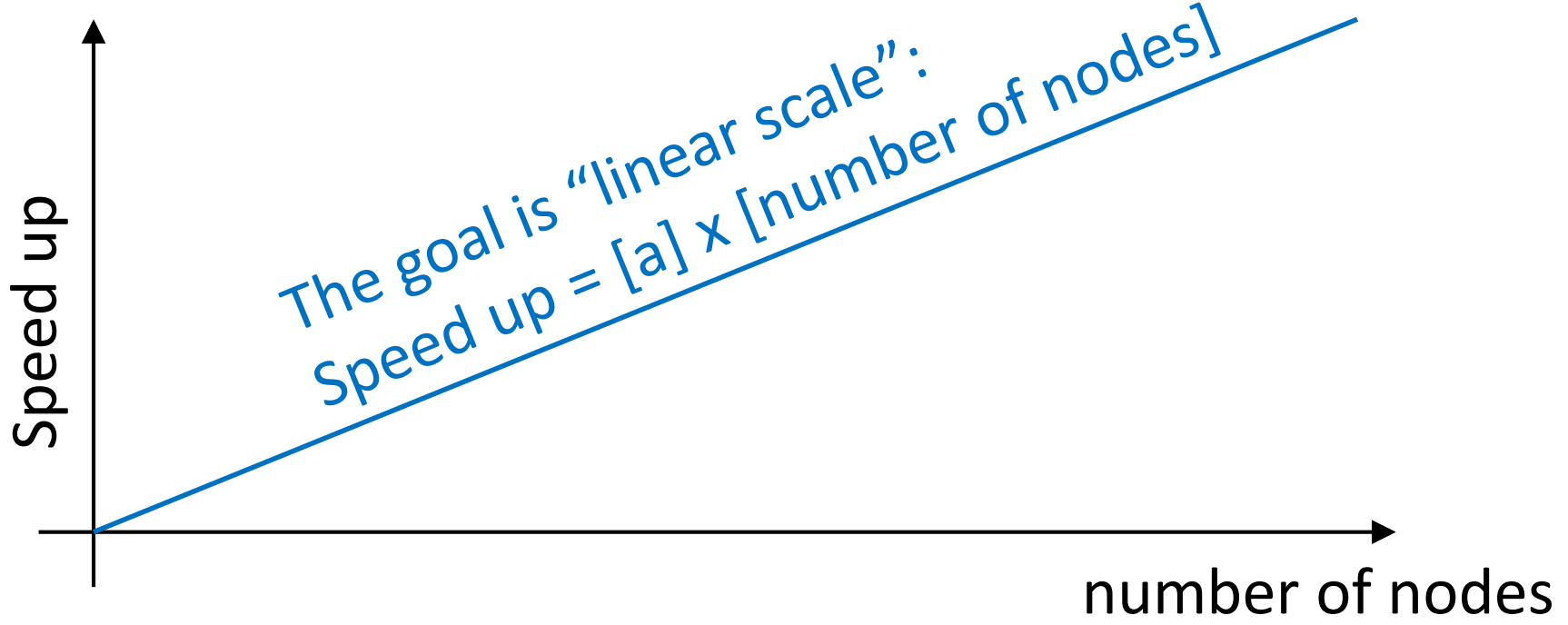


Database partitioning



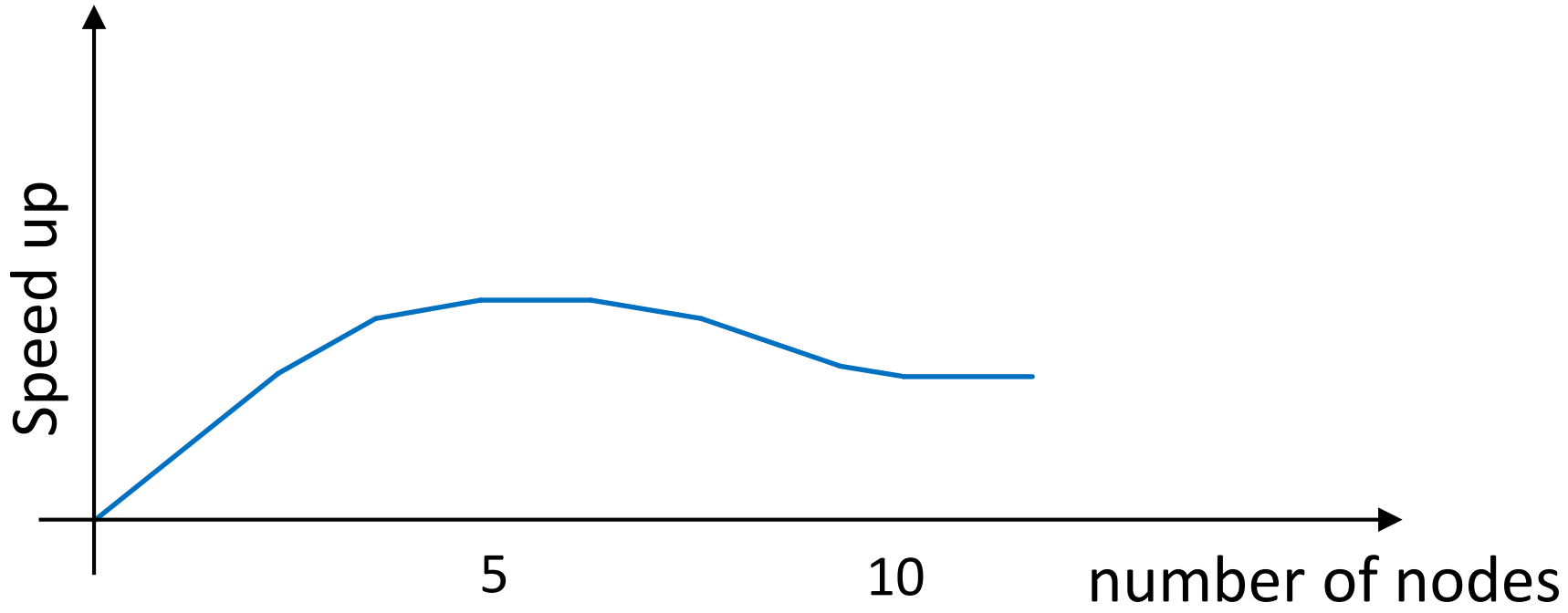
The problem of scale-out

- Concurrency
- Partitioning
- High throughput

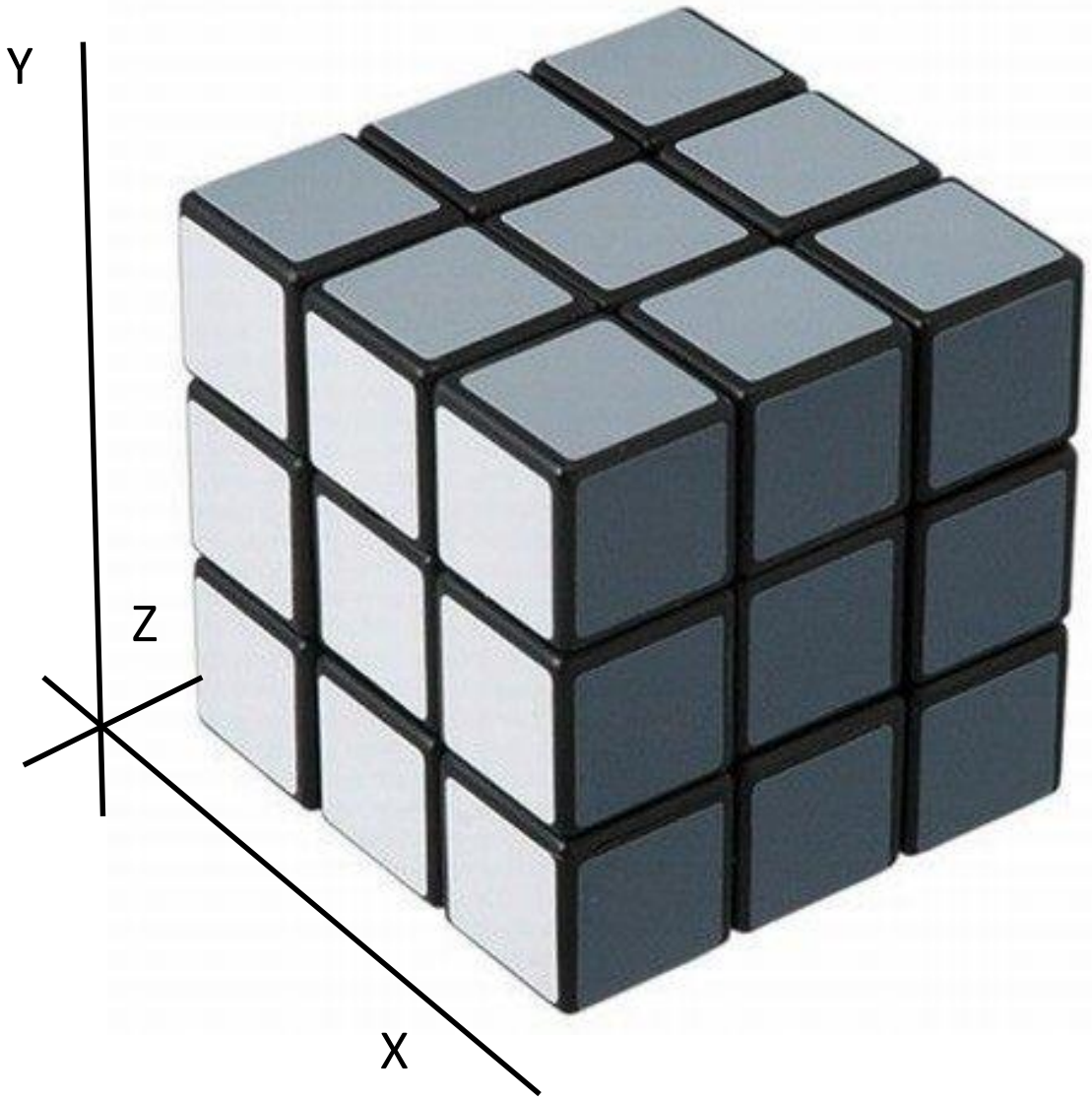


“Speed-up” of traditional approach…

- Reality is non linear
- More nodes does not mean more scale



Need more scaling options → Scale Cube (3D)



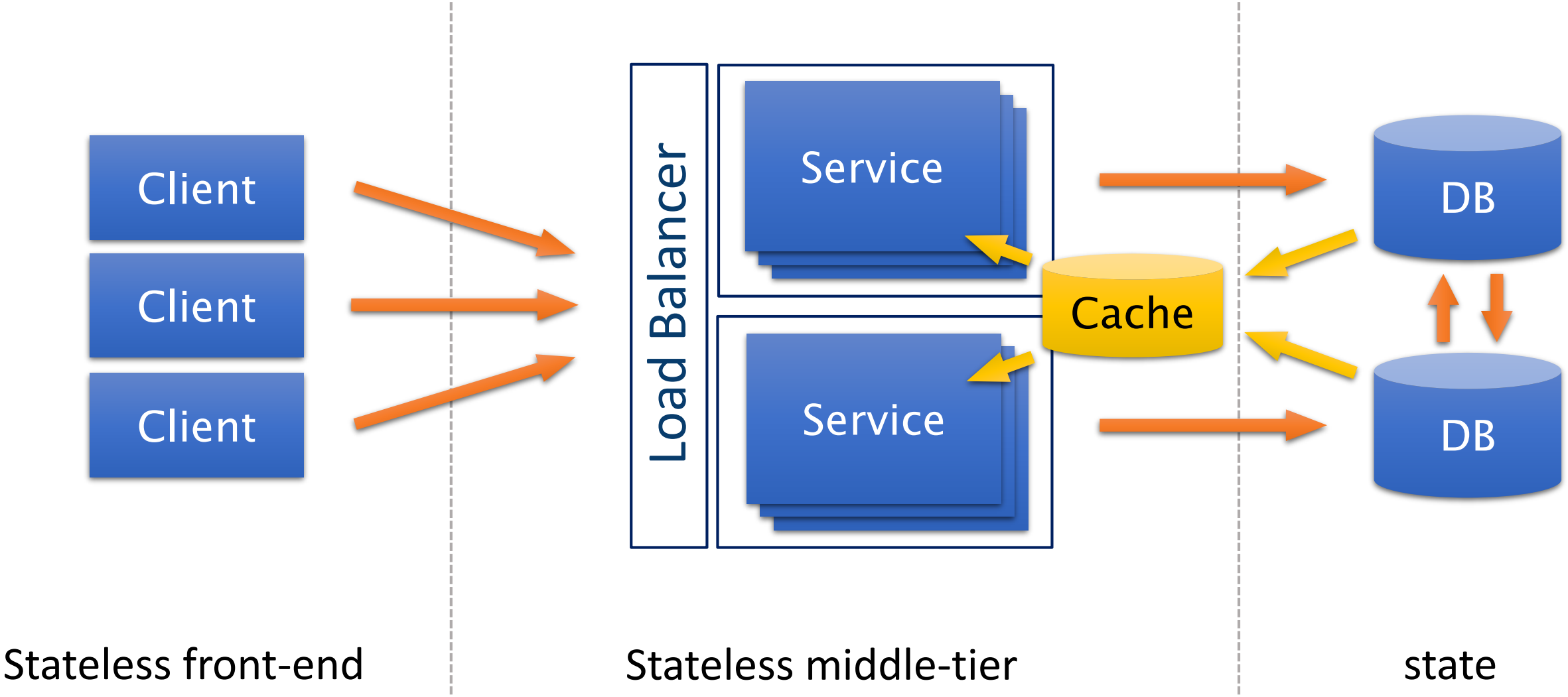
- X
  - Horizontal scale
  - More nodes
  - Network Load Balancing
- Y
  - Functional scale
  - Micro services
- Z
  - Data partitioning
  - Separate tenants
  - Separate by region

Micro Services: the goal

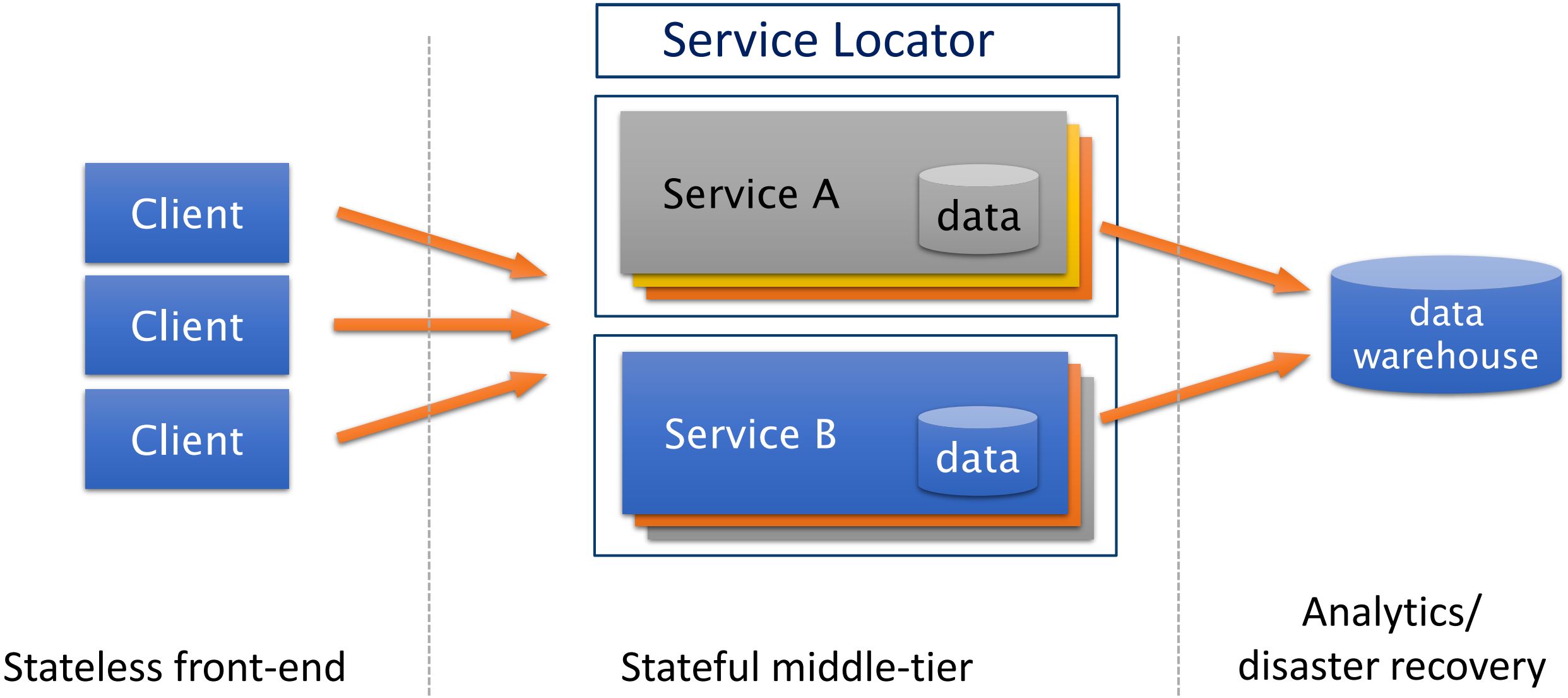
- Build easier Distributed Systems
- Automatic 3D scaling is possible

- Less skilled dev’s can build more complex systems
- Save development and operation costs
- Run same code on-premises and in Cloud

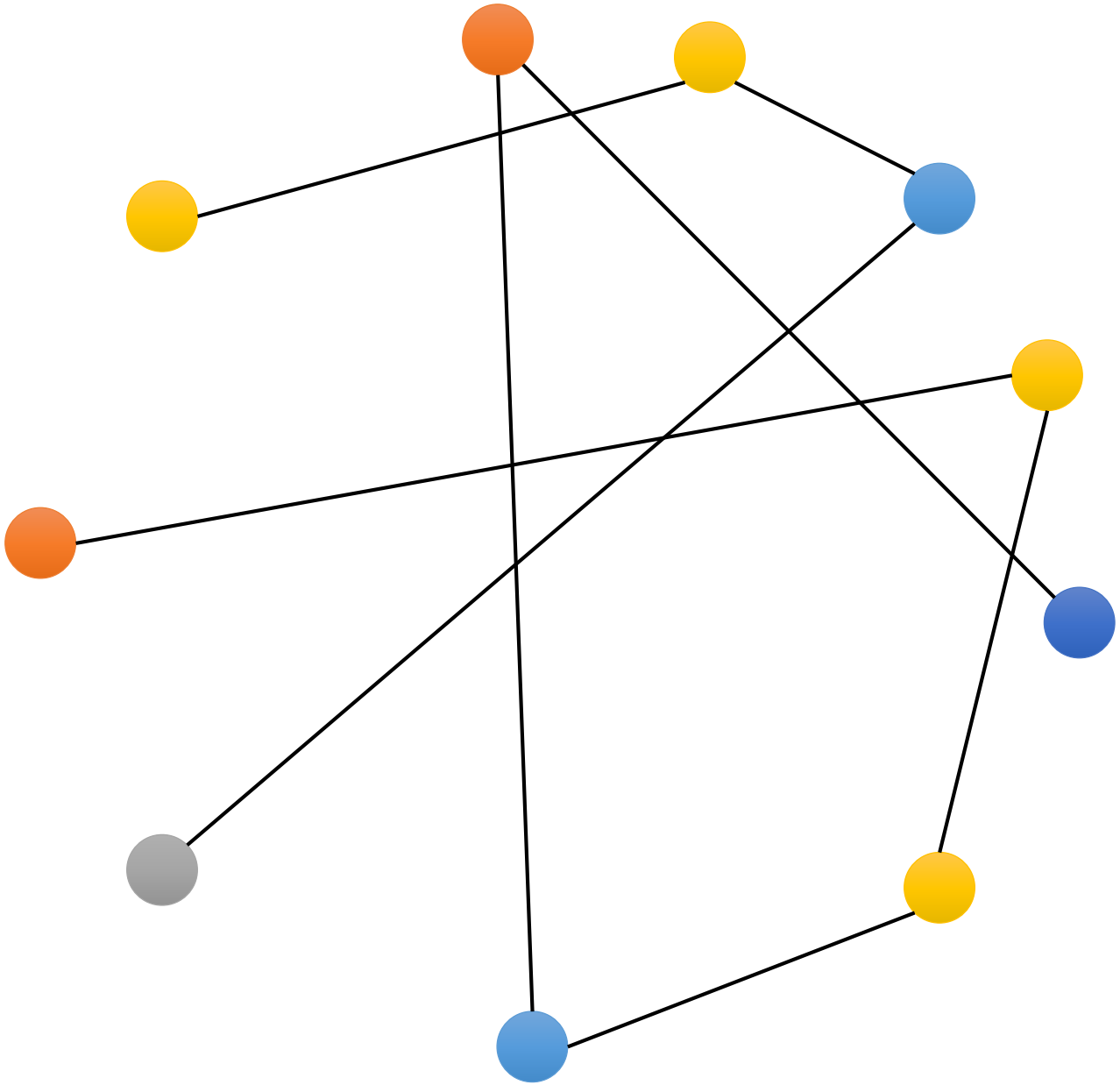
Micro Services



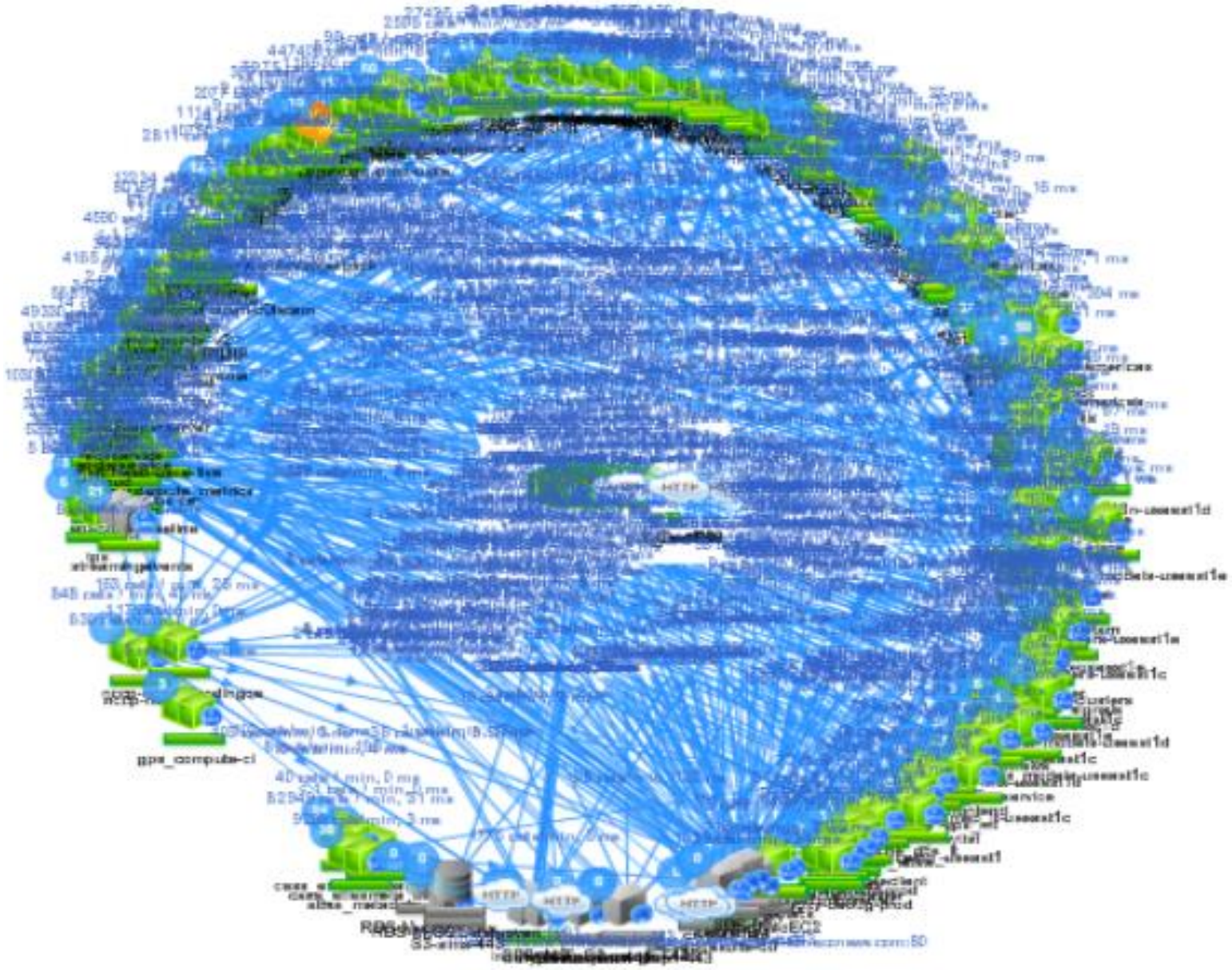
Micro Services



Micro Services



Micro Services bij Netflix



Containers in Azure

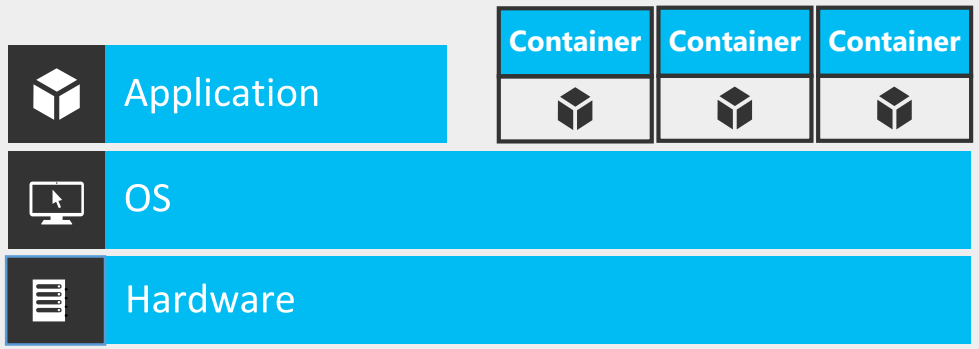
ilionx
Your partner in digital business

What is a container?

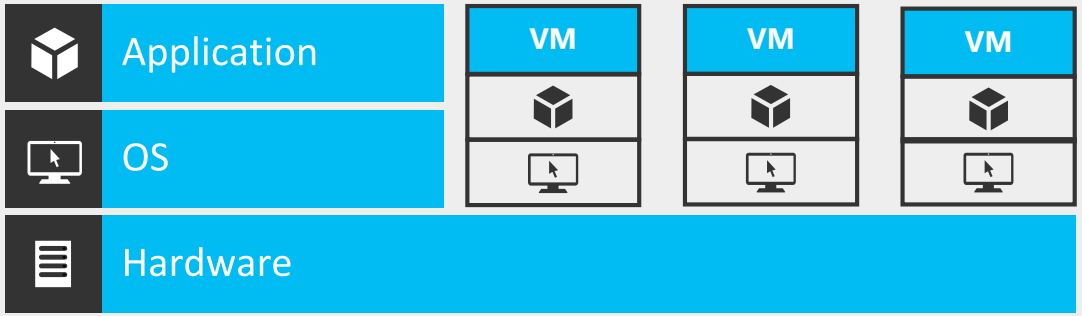
- Slice up the OS to run multiple apps on a single OS
- Typically you run one application/service per container
- Container and apps share lifecycle
- Every container has an isolated view and gets
  - Own **root directory**
  - **eth0** network interface
  - it's own **PID0**
- Shared kernel, very fast start-up, and repeatable execution
- Kernel features to accomplish this:
  - **cgroups**: limiting what you can use
  - **namespaces**: limiting what you can view
- Cannot mix operating systems

How Containers Work

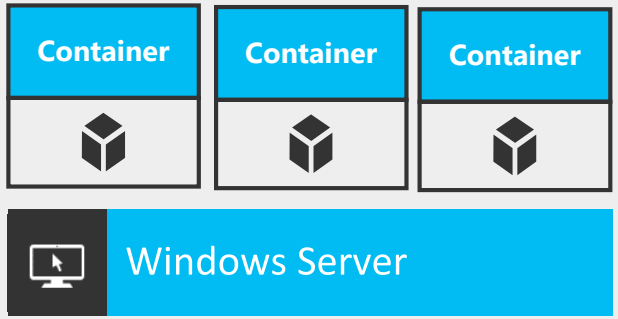
Containers = Operating system virtualization



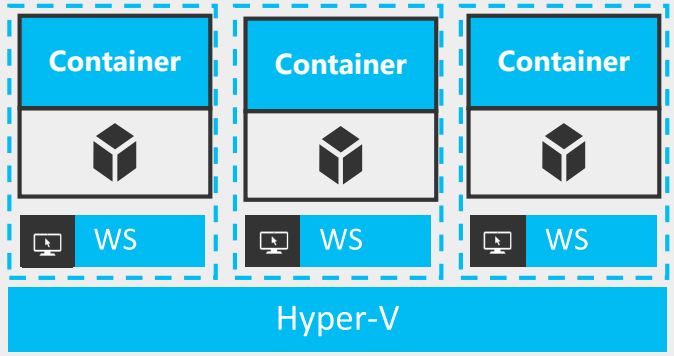
Traditional virtual machines = hardware virtualization



Windows Server containers
No different from Linux containers



Hyper-V containers
Isolation plus performance



Docker Docker Docker ...

Containers have been around for many years

Linux kernel: cgroups, namespaces

Docker Inc. did not invent them

They created open source software to build and manage containers

Docker makes containers easy

Super easy. Fast learning curve

Docker is a container format and a set of tools

Docker CLI, Docker Engine, Docker Swarm, Docker Compose, Docker Machine

Constructing a container

Container contains an application's **dependencies**

- Base image
- Added files, binaries, libraries
- App platforms. Eg – MongoDB, apache, node.js, etc.
- Configuration changes
- Networking
- Custom code

Entry Point

Can think of these as "**layers**"

- One can build new layers on top of an existing container
- Docker will pull these required dependencies to install/run

Why containers?

Benefits of a container architecture:

- Repeatable execution:
  - Immutable environment.
  - Reusable and portable code (“Build, Ship, and Run”).
- Consistency across development, test & production.
- Fast & agile app deployment; instant startup.
- Cloud portability.
- Density, partitioning, scale.
- Diverse developer framework support.
- Micro services.

Demo

Pull
Run
Build

... a container

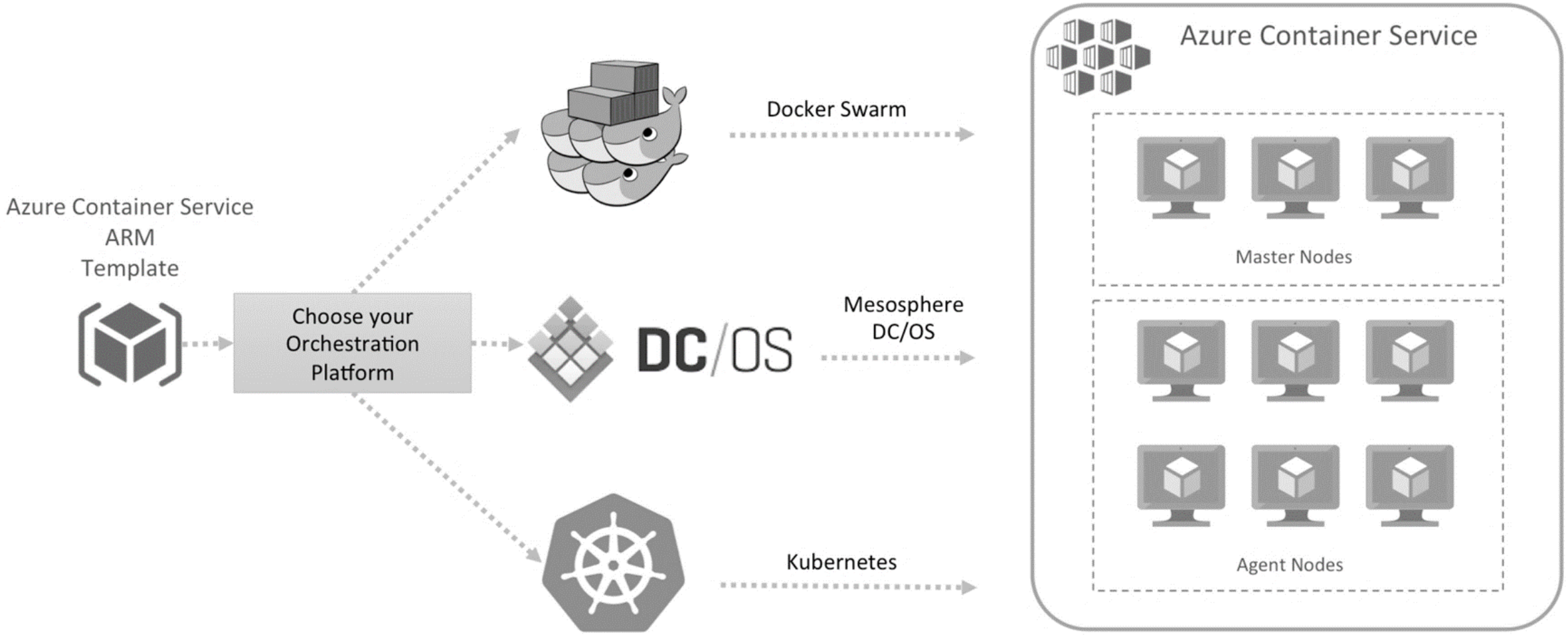

VM Host
Container
Application
App Code
Dependencies
Environment Variables
Files
Libraries
Drivers
But things get complicated quickly


VM Host
Container
Application
App Code
Dependencies
Environment Variables
Files
Libraries
Drivers
Very Quickly

Orchestrators: Container management at scale

Cluster Management Deploy and manage cluster resources	**Scheduling** When containers run	**Lifecycle & Health** Keep containers running despite failure	**Naming & Discovery** Where are my containers	**Load Balancing** Distribute traffic evenly
Scaling Make container sets elastic in number	**Image Repository** Centralized, secure container images	**Continuous Delivery** CI/CD pipeline and DevOps workflow	**Logging & Monitoring** Track events in containers and cluster	**Storage Volumes** Persistent data for containers

Container orchestrators



Selecting a Container Orchestrator

Balance of responsibility

Balance of control and responsibility depends on the category of the service

MOVE-IN READY

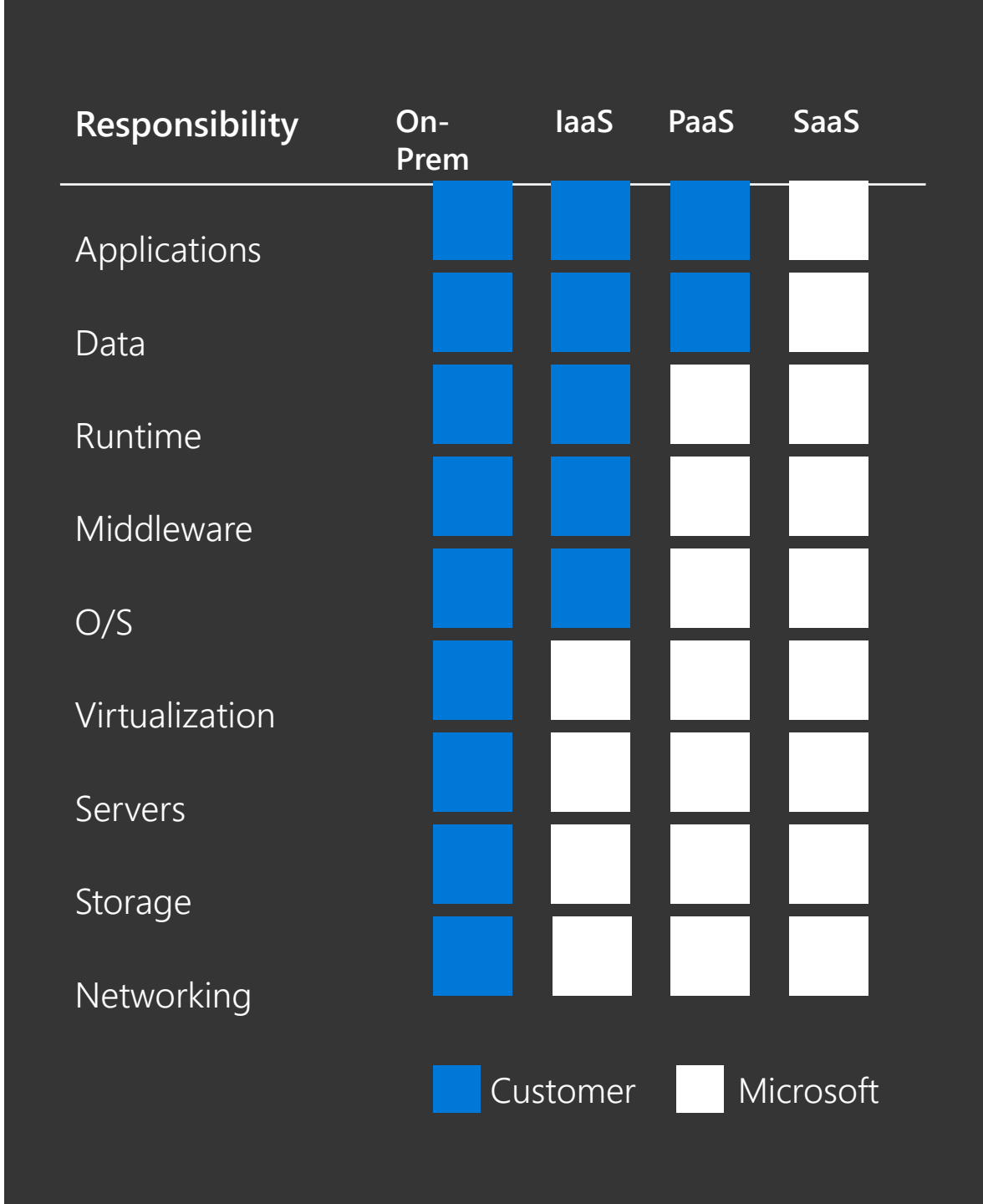
Use immediately with minimal configuration

SOME ASSEMBLY REQUIRED

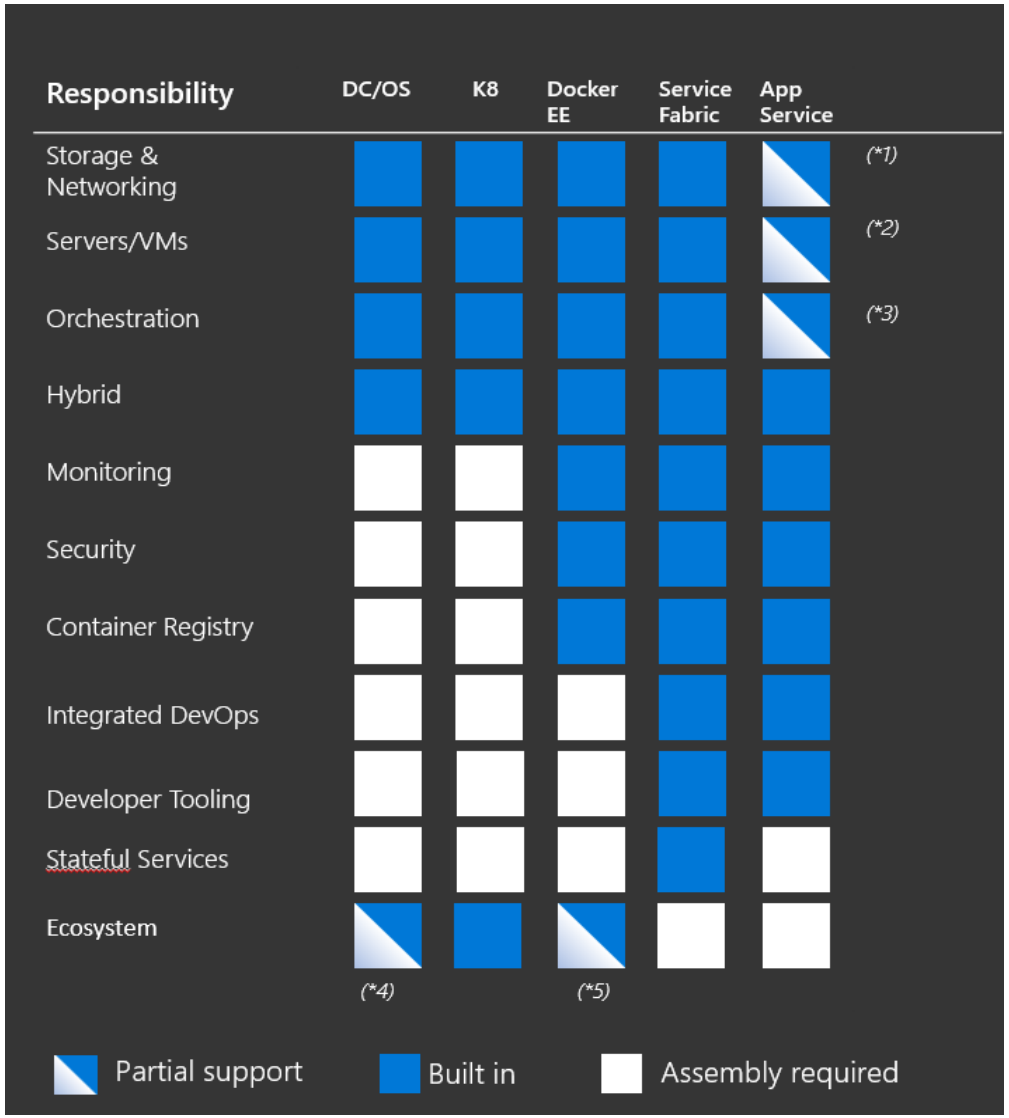
Existing services are a starting point, with additional configuration for a custom fit

BUILD FROM THE GROUND UP

Building blocks, create your own solution or apps from scratch



Comparison – container orchestrators



Notes:

*(*1) (*2) (*3) App service and "container apps" provides for orchestration of a single container apps. In the current state, this is a PaaS offering that abstracts away these concepts.*

*(*4) DC/OS has a good-sized OSS community behind it, however it pales in comparison with K8 ecosystem.*

*(*5) Docker as a container format is very popular and has good community engagement, Docker EE gets some benefits from being associated with it. However, it does not have a strong ecosystem by itself.*

*(*6) DC/OS refers to enterprise DC/OS. ACS provides open source DC/OS*

Azure Container support

Pivotal Cloud Foundry

RED HAT® OPENSHIFT Container Platform

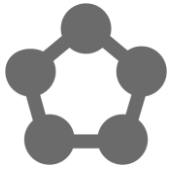
Mesos DC/OS

kubernetes

docker

Azure Container Instance

Container Service



Service Fabric

Web Apps

Batch

Why containers on Azure?

Wide choice of runtime platforms:

- Azure Container Service (ACS) and Engine (ACE)
- Managed Kubernetes Service (AKS)
- Azure Container Instance
- Web Apps for Containers

Partnerships:

- Red Hat
- Docker Inc
- Mesos
- Acquisition of Deis (Helm, Draft, Brigade)

Deep Platform Integration:

- Azure File Service Docker Volume Driver
- Docker Machine Azure Driver
- Docker Registry Azure Storage Driver
- Azure Docker VM Extension
- Extensive ARM template library

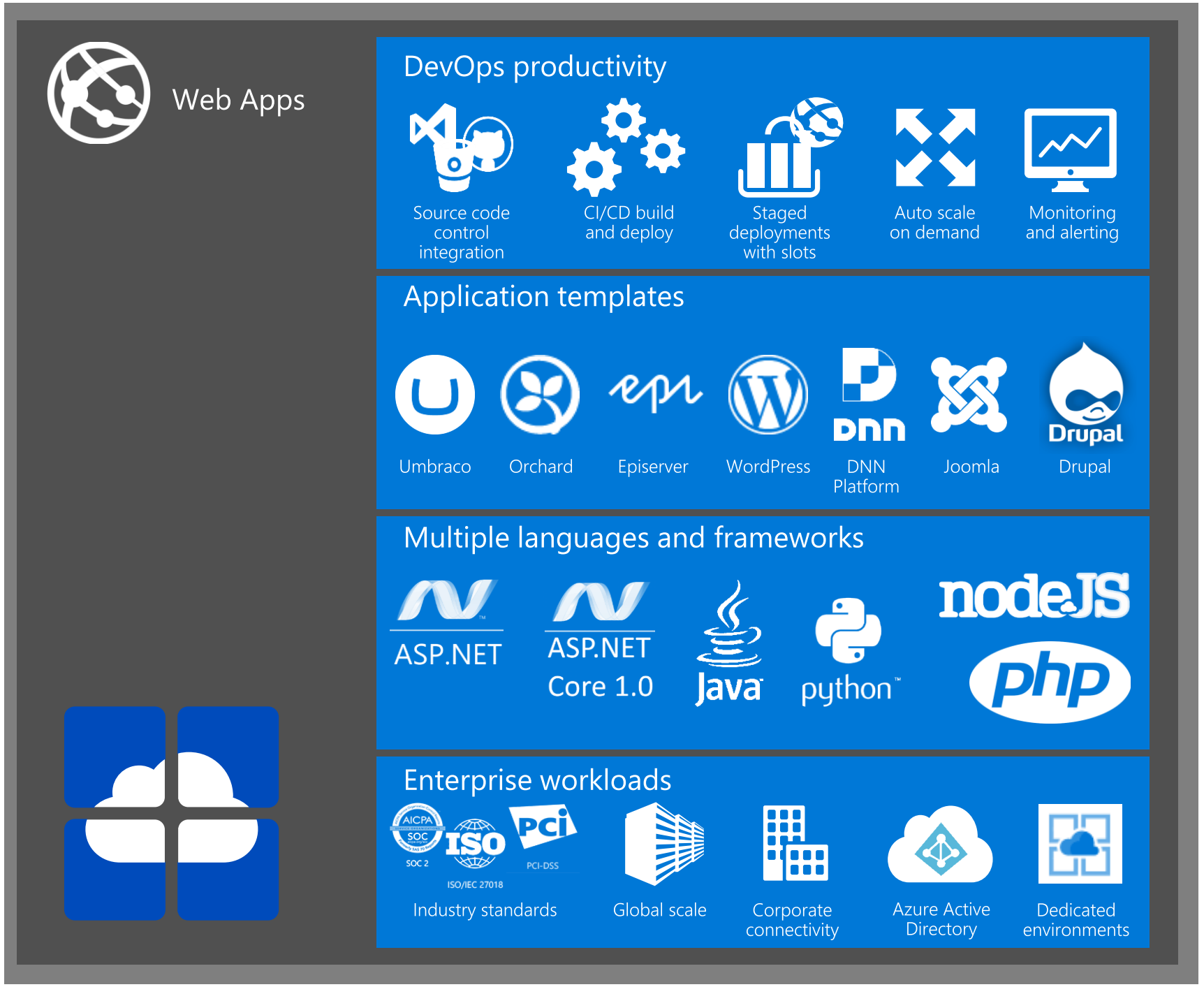
Azure Platform maturity:

- Unique platform features and SLAs
- Security Center
- Azure Active Directory
- Log Analytics

Azure Container Instances

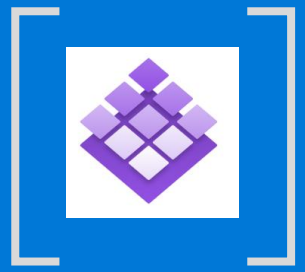
- ACI is in “Public Preview“ and offers the fastest and simplest way to run a container in Azure
- On-demand and nearly instantaneous container compute in the cloud
- ACI offers serverless container hosting on Azure
- No need to worry about managing VM’s, the OS or infrastructure underneath it
- Delivers a single container in seconds with simplicity
- Pay by the second

App Service

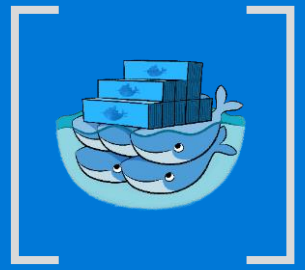

Web Apps
DevOps productivity
Source code control integration
CI/CD build and deploy
Staged deployments with slots
Auto scale on demand
Monitoring and alerting
Application templates
Umbraco
Orchard
Episerver
WordPress
DNN Platform
Joomla
Drupal
Multiple languages and frameworks
ASP.NET
ASP.NET Core 1.0
Java
python
nodeJS
php
Enterprise workloads
SOC 2
ISO/IEC 27018
PCI-DSS
Industry standards
Global scale
Corporate connectivity
Azure Active Directory
Dedicated environments

Azure Container Service

Choice of most popular container orchestrators



DC/OS



Swarm

Kubernetes

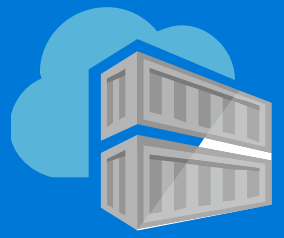
Simplest way to deploy open source container management platforms on Azure

Command Line Interface

ARM templates

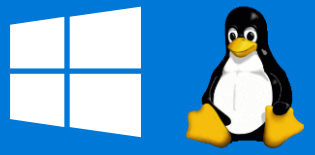
Azure Portal

Few clicks to enterprise grade container solution



Container Registry

Container Monitoring



Windows & Linux Containers

*Managed disks for persistent storage

Azure Container Service

- Managed delivery of optimized container hosting solution
- Manage container applications using popular open source tooling
- Scale and orchestrate using Mesosphere DC/OS, Docker Swarm, or Kubernetes
- Migrate container workloads to and from Azure
- Full support from Microsoft / Docker / Mesosphere
- Leverage Azure Platform capabilities
  - Azure Resource Manager
  - VM Scale Sets
  - Networking
  - Security
  - More coming soon…

What is Kubernetes

Open source project started by Google in 2014

- container-centric vs. host-centric
- schedules and runs application containers across a cluster of machines
- name originates from Greek, meaning “helmsman” or “pilot”; the root of “governor” & “cybernetic”

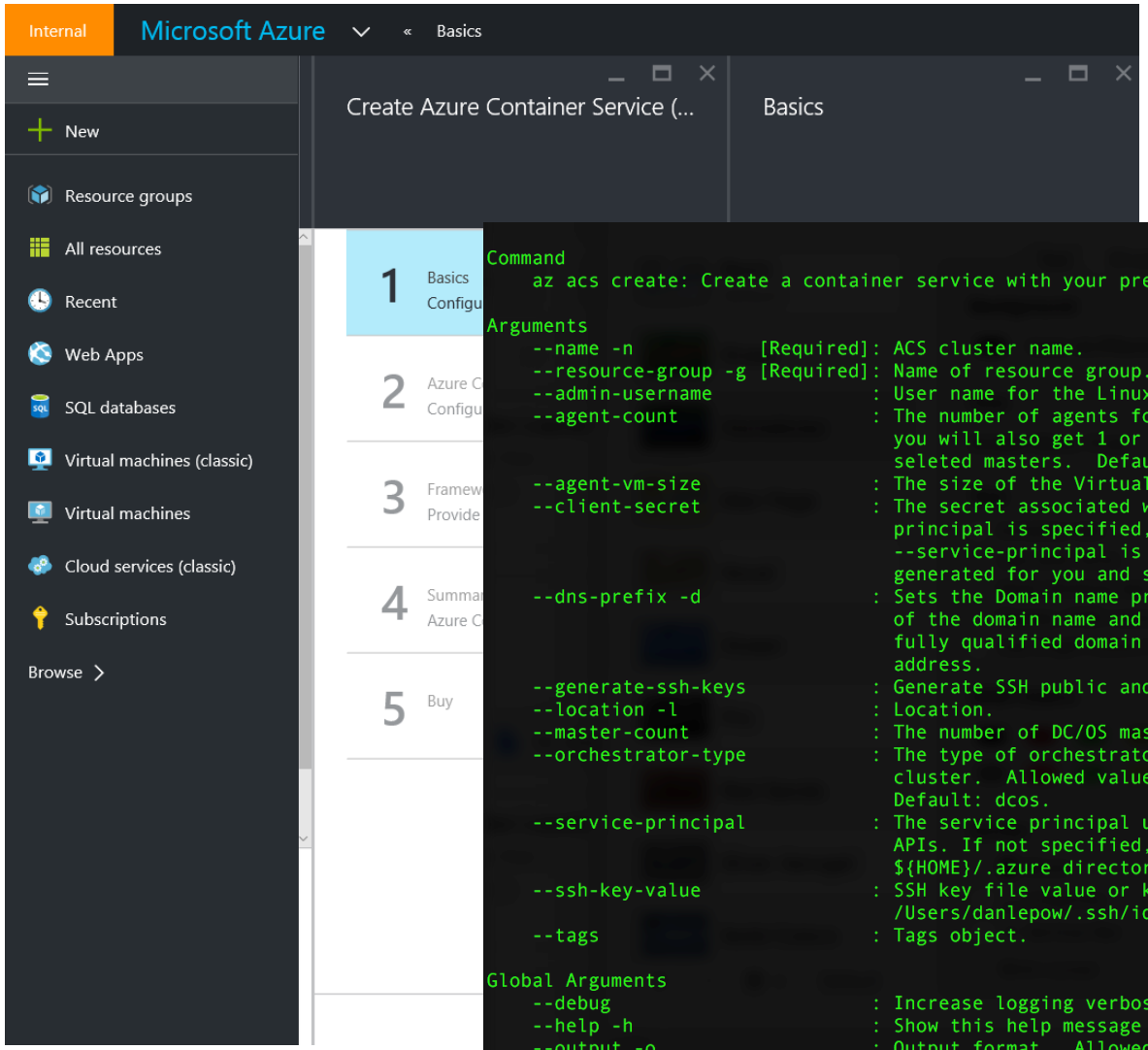
Key tenants

- portable: public, private, hybrid, multi-cloud
- extensible: modular, pluggable, hookable, composable
- self-healing: auto-placement, auto-restart, auto-replication, auto-scaling

Provides:

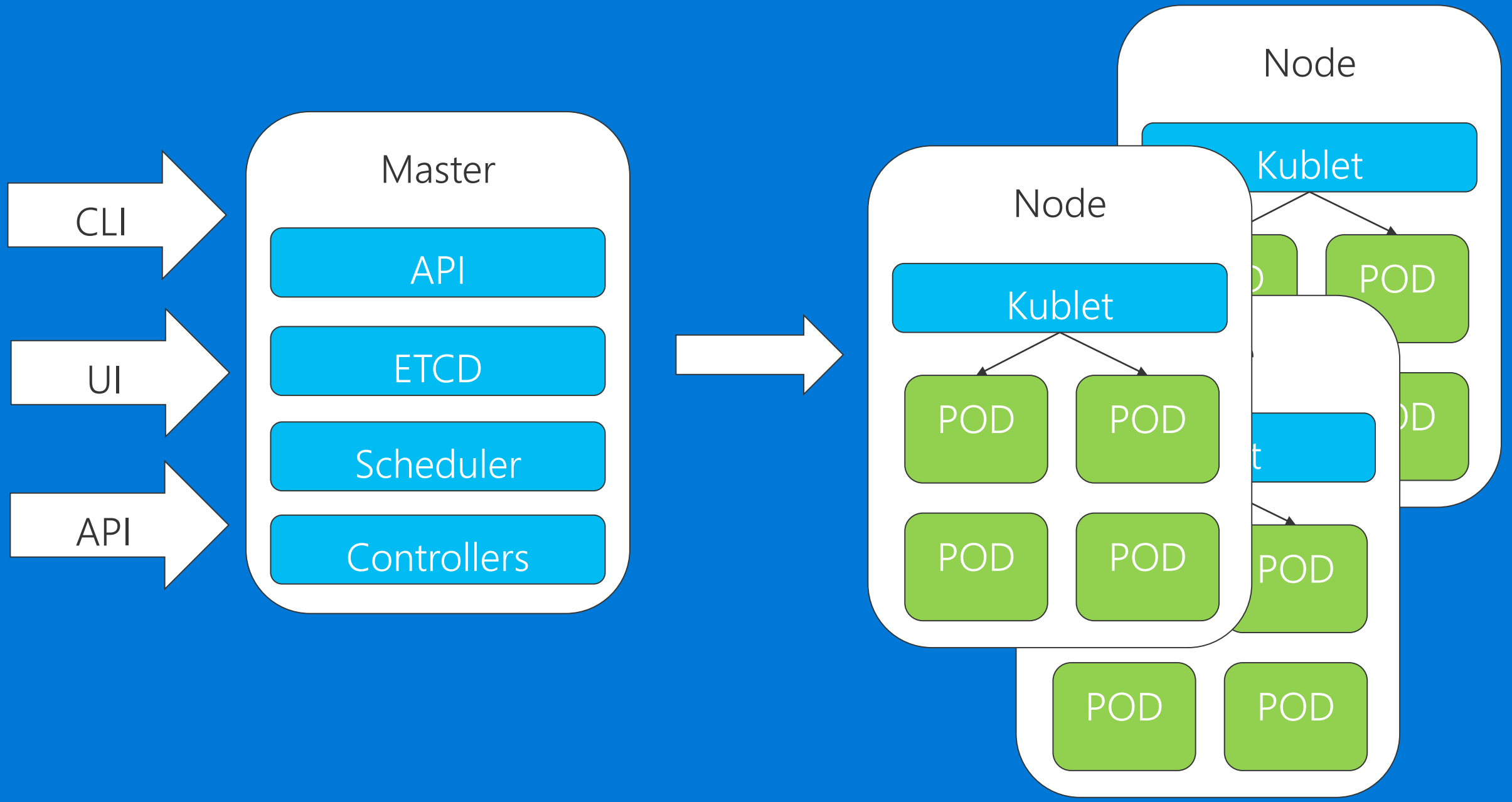
- resource monitoring, load balancing, naming & discovery, horizontal auto-scaling
- mounting storage systems, distributing secrets, application health checks, resource monitoring
- cloud provider integration (Eg - Azure networking, Azure disks)
- pods, namespaces, labels, stateful sets
- Some PaaS platforms run on top of K8s (Eg – Openshift, Deis, etc.)

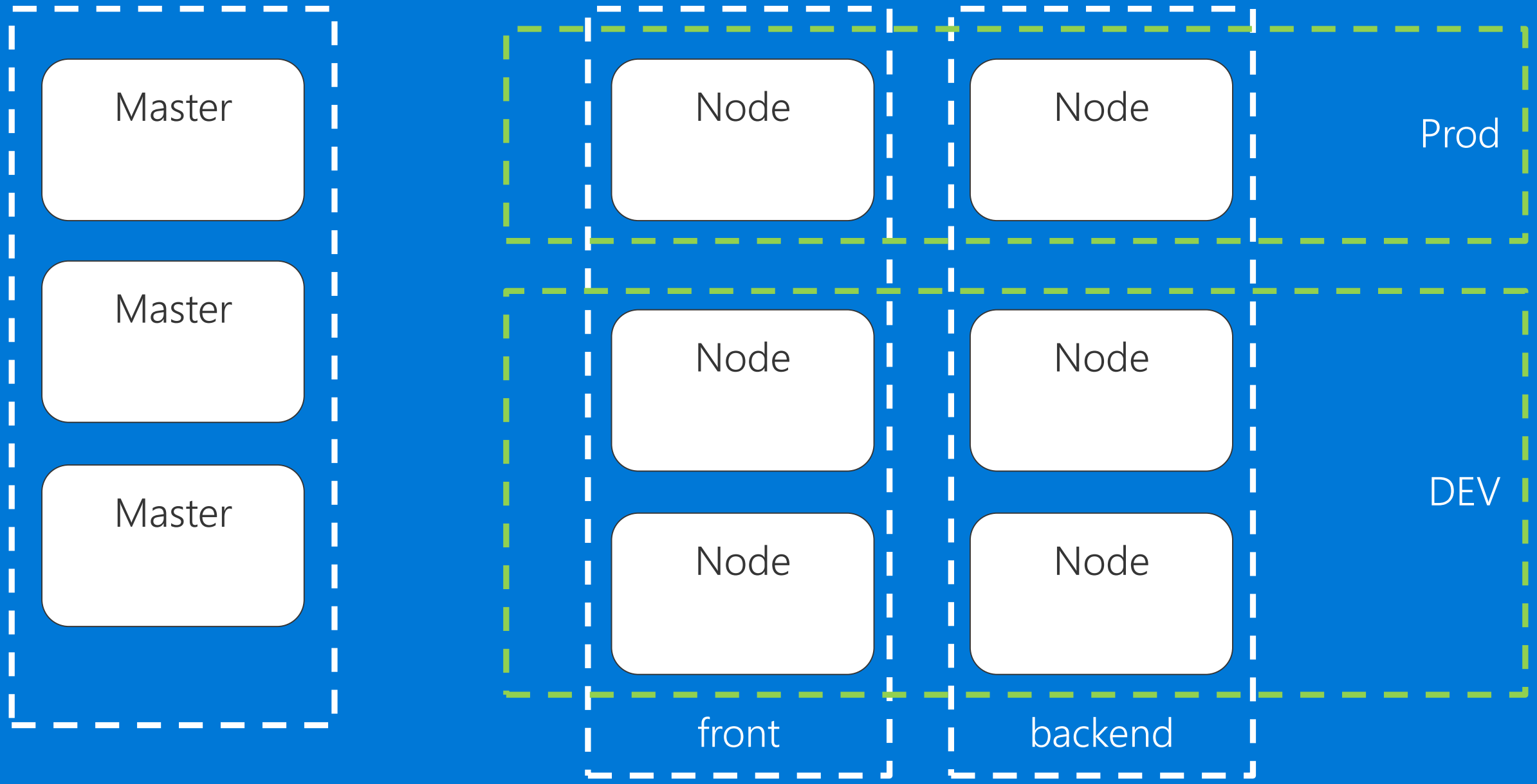
Deploy using Portal, CLI, or ARM



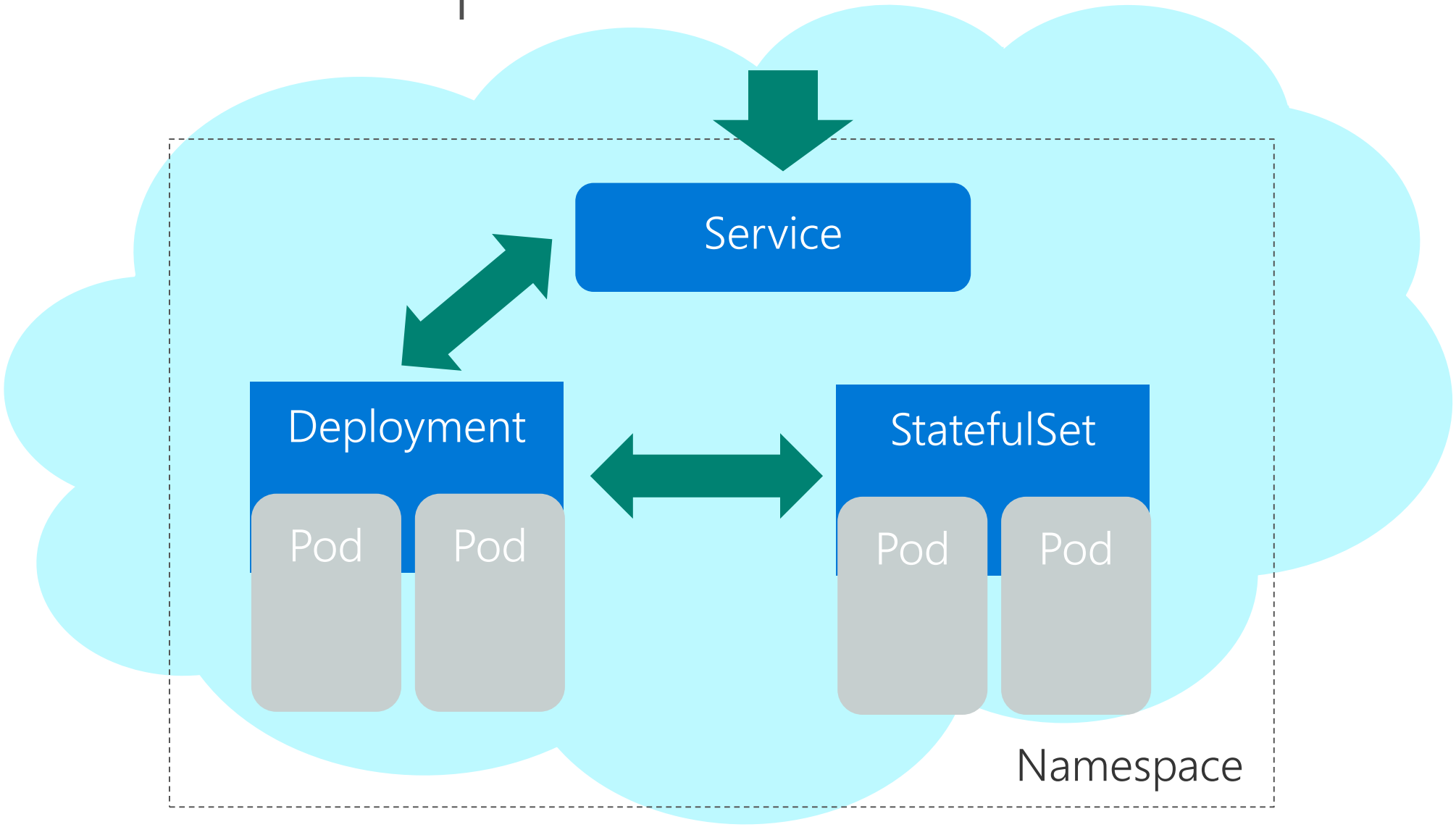

```
Command
    az acs create: Create a container service with your preferred orchestrator.

Arguments
    --name -n           [Required]: ACS cluster name.
    --resource-group -g [Required]: Name of resource group.
    --admin-username                : User name for the Linux Virtual Machines.  Default: azureuser.
    --agent-count                   : The number of agents for the cluster.  Note, for DC/OS clusters
                                      you will also get 1 or 2 public agents in addition to these
                                      seleted masters.  Default: 3.
    --agent-vm-size                 : The size of the Virtual Machine.  Default: Standard_D2_v2.
    --client-secret                 : The secret associated with the service principal. If --service-
                                      principal is specified, then secret should also be specified. If
                                      --service-principal is not specified, the secret is auto-
                                      generated for you and stored in ${HOME}/.azure/ directory.
    --dns-prefix -d                 : Sets the Domain name prefix for the cluster. The concatenation
                                      of the domain name and the regionalized DNS zone make up the
                                      fully qualified domain name associated with the public IP
                                      address.
    --generate-ssh-keys             : Generate SSH public and private key files if missing.
    --location -l                   : Location.
    --master-count                  : The number of DC/OS masters for the cluster.  Default: 3.
    --orchestrator-type             : The type of orchestrator used to manage the applications on the
                                      cluster.  Allowed values: Custom, DCOS, Kubernetes, Swarm.
                                      Default: dcos.
    --service-principal             : The service principal used for cluster authentication to Azure
                                      APIs. If not specified, it is created for you and stored in the
                                      ${HOME}/.azure directory.
    --ssh-key-value                 : SSH key file value or key file path.  Default:
                                      /Users/danlepow/.ssh/id_rsa.pub.
    --tags                          : Tags object.

Global Arguments
    --debug                         : Increase logging verbosity to show all debug logs.
    --help -h                       : Show this help message and exit.
    --output -o                     : Output format.  Allowed values: json, jsonc, list, table, tsv.
                                      Default: json.
    --query                         : JMESPath query string. See http://jmespath.org/ for more
                                      information and examples.
    --verbose                       : Increase logging verbosity. Use --debug for full debug logs.
```

```
{
    "apiVersion": "2015-11-01-preview",
    "type": "Microsoft.ContainerService/containerServices",
    "name": "MyContainerService",
    "location": "[resourceGroup().location]",
    "properties": {
      "orchestratorProfile": { "type": "mesos" },
ofile": {
: "3",
fix": "containerservicemgmt"

lProfile": [

me": "SmallPrivateCluster",
Size": "Standard_A1",
unt": "3",
sPrefix": "containerserviceapp"


rofile": {
linux",
fix": "containerservicejb"

e": {
sername": "azureuser",
assword": "password1234$",
onfiguration": {
: { "publicKeys": [ {"keyData": "AAAAB3NzaC1yc2E
```


CLI
UI
API
Master
API
ETCD
Scheduler
Controllers
Node
Kublet
POD
POD
POD
POD
Node
Kublet
POD
POD
POD
POD
POD
POD


Master
Master
Master
Node
Node
Node
Node
Node
Node
Prod
DEV
front
backend

But the Developer's View…



Demo

Aye captain!

Kubernetes

```
# login into you azure subscription
az login

# create an Azure Container Service (ACS) cluster with a kubernetes orchestrator
# az group create -n azurethacs -l westeurope
# az acs create --name k8athacs -g azurethacs --orchestrator-type kubernetes --orchestrator-version 1.8.7 --generate-ssh-keys

# create an Azure Managed Kubernets Container Service (AKS) cluster
az group create -n azureth -l westeurope
az aks create -n k8ath -g azureth -l westeurope -k 1.8.7 --generate-ssh-keys

# install the kubernets command line interface into your bash command
# az aks kubernetes install-cli

# connect to the kubernetes cluster
az aks get-credentials -g azureth -n k8ath
kubectl config get-contexts
kubectl get nodes

# SHOW ADMIN SITE
# az aks browse -n k8ath -g azureth
kubectl proxy
http://localhost:8001/ui

# deploy your container as a service onto the kubernetes cluster
kubectl create -f ./deployment.yaml
kubectl create -f ./service.yaml
kubectl get pods
```

DEPLOYMENT CONFIGURATION (deploy.yaml)

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name:  your-api-name
spec:
  replicas: 3
  strategy:
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1
    type: RollingUpdate
  template:
    metadata:
      labels:
        app:  your-app-name
        tier: your-tier-name
    spec:
      containers:
      - image:  your-docker-image-name-on-an-image-registry
        name: your-api-name
        ports:
        - containerPort:  80
          name:  api-port
```

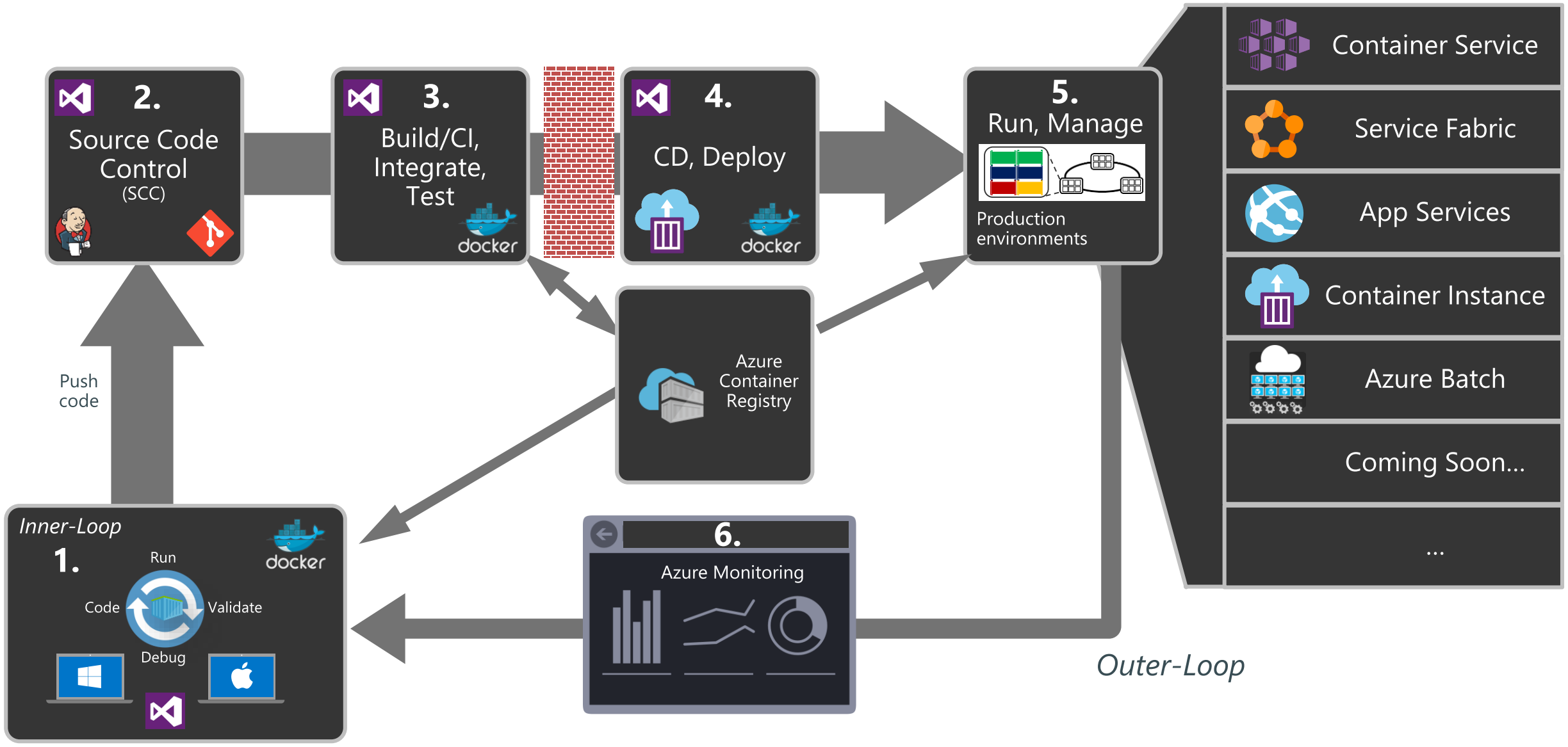
SERVICE CONFIGURATION (service.yaml)

```yaml
kind: Service
apiVersion: v1
metadata:
  name:  your-api-service-name
spec:
  selector:
    app:  your-app-name
    tier: your-tier-name
  type:  LoadBalancer
  ports:
  - name:  http
    port:  80
    targetPort:  80
```

CI/CD and DevOps workflow

Cluster Management Deploy and manage cluster resources	**Scheduling** When containers run	**Lifecycle & Health** Keep containers running despite failure	**Naming & Discovery** Where are my containers	**Load Balancing** Distribute traffic evenly
Scaling Make container sets elastic in number	**Image Repository** Centralized, secure container images	**Continuous Delivery** CI/CD pipeline and DevOps workflow	**Logging & Monitoring** Track events in containers and cluster	**Storage Volumes** Persistent data for containers

Containerized workflow



What is an Image Registry ?

- Stores container images
  - Images are **Pushed** into a registry
  - Images are **Pulled** from a registry
  - Images are **Searched** for within a registry

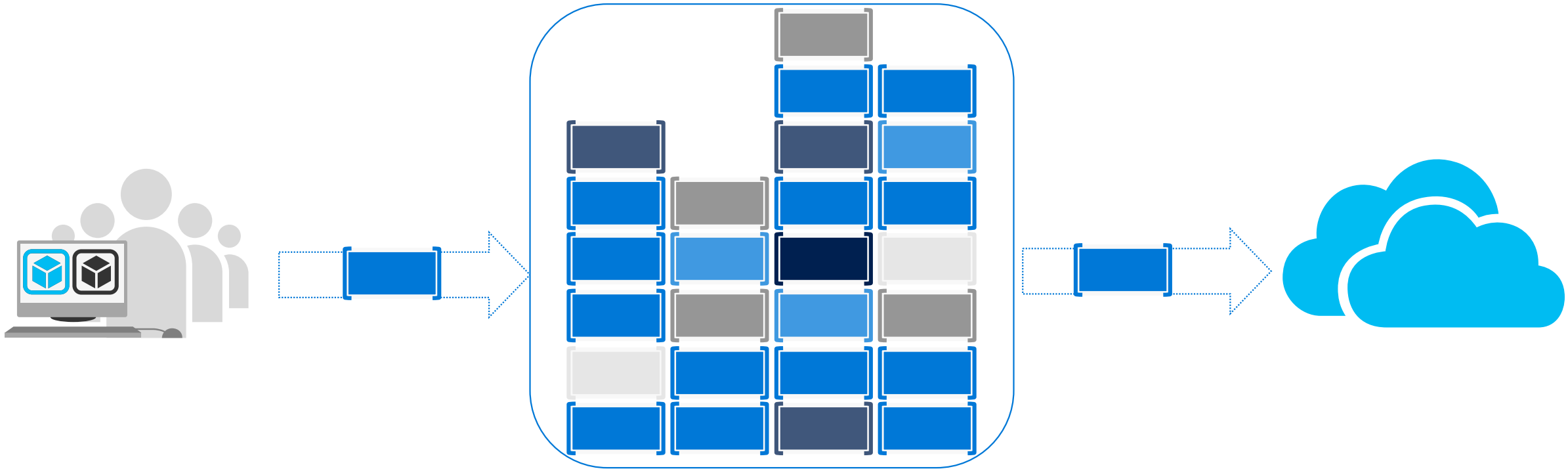
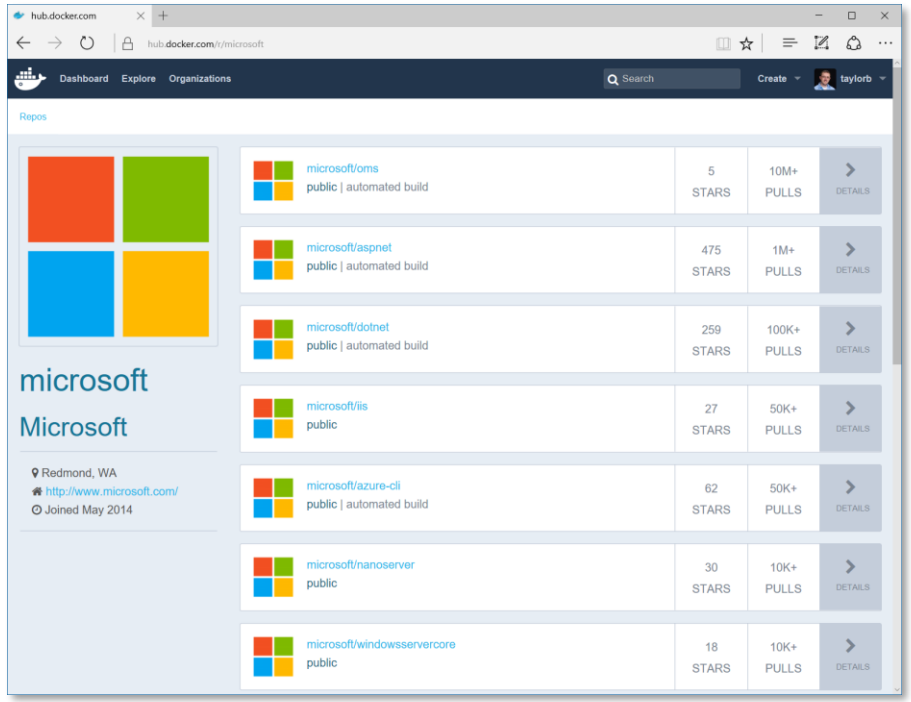


Image Registries



Docker Hub and Docker Store

- Public, Official and Private image repositories
- Granular access controls with organization support
- Automated image build support

Docker Trusted Registry

- Enterprise Grade Private Registries
- Runs on your infrastructure (on-prem or cloud)
- Active Directory and Role Based Access Controls

Azure Container Registry

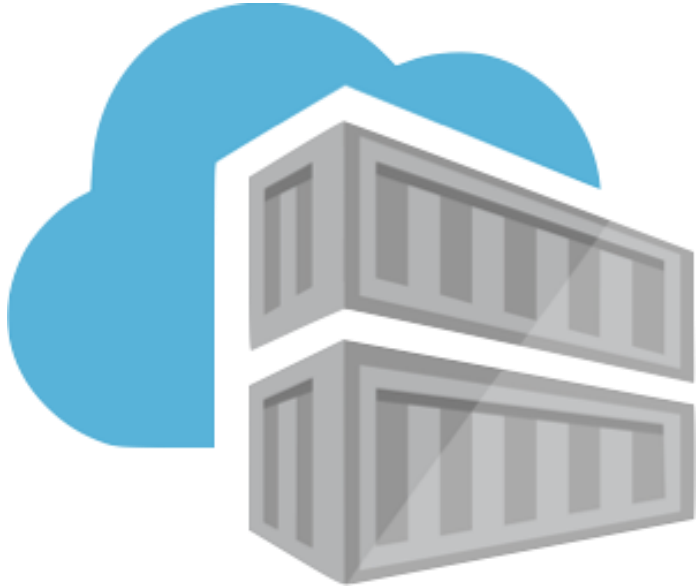
- Store and manage container images across Azure deployments
- Maintain Windows and Linux container images
- Same API and Tools as Docker Hub/Store/Registry

Docker Registry

- Open source foundation of Hub and DTR
- Runs on your infrastructure (on-prem or cloud) as a container
- https://docs.docker.com/registry and or https://github.com/docker/distribution

Azure Container Registry

- Private, secure, network-close registry for Docker images
- Azure Active Directory Integration
- Windows and Linux containers
- Command-line, VSTS & API driven



Azure Container Registry

Based on open source Docker Registry 2.0

- Backed by Azure Storage

Use Standard Docker commands
Linux and Windows containers supported

Easily distribute images to orchestration platforms and Azure services

- ACS, Docker, Kubernetes, DC/OS, etc.
- Service Fabric & Azure App Service

Access Control

- Azure AD backed service principal: Reader, Contributor, Owner

Container Image Security

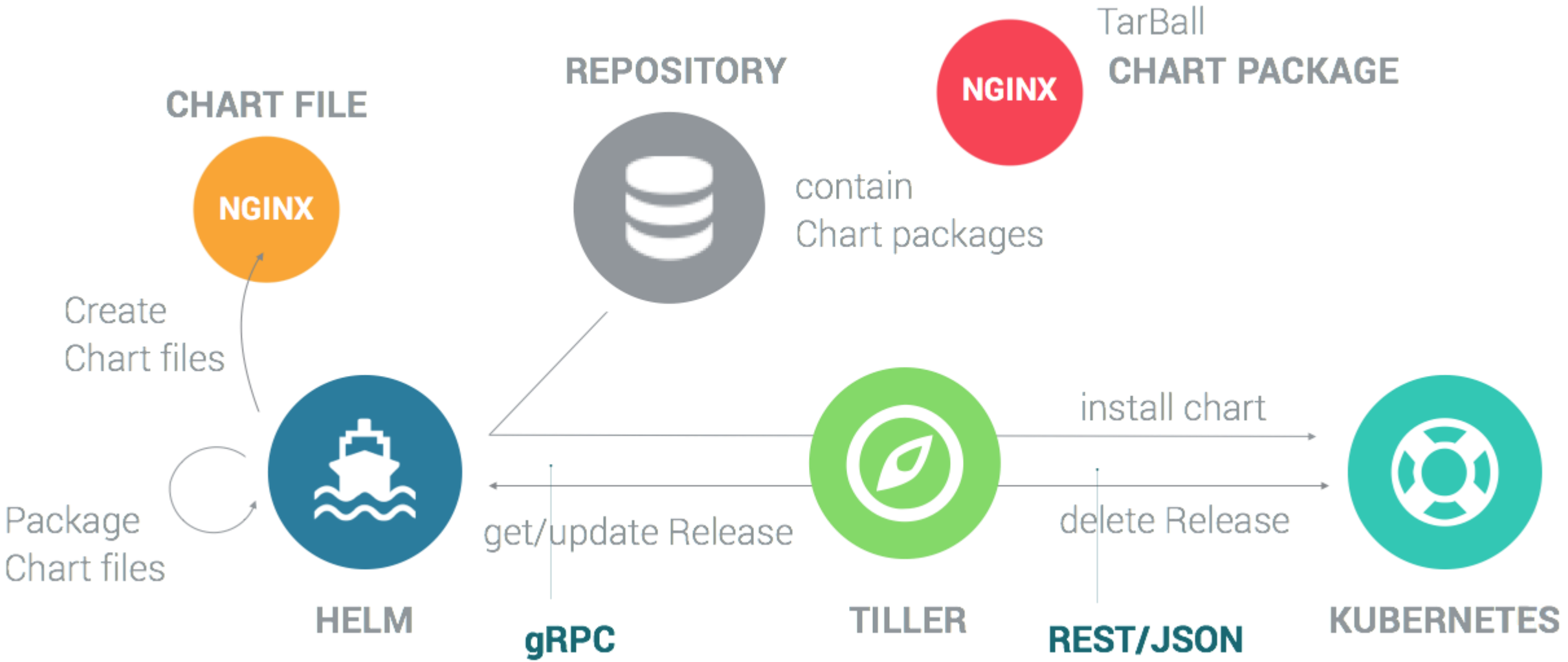
- 3rd Party Solutions (Aqua, Twistlock, etc.)
- IaaS based solutions in Azure Marketplace

Helm package manager

Charts, repositories, releases…

- Chart
  A bundle of Kubernetes resources
- Repository
  A collection of released charts
- Release
  A chart instance deployment to Kubernetes.

Helm package manager



ilionx
Hondiuslaan 46
3528 AB Utrecht
T (088) 05 90 500
E info@ilionx.com

www.ilionx.com