Survey of Cloud Computing

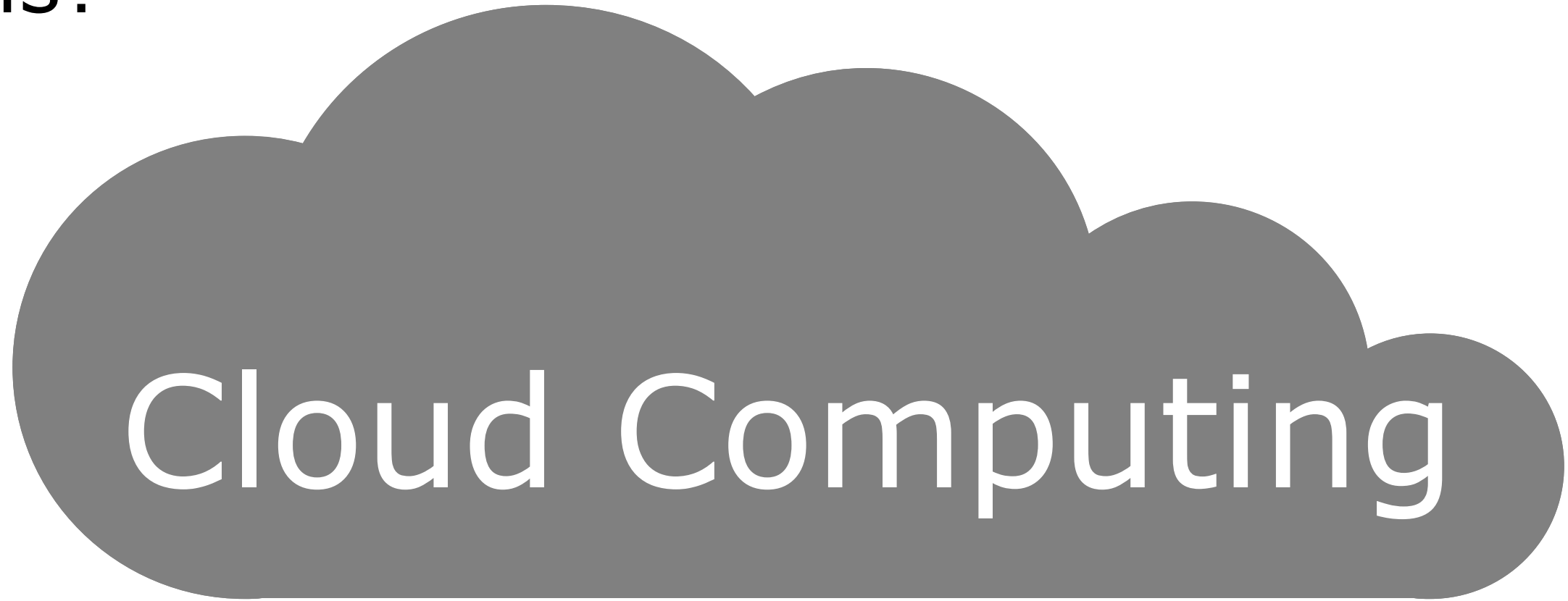
Cloud Computing Overview

What is Cloud Computing?

"The practice of using a network of remote servers hosted on the Internet to store, manage, and process data, rather than a local server or a personal computer."

Oxford Dictionary

What do you think Cloud Computing is?



Cloud Computing Perspectives

Perspectives highly influenced by roles and responsibilities within an organization

- End-User
- Application Developer
- IT Infrastructure Manager
- CIO
- CFO
- Service Provider

What is Cloud Computing? – Take 2

Further perspectives include:

- “An approach to computing that’s about Internet scale and connecting to a variety of devices and endpoints.”
- “Treating hardware and software resources as a utility.”
- “A way to save a ton of money by only paying for what you need.”
- “A way to scale huge when you need something done fast.”

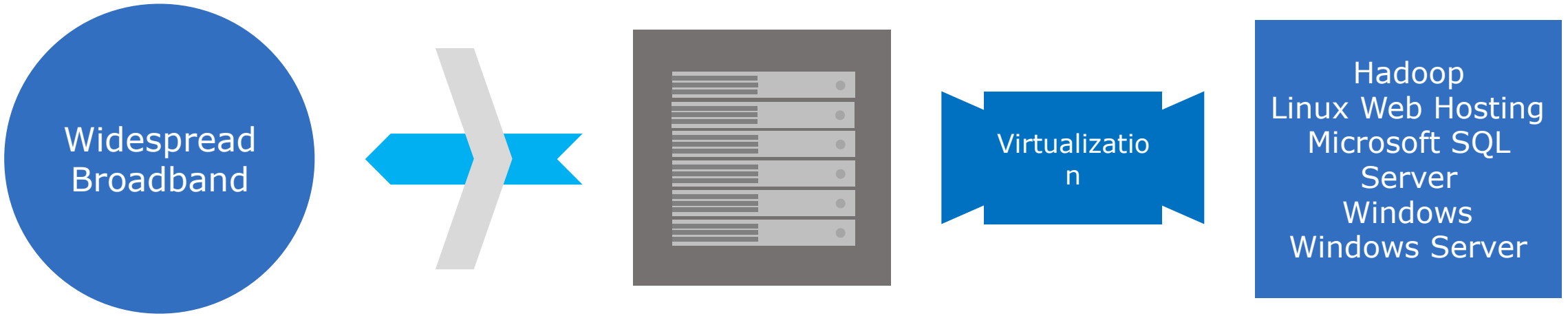
Evolution of Cloud Computing

Order of Evolution

Stage	Characteristics
Grid Computing	Solving large problems with parallel computing Made mainstream by Global Alliance
Utility Computing	Computing resources offered as a metered service Late 1990s
Software as a Service	Subscription-based software accessed over the Internet Gained momentum after 2001
Cloud Computing	Next-generation datacenters with virtualization technology Full stack of service - IasS, PaaS, & SaaS

Key Enabling Technologies

- Ubiquitous fast wide-area networks
- Powerful and inexpensive servers
- High-performance virtualization technology



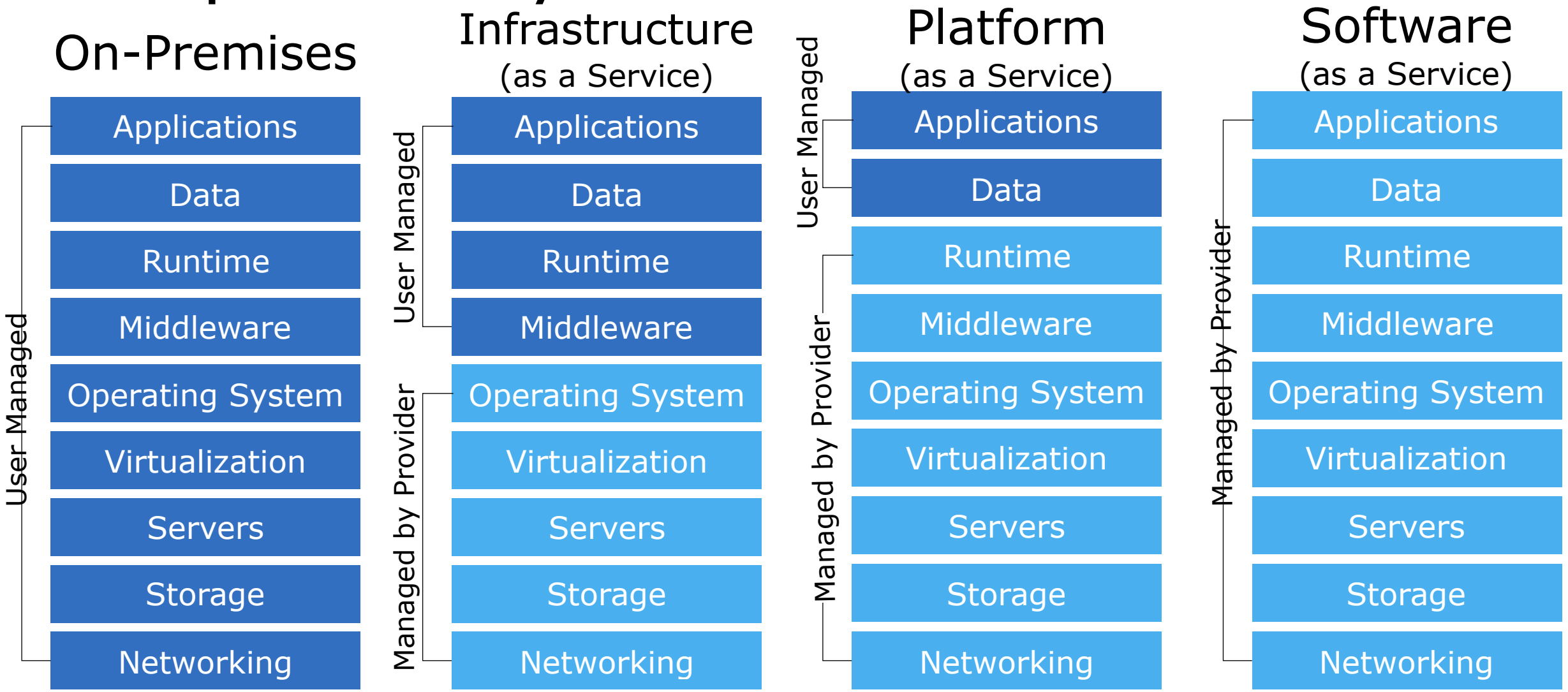
Five Key Cloud Characteristics

- On-demand self-service
- Ubiquitous network access
- Location-independent resource pooling
- Rapid elasticity
- Pay for what you use

Cloud Computing Service Models

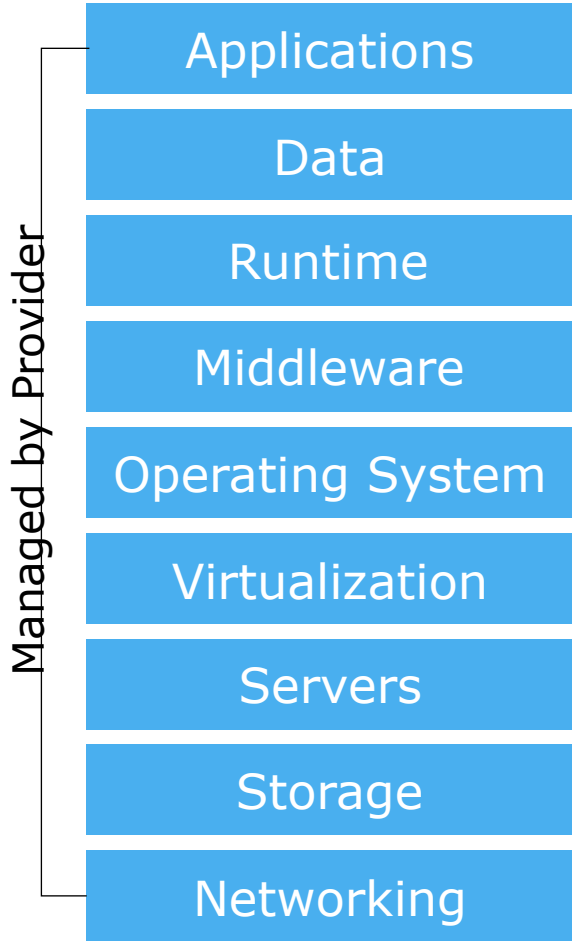
Model	Description
Software as a Service (SaaS)	Consume it End-User Applications delivered as a service, rather than by on-premises software
Platform as a Service (PaaS)	Build on it Application platform or middleware provided as a service on which developers can build and deploy custom applications
Infrastructure as a Service (IaaS)	Migrate to it Computing, storage, or other IT infrastructure provided as a service, rather than as a dedicated capability

Service Model Division of Responsibility



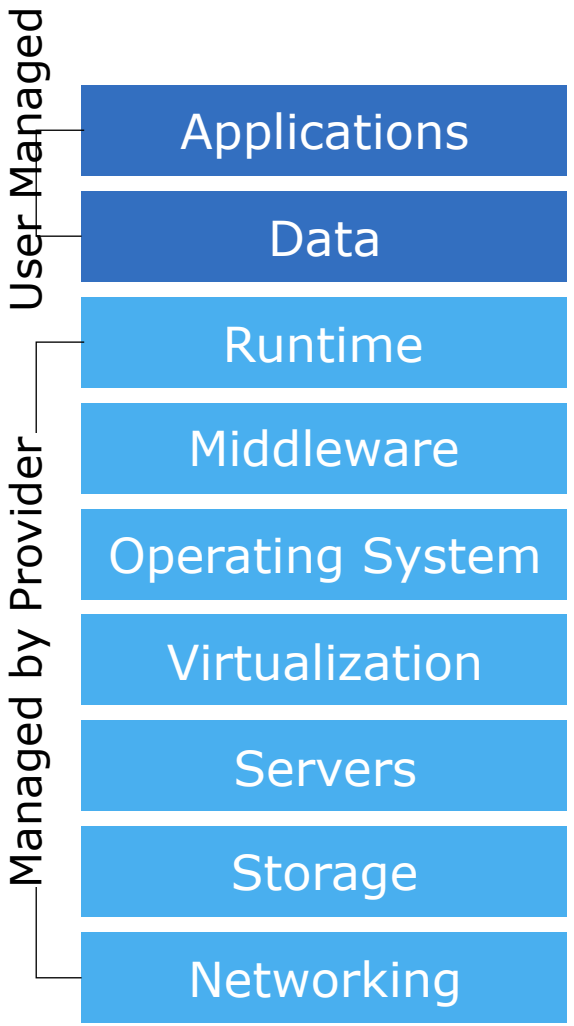
Software as a Service (SaaS)

- Internet hosted software
- Full vendor maintenance
- No upfront cost
- Pay for services as they are consumed



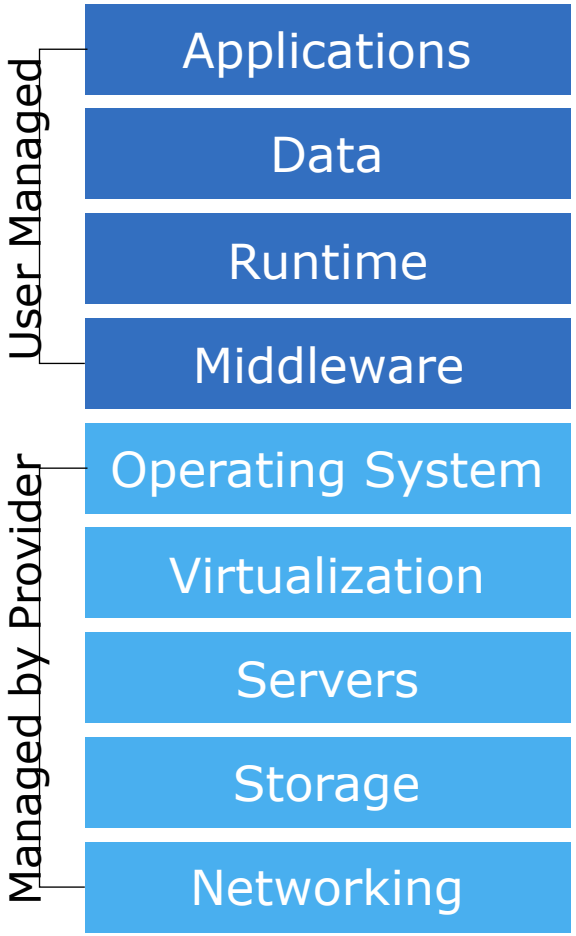
Platform as a Service (PaaS)

- Delivers and manages various development environments
- Environment and tools can be easily provisioned and torn down

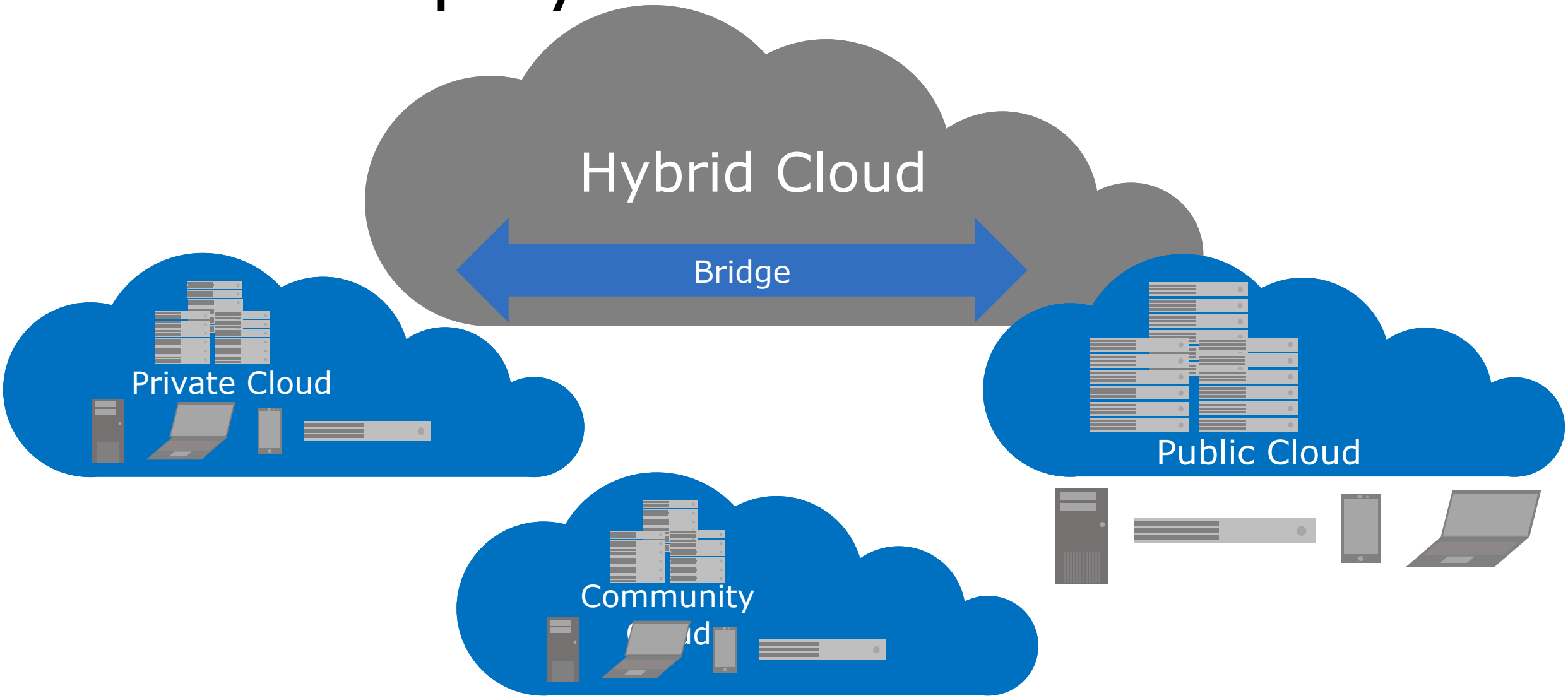


Infrastructure as a Service (IaaS)

- Dedicated virtual machines (VMs)
- Users configure server type, operating system, storage, network, etc.
- Scale up and down



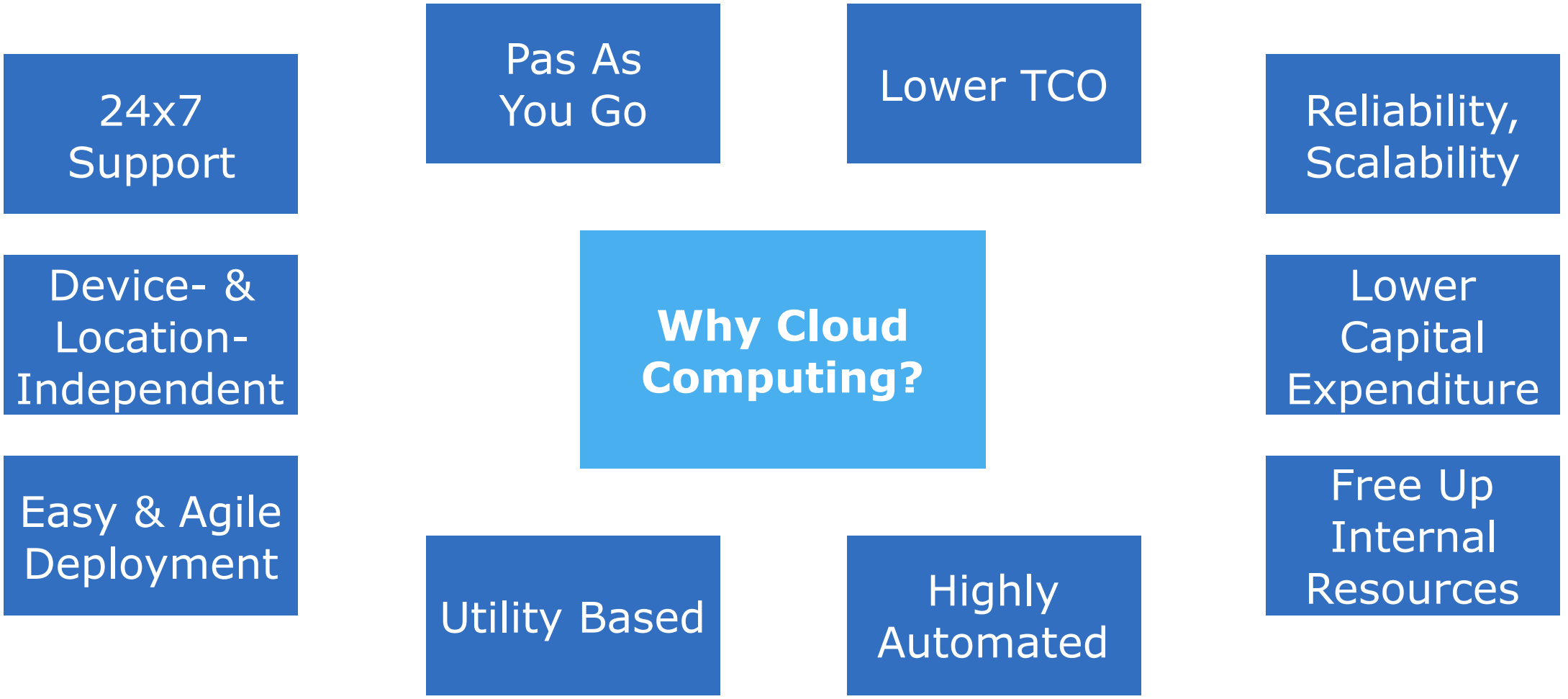
Cloud Deployment Model



Cloud Deployment Models – Advantages & Characteristics

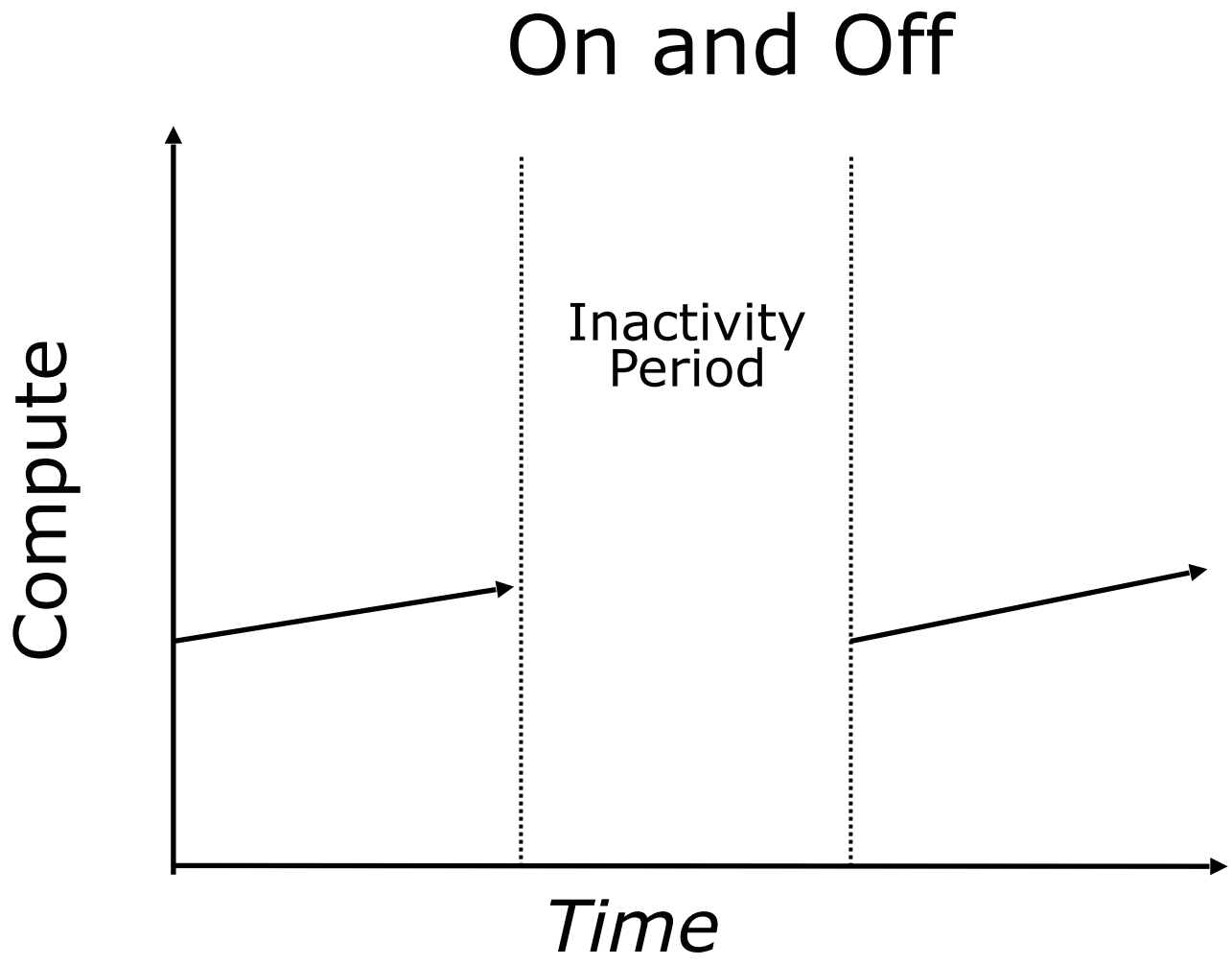
Model	Advantages and Characteristics
Public	Shifts capital expense to operating expense Offers pay-as-you-go pricing Supports multiple tenants
Private	Leverages existing capital expense Can help reduce operating costs Intended for a single tenant
Hybrid	Bridges one or more community, private, or public clouds Allows manipulation of CapEx and OpEx to optimize cost Supports resource portability
Community	Allows sharing of CapEx and OpEx to reduce costs Brings together groups with a common interest Supports resource portability

Why Cloud Computing?



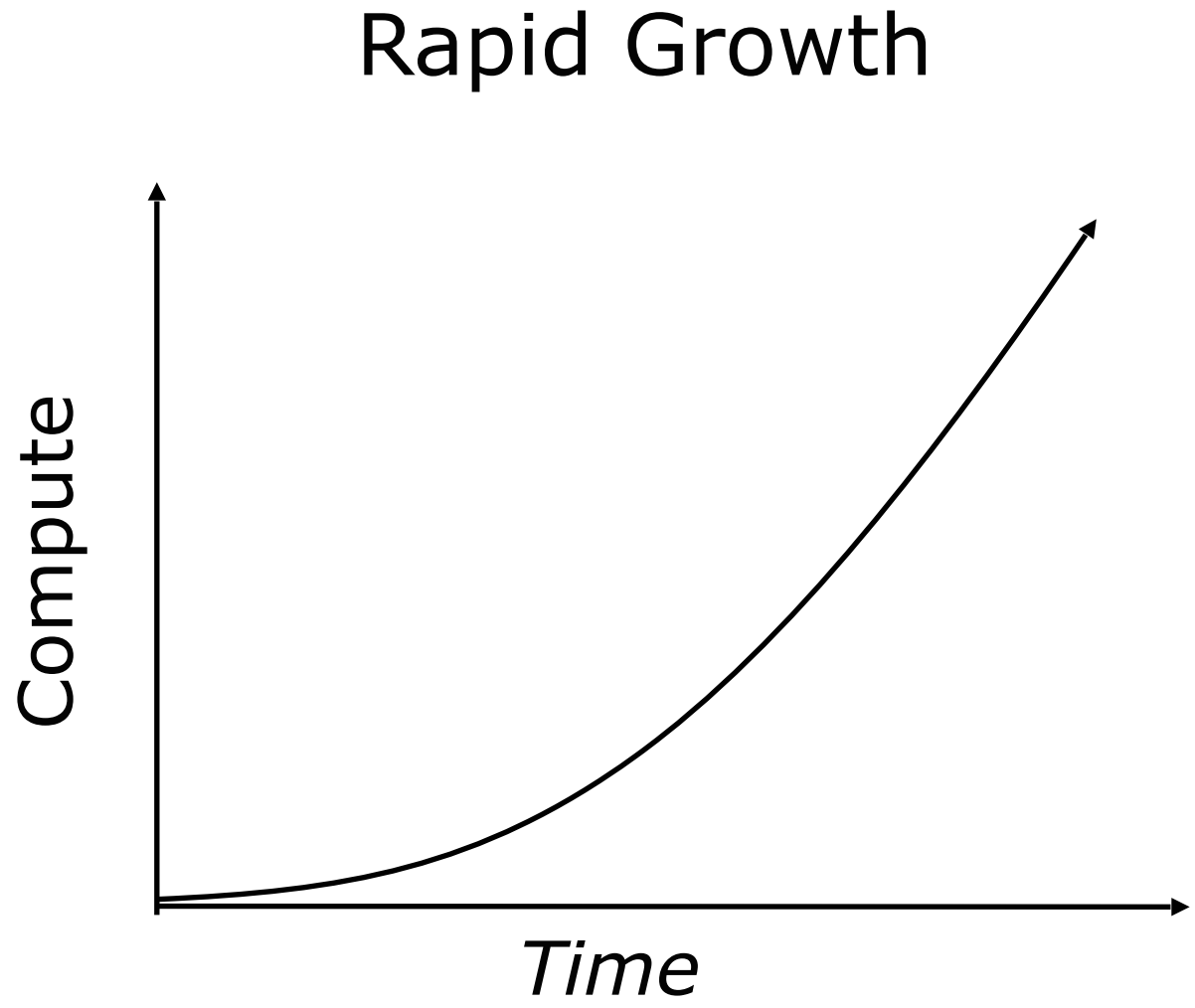
Typical Computing Pattern

- On & off workloads
  - Batch jobs
- Wasted Capacity
- Time to market can be cumbersome



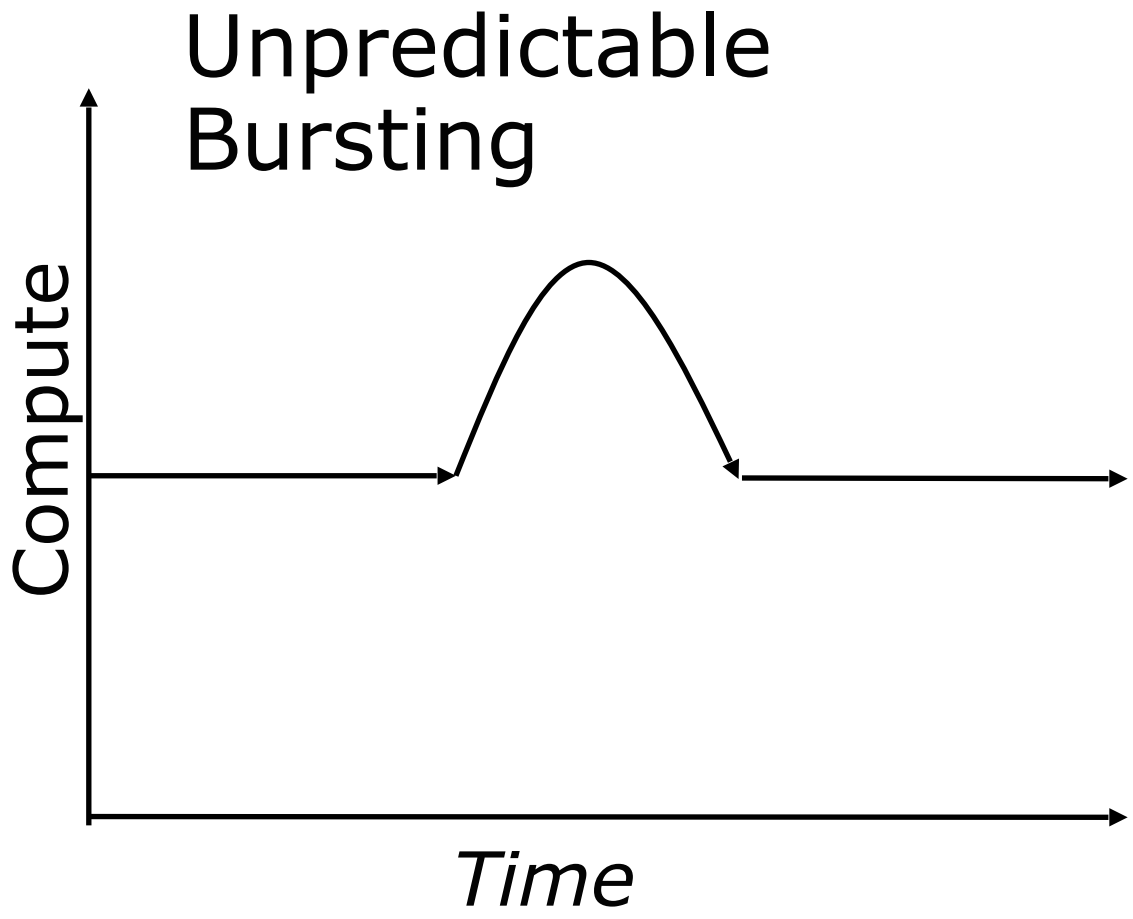
Typical Computing Pattern

- Rapidly growing company
- Major challenge for IT dept. to keep up with growth
- Potential loss of business opportunity
- Potential customer service problems



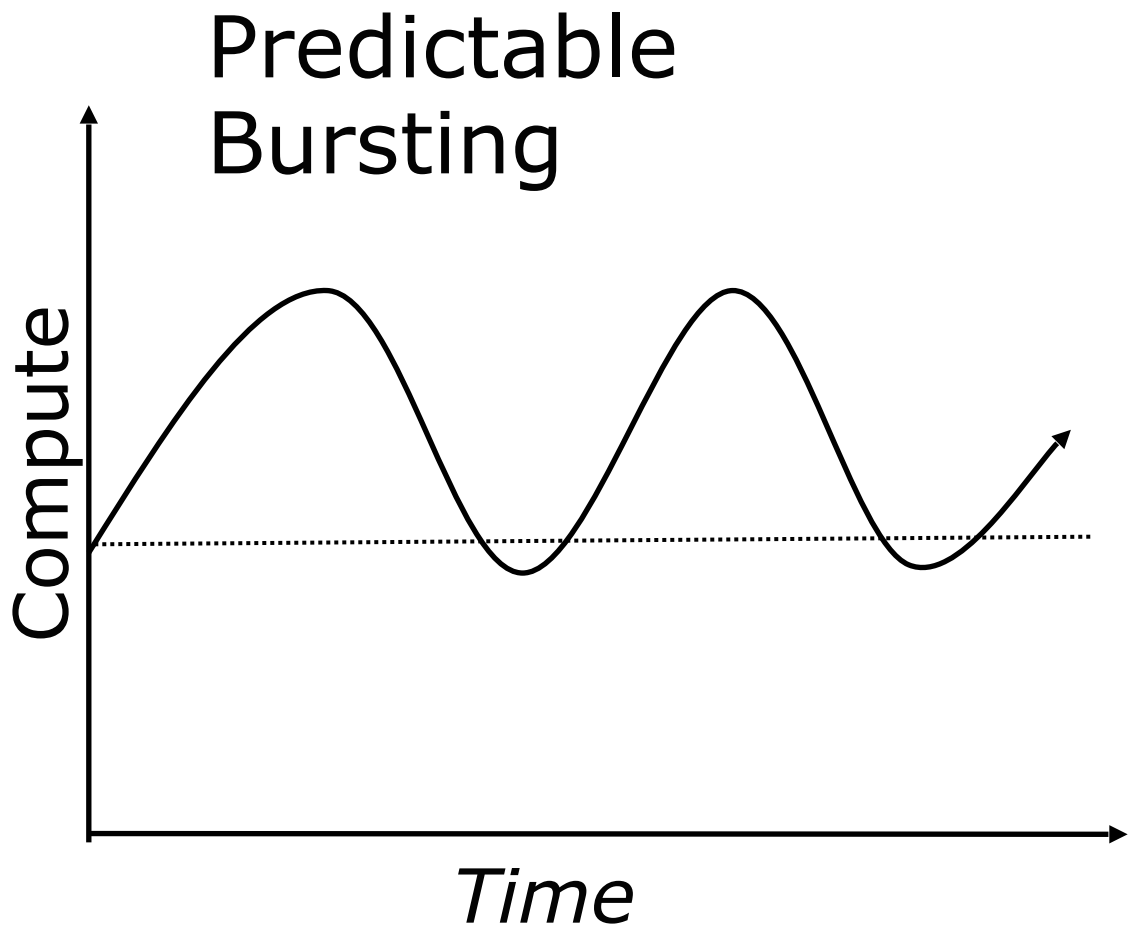
Typical Computing Pattern

- Unexpected peak in demand
- Loss of business opportunity
- Wasted capacity if demand wanes

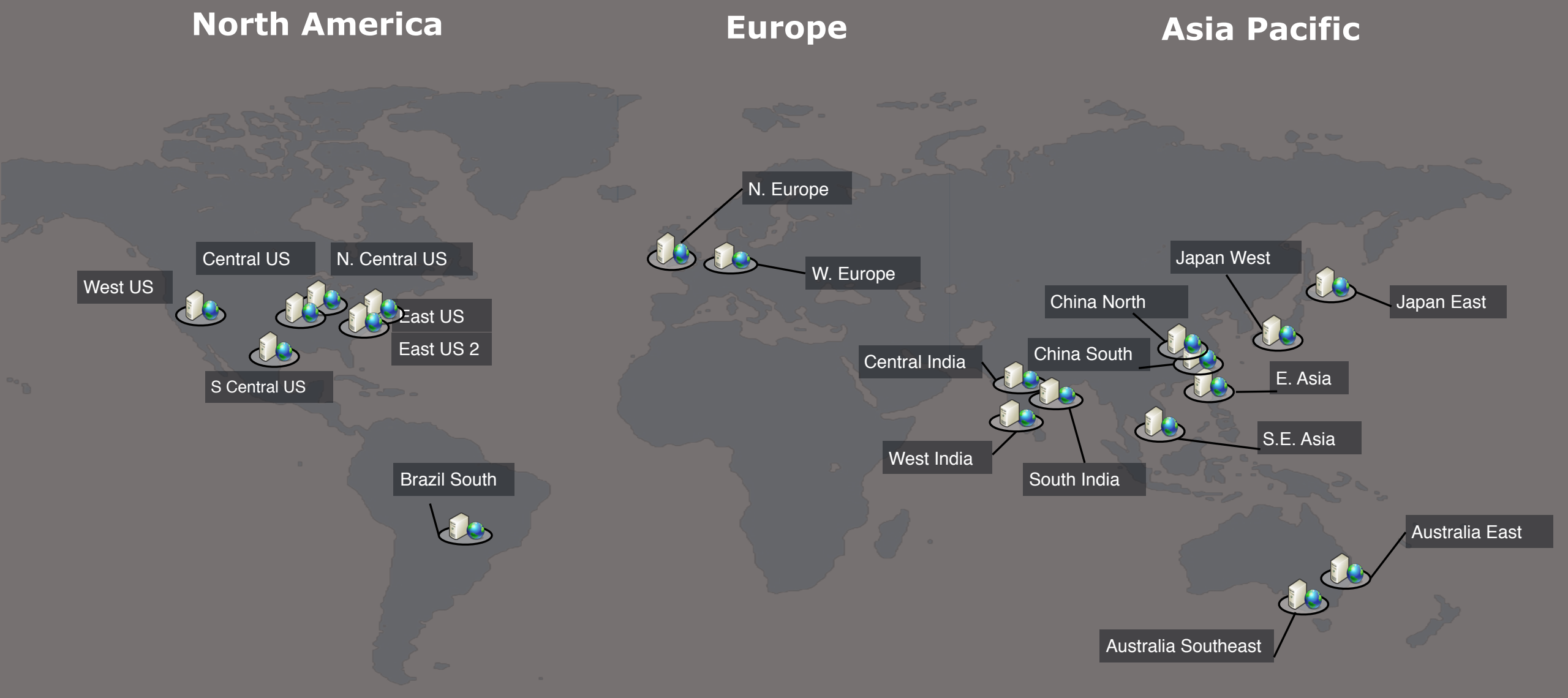


Typical Computing Pattern

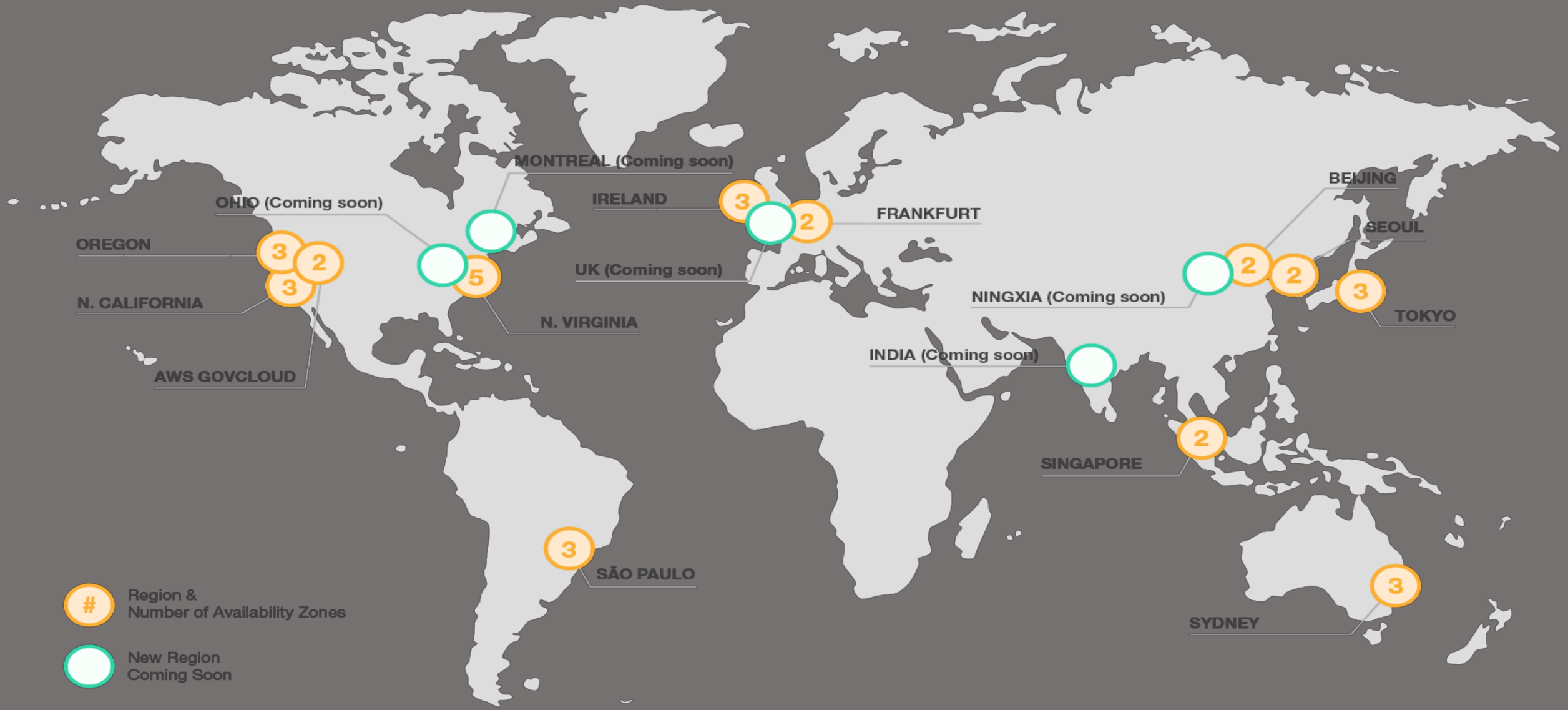
- Seasonal peaks and troughs
- Provisioning dilemma
  - Wasted capacity or
  - Loss of business



Azure Datacenter Regions



Amazon AWS Datacenter Regions

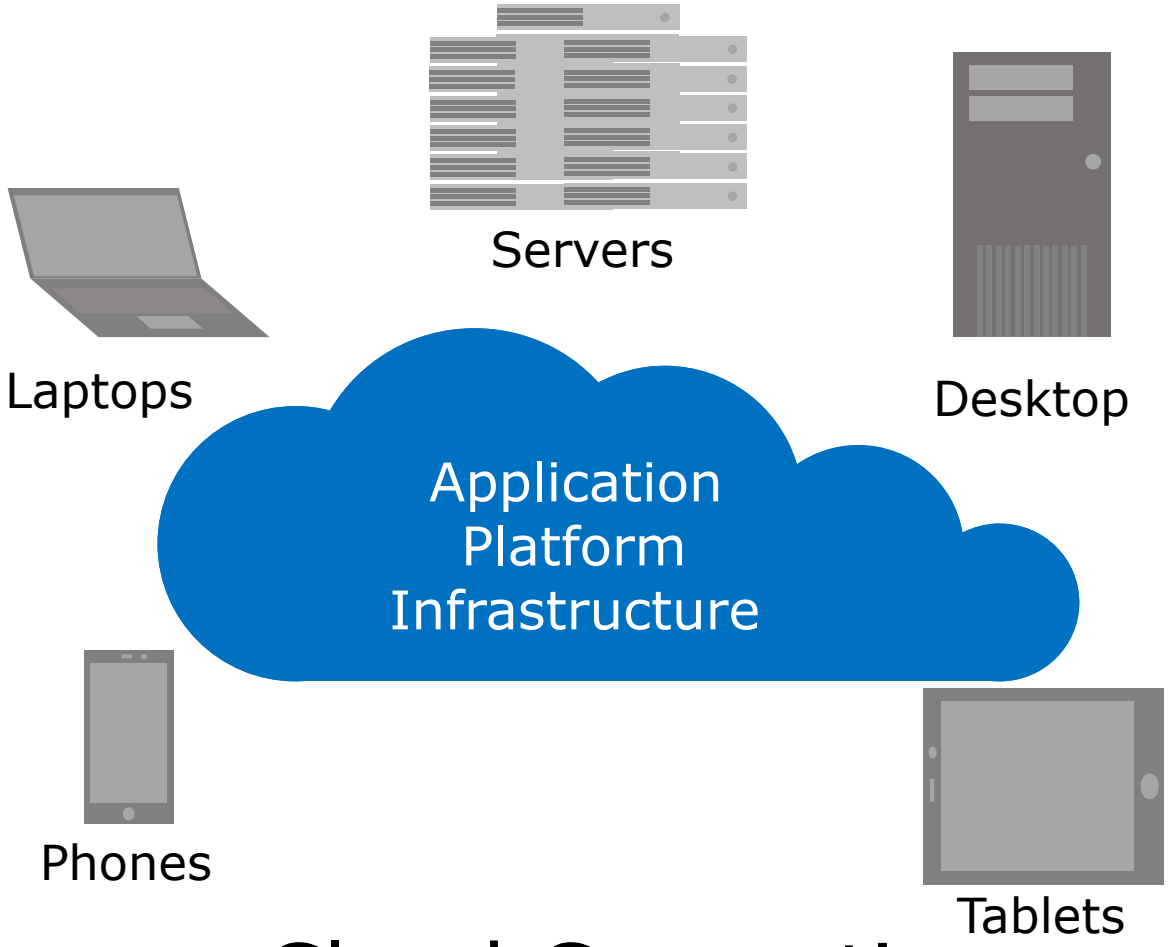


Cloud Computing Examples

- A large enterprise quickly & economically deploys new internal applications to its distributed workforce.
- An e-commerce website accommodates sudden demand for a “hot” product caused by a viral buzz.
- A pharmaceutical research firm executes large-scale simulations using computing power provided by cloud vendors.
- A media company serves unlimited video, music, and other media to their worldwide customer base.

Cloud Computing Nutshell

- End-users connect over the Internet to the cloud from their own personal computers or portable devices in order to access services.
- To the end-user, the underlying infrastructure such as the hardware, operating system, etc., is invisible



Cloud Computing

Cloud Vendor - Azure & AWS

Microsoft Azure and Amazon Web Services (AWS) offer broad and deep capabilities with global coverage

Category	Azure Service	AWS Service
Computing infrastructure	Virtual Machines	EC2
Object storage infrastructure	Blob Storage	S3
Networking	Virtual Network	Virtual Private Cloud
Relational database-as-a-service	SQL Database	RDS
NoSQL document database	DocumentDB	DynamoDB
Big data processing	HDInsight	Elastic MapReduce (EMR)
Visualization	Power BI	QuickSight

Cloud Vendor - Bluemix & Google

IBM Bluemix and Google Cloud each offer and deploy applications on highly-scalable and reliable infrastructure

Category	Bluemix	Google Service
Computing infrastructure	Virtual Server, Containers	Compute Engine
Object storage infrastructure	Object, Block Storage	Cloud Storage
Networking	Virtual Private Network	Cloud Virtual Network
Relational database-as-a-service	SQL Database	Cloud SQL
NoSQL document database	MongoDB	Cloud Datastore, Bigtable
Big data processing	Analytics for Apache Hadoop	BigQuery, Cloud Dataproc
Visualization		

Vendor Lock-In

Companies that adopt cloud computing must be wary of potential vendor lock-in issues

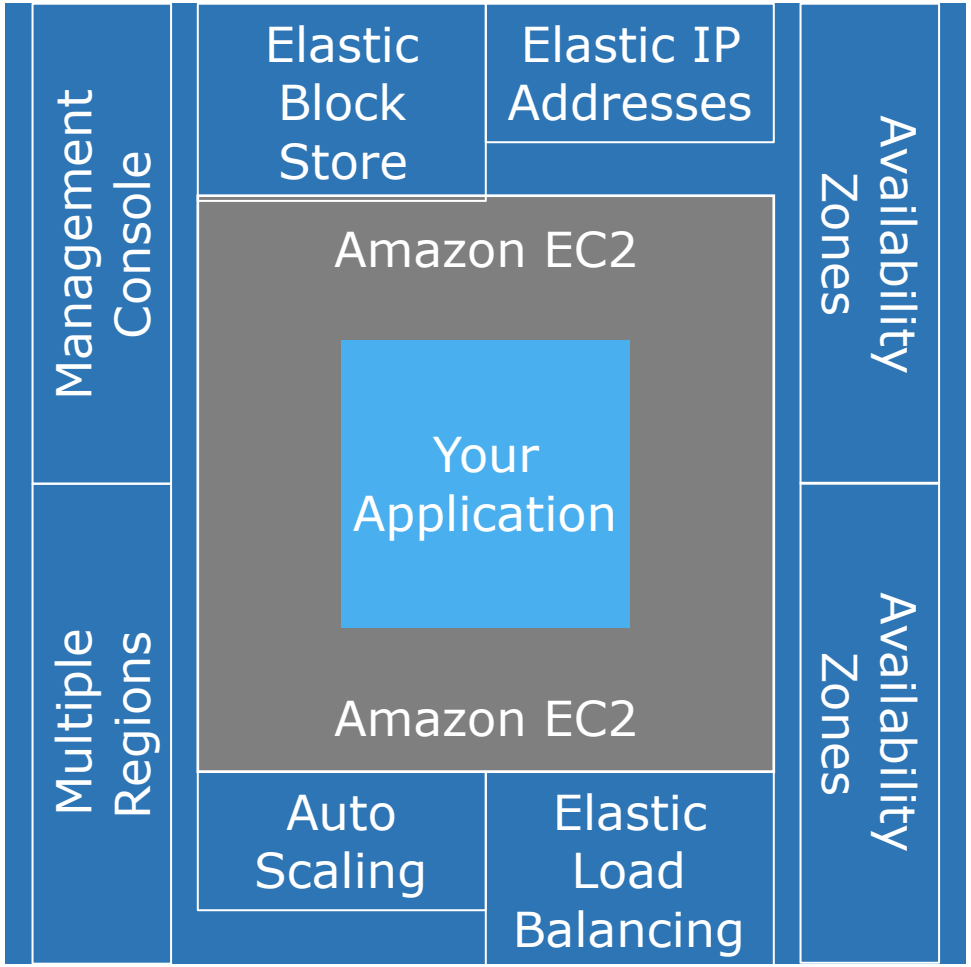
- Company’s entire data is stored with a single vendor’s cloud storage
- Company relies on a single vendor for all of its computations
- Changing vendors can be very costly

Survey of Cloud Computing

Cloud Computing Services

AWS EC2 and S3

- Amazon Elastic Cloud Computing
  - Resizable and elastic compute capacity in the cloud
  - Abundant community images
- Amazon Simple Storage Service
  - Online file storage web service
  - Accessible through REST, SOAP. and BitTorrent
- Elastic load balancing
- Automatic scaling

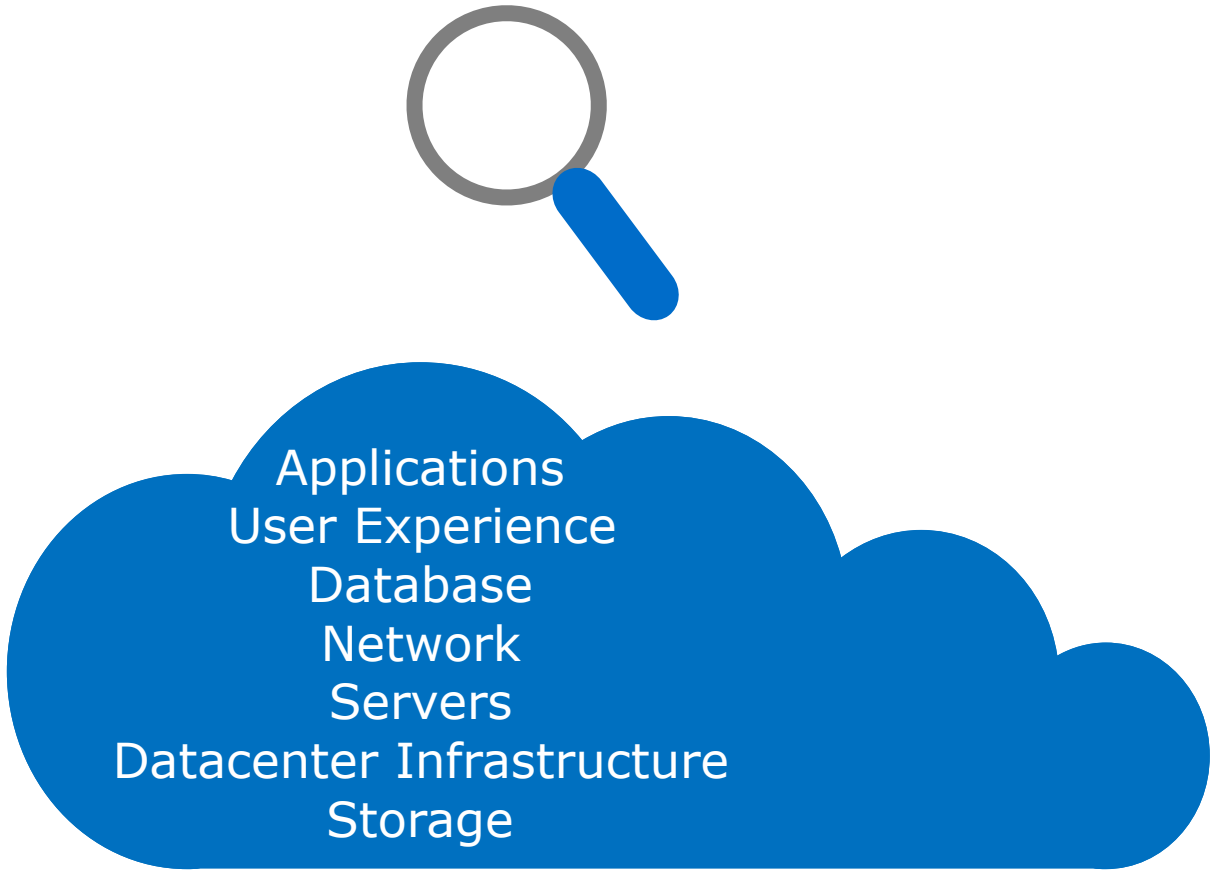


Desktop (as a service) Cloud Service

- Provides and manages a virtual desktops
- Allows smaller companies who find Virtual Desktop Infrastructure (VDI) to be cost prohibitive to deliver similar services
- Quickly deploy new solutions across the entire enterprise located in multiple regions

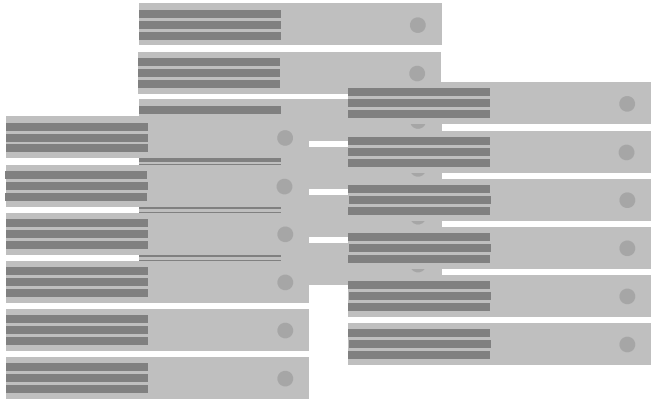
Monitoring (as a service) Cloud Service

- Consists of tools and applications meant to monitor certain aspects of an application, server, system, or any other IT component
- State monitoring is the most common service
  - The state of a component is constantly evaluated and results are displayed in real time



Metal (as a service) Cloud Service

- A bare metal provisioning system to rapidly deploy physical servers
- *Flexibility* of cloud computing with *power* of actual physical servers
- Serves as a layer underneath IaaS

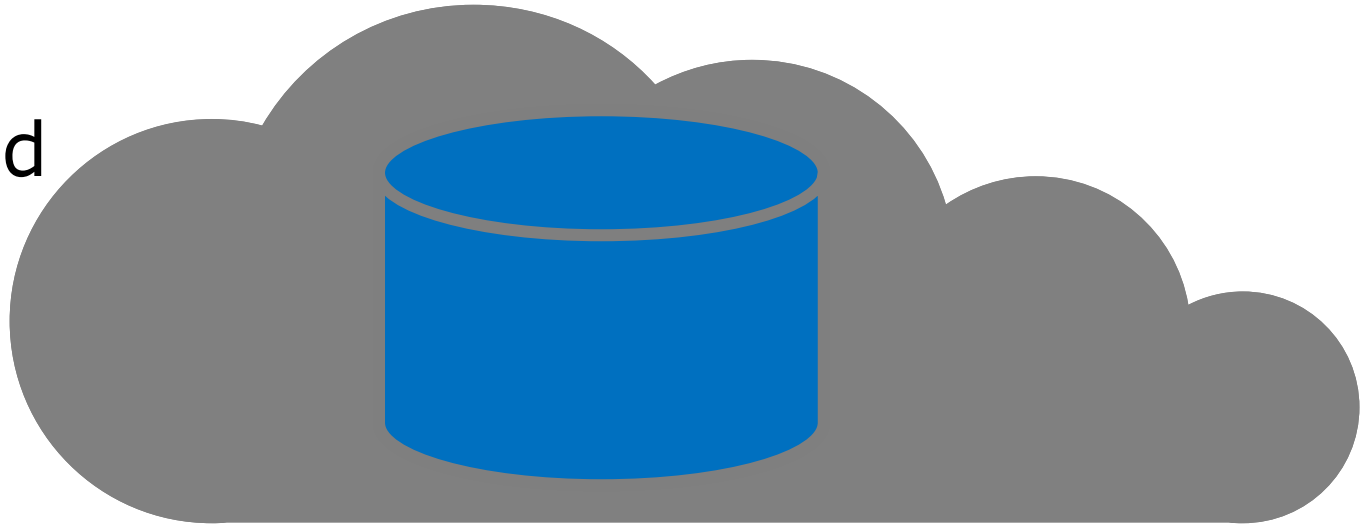


Communication (as a service) Cloud Service

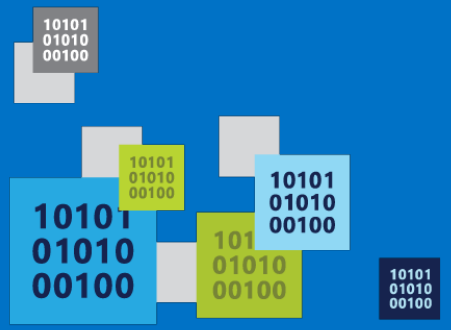
- Includes enterprise communication solutions such as VoIP (Voice over IP), instant messaging (IM) and video conferencing tha can be leased
- Vendor is responsible for all hardware and software and offers guaranteed Quality of Service

Database (as a service) Cloud Service

- Cloud-based approach to the storage and management of structured and unstructured data
- Rather than offering raw storage platforms, this service offers functionality of database platforms such as SQL Server, MySQL, Oracle, and NoSQL

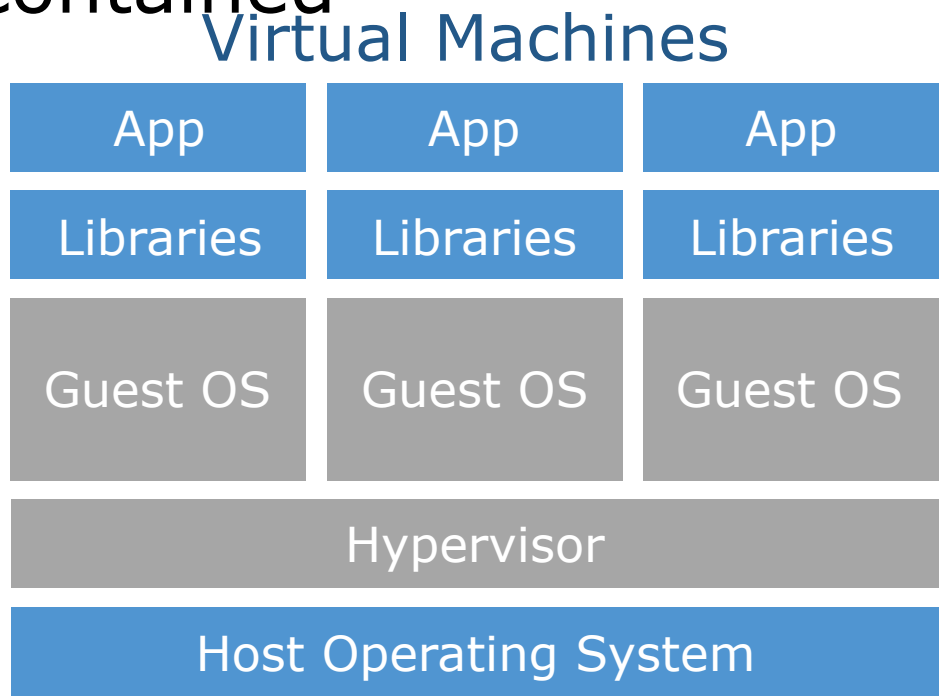


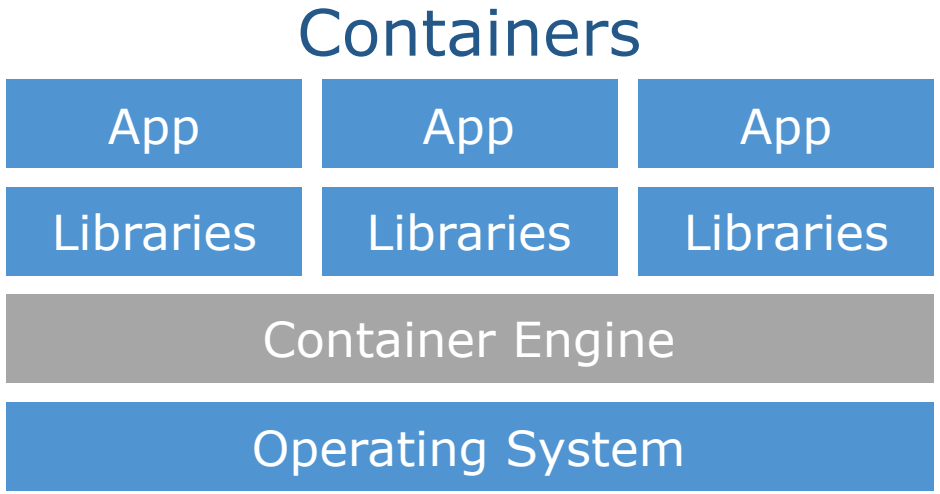
Building Cloud Solutions



Containers

- Lightweight alternative to virtual machines
- Smaller, less expensive, faster to start up, and self-contained



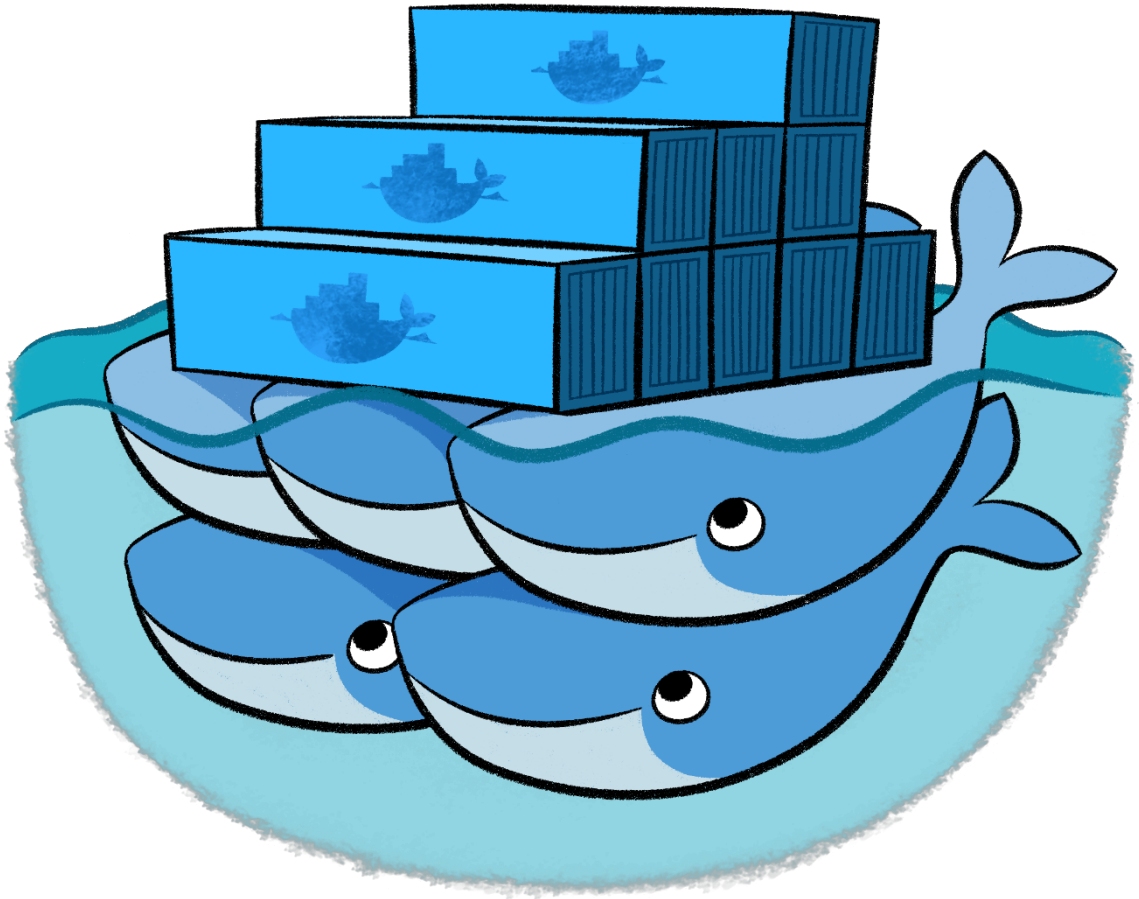


Docker

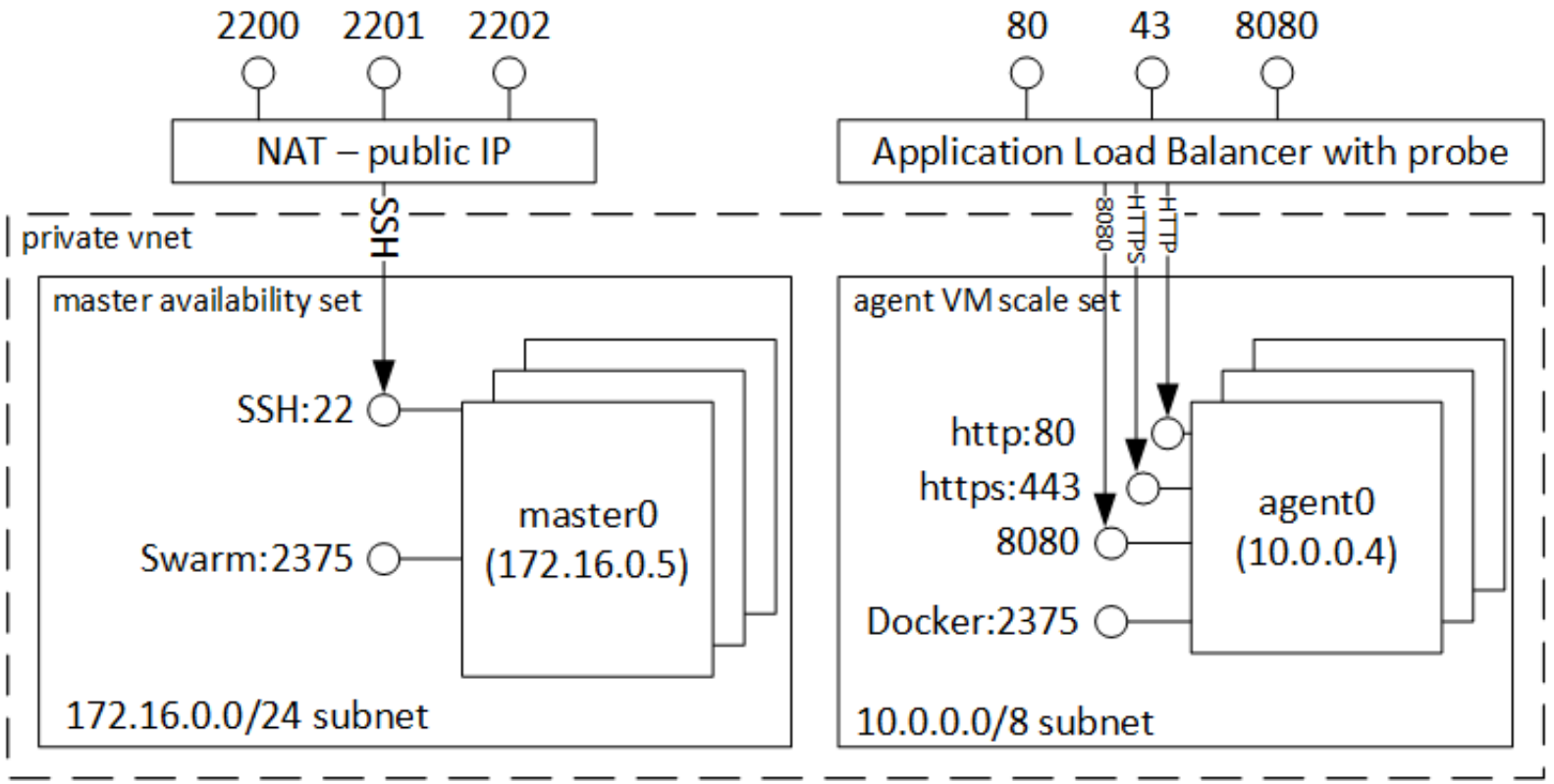
- Leading open-source containerization platform

  Docker containers wrap up a piece of software in a complete filesystem that contains everything it needs to run: code, runtime, system tools, system libraries – anything you can install on a server. This guarantees that it will always run the same, regardless of the environment it is running in

- Supported natively in Azure via Azure Container Service

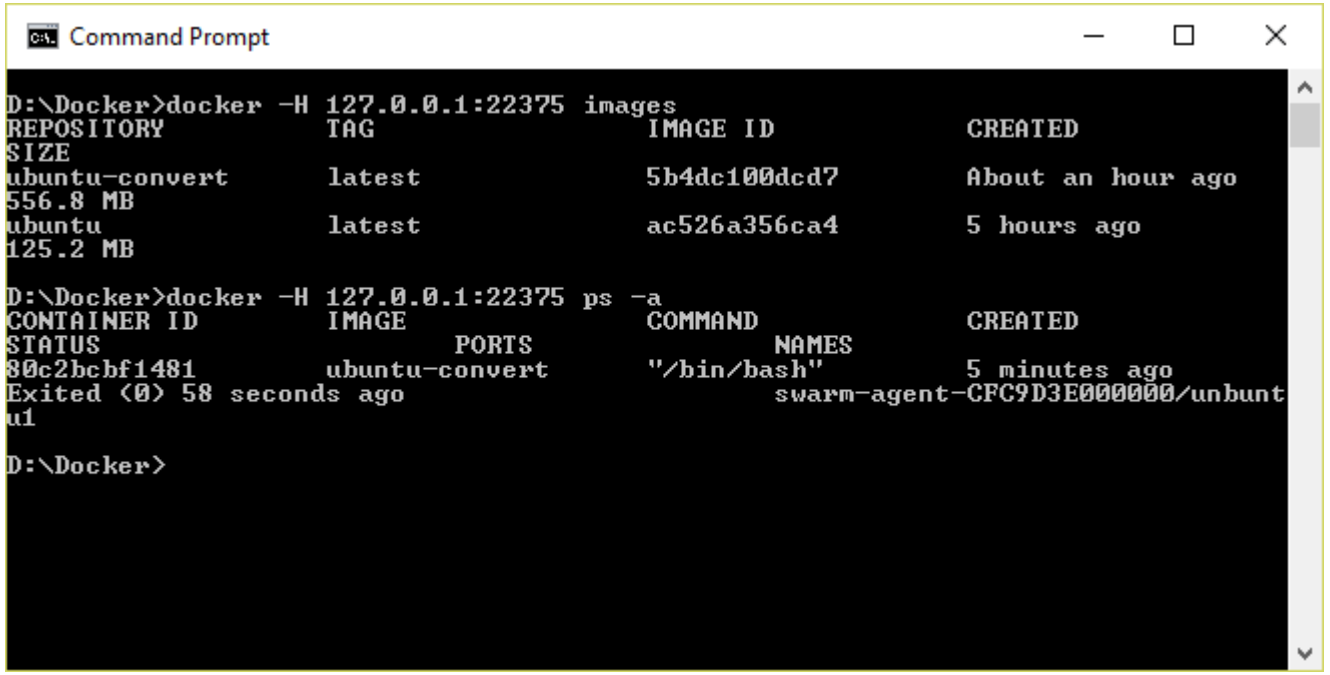


Clustering with Docker Swarm



Docker Client

- Command-line interface for Docker, available for Linux, OS X, and Windows (available separately or as part of Docker Toolbox)



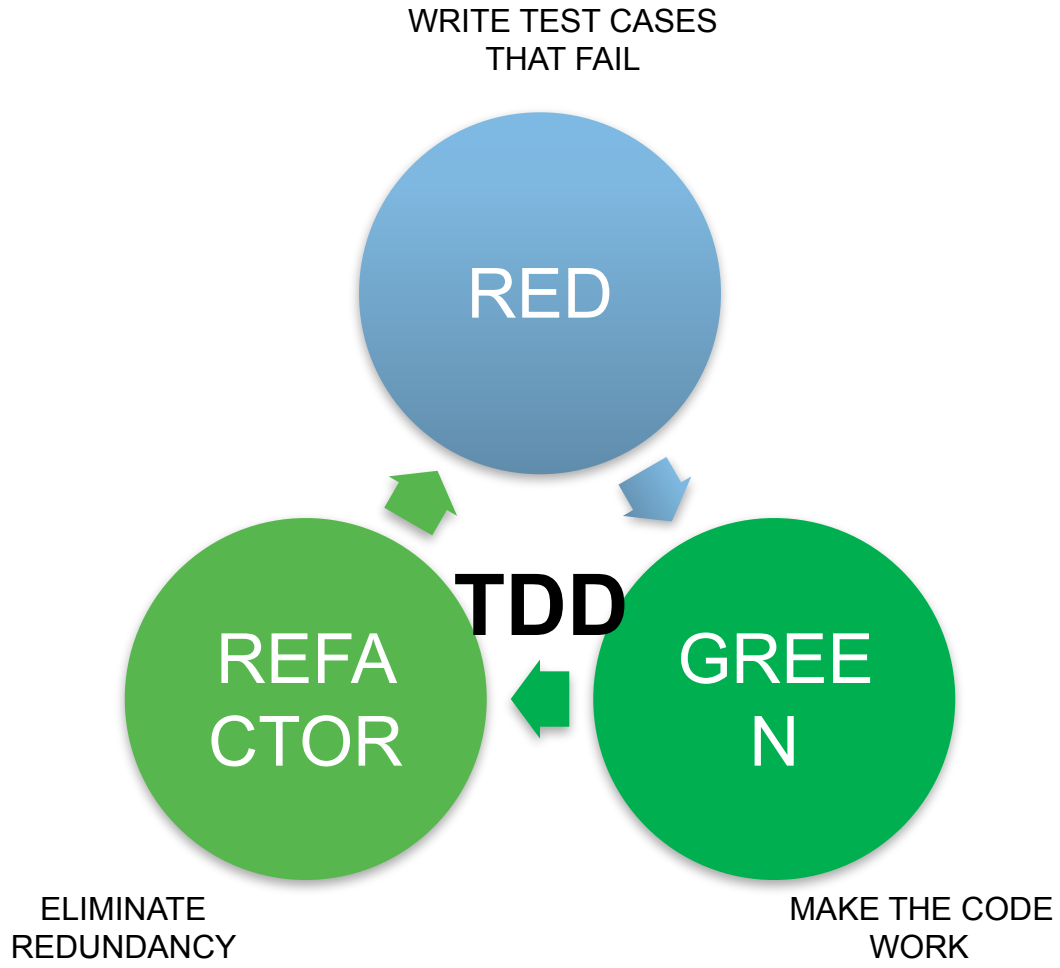
DevOps and Other Frameworks

Agile

- Tip to tail: Dev to Customer
  - Pain points
- The three pillars of the Agile stool:
  - Automation
  - Testing
  - Skill
- DevOps fits like a glove

Dev/Test Cycle

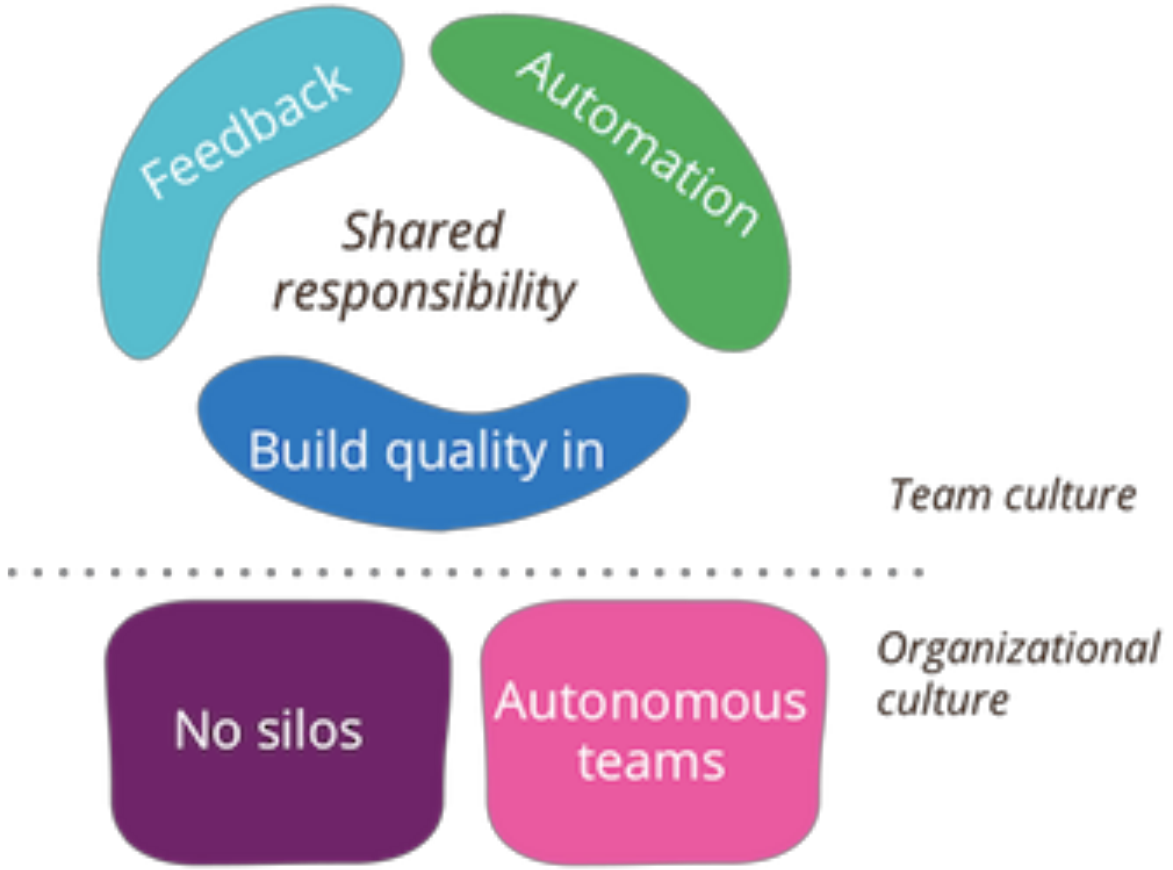
- Zealots of automation
- Time is short—getting to market needs a well-paved path
- Nothing manual
- Especially Q/A



DevOps Culture

Culture

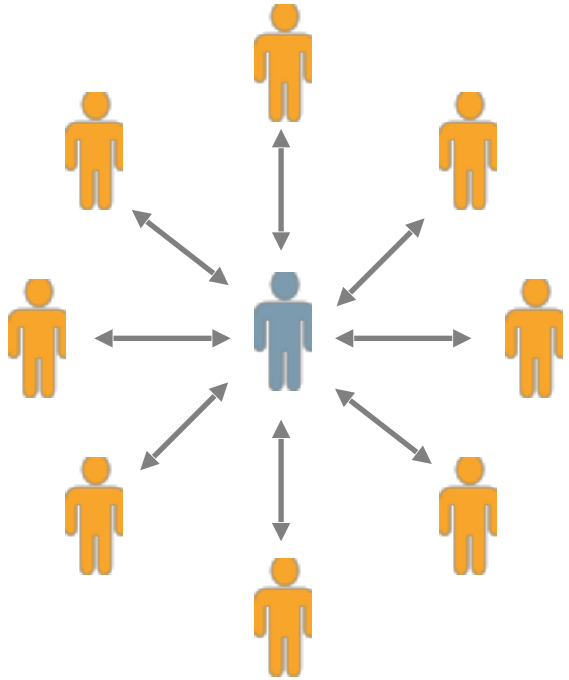
- Thinking in Systems
- Feedback Loops
- Continuous experimentation



Foundations

- Version control
- Automated like a zealot
- Visibility:
  - Everything in the open
  - Transparency
- Measure everything

Organization Items



GROUPING DYNAMIC

Operational Silos Centralized Management

From Silo to Open:

- Do not underestimate

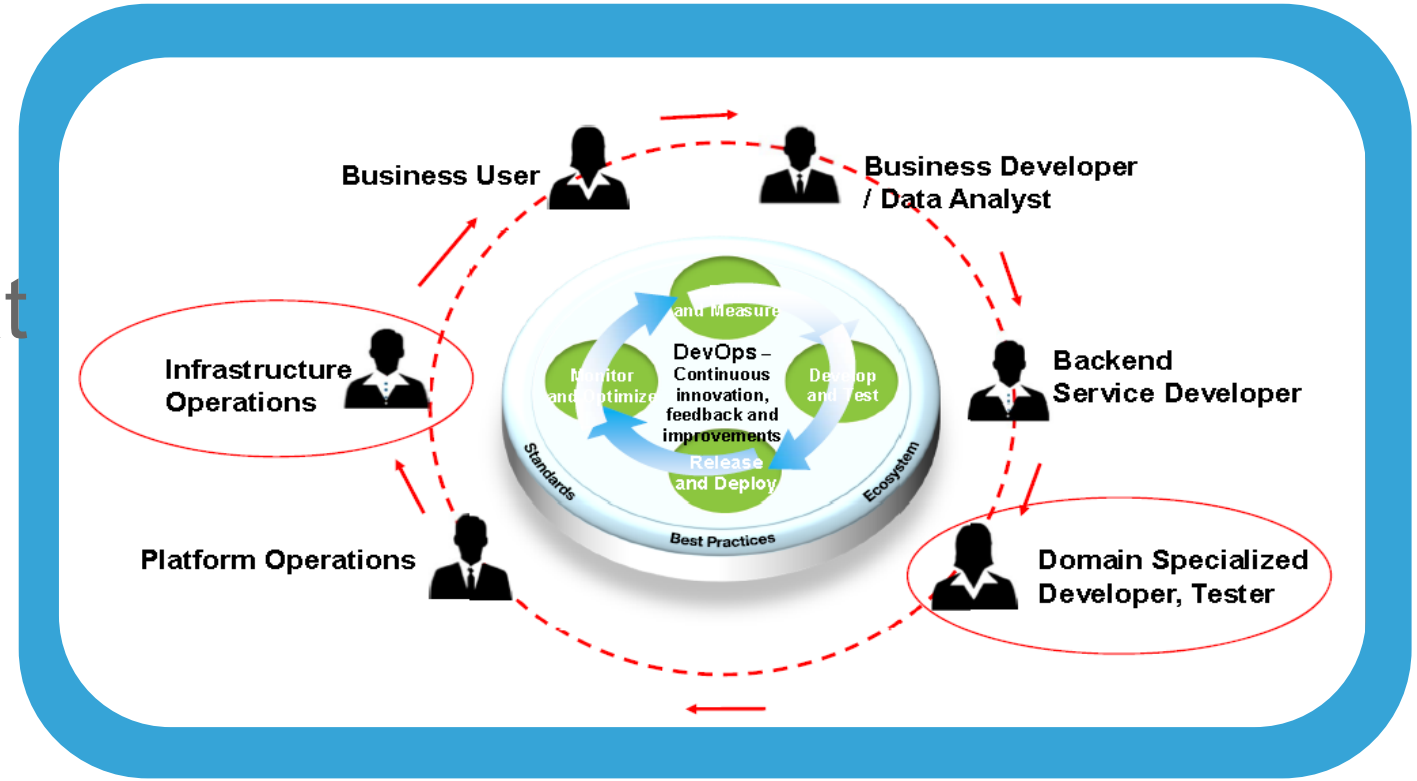
Transparency required

TEAMING MODEL

Cross-functional Team
Self-managed; Product-aligned

Stakeholders

- Who needs to buy in?
- Most DevOps movements will start at the grassroots level but will not succeed without management support



Roles

- Roles have to be re-adjusted
- Collapsed
- The Specialized Generalist

DevOps Practices

Roles

Make environments available early in the development process:

- Common Dev, QA and Prod environment creation process

The Bottlenecks in the Flow

- Environment creation
- Code deployment
- Test setup and run (menti)
- Overly tight architecture
- Development
- Product management

“In November 2011, running even the most minimal test for CloudFoundry required deploying to 45 virtual machines, which took a half hour. This was way too long, and also prevented developers from testing on their own workstations.

By using containers, within months, we got it down to 18 virtual machines so that any developer can deploy the entire system to single VM in six minutes.”

Elisabeth Hendrickson, Director of Quality Engineering, Pivotal Labs

Google Dev and Ops (2013)

- 15,000 engineers, working on 4,000+ projects
- All code is checked into one source tree (billions of files!)
- 5,500 code commits/day
- 75 million test cases are run daily

“Automated tests transform fear into boredom.”

Eran Messeri, Google

Inject Failures Often

The Netflix Tech Blog

We've sometimes referred to the Netflix software architecture in AWS as our Rambo Architecture. Each system has to be able to succeed, no matter what, even all on its own. We're designing each distributed system to expect and tolerate failure from other systems on which it depends.

One of the first systems our engineers built in AWS is called the Chaos Monkey. The Chaos Monkey's job is to randomly kill instances and services within our architecture. If we aren't constantly testing our ability to succeed despite failure, then it isn't likely to work when it matters most – in the event of an unexpected outage.

NETFLIX

You Don’t Choose Chaos Monkey—Chaos Monkey Chooses You

The 2014 AWS Reboot

“When we got the news about the emergency EC2 reboots, our jaws dropped. When we got the list of how many Cassandra nodes would be affected, I felt ill. Then I remembered all the Chaos Monkey exercises we’ve gone through. My reaction was, ’Bring it on!’.”

Christos Kalantzis, Netflix Cloud DB Engineering

Add Ops into Dev

Enhance Service Design with Operational Knowledge:

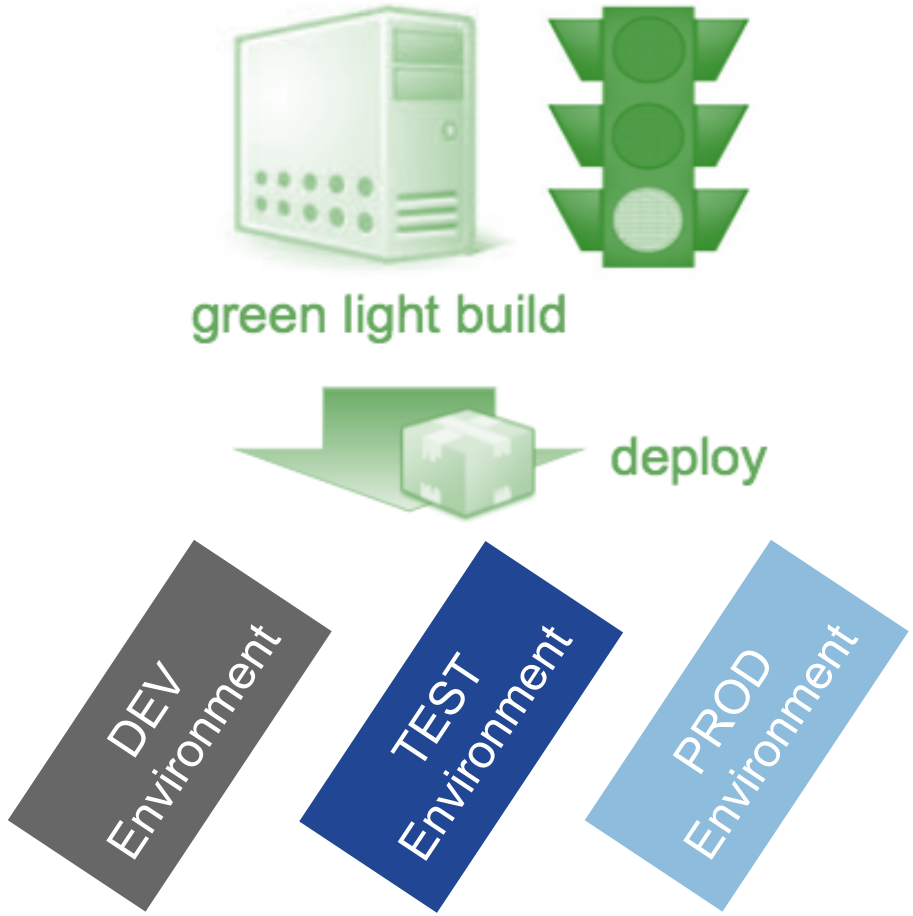
- Reliability
- Performance
- Security
- Test them

Build feedback paths back from Production:

- Foster a culture of responsibility
  - Whether your code passes test, gets deployed, and stays up for users is your responsibility – not someone else
- Make Development better with Ops:
  - Production-like environments
  - Power tooling

Continuous Deployment

- Extension of continuous integration
- First: automate deployment
- Application always known to be in a ”deployable” state



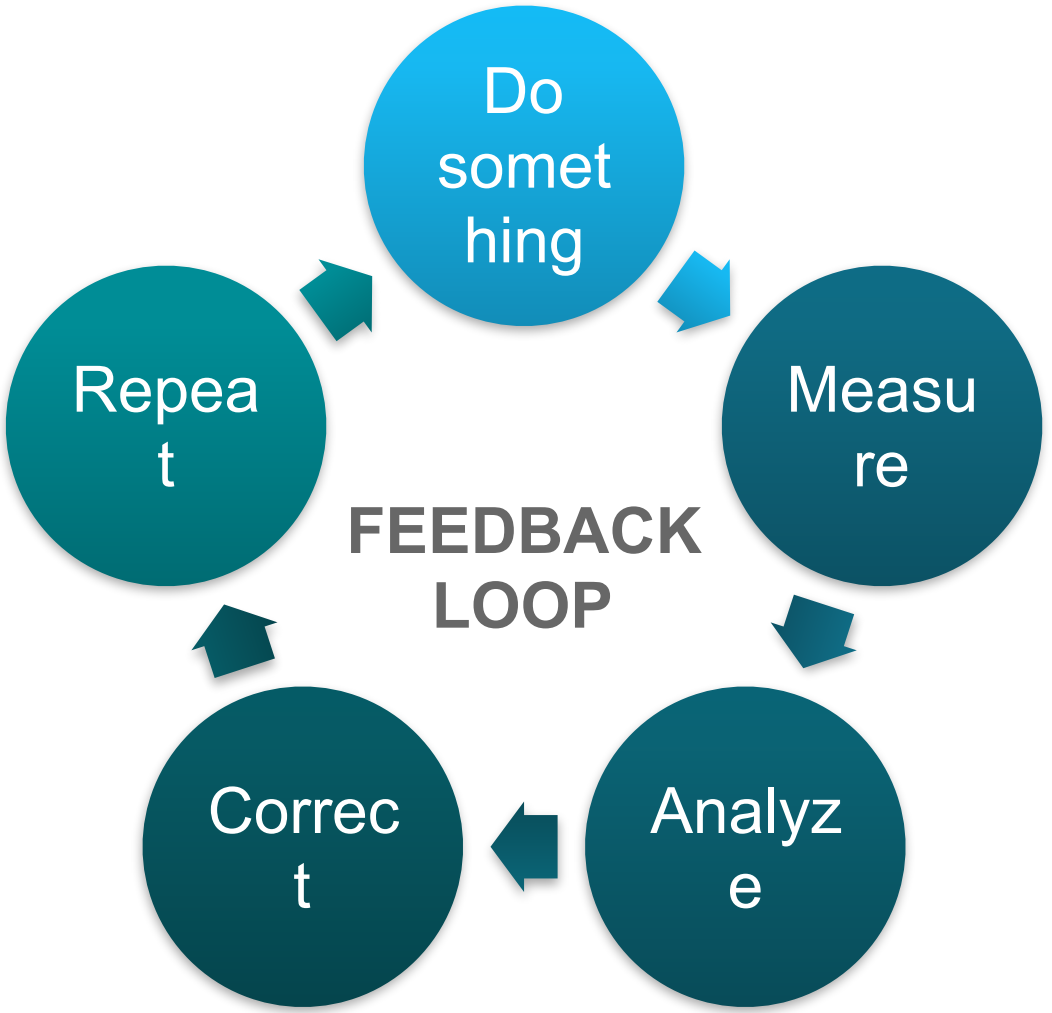
DevOps for IT Automation

One-step Environment Creation

- Need a common environment build process
- For development Q/A and production
- The environment will evolve as development proceeds
- The longer you wait to have a common environment build process, the harder it is to create one

Automating Feedback Loops

- Capture as much data as possible at the source
- The issue becomes the data



Break Things Early

- Consistency in code, environments and configuration
- ASSERTs to catch misconfigurations
- Static code analysis and testing becomes part of the continuous integration and deployment

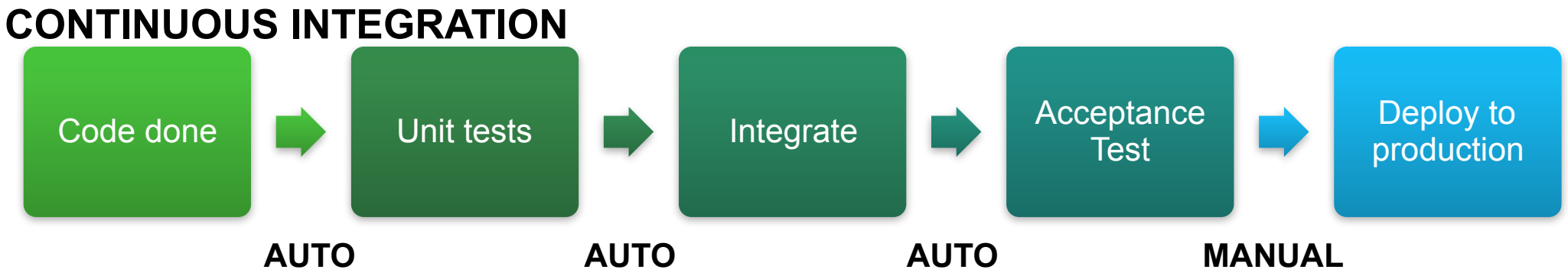
DevOps Lifecycles

Principles of Continuous Integration

- Code repository
- Automate the build
- Make the build self-testing
- Everyone commits to the main-line every day
- Every commit is built
- Build must be fast
- Test in a clone of the production environment
- Make it easy to get the latest deliverables
- Everyone can see the results of the latest build
- Automated deployments

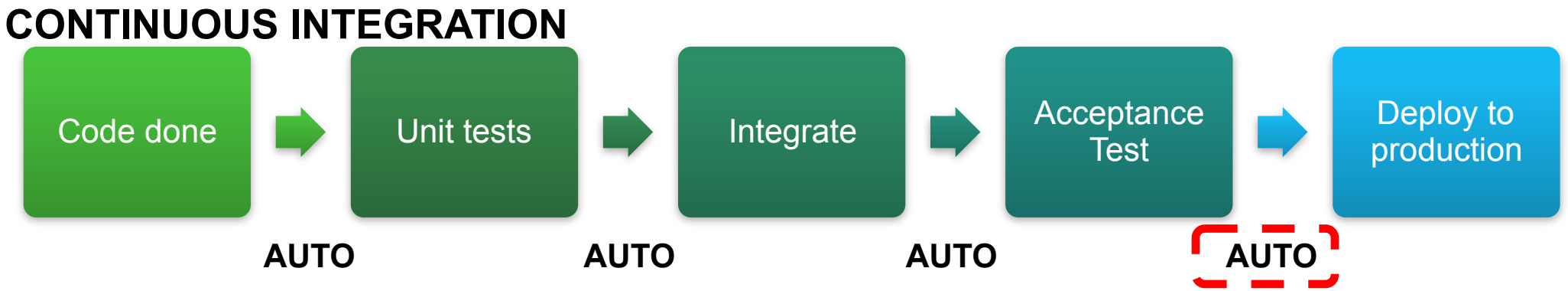
How CI Improves Efficiency

- Rapid feedback
- Reduce technical debt
- Visibility
- Builds automated
- Precursor to continuous delivery and deployment



Keys to Continuous Delivery

- Minimize shock
- Avoid off-hour, high risk, expensive deployments
- Know your rollback plan
- Build in health checks

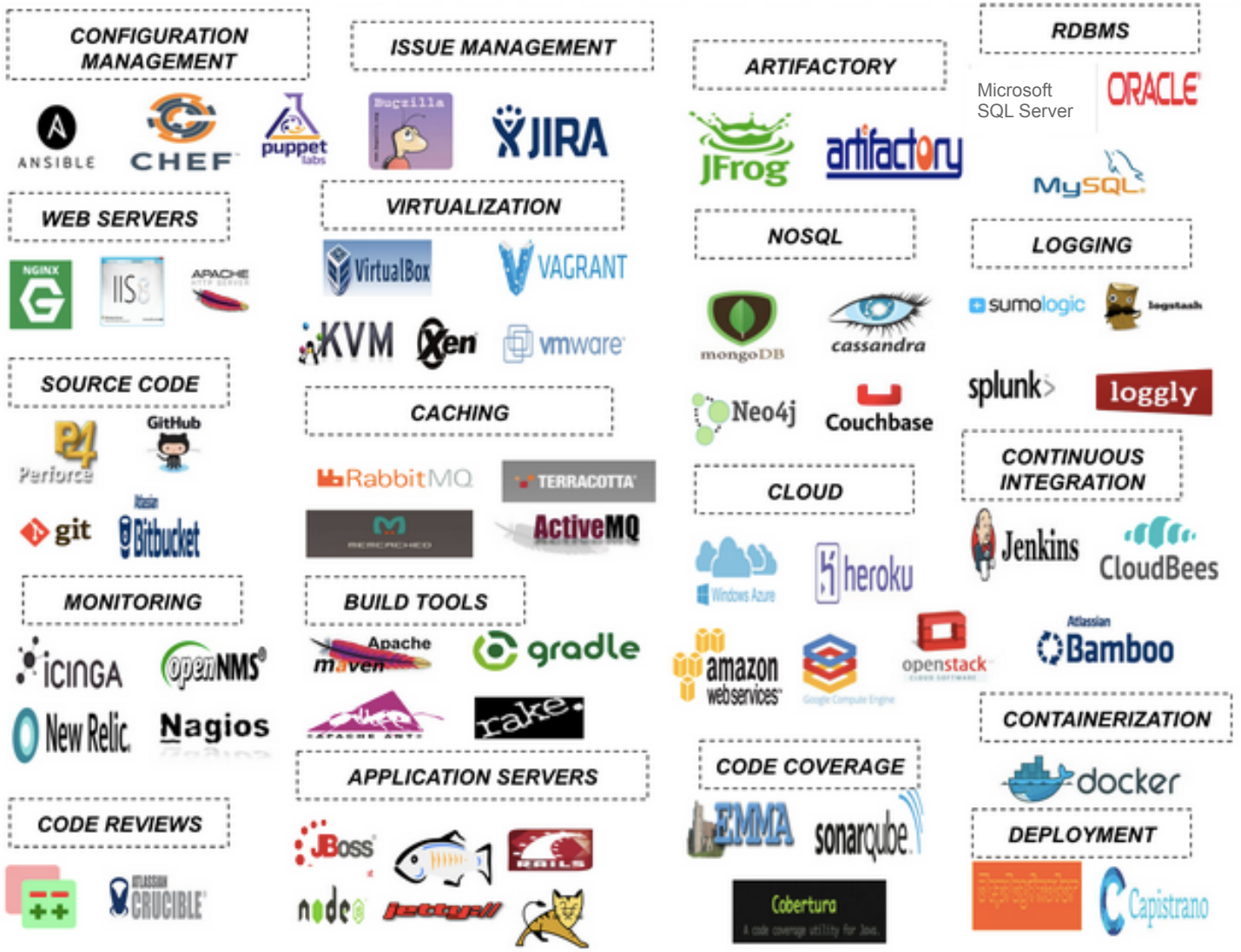


Feedback Loops

- Understanding and responding to the needs of all customers (internal and external)
- Shorten feedback loops
- Feedback = quality

Types of Tools

- Build tools
- Test tools
- General automation
- Configuration management



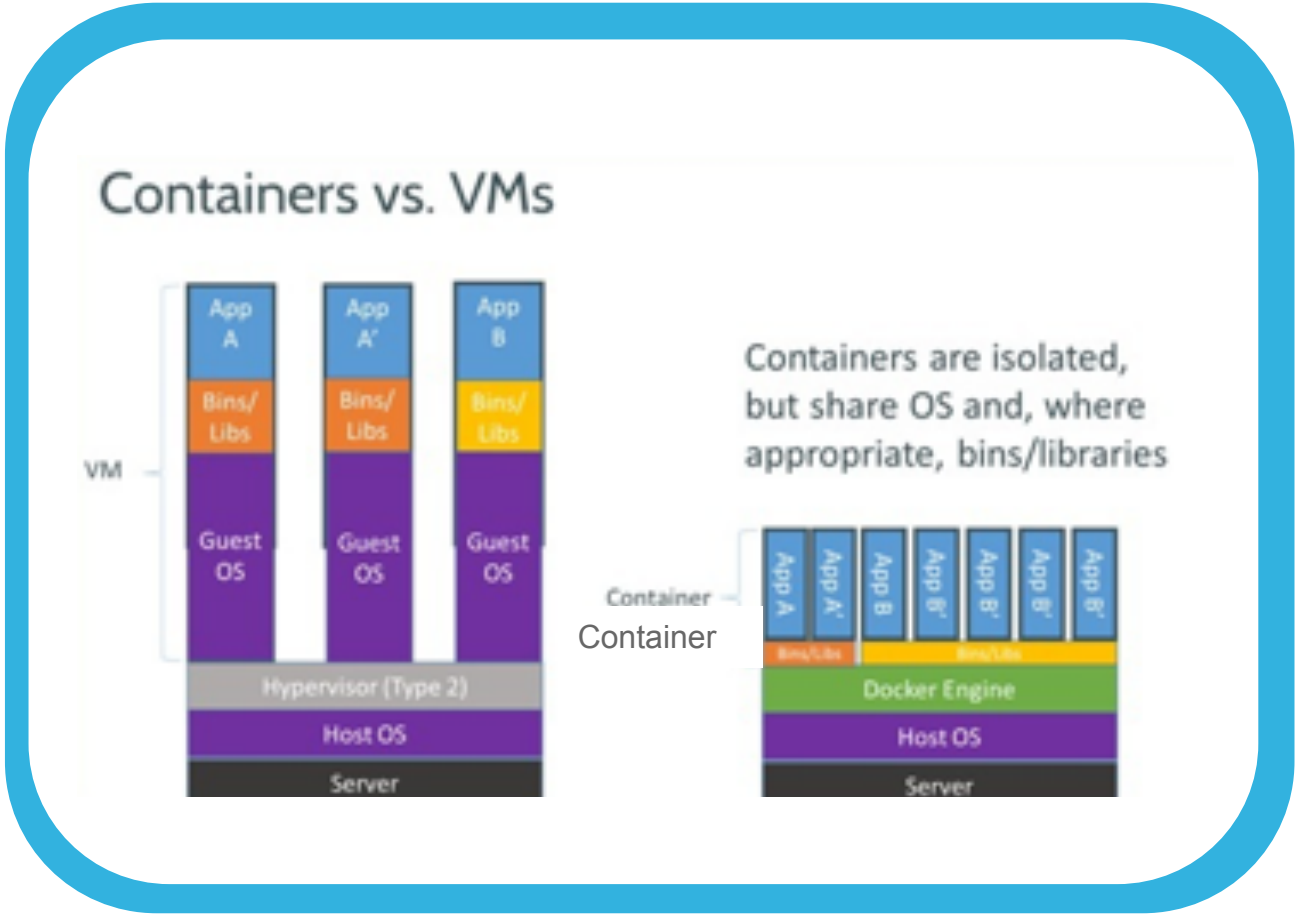
DevOps Related Technology

Virtualization

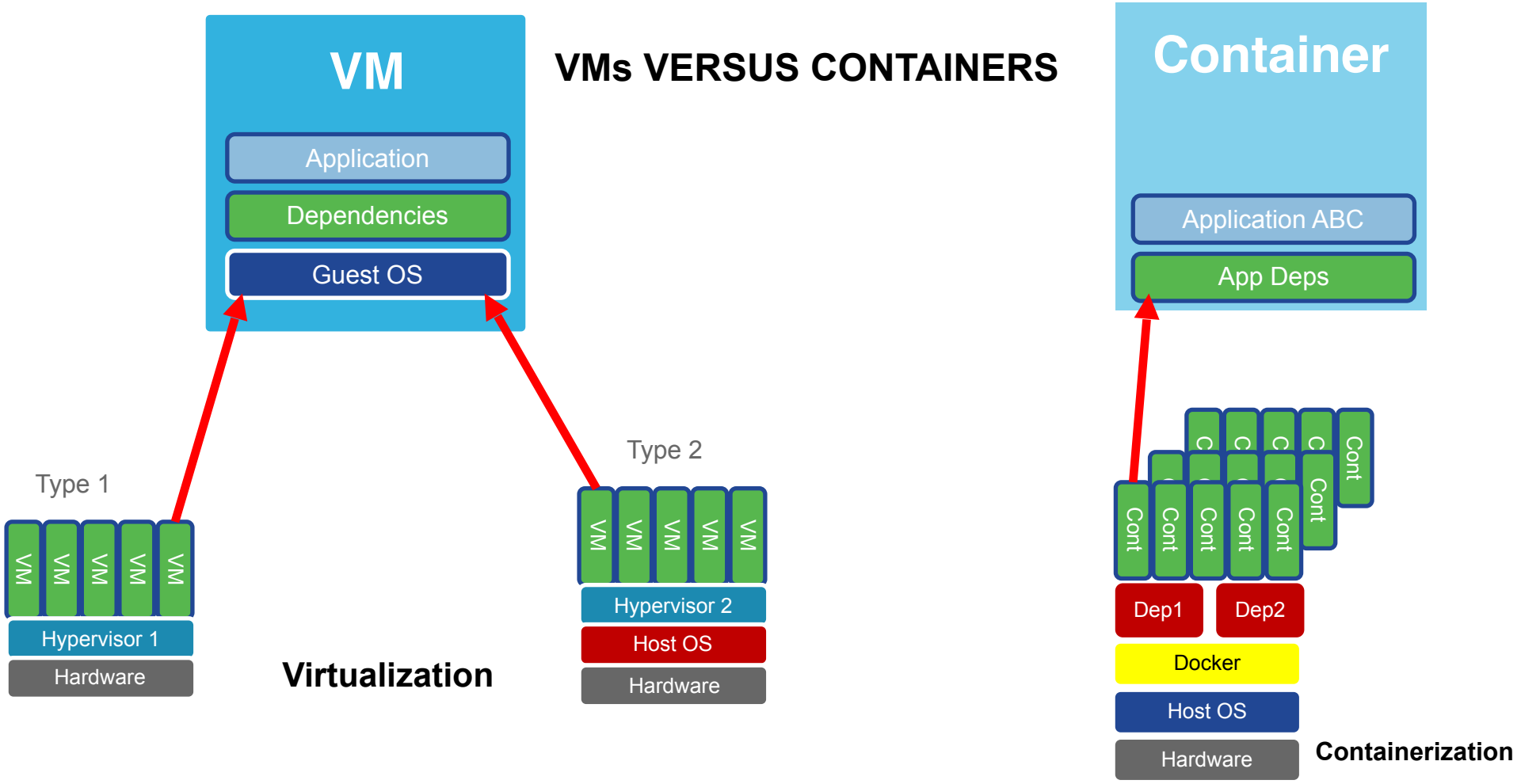
- Been around for a while
- Think VMware ESX
- Virtual Machines

Containers (Docker)

- Open source engine that automates the deployment of any application as a lightweight, portable, self-sufficient container that will run virtually anywhere.
- Based on LXC (Linux Container) and AUFS (Union FS), easy to use.
- Similar to VM as end-user with different features



Docker (What is it?)



Current Container Issues

- Networking
- Service Orchestration
- Security

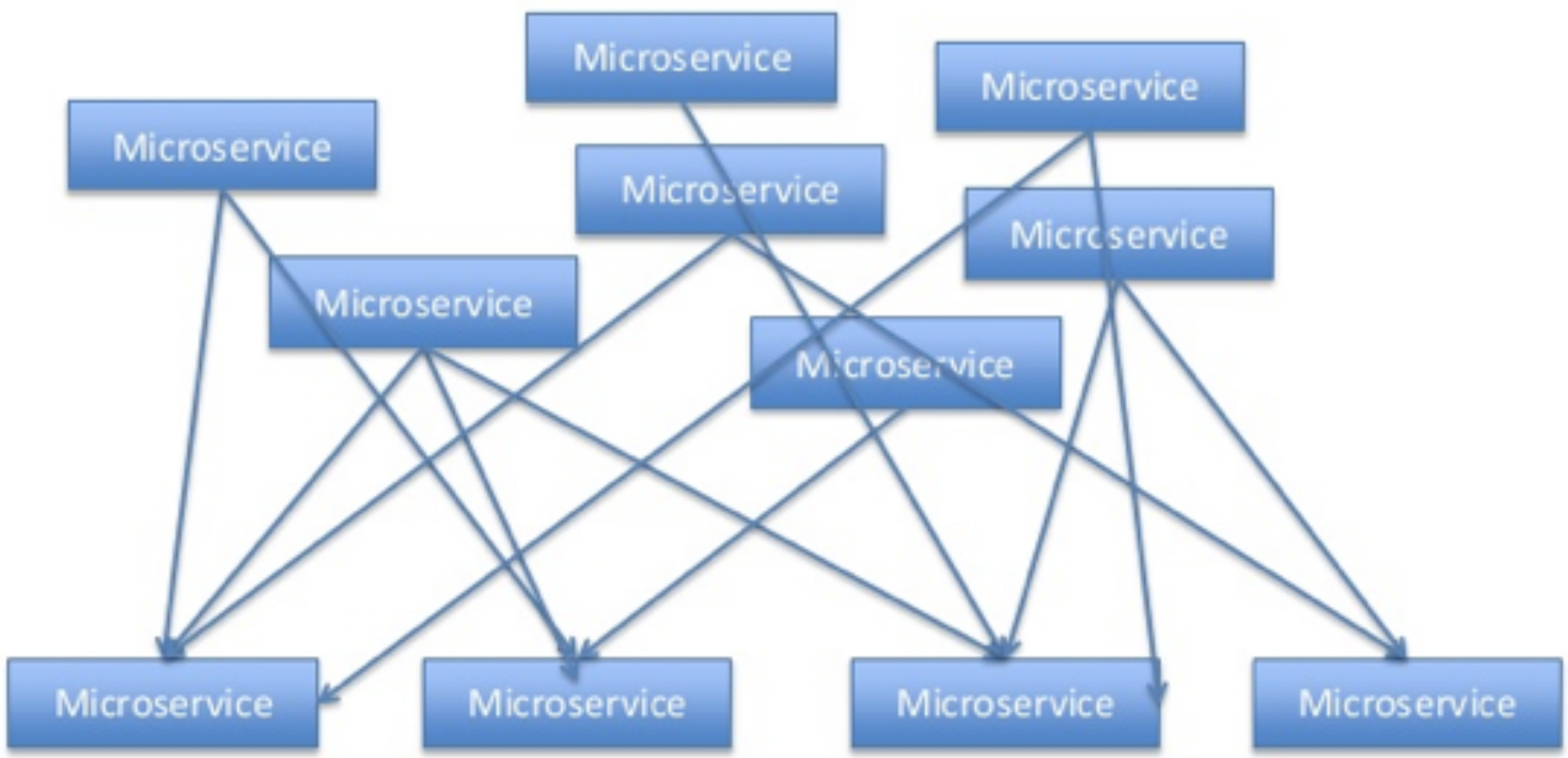
Docker Alternatives

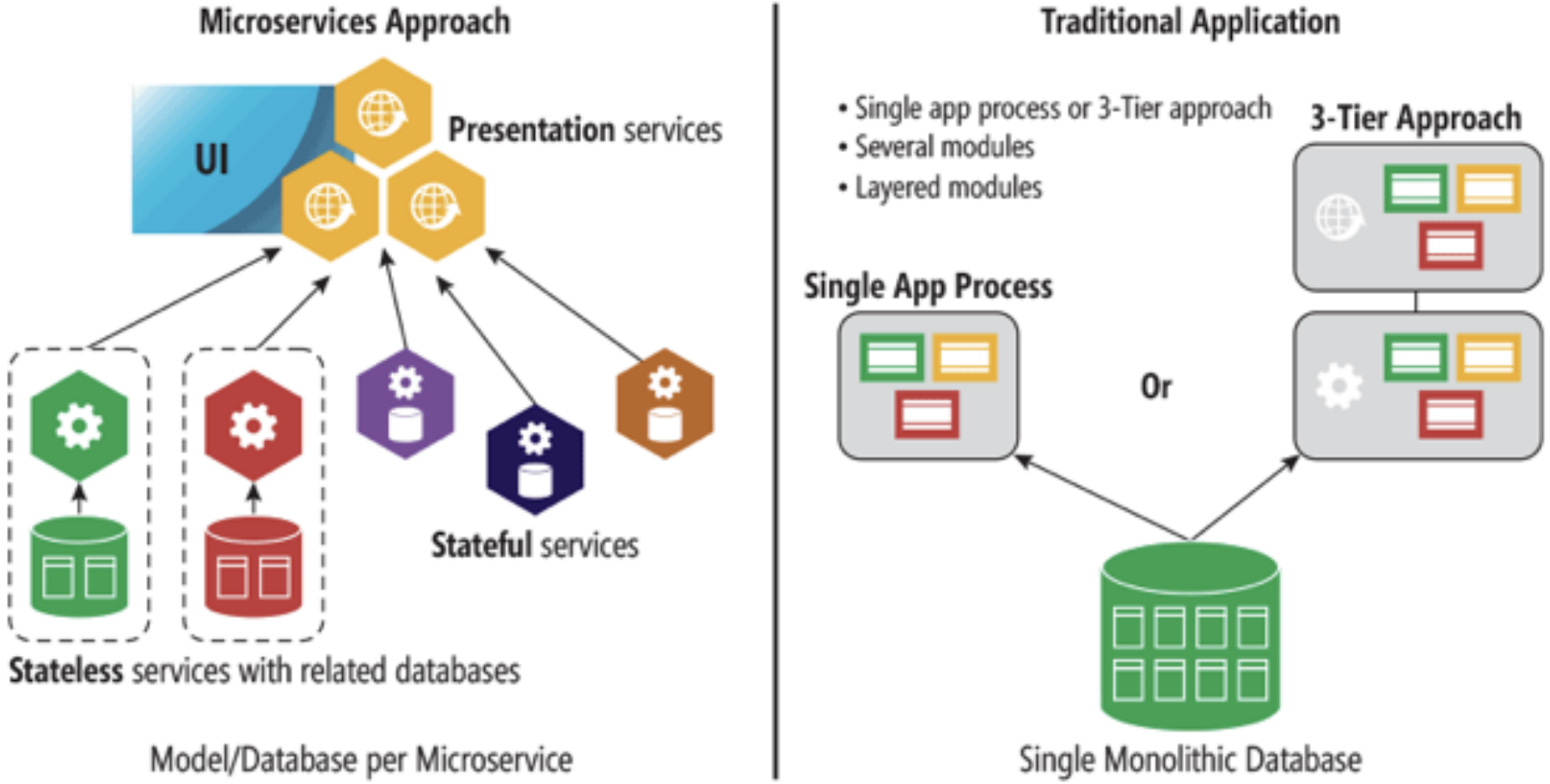
- Rocket
- Microsoft Drawbridge
- LXD (Canonical)

Microservices

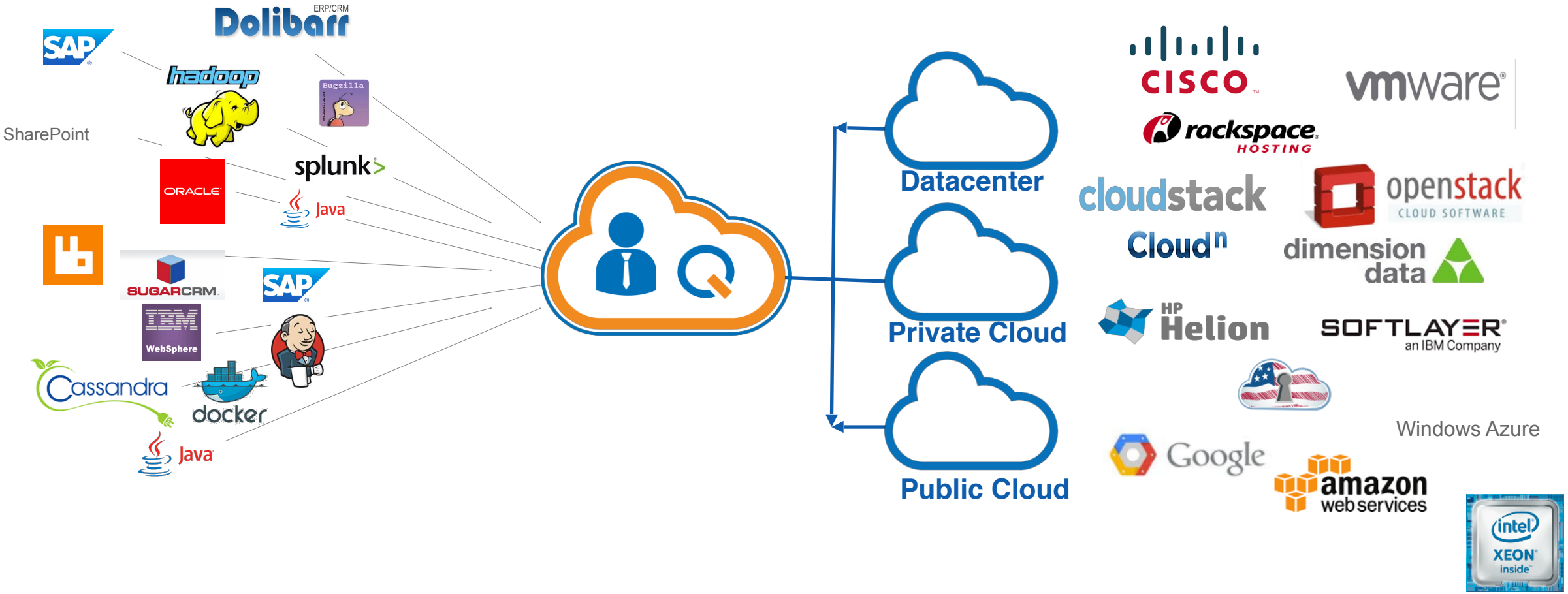
- Architectural pattern based on a stateless service model
- Granular services targeted at providing a discrete piece of application functionality
- Couples with the container model promoted by Docker
- Services are elastic and provisioned on demand in a container as needed

Distributed Service Layer Microservices Architecture




Microservices Approach
UI
Presentation services
Stateful services
Stateless services with related databases
Model/Database per Microservice
Traditional Application
• Single app process or 3-Tier approach
• Several modules
• Layered modules
3-Tier Approach
Single App Process
Or
Single Monolithic Database

CliQr®


ERP/CRM
Dolibarr
SAP
hadoop
Bugzilla
SharePoint
splunk>
ORACLE
Java
SUGARCRM
SAP
IBM
WebSphere
Cassandra
docker
Java
Datacenter
Private Cloud
Public Cloud
CISCO
vmware
rackspace
HOSTING
cloudstack
openstack
CLOUD SOFTWARE
Cloudn
dimension data
HP Helion
SOFTLAYER
an IBM Company
Google
Windows Azure
amazon web services
intel
XEON
inside

Common Use Cases

Application Migration & Management

- New and existing applications
- Start with single application, single cloud
- Workload migration. Datacenter consolidation

DevOps and Continuous Delivery

- Self-service on demand environments
- Tool-chain automated deployment
- Hybrid cloud pipeline

Dynamic Capacity Augmentation

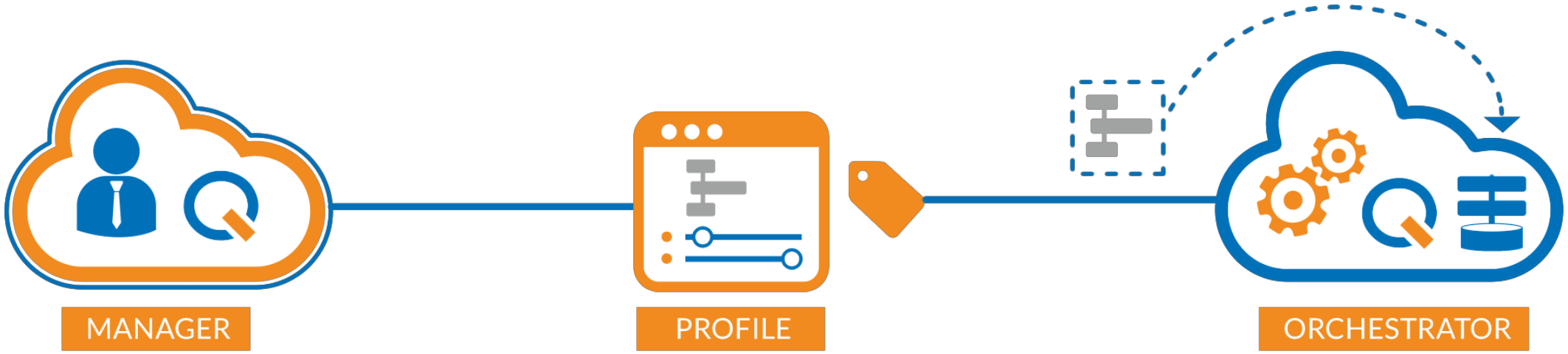
- High performance, batch and cluster
- Horizontal scaling
- Cross-environment bursting

IT as a Service with Governance

- On-demand self-service marketplace
- Single pane of glass – financial controls, usage metering, multi-tenant governance

CliQr®

CloudCenter



CloudCenter

- ✓ Single platform solution
- ✓ Unique application-defined technology
- ✓ Abstracts application from underlying cloud environment
- ✓ Ensures infrastructure adapts to meet the needs of each application

No need to:

- ✓ Write cloud specific scripting
- ✓ Write orchestration workflows
- ✓ Modify application code

CliQr®

SaaS

Software as a Service

Email
CRM
Collaborative
ERP

CONSUME

PaaS

Platform as a Service

Application
Development
Decision Support
Web
Streaming

BUILD ON IT

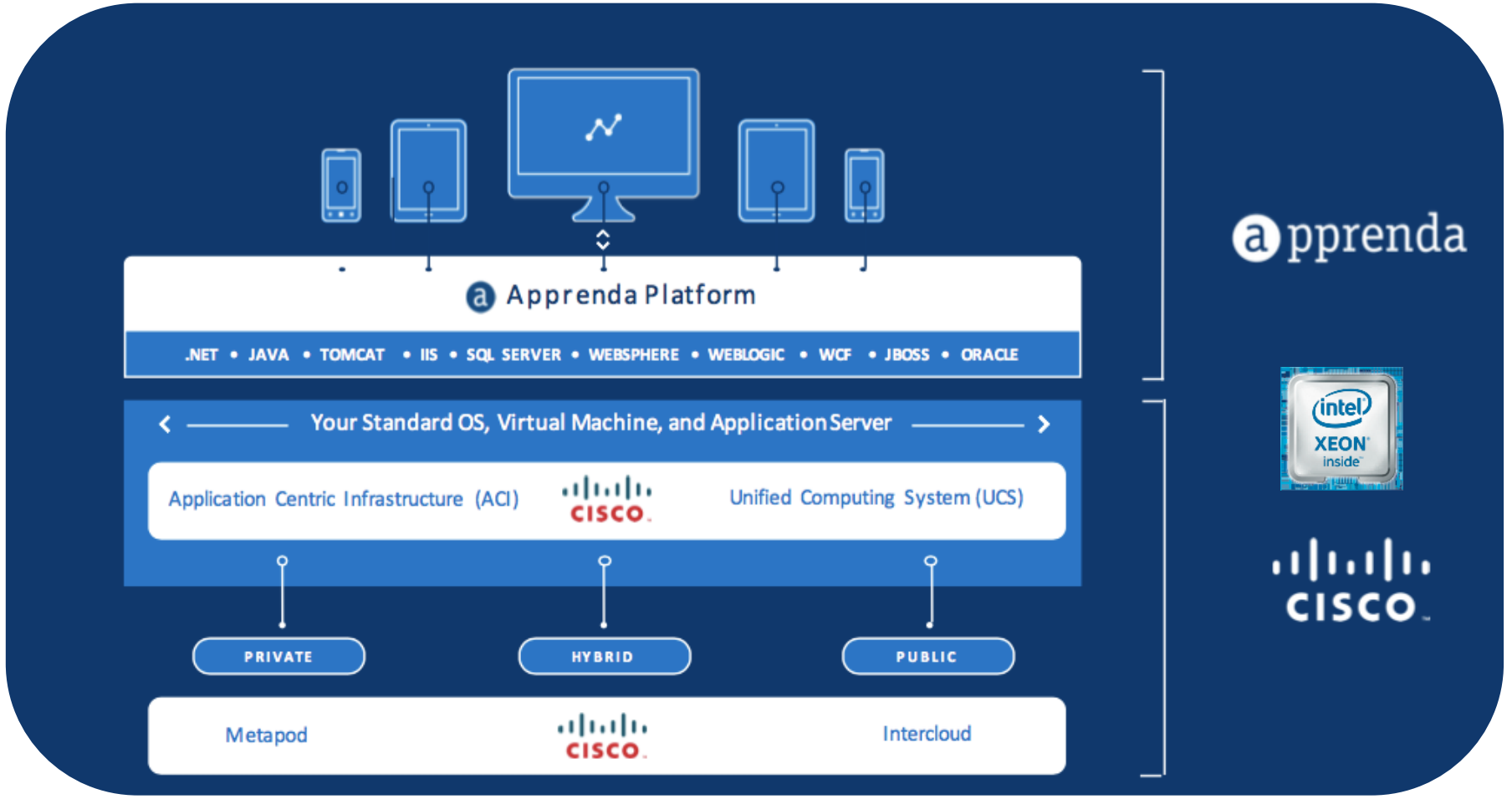
IaaS

Infrastructure as a Service

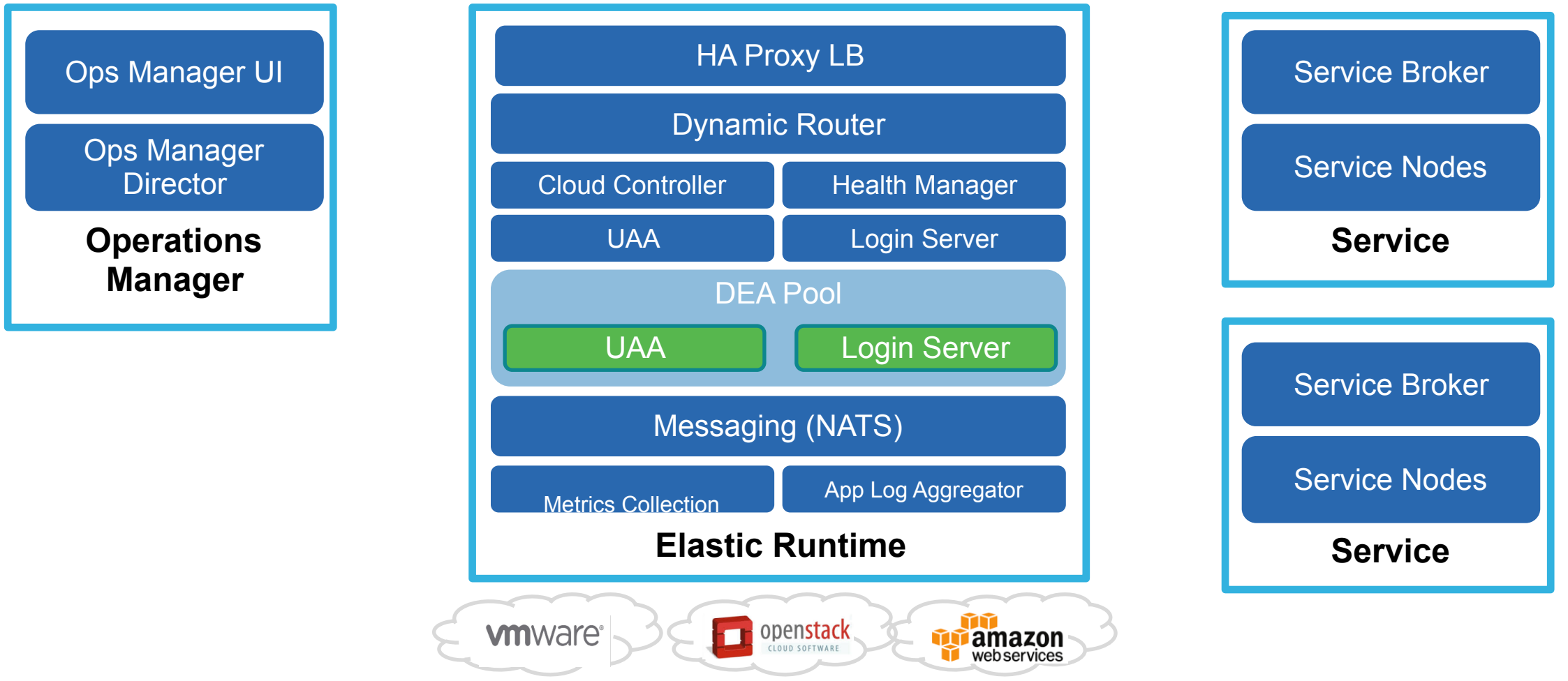
Caching
Legacy | File
Networking |
Technical
Security | SysMgmt

MIGRATE TO IT

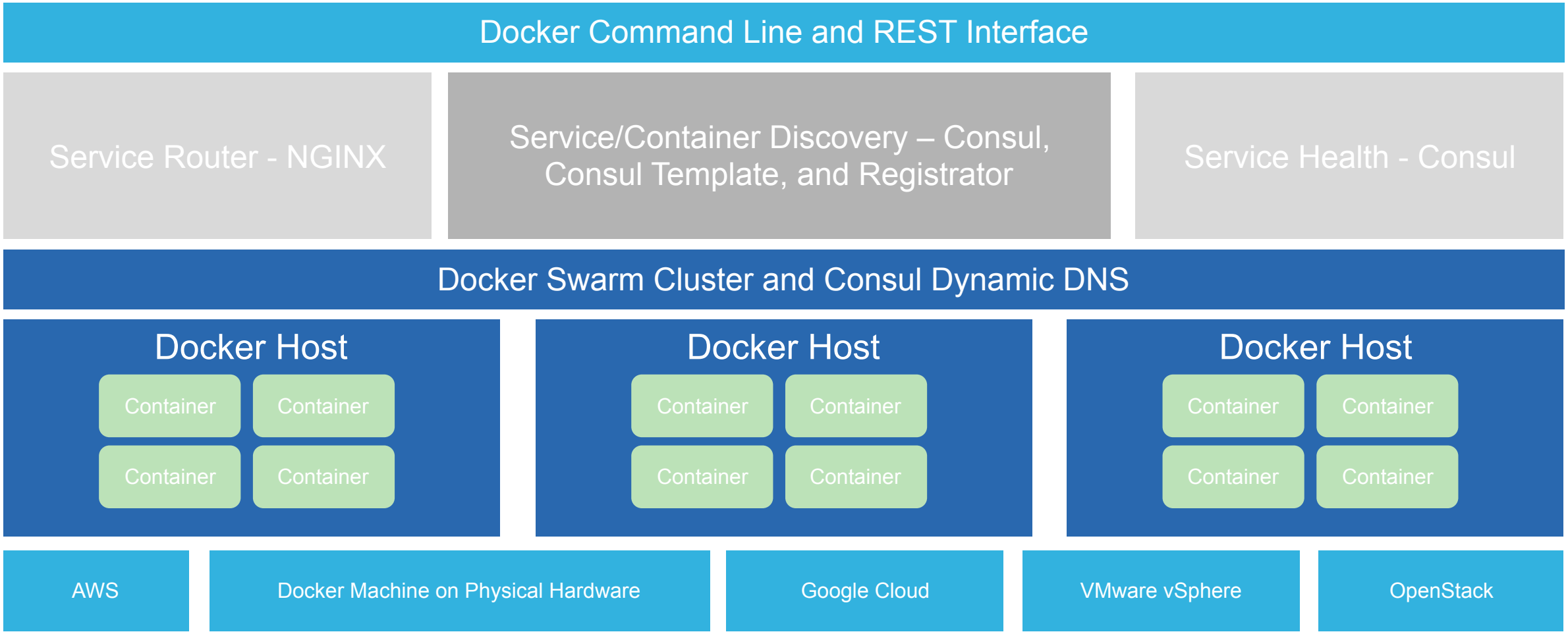
apprenda


Apprenda Platform
.NET • JAVA • TOMCAT • IIS • SQL SERVER • WEBSPHERE • WEBLOGIC • WCF • JBOSS • ORACLE
Your Standard OS, Virtual Machine, and Application Server
Application Centric Infrastructure (ACI)
CISCO
Unified Computing System (UCS)
PRIVATE
HYBRID
PUBLIC
Metapod
CISCO
Intercloud
apprenda
intel
XEON
inside
CISCO

Pivotal CF Architecture



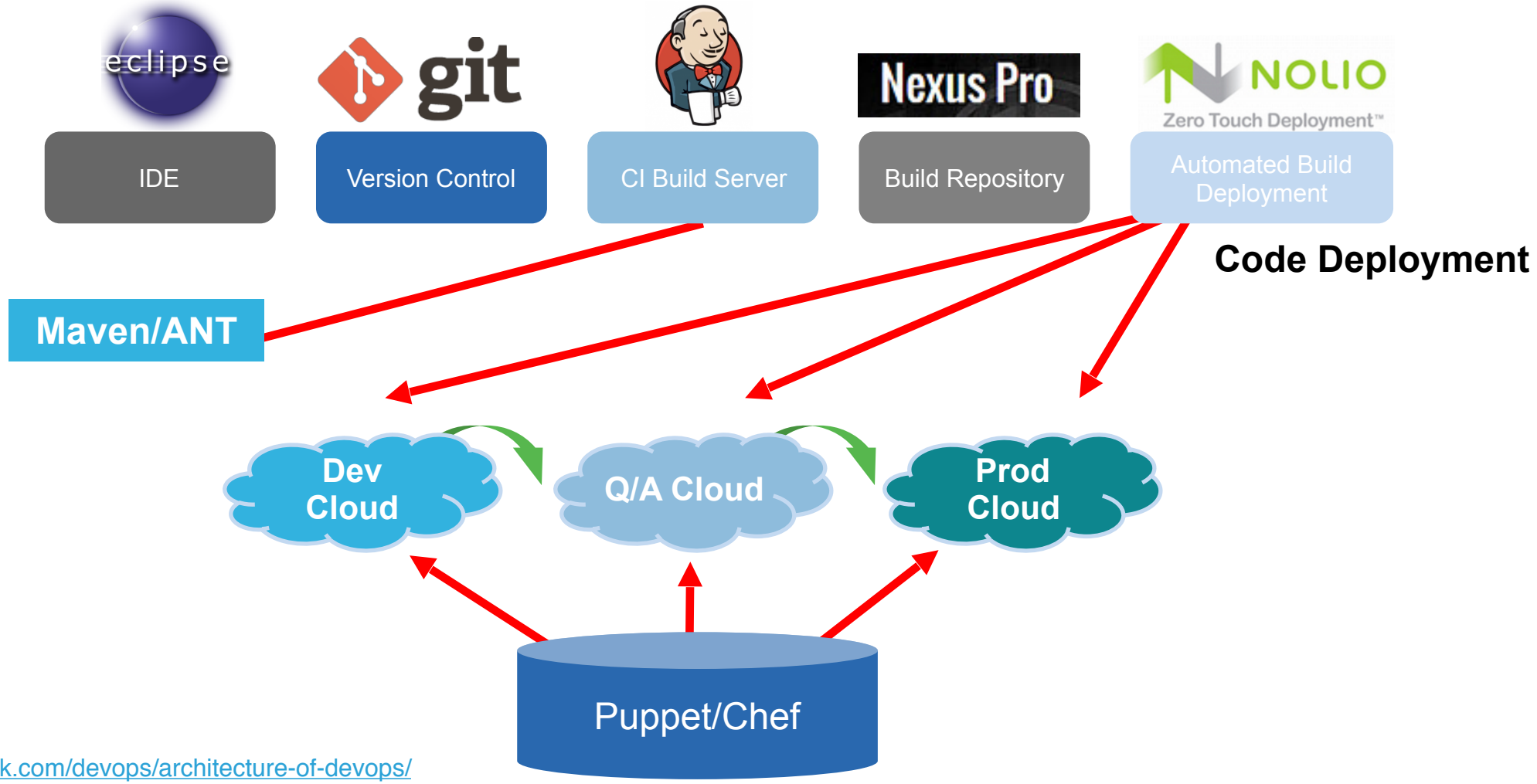
Back to Docker…



Core DevOps Technology

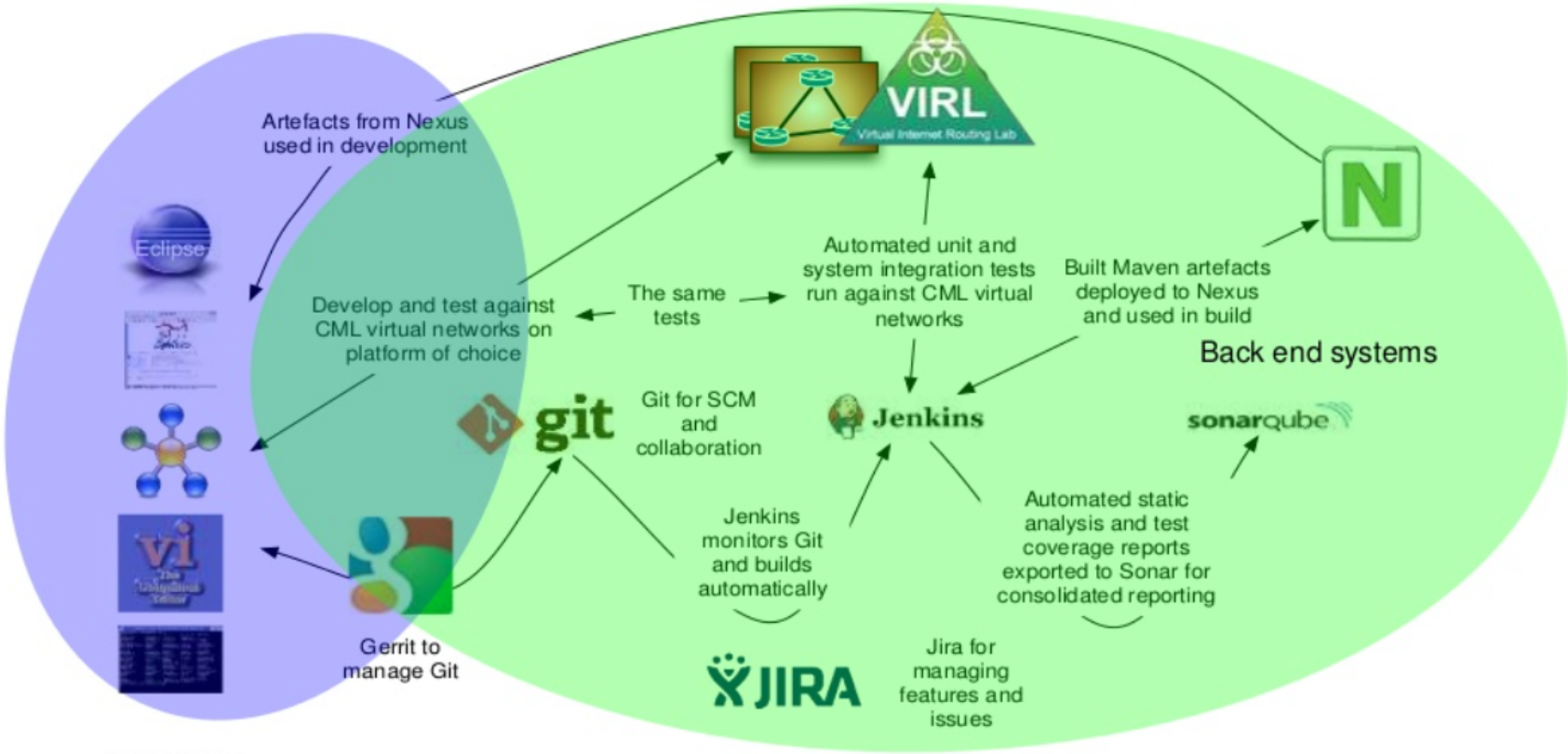
Common Elements of the Software Supply Chain

Example Reference Architecture for DevOps



SOURCE: http://agiletrick.com/devops/architecture-of-devops/

According to Cisco



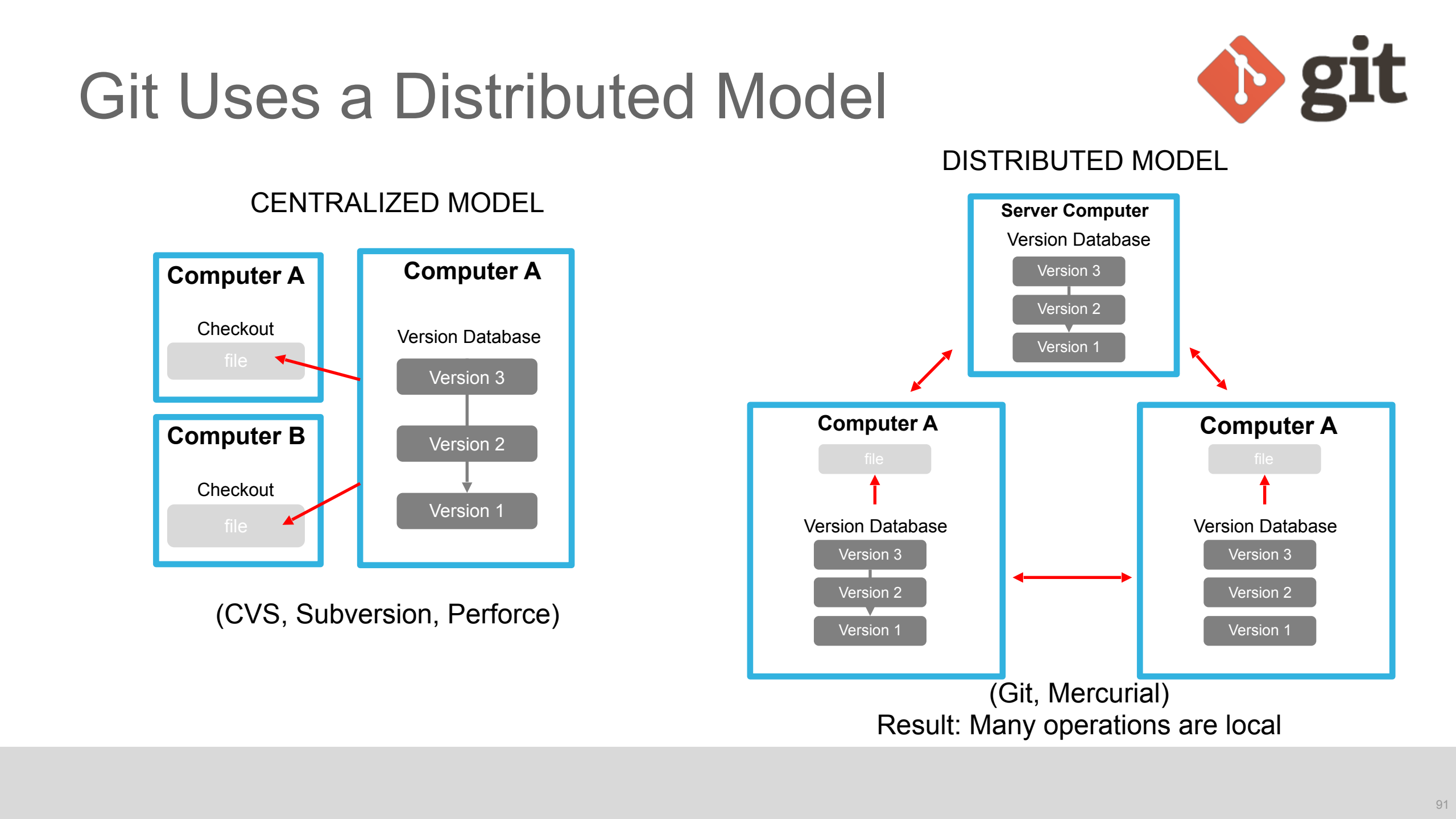
General Notes

- CAPS (Chef, Ansible, Puppet, Salt) are mainly for centrally controlling what lives inside a large number of instances. I.e. processes, files, etc.
- Terraform and CloudFormation are mainly for creating instances themselves (and other cloud resources like load balancers etc).

Git

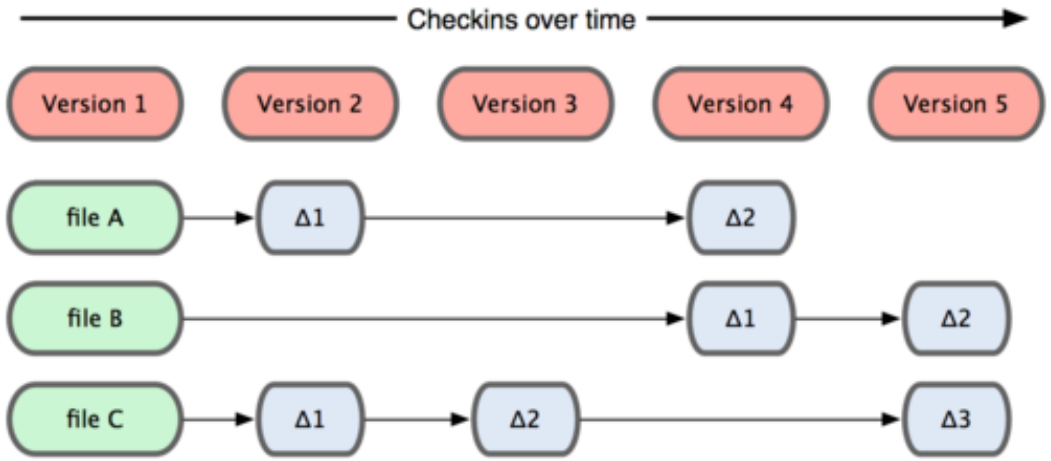
- Source Control
- The backbone for open source

Git Uses a Distributed Model

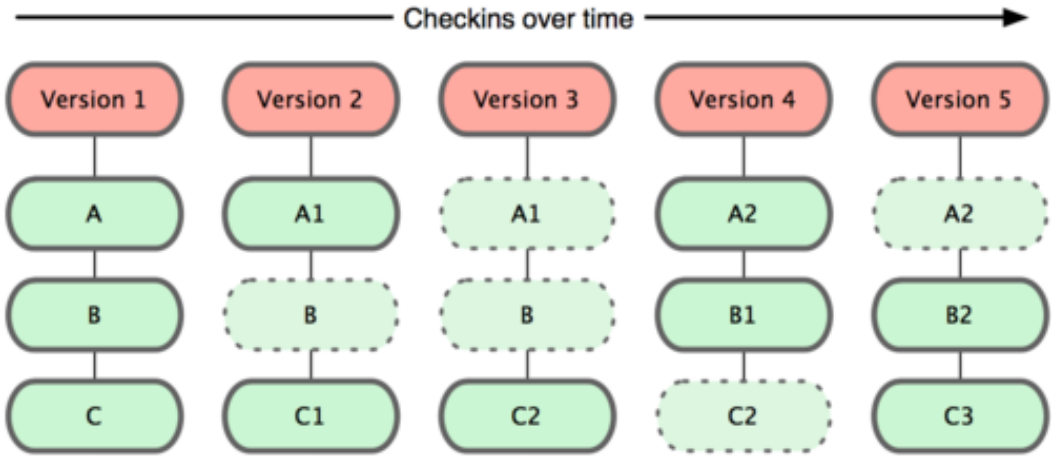


Git Takes Snapshots

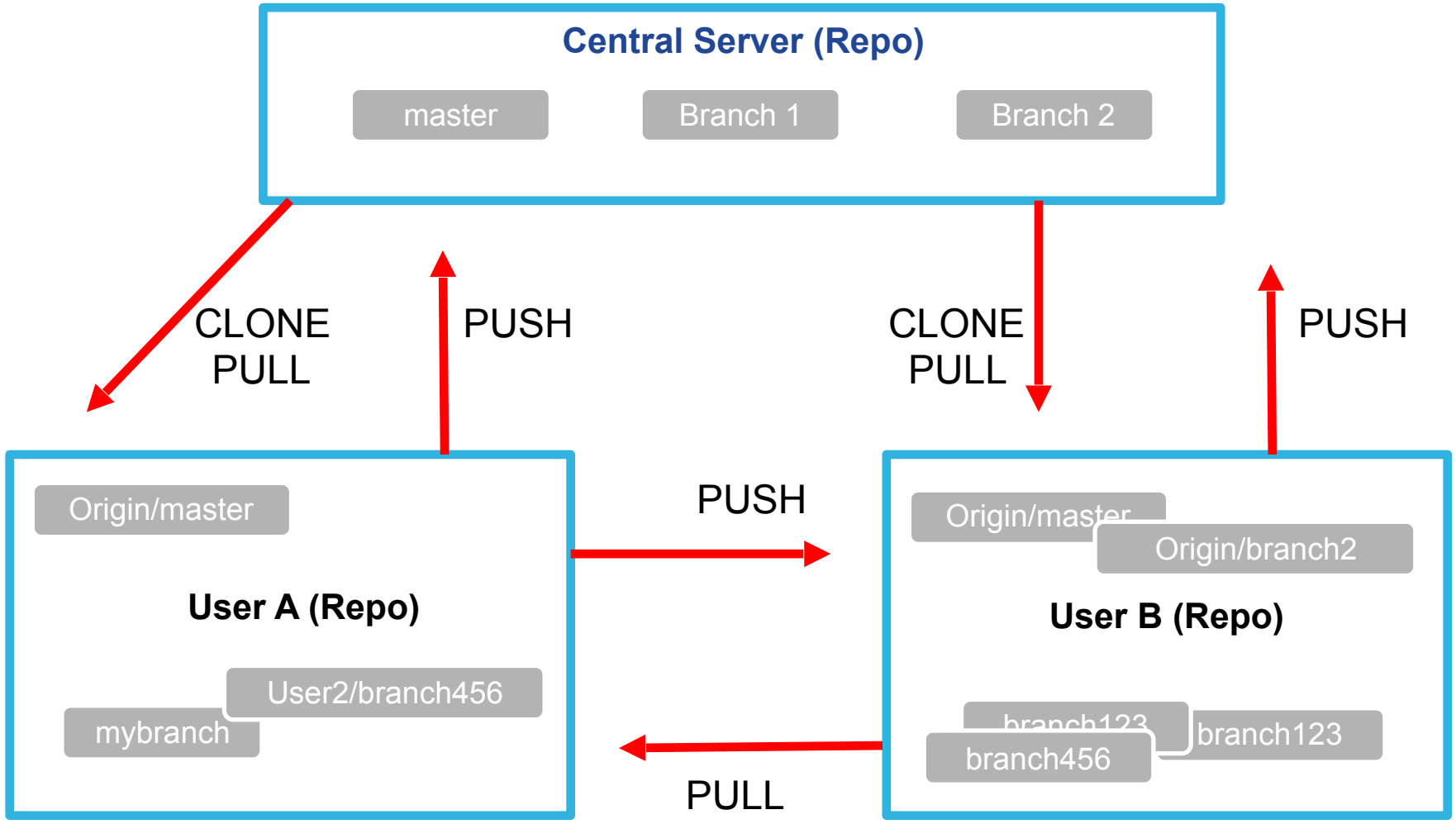
SUBVERSION



GIT



How Git Does It

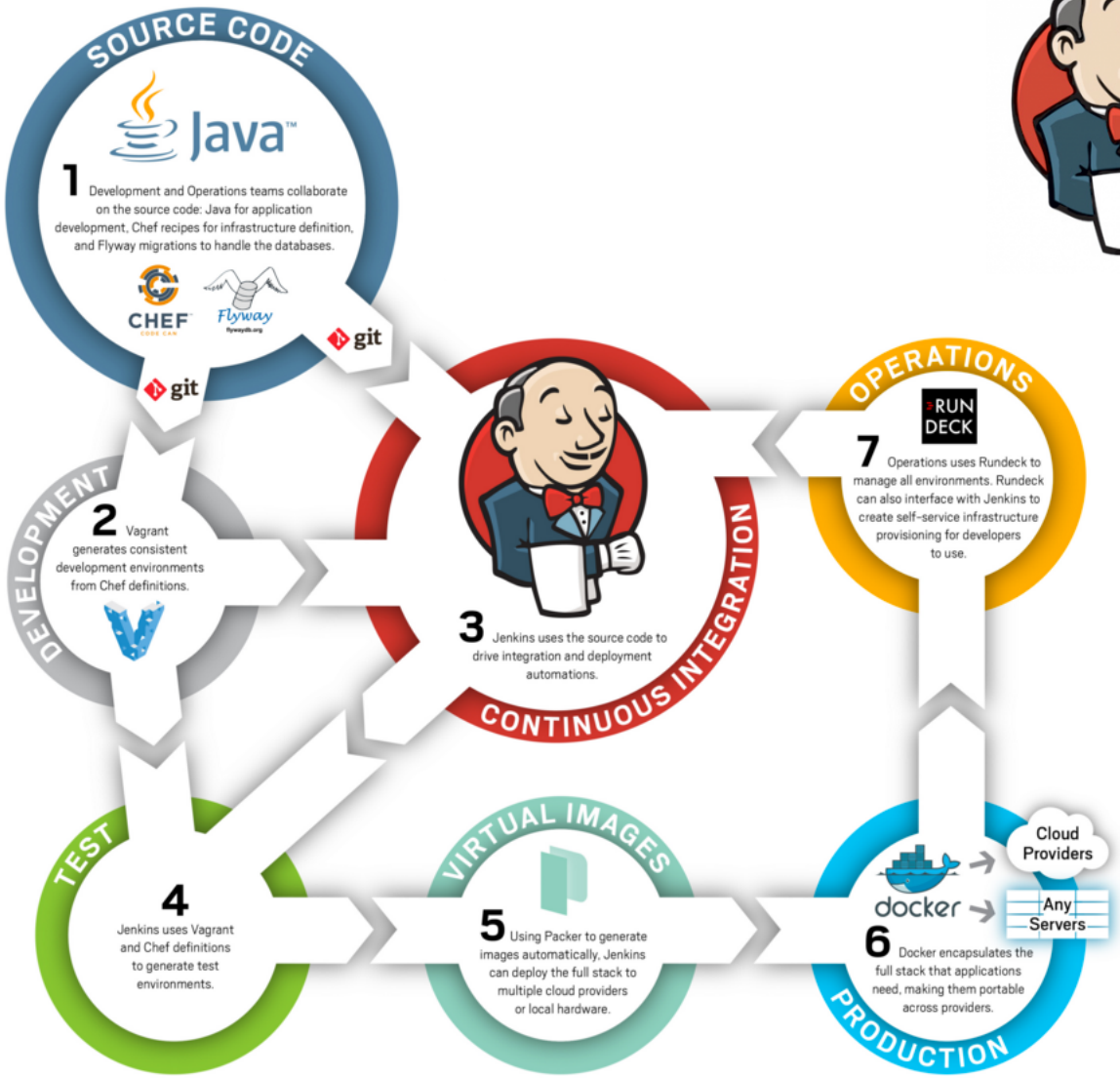


Jenkins

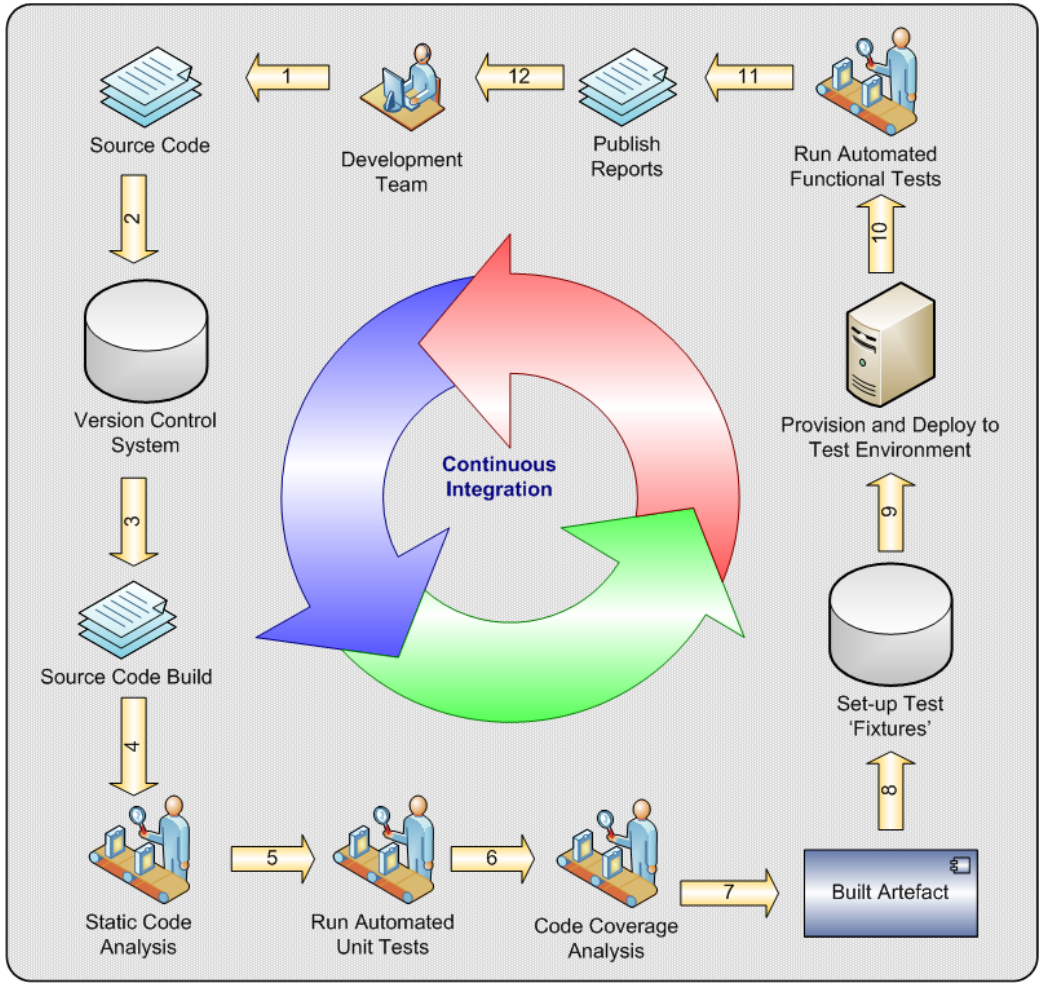
Continuous integration server:

- Coordinate “jobs” to automate the building of applications and environments
- Scales using master/slave architecture
- Integrates into all major DevOps and code build tools

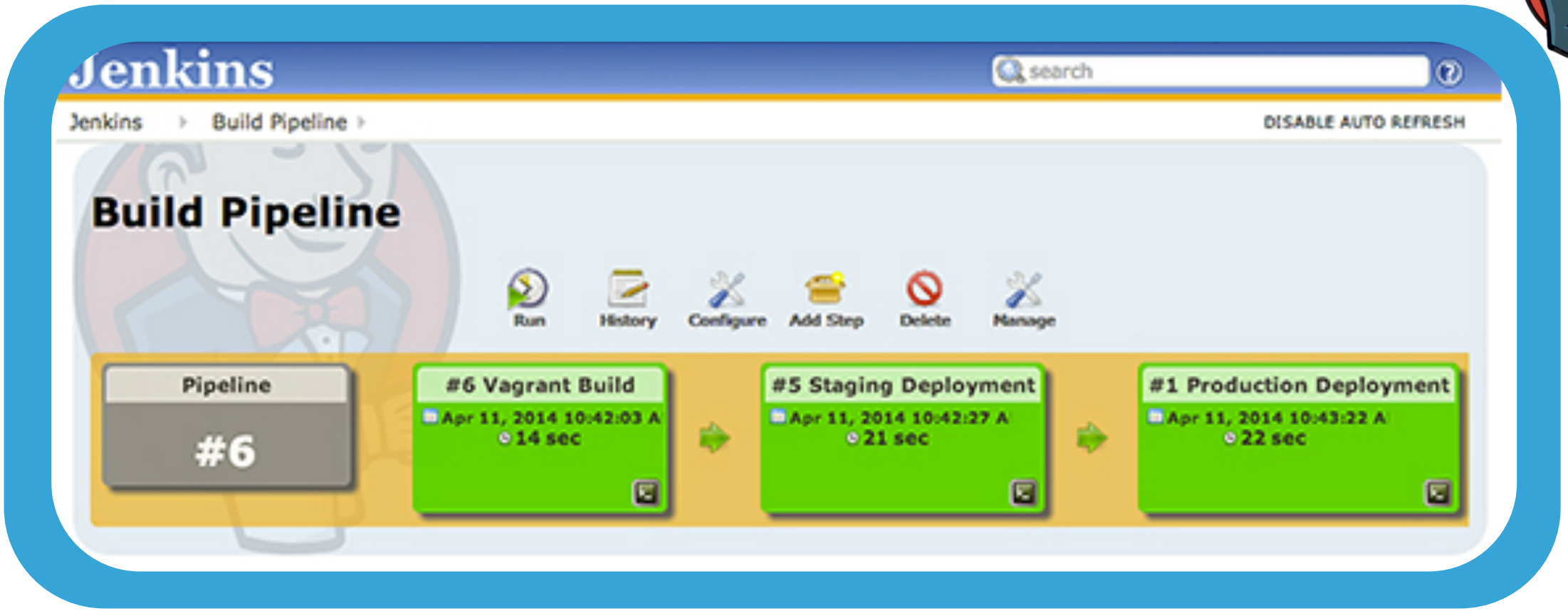
DevOps practices accelerate the rate of releases by increasing collaboration and reducing friction between developers and IT operations professionals. Open source tools help you to define good DevOps strategies and implement continuous deployment for projects of any size.



Typical CI Workflow using Jenkins



Jenkins Pipelines



Vagrant

- Provisioning tool for Dev and Test environments
- Why use Vagrant?
  - Quick
  - Easily replicate production on a Dev box
- Many use cases for developers, operations as well as Q/A

Vagrant: Basics

- Command line utility
- Uses a vagrant file to describe the virtual environment you want to set up
- Uses a simple abstraction from underlying complexities:
  - vagrant up
  - vagrant ssh
  - vagrant halt
  - Etc.

Vagrant: Basics (Cont.)

- The Box:
  - Basic building block
  - Uses shared “repo”
- Providers:
  - Supports AWS, VirtualBox, Vmware, etc.

Vagrant: When

- Need to easily provision development environments

Vagrant: Pros

- Initial set up to get running is quick and easy
- Wide availability of boxes to provision your environment

Vagrant: Cons

- Mysterious performance issues
- Some features on certain platforms (cough windows) can be a pain to implement

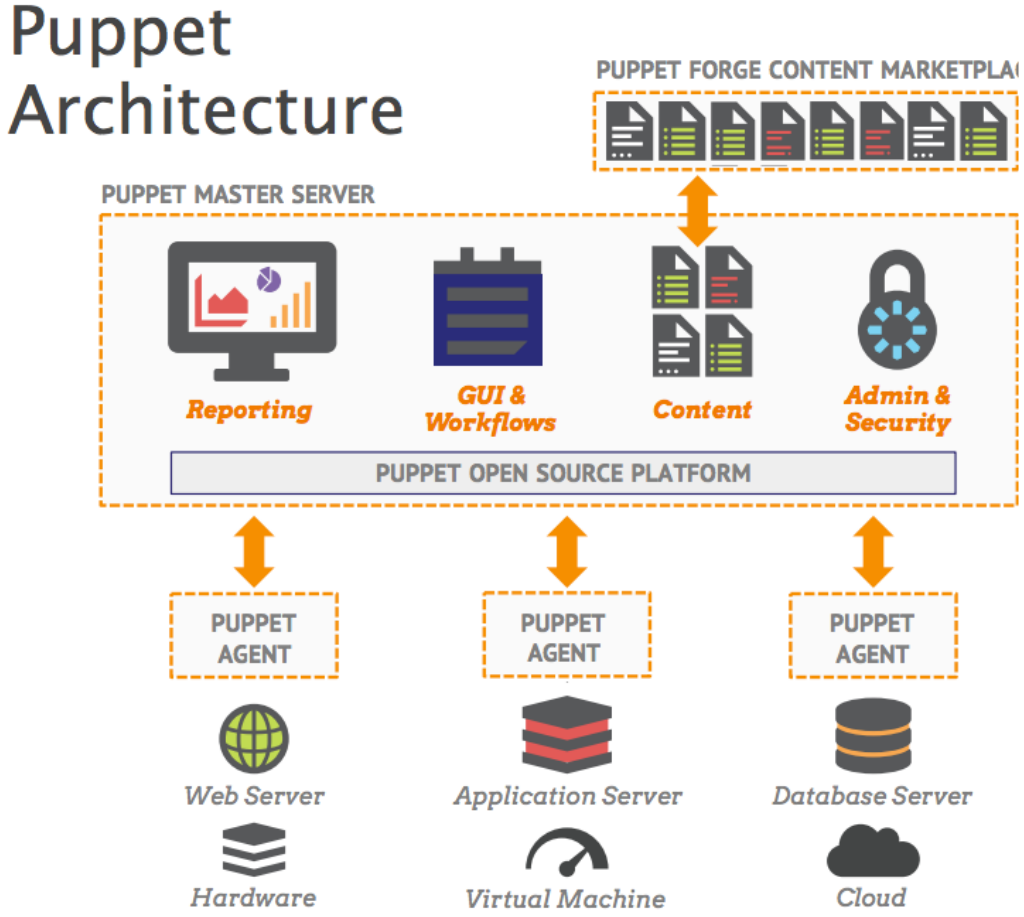
Puppet: What is Puppet?

- Think of it as a language
- Describe state, not steps
- Paint a picture of your ideal and most clean system

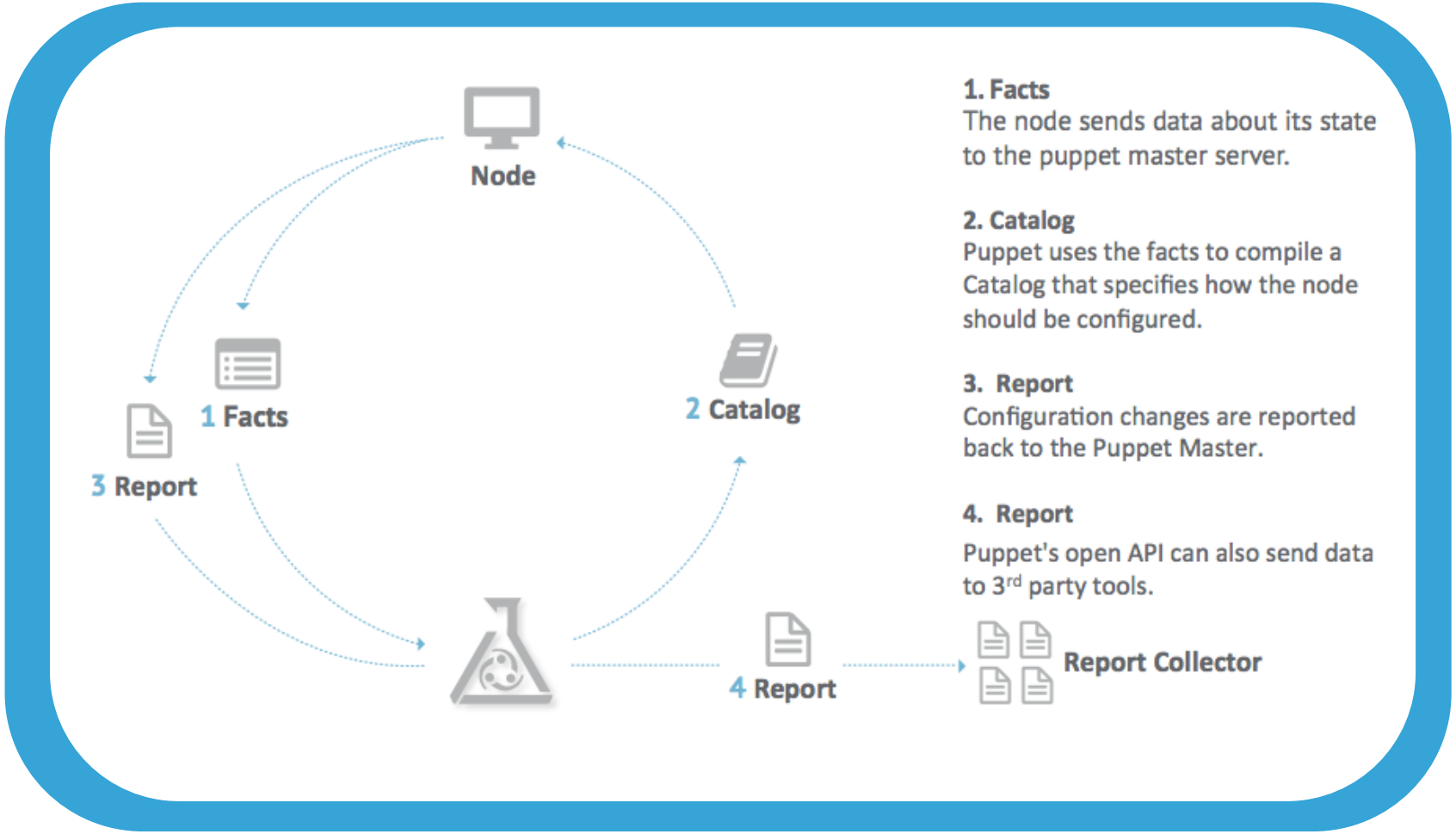
Puppet

- Automation of System Administration tasks
- Uses a declarative language to automate mundane admin tasks
- Seen as the tool of choice for pure configuration management
- Written in Ruby

Puppet: Architecture



Puppet: Puppet Run



Puppet: When

- Stability and maturity are key factors
- Good for large organizations with a heterogeneous environment and breadth of skills on the team

Puppet: Pros

- Mature interface and runs on nearly every OS
- Simple installation and setup
- Complete Web UI in this space
- Strong reporting capabilities
- Well-established support community

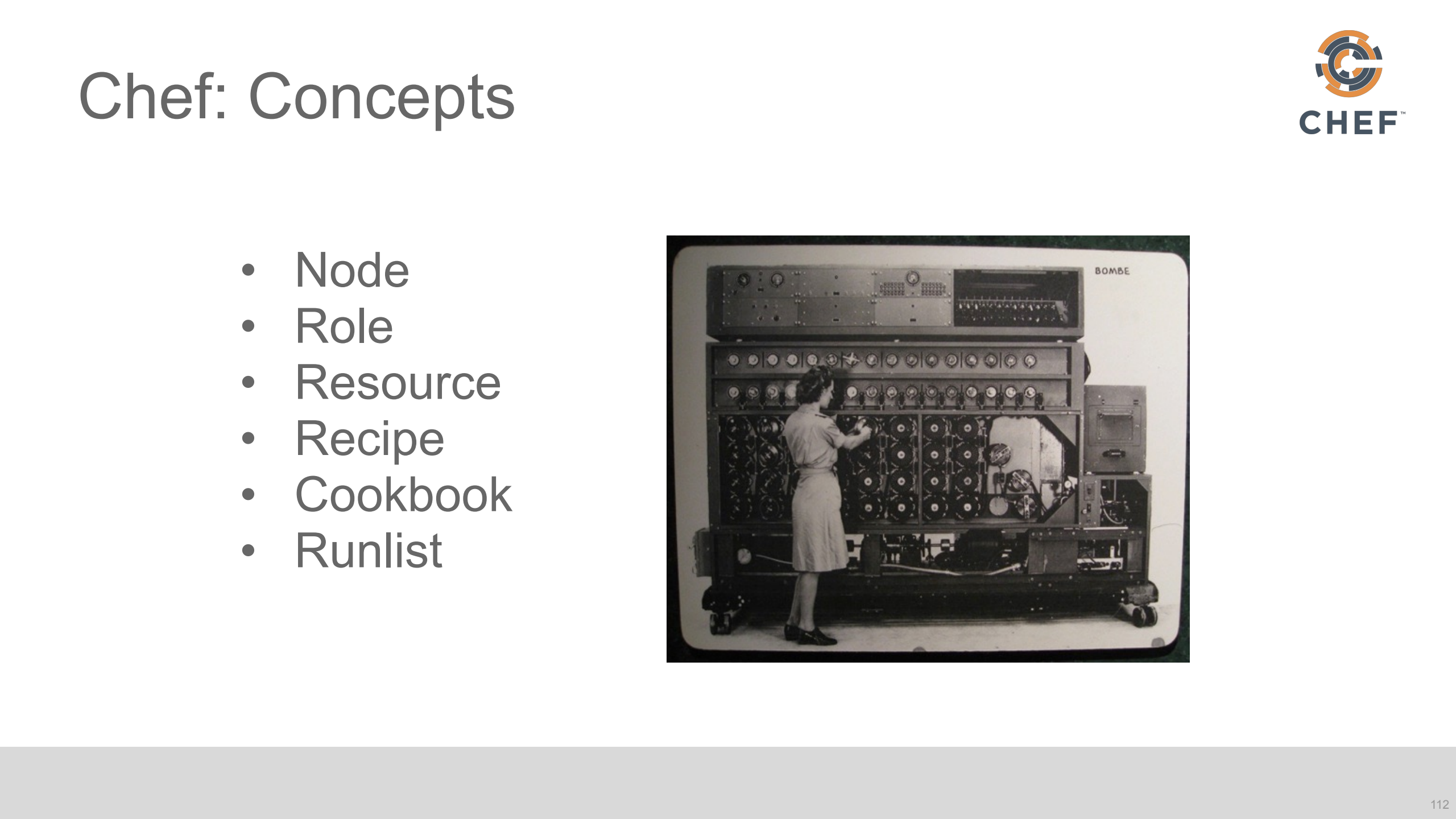
Puppet: Cons

- Advanced tasks will lead you to use the CLI, which is Ruby-based
- Due to the DSL and a non-simplistic design, a Puppet code base can grow large, complex, and will be hard for new team members to become productive on quickly
- Model-driven Pure-Ruby version support is being scaled back
- Versus code-driven approach

Chef

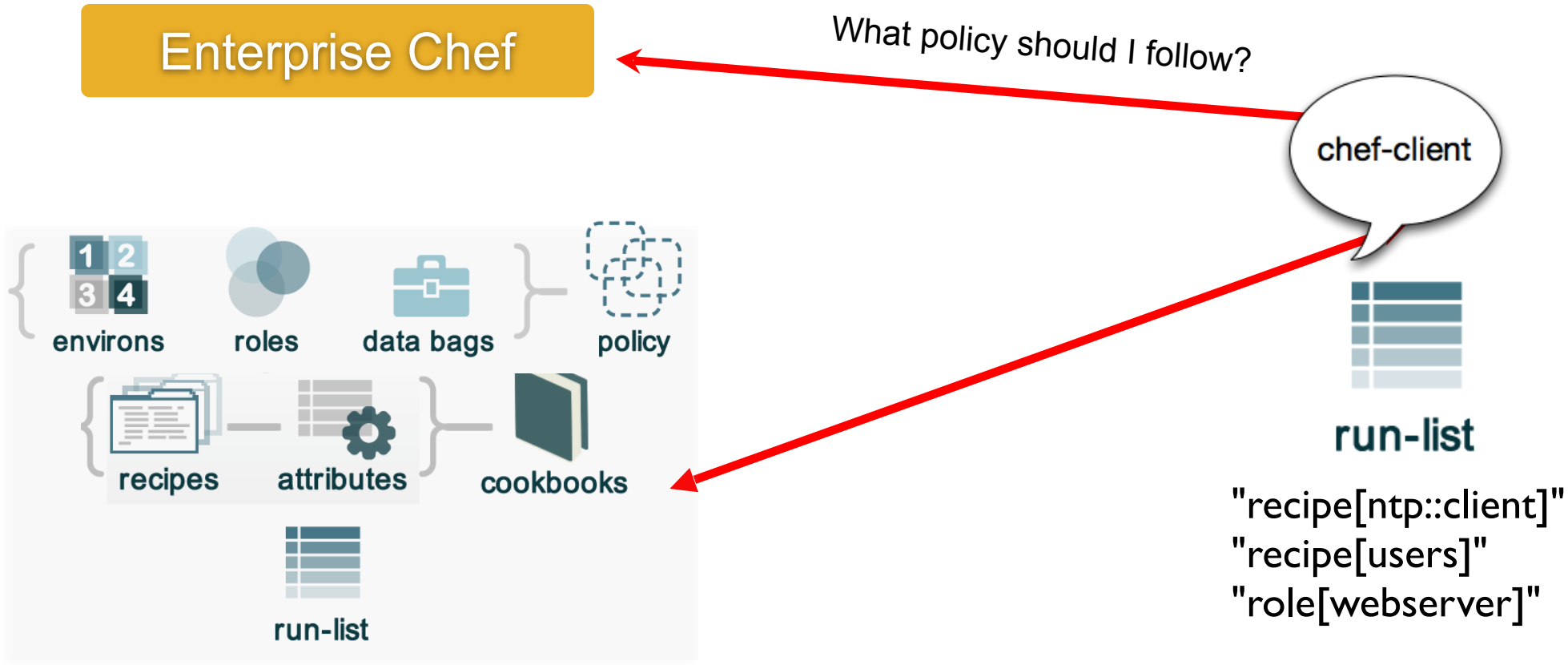
- Chef is the ninja of DevOps tools
- Comprehensive utility for:
  - provisioning
  - managing
  - automating entire infrastructures
- Uses a client/server architecture to facilitate application of “recipes” to infrastructure components
- Written in Ruby

Chef: Concepts



- Node
- Role
- Resource
- Recipe
- Cookbook
- Runlist

Chef: Run List



Chef: When/Why

- Organizations looking for a more mature solution for a heterogeneous environment
- Good for development-focused teams

Chef: Cons

- Not simple—can lead to large code bases and complex environments
- Doesn’t support push functionality
- Steep learning curve

Chef: Cons

- “Knife” tool eases installation burdens
- Large collection of modules and configuration recipes
- Heavy focus on Git gives it strong version control capabilities
- Code-driven approach gives you more control configurations

Saltstack

- CLI-based tool that can be set up as a master-slave model or a non-centralized model.
- Python
- Push method of communication with clients
- Provides for grouping of clients and configuration templates

Saltstack: When

- Scalability and resiliency are a big concern.
- Oriented to system administrators due to its usability

Saltstack: Pros

- After setup, fairly straightforward
- DSL is robust with features
- I/O and configs are consistent-> YAML.
- Strong community
- High scalability and resiliency in the master model
- Transparency is excellent. Easy to see what's happening internally

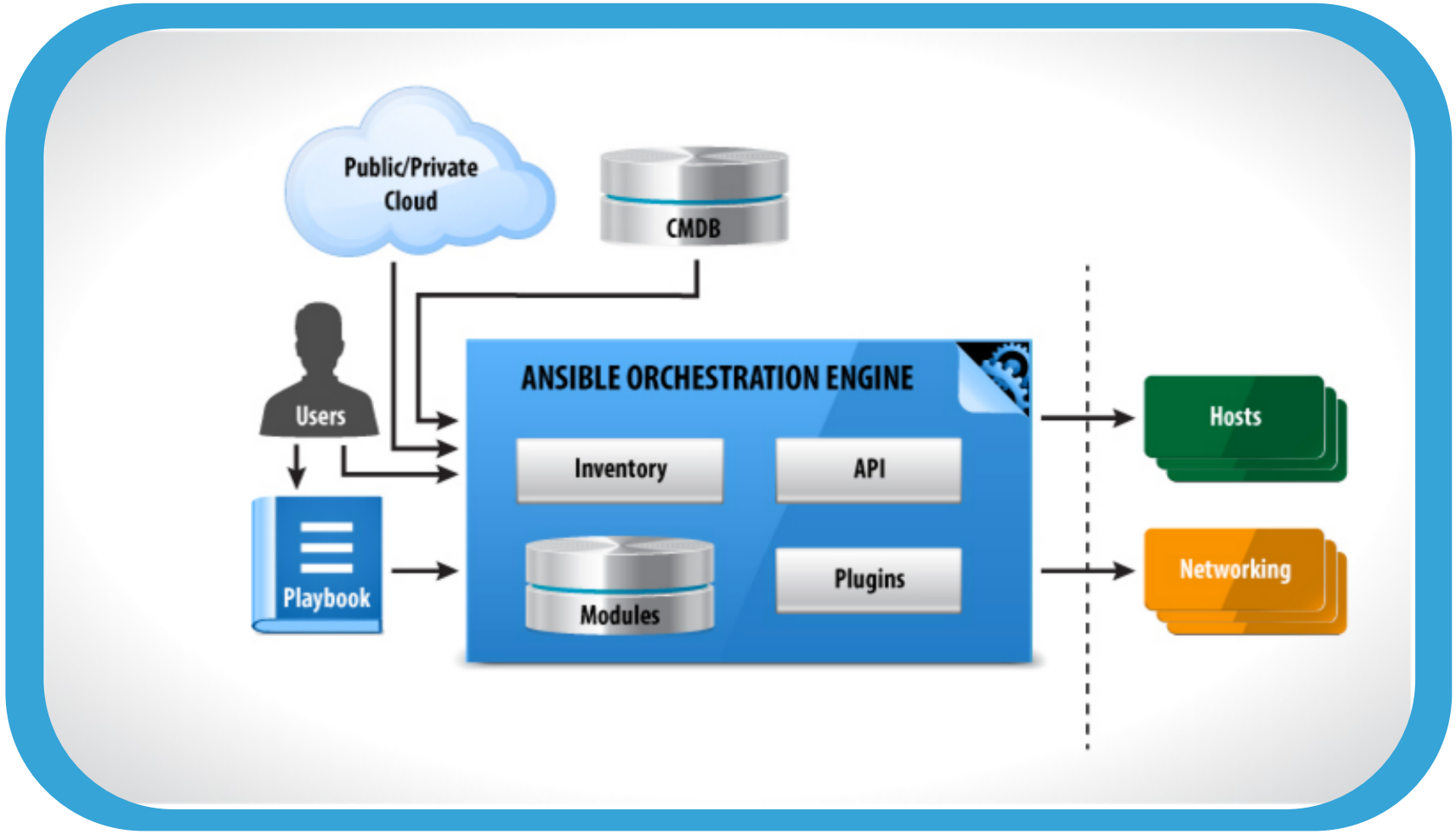
Saltstack: Cons

- Documentation is weak
- Incomplete web UI
- Non Linux OS support weak
- Difficult to set up and to pick up for new users.

Ansible

- Agentless configuration management tool
- Written in Python
- Multi-node (like chef)
- Seeks to be simple
- Similar to Vagrant

Ansible: Architecture



Ansible: Key Concepts

- Module
- Playbook

Ansible: When/Why

- Getting up and running quickly
- Ease is important to you
- You don’t want to install agents on remote nodes or managed servers
- It’s good if your focus is more on the system admin side.

Ansible: Pros

- SSH-based, so it doesn’t require installing any agents on remote nodes.
- Easy learning curve thanks to the use of YAML.
- Playbook structure is simple and clearly structured.
- Has a variable registration feature that enables tasks to register variables for later tasks
- Much more streamlined code base than some other tools

Ansible: Cons

- Less powerful than tools based in other programming languages.
- Does its logic through its DSL, which means checking in on the documentation frequently until you learn it
- Variable registration is required for even basic functionality, which can make easier tasks more complicated
- Introspection is poor. Difficult to see the values of variables within the playbooks
- No consistency between formats of input, output, and config files
- Struggles with performance speed at times.

Teraform

- Higher level abstraction of CM
- Written in G (golang)
- Focuses on data center wide operations
- Integrates with other CM tools like Chef and Puppet
- From the makers of Vagrant

Teraform: Configuration

Like Vagrant:

- Command line tool
- Uses text files to describe the infrastructure

Teraform: When

- The planning phase plain and simple:
  - Killer feature
- When the holes in a young product are not in areas you need support. (Missing resources)

Teraform: Pros

- Supports many cloud providers and can run on your local machine
- Vendor neutral
- Separate planning and execution phase—killer feature
- Young—stability can be an issue

Teraform: Cons

- Limited support for many AWS components
- Issues with state management—state file sharing between developers is chaos
- DSL is young and missing some functionality

Cloud Foundry

- Provision operating systems and middleware.
- Manage network security safe guards to prevent security threats.
- Manage application connections to external sources including databases and legacy middleware.
- Provides 4 levels of HA, as well as built in load balancing for scale.

Cloud Foundry (Cont.)

- Supports multi-tenant environments—each line of business can operate with a discrete quota and isolated system access.
- Provision next generation data services including NOSQL databases, traditional databases and Hadoop clusters.
- We provide horizontal and vertical scaling for the underlying IaaS so that you can scale your infrastructure in lock step with your Business.
- Provides a built-in log aggregation service.

Cloud Foundry: Open Source Capabilities

CLOUD FOUNDRY™

Automatic AppServer & OS Configuration with Buildpacks ("just push your app")

Application Containerization & Cluster Scheduling

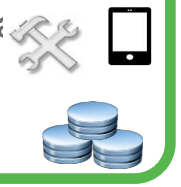
Application Network Security Groups

Application to Services Binding and Access

App Health Mng, Load Balancing, Rapid Scaling, Availability Zones

Policy, Identity and Roles Management

Native & Extended Data Mobile and Platform Services



IaaS Provisioning, Scaling & Configuration

Basic logging as a service

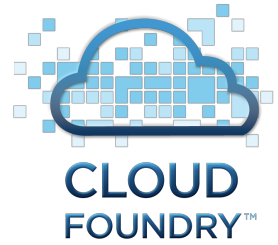
Deploy, Operate, Update & Scale with minimal downtime

vmware

amazon web services™

openstack™

Cloud Foundry: Components



BOSH CLI
CF CLI

HA Proxy LB

Dynamic Router

Cloud Controller | Health Manager

UAA | Login Server

DEA Pool

Apps | Apps

Messaging (NATS)

Metrics Collection | Log Aggregator

Elastic Runtime

Service Broker

Service Nodes

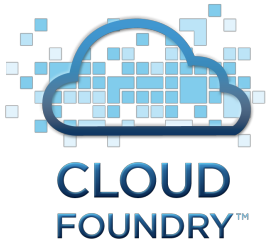
Service

Service Broker

Service Nodes

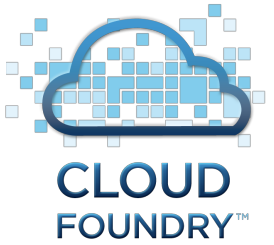
Service

Cloud Foundry: When



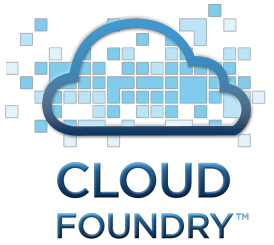
- You are running on AWS
- Significant number of ready to go resource definitions

Cloud Foundry: Pros



- Support for all AWS components
- No state management issues
- Templates written in son and very easy to understand
- Very stable

Cloud Foundry: Cons



- AWS only and is a hosted service
- Can makes changes without prior notice - lack of transparency
- Issues with rollback mechanism

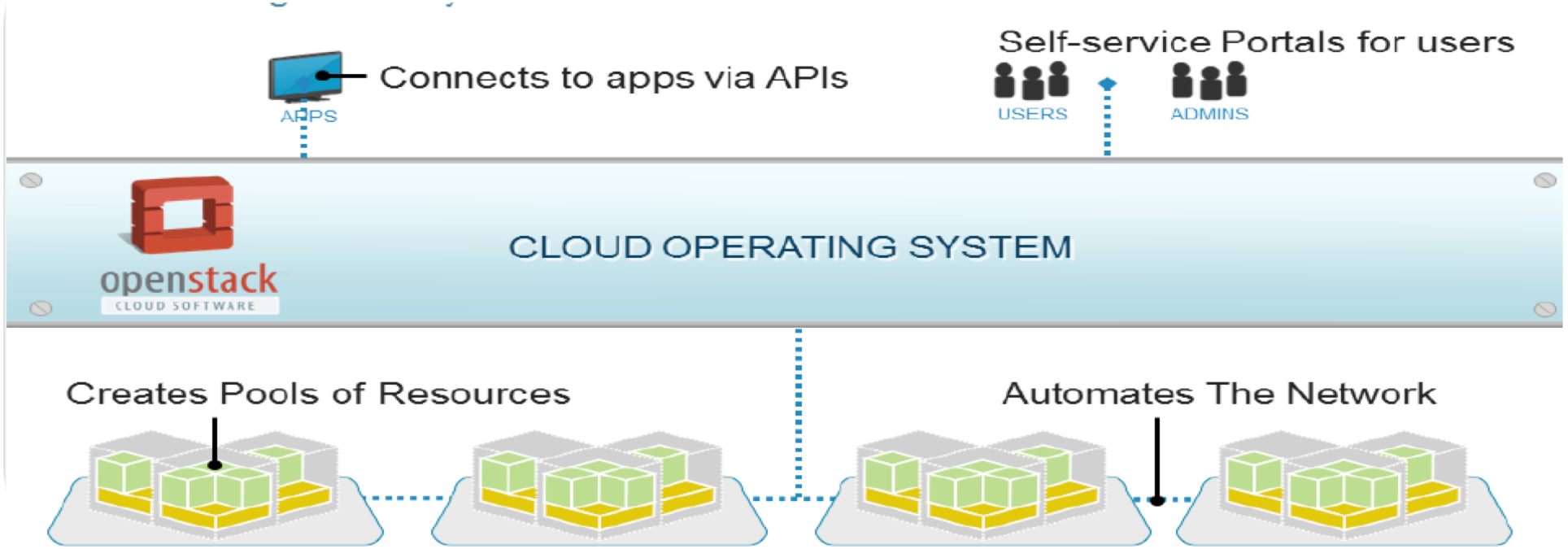
OpenStack

- Software platform for cloud computing
- Infrastructure-as-a-service (IaaS)
- The software platform consists of interrelated components that control hardware pools of processing, storage, and networking resources throughout a data center.

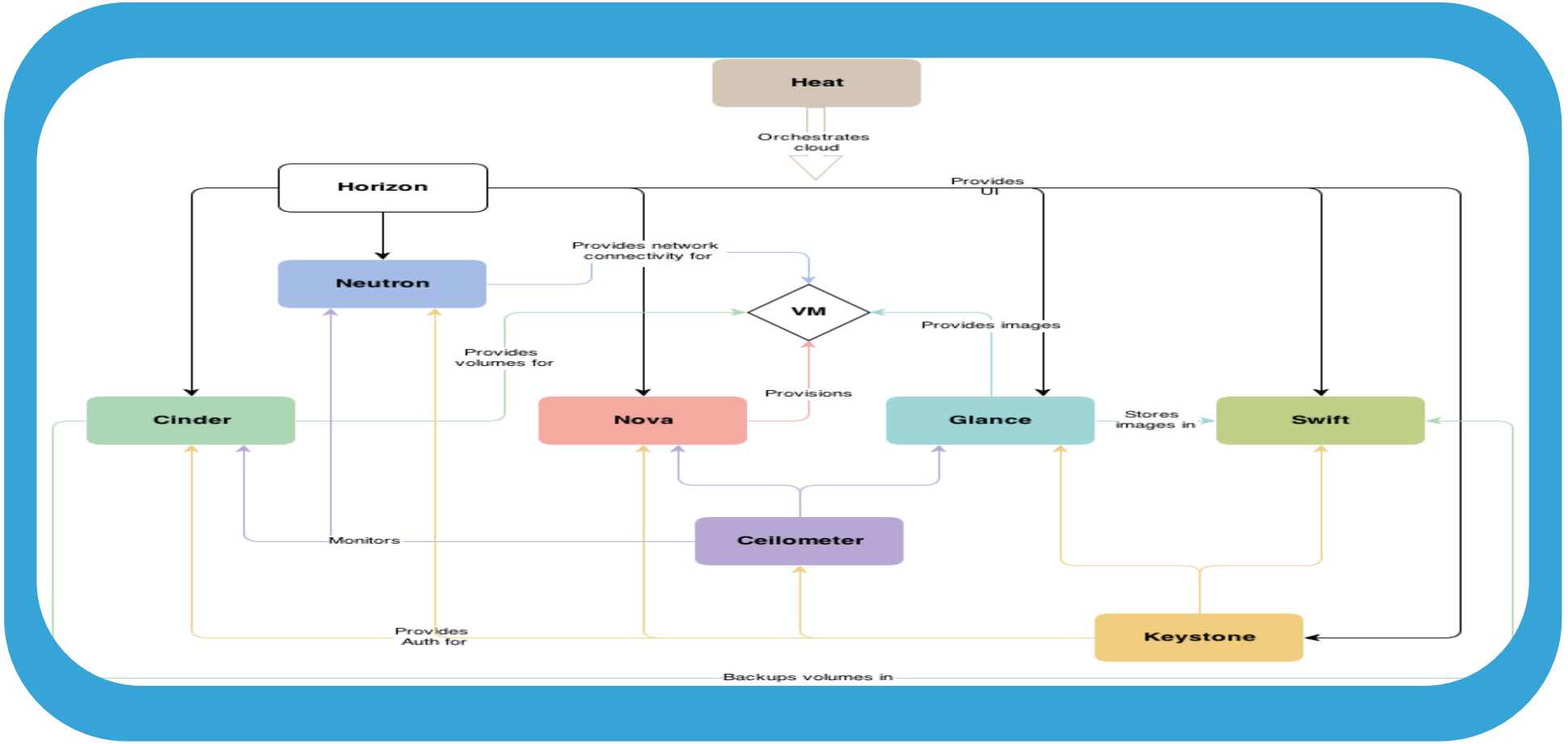
Automation and Orchestration of IT Resources

Solution: OpenStack, the Cloud Operating System

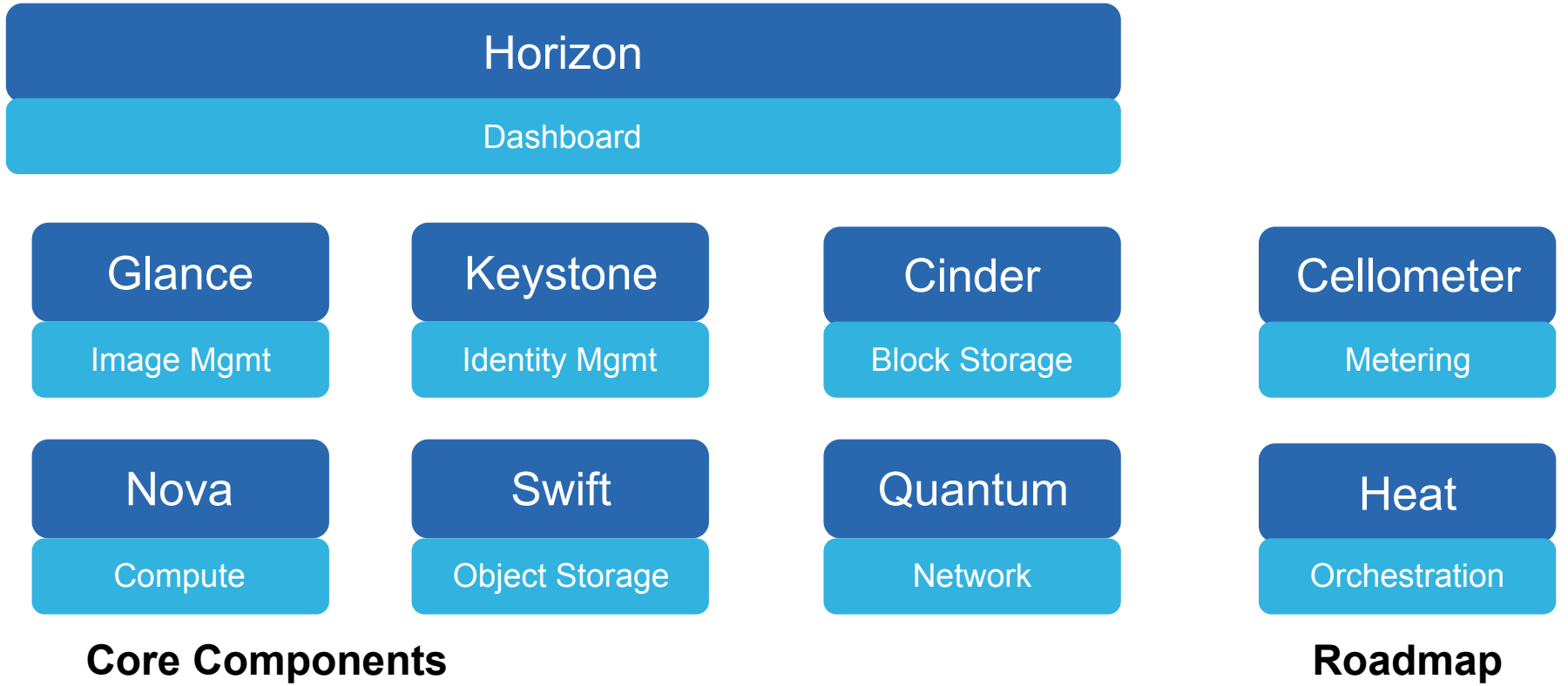
A new management layer that adds automation and control



In a Loosely Coupled Architecture



By Leveraging Various Open Source Projects



Rackspace Private Cloud Reference Architecture